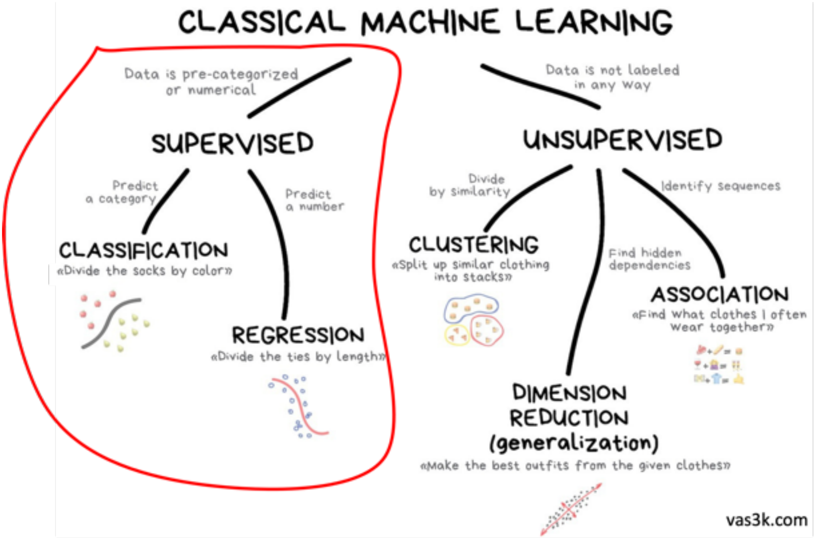
Learning with Linear Models: Foundations of Machine Learning

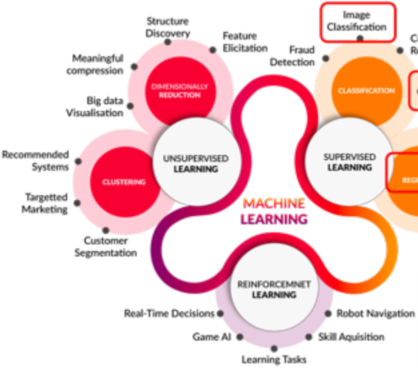
Mário A. T. Figueiredo

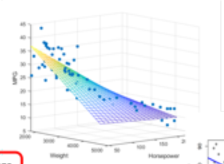
15th Lisbon Machine Learning Summer School, LxMLS 2025

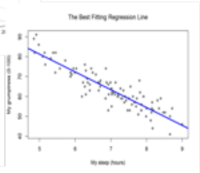

CLASSICAL MACHINE LEARNING
Data is pre-categorized or numerical
Data is not labeled in any way
SUPERVISED
UNSUPERVISED
Predict a category
Predict a number
Divide by similarity
Identify sequences
CLASSIFICATION
«Divide the socks by color»
REGRESSION
«Divide the ties by length»
CLUSTERING
«Split up similar clothing into stacks»
Find hidden dependencies
ASSOCIATION
«Find what clothes I often wear together»
DIMENSION REDUCTION (generalization)
«Make the best outfits from the given clothes»
vas3k.com

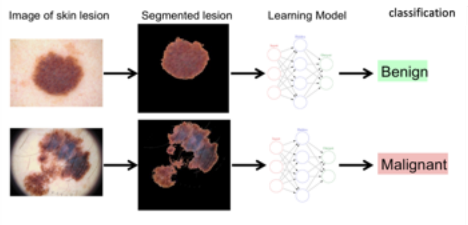
Supervised Learning

Types of Machine Learning









Why Study Linear Models?

- In 2025, deep neural networks (DNNs) are ubiquitous!

Why Study Linear Models?

- In 2025, deep neural networks (DNNs) are ubiquitous!
- Why a lecture on linear models?
  - ✓ Underlying machine learning (ML) core concepts are the same.

Why Study Linear Models?

- In 2025, deep neural networks (DNNs) are ubiquitous!
- Why a lecture on linear models?
  - ✓ Underlying machine learning (ML) core concepts are the same.
  - ✓ Theory (statistics and optimization) is easier to understand.

Why Study Linear Models?

- In 2025, deep neural networks (DNNs) are ubiquitous!
- Why a lecture on linear models?
  - ✓ Underlying machine learning (ML) core concepts are the same.
  - ✓ Theory (statistics and optimization) is easier to understand.
  - ✓ Still widely used (specially if data is scarce)

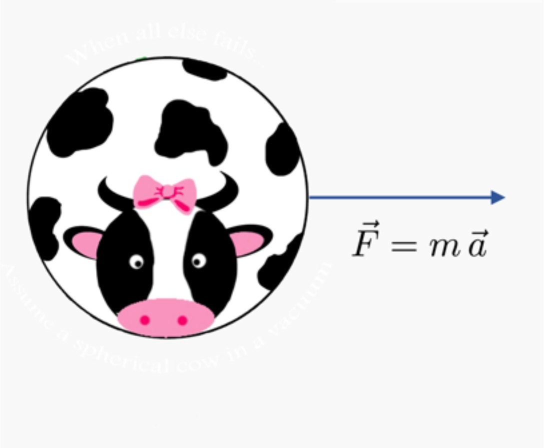
Why Study Linear Models?

- In 2025, deep neural networks (DNNs) are ubiquitous!
- Why a lecture on linear models?
  - ✓ Underlying machine learning (ML) core concepts are the same.
  - ✓ Theory (statistics and optimization) is easier to understand.
  - ✓ Still widely used (specially if data is scarce)
  - ✓ They are a component of DNNs.

Why Study Linear Models?

- In 2025, deep neural networks (DNNs) are ubiquitous!
- Why a lecture on linear models?
  - ✓ Underlying machine learning (ML) core concepts are the same.
  - ✓ Theory (statistics and optimization) is easier to understand.
  - ✓ Still widely used (specially if data is scarce)
  - ✓ They are a component of DNNs.
  - ✓ Natural starting point for studying ML.

Spherical Cow



Good Advice

Eduardo Ordax · 2nd

Generative AI Lead @ AWS (90k+) | Startup Advisor | Public ...

3d · Edited ·

+ Follow

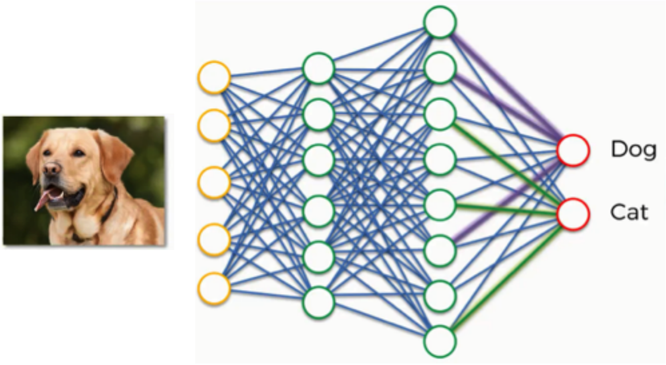
Math Is All You Need! (Or at Least, the Best Place to Start for AI)

I recently came across this advice:

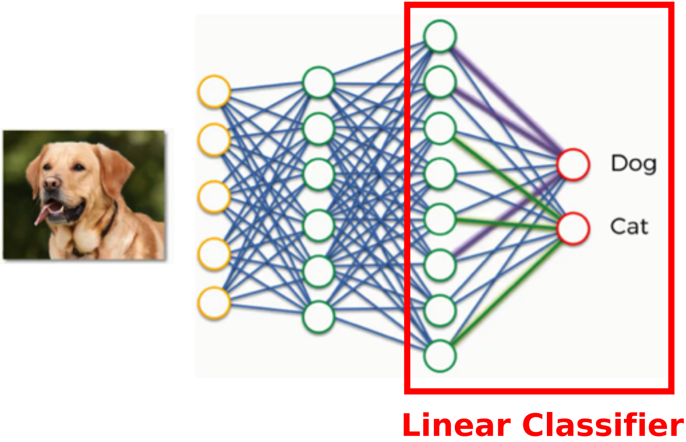
"Don't get an AI degree—the curriculum will be outdated before you graduate. Instead, build a strong foundation in math, statistics, or physics, and stay up to date with AI through code-focused books, blogs, and research papers."

In short, if you're solid in math and willing to refine your coding skills, you'll be a valuable asset to any top AI lab.

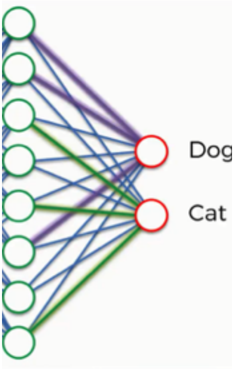
Linear Classifiers and Neural Networks



Linear Classifiers and Neural Networks

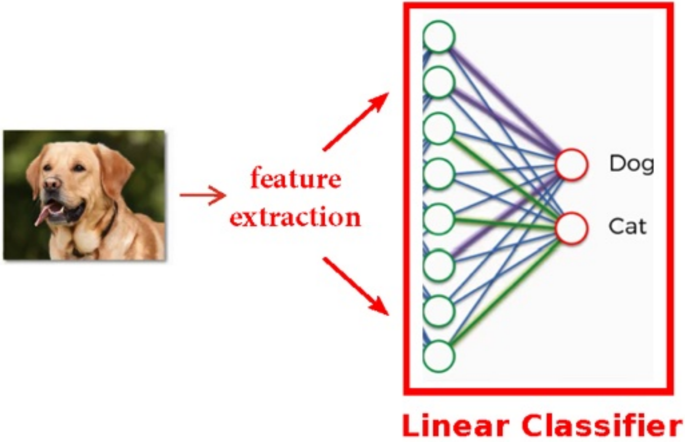


Linear Classifiers and Neural Networks



Linear Classifier

Linear Classifiers and Neural Networks



Outline

1. Introduction

2. Regression

3. Classification

4. Optimization for Supervised Learning

Inputs and Outputs

- Input $\boldsymbol{x} \in \mathcal{X}$
  - ✓ e.g., a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, features of a car, features of a house, ...

Inputs and Outputs

- Input $\boldsymbol{x} \in \mathcal{X}$
  - ✓ e.g., a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, features of a car, features of a house, ...
- Output $\boldsymbol{y} \in \mathcal{Y}$
  - ✓ e.g., fake/true, spam/legitimate, an identity, a diagnostic, fraud/legitimate, fuel consumption, price, ...

Inputs and Outputs

- Input $\boldsymbol{x} \in \mathcal{X}$
  - ✓ e.g., a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, features of a car, features of a house, ...
- Output $\boldsymbol{y} \in \mathcal{Y}$
  - ✓ e.g., fake/true, spam/legitimate, an identity, a diagnostic, fraud/legitimate, fuel consumption, price, ...
- Input/output pair: $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$

Inputs and Outputs

- Input $\boldsymbol{x} \in \mathcal{X}$

  ✓ e.g., a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, features of a car, features of a house, ...

- Output $\boldsymbol{y} \in \mathcal{Y}$

  ✓ e.g., fake/true, spam/legitimate, an identity, a diagnostic, fraud/legitimate, fuel consumption, price, ...

- Input/output pair: $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$

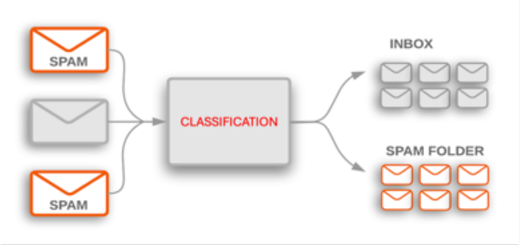
  ✓ e.g., a **news article** together with a **topic**

  ✓ e.g., a **sentence** together with its **translation**

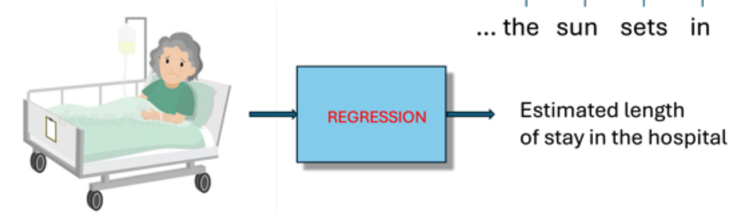
  ✓ e.g., a **sequence of words (tokens)** together with the **next word**

  ✓ e.g., an **image** partitioned into **segmentation regions**

Decisions


SPAM
SPAM
CLASSIFICATION
INBOX
SPAM FOLDER


PREDICTION
... the sun sets in the west


REGRESSION
Estimated length
of stay in the hospital

Optimal Decisions

- Goal: find a "good" decision function: $h : \mathcal{X} \rightarrow \mathcal{Y}$

Optimal Decisions

- Goal: find a "good" decision function: $h : \mathcal{X} \to \mathcal{Y}$
- Ideal situation: joint distribution $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$ is known.

Optimal Decisions

- Goal: find a "good" decision function: $h : \mathcal{X} \to \mathcal{Y}$
- Ideal situation: joint distribution $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y})$ is known.
- We know how to assess decisions, *i.e.*, we have a loss function:

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = \text{loss of deciding } \widehat{\boldsymbol{y}} \text{ if the truth is } \boldsymbol{y}$$

Optimal Decisions

- Goal: find a "good" decision function: $h : \mathcal{X} \to \mathcal{Y}$
- Ideal situation: joint distribution $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y})$ is known.
- We know how to assess decisions, *i.e.*, we have a loss function:

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = \text{loss of deciding } \widehat{\boldsymbol{y}} \text{ if the truth is } \boldsymbol{y}$$

- Optimal decision functions minimize the expected loss or risk:

$$\begin{aligned} h^* &= \arg\min_{h\in\mathcal{H}} \mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[L(\boldsymbol{Y}, h(\boldsymbol{X}))\big] \\ &= \arg\min_{h\in\mathcal{H}} \int_{\mathcal{X}}\int_{\mathcal{Y}} f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y}) L(\boldsymbol{y}, h(\boldsymbol{x}))\, d\boldsymbol{y}\, d\boldsymbol{x}, \end{aligned}$$

where $\mathcal{H}$ is some set of allowed functions.

Optimal Decisions

- Goal: find a "good" decision function: $h : \mathcal{X} \to \mathcal{Y}$
- Ideal situation: joint distribution $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$ is known.
- We know how to assess decisions, *i.e.*, we have a loss function:

$$L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = \text{loss of deciding } \widehat{\boldsymbol{y}} \text{ if the truth is } \boldsymbol{y}$$

- Optimal decision functions minimize the expected loss or risk:

$$\begin{aligned} h^* &= \arg\min_{h \in \mathcal{H}} \mathbb{E}_{\boldsymbol{X},\boldsymbol{Y}}\big[L(\boldsymbol{Y}, h(\boldsymbol{X}))\big] \\ &= \arg\min_{h \in \mathcal{H}} \int_{\mathcal{X}} \int_{\mathcal{Y}} f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y}) L(\boldsymbol{y}, h(\boldsymbol{x}))\, d\boldsymbol{y}\, d\boldsymbol{x}, \end{aligned}$$

  where $\mathcal{H}$ is some set of allowed functions.

- Unfortunately, $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$ is seldom known: use supervised learning

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...
- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1,\boldsymbol{y}_1), ..., (\boldsymbol{x}_n,\boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad\qquad (\boldsymbol{x}_i \in \mathcal{X},\; \boldsymbol{y}_i \in \mathcal{Y})$$

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...

- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad\qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

- Same goal: learn a predictor/decision function $h : \mathcal{X} \to \mathcal{Y}$.

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y})$, ...

- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad\qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

- Same goal: learn a predictor/decision function $h : \mathcal{X} \to \mathcal{Y}$.

- Two standard approaches:

  ✓ Generative: estimate $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x}, \boldsymbol{y})$ from $\mathcal{D}$; go back to the previous slide.

  ✓ Discriminative: replace the expected risk with the empirical risk,

$$h^* = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(\boldsymbol{y}_i, h(\boldsymbol{x}_i)) \qquad \text{(empirical risk minimization – ERM)}$$

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...

- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1,\boldsymbol{y}_1), ..., (\boldsymbol{x}_n,\boldsymbol{y}_n) \in \mathcal{X}\times\mathcal{Y} \qquad\qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

- Same goal: learn a predictor/decision function $h : \mathcal{X} \to \mathcal{Y}$.

- Two standard approaches:

  ✓ Generative: estimate $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$ from $\mathcal{D}$; go back to the previous slide.

  ✓ Discriminative: replace the expected risk with the empirical risk,

$$h^* = \arg\min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} L(\boldsymbol{y}_i, h(\boldsymbol{x}_i)) \qquad \text{(empirical risk minimization – ERM)}$$

- This lecture focuses on discriminative supervised learning.

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...
- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad\qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...
- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad\qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

- Same goal: learn a predictor/decision function $h : \mathcal{X} \rightarrow \mathcal{Y}$.

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...

- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1,\boldsymbol{y}_1), ..., (\boldsymbol{x}_n,\boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

- Same goal: learn a predictor/decision function $h : \mathcal{X} \to \mathcal{Y}$.

- Two standard approaches:

  ✓ Generative: estimate $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$ from $\mathcal{D}$; go back to the previous slide.

  ✓ Discriminative: replace the expected risk with the empirical risk,

$$h^* = \arg\min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} L(\boldsymbol{y}_i, h(\boldsymbol{x}_i)) \qquad \text{(empirical risk minimization – ERM)}$$

Supervised Learning

- Rather than knowing $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$, ...
- ... we have a collection of input/output pairs (training data)

$$\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n) \in \mathcal{X} \times \mathcal{Y} \qquad (\boldsymbol{x}_i \in \mathcal{X},\ \boldsymbol{y}_i \in \mathcal{Y})$$

- Same goal: learn a predictor/decision function $h : \mathcal{X} \to \mathcal{Y}$.
- Two standard approaches:
  - ✓ Generative: estimate $f_{\boldsymbol{X},\boldsymbol{Y}}(\boldsymbol{x},\boldsymbol{y})$ from $\mathcal{D}$; go back to the previous slide.
  - ✓ Discriminative: replace the expected risk with the empirical risk,

$$h^* = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(\boldsymbol{y}_i, h(\boldsymbol{x}_i)) \qquad \text{(empirical risk minimization – ERM)}$$

- This lecture focuses on discriminative supervised learning.

What about self-supervised learning?

What about self-supervised learning?

- In its most basic form, it's just supervised learning with programmatically defined training outputs.

What about self-supervised learning?

- In its most basic form, it's just supervised learning with programmatically defined training outputs.
- Classical example: next word prediction.

Regression, Classification, and Variants

- Regression: quantitative $\mathcal{Y}$. Classification: categorical $\mathcal{Y}$ (no order).

Regression, Classification, and Variants

- Regression: quantitative $\mathcal{Y}$. Classification: categorical $\mathcal{Y}$ (no order).
- Simple regression: $\mathcal{Y} = \mathbb{R}$, or $\mathcal{Y} = [0,\, 1]$, or $\mathcal{Y} = \mathbb{R}_+$, or ...
  - ✓ e.g., given a news article, how much time a user will spend reading it?

Regression, Classification, and Variants

- Regression: quantitative $\mathcal{Y}$. Classification: categorical $\mathcal{Y}$ (no order).
- Simple regression: $\mathcal{Y} = \mathbb{R}$, or $\mathcal{Y} = [0, 1]$, or $\mathcal{Y} = \mathbb{R}_+$, or ...
  - ✓ e.g., given a news article, how much time a user will spend reading it?
- Multivariate/multiple regression: $\mathcal{Y} = \mathbb{R}^K$, or $\mathcal{Y} = \mathbb{R}_+^K$, or $\mathcal{Y} = \Delta_K$, ...
  - ✓ e.g., denoise an image, estimate class probabilities, ...

Regression, Classification, and Variants

- Regression: quantitative $\mathcal{Y}$. Classification: categorical $\mathcal{Y}$ (no order).
- Simple regression: $\mathcal{Y} = \mathbb{R}$, or $\mathcal{Y} = [0,\, 1]$, or $\mathcal{Y} = \mathbb{R}_+$, or ...
  - ✓ e.g., given a news article, how much time a user will spend reading it?
- Multivariate/multiple regression: $\mathcal{Y} = \mathbb{R}^K$, or $\mathcal{Y} = \mathbb{R}_+^K$, or $\mathcal{Y} = \Delta_K$, ...
  - ✓ e.g., denoise an image, estimate class probabilities, ...
- Binary classification: $\mathcal{Y} = \{0,\, 1\}$, or $\mathcal{Y} = \{a, b\}$, ...
  - ✓ e.g., spam detection, fraud detection, target detection, ...

Regression, Classification, and Variants

- Regression: quantitative $\mathcal{Y}$. Classification: categorical $\mathcal{Y}$ (no order).
- Simple regression: $\mathcal{Y} = \mathbb{R}$, or $\mathcal{Y} = [0,\, 1]$, or $\mathcal{Y} = \mathbb{R}_+$, or ...
  - ✓ e.g., given a news article, how much time a user will spend reading it?
- Multivariate/multiple regression: $\mathcal{Y} = \mathbb{R}^K$, or $\mathcal{Y} = \mathbb{R}_+^K$, or $\mathcal{Y} = \Delta_K$, ...
  - ✓ e.g., denoise an image, estimate class probabilities, ...
- Binary classification: $\mathcal{Y} = \{0,\, 1\}$, or $\mathcal{Y} = \{a, b\}$, ...
  - ✓ e.g., spam detection, fraud detection, target detection, ...
- Multi-class classification: $\mathcal{Y} = \{1, 2, \ldots, K\}$ (order is irrelevant!)
  - ✓ e.g., topic classification, image classification, word prediction, ...

Regression, Classification, and Variants

- Regression: quantitative $\mathcal{Y}$. Classification: categorical $\mathcal{Y}$ (no order).
- Simple regression: $\mathcal{Y} = \mathbb{R}$, or $\mathcal{Y} = [0,\, 1]$, or $\mathcal{Y} = \mathbb{R}_+$, or ...
  - ✓ e.g., given a news article, how much time a user will spend reading it?
- Multivariate/multiple regression: $\mathcal{Y} = \mathbb{R}^K$, or $\mathcal{Y} = \mathbb{R}_+^K$, or $\mathcal{Y} = \Delta_K$, ...
  - ✓ e.g., denoise an image, estimate class probabilities, ...
- Binary classification: $\mathcal{Y} = \{0,\, 1\}$, or $\mathcal{Y} = \{a, b\}$, ...
  - ✓ e.g., spam detection, fraud detection, target detection, ...
- Multi-class classification: $\mathcal{Y} = \{1, 2, \ldots, K\}$ (order is irrelevant!)
  - ✓ e.g., topic classification, image classification, word prediction, ...
- Structured classification: $\mathcal{Y}$ exponentially large and structured
  - ✓ e.g., machine translation, caption generation, image segmentation, ...

Feature Representations

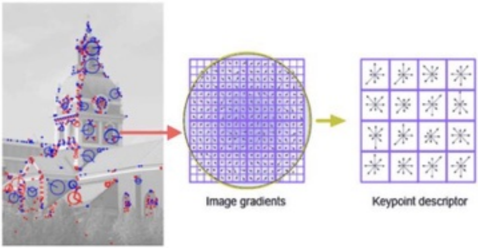
- Feature engineering is (was?) an important step for linear models:

Feature Representations

- Feature engineering is (was?) an important step for linear models:
  - ✓ Bag-of-words features for text, parts-of-speech, ...

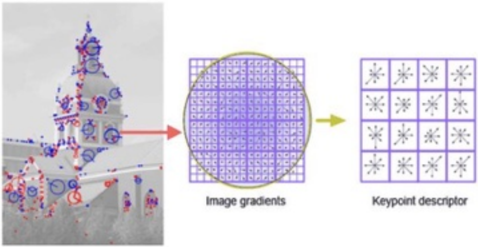
Feature Representations

- Feature engineering is (was?) an important step for linear models:
  - ✓ Bag-of-words features for text, parts-of-speech, ...
  - ✓ SIFT features and wavelet representations in computer vision



Feature Representations

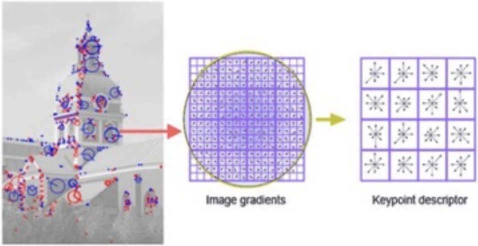
- Feature engineering is (was?) an important step for linear models:
  - ✓ Bag-of-words features for text, parts-of-speech, ...
  - ✓ SIFT features and wavelet representations in computer vision



  - ✓ Other categorical, Boolean, continuous features, ...

Feature Representations

- Feature engineering is (was?) an important step for linear models:
  - ✓ Bag-of-words features for text, parts-of-speech, ...
  - ✓ SIFT features and wavelet representations in computer vision



  - ✓ Other categorical, Boolean, continuous features, ...
  - ✓ Decades of research in machine learning, natural language processing, computer vision, image analysis, speech processing, ...

Feature Representations

- Feature represent information about an "object" x

Feature Representations

- Feature represent information about an "object" $\boldsymbol{x}$
- Typical approach: a feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^d$

Feature Representations

- Feature represent information about an "object" $\boldsymbol{x}$
- Typical approach: a feature map $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathbb{R}^d$
- $\boldsymbol{\phi}(\boldsymbol{x})$ is a (maybe high-dimensional) feature vector

Feature Representations

- Feature represent information about an "object" $\boldsymbol{x}$
- Typical approach: a feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^d$
- $\boldsymbol{\phi}(\boldsymbol{x})$ is a (maybe high-dimensional) feature vector
- Feature vectors may mix categorical and continuous features

Feature Representations

- Feature represent information about an "object" $\boldsymbol{x}$
- Typical approach: a feature map $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathbb{R}^d$
- $\boldsymbol{\phi}(\boldsymbol{x})$ is a (maybe high-dimensional) feature vector
- Feature vectors may mix categorical and continuous features
- Categorical features are often reduced to one-hot binary features:

$$\boldsymbol{e}_y := (0, \ldots, 0, \underbrace{1}_{\text{position } y}, 0, \ldots, 0) \in \{0,\, 1\}^K \text{ represents class } y$$

Feature Representations

- Feature represent information about an "object" $\boldsymbol{x}$
- Typical approach: a feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^d$
- $\boldsymbol{\phi}(\boldsymbol{x})$ is a (maybe high-dimensional) feature vector
- Feature vectors may mix categorical and continuous features
- Categorical features are often reduced to one-hot binary features:

$$\boldsymbol{e}_y := (0, \ldots, 0, \underbrace{1}_{\text{position } y}, 0, \ldots, 0) \in \{0,\, 1\}^K \text{ represents class } y$$

- Categorical features (e.g., words in a sentence) may be represented by vectors (embeddings).

Representation/Feature Engineering vs Learning

- Feature engineering (FE) is "alchemy":

  ✓ it requires deep domain knowledge
  (linguistics in NLP, vision in computer vision, ...)

  ✓ usually very time-consuming

Representation/Feature Engineering vs Learning

- Feature engineering (FE) is "alchemy":
  - ✓ it requires deep domain knowledge
    (linguistics in NLP, vision in computer vision, ...)
  - ✓ usually very time-consuming
- FE allows incorporating knowledge, it is a form of inductive bias

Representation/Feature Engineering vs Learning

- Feature engineering (FE) is "alchemy":
  - ✓ it requires deep domain knowledge (linguistics in NLP, vision in computer vision, ...)
  - ✓ usually very time-consuming
- FE allows incorporating knowledge, it is a form of inductive bias
- FE is still widely used in practice, namely in data-scarce scenarios

Representation/Feature Engineering vs Learning

- Feature engineering (FE) is "alchemy":
  - ✓ it requires deep domain knowledge
    (linguistics in NLP, vision in computer vision, ...)
  - ✓ usually very time-consuming
- FE allows incorporating knowledge, it is a form of inductive bias
- FE is still widely used in practice, namely in data-scarce scenarios
- Modern alternative: representation learning a.k.a. deep learning

Representation/Feature Engineering vs Learning

- Feature engineering (FE) is "alchemy":
  - ✓ it requires deep domain knowledge
    (linguistics in NLP, vision in computer vision, ...)
  - ✓ usually very time-consuming
- FE allows incorporating knowledge, it is a form of inductive bias
- FE is still widely used in practice, namely in data-scarce scenarios
- Modern alternative: representation learning a.k.a. deep learning

Monday's lecture

Outline

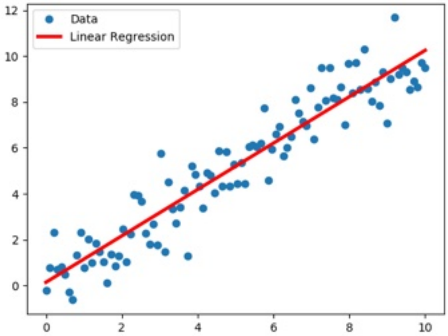
1 Introduction

2 Regression

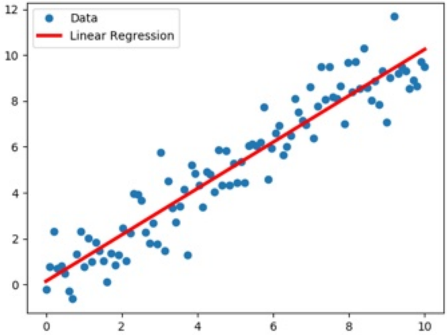
3 Classification

4 Optimization for Supervised Learning

Linear Regression: A Picture



Linear Regression: A Picture



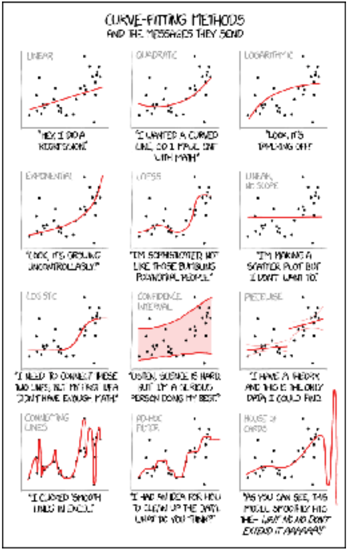
"When you're fundraising, it's AI.

When you're hiring, it's ML.

When you're implementing, it's just linear regression" (B. Schwartz)

Linear (Nonlinear) Regression

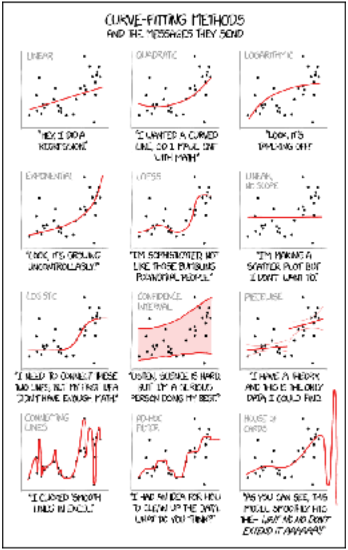
- "Linear" regression may be nonlinear (more later)



xkcd.com

Linear (Nonlinear) Regression

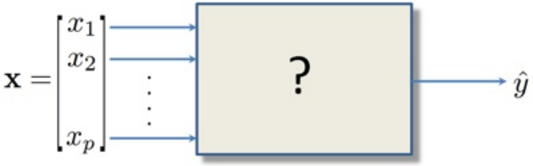
- "Linear" regression may be nonlinear (more later)
- Beware the inductive bias



xkcd.com

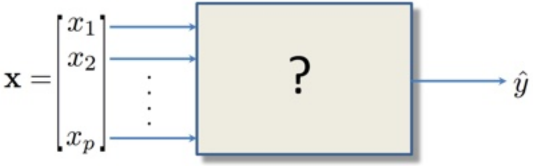
Regression

- In a nutshell: build a "machine" that predicts/estimates/guesses a quantity y from of other "quantities" / "features" $x_1, \ldots, x_p$



Regression

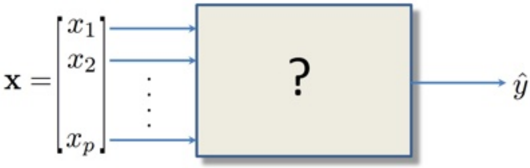
- In a nutshell: build a "machine" that predicts/estimates/guesses a quantity y from of other "quantities" / "features" $x_1,\ldots,x_p$



- Fundamental tool in data analysis, thus in much of science (biology, social sciences, economics, physics, ...) and engineering.

Regression

- In a nutshell: build a "machine" that predicts/estimates/guesses a quantity y from of other "quantities" / "features" $x_1,...,x_p$



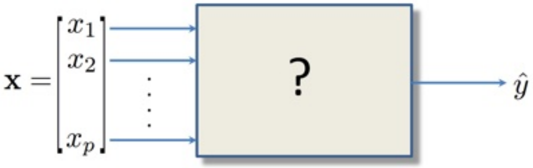
- Fundamental tool in data analysis, thus in much of science (biology, social sciences, economics, physics, ...) and engineering.
- Learning/training: given a collection of examples (training data)

$$\mathcal{D} = \big((\boldsymbol{x}_1, y_2), ..., (\boldsymbol{x}_n, y_n)\big)$$

..find the "best" predictor/decision function $h \in \mathcal{H}$.

Regression

- In a nutshell: build a "machine" that predicts/estimates/guesses a quantity y from of other "quantities" / "features" $x_1,...,x_p$



- Fundamental tool in data analysis, thus in much of science (biology, social sciences, economics, physics, ...) and engineering.
- Learning/training: given a collection of examples (training data)

$$\mathcal{D} = \big((\boldsymbol{x}_1, y_2), ..., (\boldsymbol{x}_n, y_n)\big)$$

..find the "best" predictor/decision function $h \in \mathcal{H}$.

- Notation: **bold** = vector or matrix (e.g. $\boldsymbol{x}$, $\boldsymbol{X}$).

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j$$

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j = w_0 + \boldsymbol{w}^T \boldsymbol{x}$$

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j = w_0 + \boldsymbol{w}^T \boldsymbol{x} = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle$$

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j = w_0 + \boldsymbol{w}^T \boldsymbol{x} = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$$

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j = w_0 + \boldsymbol{w}^T \boldsymbol{x} = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$$

- Standard loss: $L(y, \hat{y}) = (y - \hat{y})^2$ (why? what assumptions?)

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j = w_0 + \boldsymbol{w}^T \boldsymbol{x} = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$$

- Standard loss: $L(y, \hat{y}) = (y - \hat{y})^2$ (why? what assumptions?)

- Empirical risk and residual sum of squares (RSS)

$$R_{\text{emp}}[\boldsymbol{w}, w_0] = \frac{1}{n} \sum_{i=1}^{n} (\underbrace{w_0 + \boldsymbol{w}^T \boldsymbol{x}_i}_{\hat{y}_i} - y_i)^2 = \frac{1}{n} \text{RSS}(\boldsymbol{w}, w_0)$$

Linear Regression

- $\mathcal{H}$ only contains linear (affine) functions:

$$h(\boldsymbol{x}) = w_0 + \sum_{j=1}^{p} w_j x_j = w_0 + \boldsymbol{w}^T \boldsymbol{x} = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$$

- Standard loss: $L(y, \hat{y}) = (y - \hat{y})^2$ (why? what assumptions?)

- Empirical risk and residual sum of squares (RSS)

$$R_{\mathsf{emp}}[\boldsymbol{w}, w_0] = \frac{1}{n} \sum_{i=1}^{n} (\underbrace{w_0 + \boldsymbol{w}^T \boldsymbol{x}_i}_{\hat{y}_i} - y_i)^2 = \frac{1}{n} \mathsf{RSS}(\boldsymbol{w}, w_0)$$

- Empirical risk minimization (ERM) = least squares (LS) regression

$$(\hat{\boldsymbol{w}}, \hat{w}_0)_{\mathsf{ERM}} = (\hat{\boldsymbol{w}}, \hat{w}_0)_{\mathsf{LS}} = \arg\min_{\boldsymbol{w}, w_0} R_{\mathsf{emp}}[\boldsymbol{w}, w_0]$$

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.
- The $y_1, ..., y_n$ are conditionally independent, given $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.
- The $y_1, ..., y_n$ are conditionally independent, given $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.
- Each y_i is a Gaussian noisy version of a "clean" value $w_0 + \boldsymbol{w}^T \boldsymbol{x}_i$

$$Y_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i + N_i, \quad \text{where } N_i \sim \mathcal{N}(0, \sigma^2)$$

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.
- The $y_1, ..., y_n$ are conditionally independent, given $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.
- Each y_i is a Gaussian noisy version of a "clean" value $w_0 + \boldsymbol{w}^T\boldsymbol{x}_i$

$$Y_i = w_0 + \boldsymbol{w}^T\boldsymbol{x}_i + N_i, \quad \text{where } N_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus, $f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i) = \mathcal{N}(y_i|w_0 + \boldsymbol{w}^T\boldsymbol{x}_i, \sigma^2)$

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.
- The $y_1, ..., y_n$ are conditionally independent, given $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.
- Each y_i is a Gaussian noisy version of a "clean" value $w_0 + \boldsymbol{w}^T \boldsymbol{x}_i$

$$Y_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i + N_i, \quad \text{where } N_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus, $f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i) = \mathcal{N}(y_i | w_0 + \boldsymbol{w}^T \boldsymbol{x}_i, \sigma^2)$
- Likelihood

$$f_{Y_1,...,Y_n}(y_1, ..., y_n | \boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{w}, w_0, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(y_i | w_0 + \boldsymbol{w}^T \boldsymbol{x}_i, \sigma^2)$$

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.
- The $y_1, ..., y_n$ are conditionally independent, given $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.
- Each y_i is a Gaussian noisy version of a "clean" value $w_0 + \boldsymbol{w}^T \boldsymbol{x}_i$

$$Y_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i + N_i, \quad \text{where } N_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus, $f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i) = \mathcal{N}(y_i|w_0 + \boldsymbol{w}^T \boldsymbol{x}_i, \sigma^2)$
- Likelihood and log-likelihood function

$$f_{Y_1,...,Y_n}(y_1, ..., y_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{w}, w_0, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(y_i|w_0 + \boldsymbol{w}^T \boldsymbol{x}_i, \sigma^2)$$

$$\log f_{Y_1,...,Y_n}(y_1, ..., y_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{w}, w_0, \sigma^2) = K - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (w_0 + \boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$

Linear Regression: Statistical Assumptions

- The inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are seen as deterministic, given.
- The $y_1, ..., y_n$ are conditionally independent, given $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$.
- Each y_i is a Gaussian noisy version of a "clean" value $w_0 + \boldsymbol{w}^T \boldsymbol{x}_i$

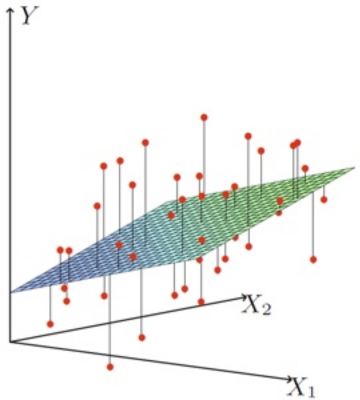
$$Y_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i + N_i, \quad \text{where } N_i \sim \mathcal{N}(0, \sigma^2)$$

- Thus, $f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i) = \mathcal{N}(y_i|w_0 + \boldsymbol{w}^T \boldsymbol{x}_i, \sigma^2)$
- Likelihood and log-likelihood function

$$f_{Y_1,...,Y_n}(y_1, ..., y_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{w}, w_0, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(y_i|w_0 + \boldsymbol{w}^T \boldsymbol{x}_i, \sigma^2)$$
$$\log f_{Y_1,...,Y_n}(y_1, ..., y_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{w}, w_0, \sigma^2) = K - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (w_0 + \boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$

- Maximum likelihood (ML) estimate: $(\hat{\boldsymbol{w}}, \hat{w}_0)_{\text{ML}} = (\hat{\boldsymbol{w}}, \hat{w}_0)_{\text{ERM}}$

Linear Regression: Another Picture



Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.

From: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", Springer, 2009.

Linear Regression: Getting Rid of w_0

- Replace each original $\boldsymbol{x}_i$ with $\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \in \mathbb{R}^{p+1}$

Linear Regression: Getting Rid of w_0

- Replace each original $\boldsymbol{x}_i$ with $\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \in \mathbb{R}^{p+1}$

- Let $\boldsymbol{w}$ now denote a $(p+1)$-dimensional vector: $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1}$

Linear Regression: Getting Rid of w_0

- Replace each original $\boldsymbol{x}_i$ with $\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \in \mathbb{R}^{p+1}$

- Let $\boldsymbol{w}$ now denote a $(p+1)$-dimensional vector: $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1}$

- The offset/bias w_0 is absorbed into $\boldsymbol{w}^T \boldsymbol{x}_i$, thus

$$\hat{\boldsymbol{w}}_{\mathsf{LS}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

Linear Regression: Getting Rid of w_0

- Replace each original $\boldsymbol{x}_i$ with $\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \in \mathbb{R}^{p+1}$

- Let $\boldsymbol{w}$ now denote a $(p+1)$-dimensional vector: $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1}$

- The offset/bias w_0 is absorbed into $\boldsymbol{w}^T \boldsymbol{x}_i$, thus

$$\hat{\boldsymbol{w}}_{\mathsf{LS}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

- From now on, we will mostly ignore w_0.

Linear Regression: Vector Notation

- Least squares regression,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

Linear Regression: Vector Notation

- Least squares regression,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

where $\boldsymbol{X}$ is the design matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n\times p} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

Linear Regression: Vector Notation

- Least squares regression,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

where $\boldsymbol{X}$ is the design matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n\times p} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

- Gradient: $\nabla_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 2\,\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$

Linear Regression: Vector Notation

- Least squares regression,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

where $\boldsymbol{X}$ is the design matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n\times p} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

- Gradient: $\nabla_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 2\,\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$
- Equating to zero,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \text{solution}_{\boldsymbol{w}} \left(\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0\right)$$

Linear Regression: Vector Notation

- Least squares regression,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

where $\boldsymbol{X}$ is the design matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n\times p} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

- Gradient: $\nabla_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 2\,\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$
- Equating to zero,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \mathrm{solution}_{\boldsymbol{w}}\left(\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0\right) = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Linear Regression: Vector Notation

- Least squares regression,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T\boldsymbol{w})^2 = \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

where $\boldsymbol{X}$ is the design matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n\times p} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

- Gradient: $\nabla_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = 2\,\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$
- Equating to zero,

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \text{solution}_{\boldsymbol{w}}\left(\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0\right) = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

... if $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, *i.e.*, $\texttt{rank}(\boldsymbol{X}) = p$, requiring $n \geq p$.

A Classic: Coefficient of Determination R^2

- Total sum of squares: $\mathsf{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2$ (variance $\times n$)

A Classic: Coefficient of Determination R^2

- Total sum of squares: TSS $= \sum_{i=1}^n (y_i - \bar{y})^2$ (variance $\times n$)
- Sum of squared residuals: SSR $= \sum_{i=1}^n (y_i - \hat{\boldsymbol{w}}^T \boldsymbol{x}_i)^2$

A Classic: Coefficient of Determination R^2

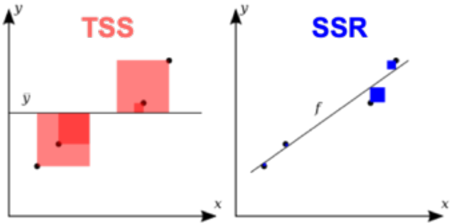
- Total sum of squares: TSS $= \sum_{i=1}^{n}(y_i - \bar{y})^2$ (variance $\times n$)
- Sum of squared residuals: SSR $= \sum_{i=1}^{n}(y_i - \hat{\boldsymbol{w}}^T \boldsymbol{x}_i)^2$
- Coefficient of determination:

$$R^2 = 1 - \frac{SSR}{TSS} = 1 - FVU \quad (1 - \text{fraction of variance unexplained})$$

A Classic: Coefficient of Determination R^2

- Total sum of squares: TSS $= \sum_{i=1}^{n}(y_i - \bar{y})^2$ (variance $\times n$)
- Sum of squared residuals: SSR $= \sum_{i=1}^{n}(y_i - \hat{\boldsymbol{w}}^T \boldsymbol{x}_i)^2$
- Coefficient of determination:

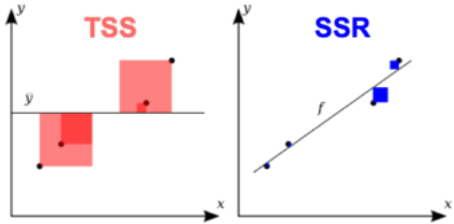
$$R^2 = 1 - \frac{SSR}{TSS} = 1 - FVU \quad (1 - \text{fraction of variance unexplained})$$

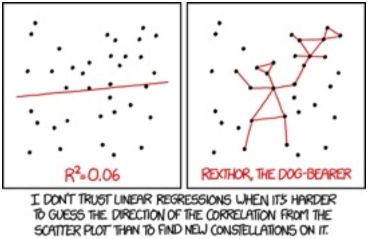


A Classic: Coefficient of Determination R^2

- Total sum of squares: TSS $= \sum_{i=1}^{n}(y_i - \bar{y})^2$ (variance $\times n$)
- Sum of squared residuals: SSR $= \sum_{i=1}^{n}(y_i - \hat{\boldsymbol{w}}^T \boldsymbol{x}_i)^2$
- Coefficient of determination:

$$R^2 = 1 - \frac{SSR}{TSS} = 1 - FVU \quad (1 - \text{fraction of variance unexplained})$$





The Geometry of Linear Regression

- Predicted values at the sampled points:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \underbrace{\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T}_{\text{hat matrix } \boldsymbol{P} \in \mathbb{R}^{n\times n}}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

The Geometry of Linear Regression

- Predicted values at the sampled points:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \underbrace{\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T}_{\text{hat matrix } \boldsymbol{P} \in \mathbb{R}^{n\times n}}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

- Matrix $\boldsymbol{P}$ is a projection matrix; it is idempotent, $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$:

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{P}$$

The Geometry of Linear Regression

- Predicted values at the sampled points:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \underbrace{\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T}_{\text{hat matrix } \boldsymbol{P} \in \mathbb{R}^{n\times n}}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

- Matrix $\boldsymbol{P}$ is a projection matrix; it is idempotent, $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$:

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{P}$$

- Clearly, $\hat{\boldsymbol{y}} \in \mathrm{range}(\boldsymbol{X})$ (span of the columns of $\boldsymbol{X}$); in fact,

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{X}\Big(\underbrace{\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2}_{\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y})}\Big)$$

The Geometry of Linear Regression

- Predicted values at the sampled points:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \underbrace{\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T}_{\text{hat matrix } \boldsymbol{P} \in \mathbb{R}^{n\times n}}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

- Matrix $\boldsymbol{P}$ is a projection matrix; it is idempotent, $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$:

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{P}$$

- Clearly, $\hat{\boldsymbol{y}} \in \mathrm{range}(\boldsymbol{X})$ (span of the columns of $\boldsymbol{X}$); in fact,

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{X}\underbrace{\left(\arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2\right)}_{\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y})} = \arg\min_{\boldsymbol{z}\in\mathrm{range}(\boldsymbol{X})} \|\boldsymbol{y} - \boldsymbol{z}\|_2^2$$

The Geometry of Linear Regression

- Predicted values at the sampled points:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) = \underbrace{\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T}_{\text{hat matrix } \boldsymbol{P} \in \mathbb{R}^{n\times n}}\boldsymbol{y} = \boldsymbol{P}\boldsymbol{y}$$

- Matrix $\boldsymbol{P}$ is a projection matrix; it is idempotent, $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$:

$$\boldsymbol{P}\boldsymbol{P} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T = \boldsymbol{P}$$

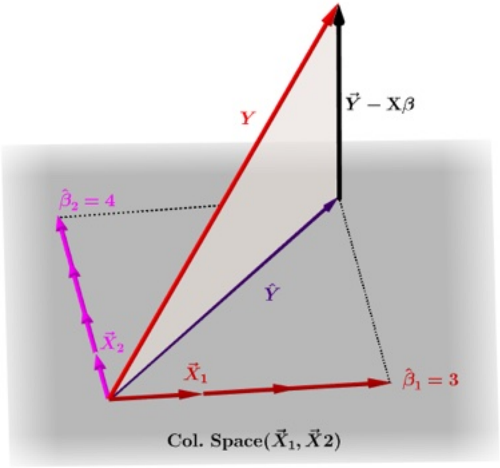
- Clearly, $\hat{\boldsymbol{y}} \in \operatorname{range}(\boldsymbol{X})$ (span of the columns of $\boldsymbol{X}$); in fact,

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{X}\underbrace{\left(\arg\min_{\boldsymbol{w}}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2\right)}_{\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y})} = \arg\min_{\boldsymbol{z}\in\operatorname{range}(\boldsymbol{X})}\|\boldsymbol{y} - \boldsymbol{z}\|_2^2$$

i.e., the orthogonal projection onto $\operatorname{range}(\boldsymbol{X})$.

Geometry of Linear Regression: Euclidean Projection

This picture is in $\mathbb{R}^n$



Going Non-Linear

- To express non-linearities, just replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$,

$$\boldsymbol{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^d, \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{d-1}(\boldsymbol{x}) \end{bmatrix} \quad \text{(typically } \phi_0(\boldsymbol{x}) = 1\text{)}$$

Going Non-Linear

- To express non-linearities, just replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$,

$$\boldsymbol{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^d, \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{d-1}(\boldsymbol{x}) \end{bmatrix} \quad \text{(typically } \phi_0(\boldsymbol{x}) = 1\text{)}$$

- Components of ϕ often called features, and ϕ a feature map.

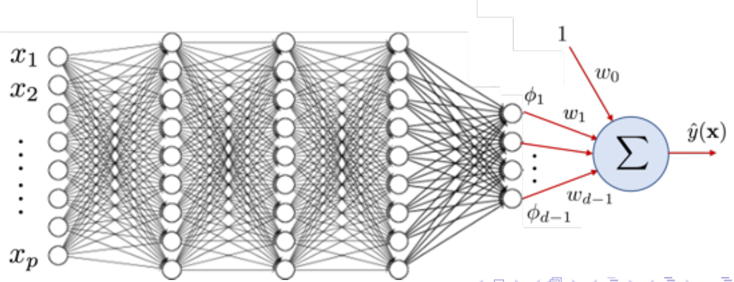
Going Non-Linear

- To express non-linearities, just replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$,

$$\boldsymbol{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^d, \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{d-1}(\boldsymbol{x}) \end{bmatrix} \quad \text{(typically } \phi_0(\boldsymbol{x}) = 1\text{)}$$

- Components of ϕ often called features, and ϕ a feature map.

- E.g., final layer of a deep network:



Going Non-Linear (but staying linear)

- To express non-linearities, just replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$,

$$\boldsymbol{\phi} : \mathbb{R}^p \rightarrow \mathbb{R}^d, \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{d-1}(\boldsymbol{x}) \end{bmatrix} \quad \text{(typically } \phi_0(\boldsymbol{x}) = 1\text{)}$$

Going Non-Linear (but staying linear)

- To express non-linearities, just replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$,

$$\boldsymbol{\phi} : \mathbb{R}^p \to \mathbb{R}^d, \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{d-1}(\boldsymbol{x}) \end{bmatrix} \quad \text{(typically } \phi_0(\boldsymbol{x}) = 1)$$

- The LS criterion becomes

$$\begin{aligned} \hat{\boldsymbol{w}}_{\mathsf{LS}} &= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i))^2 \\ &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \end{aligned}$$

re where the design matrix $\boldsymbol{X}$ is now

$$\boldsymbol{X} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \cdots & \phi_{d-1}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_n) & \cdots & \phi_{d-1}(\boldsymbol{x}_n) \end{bmatrix} \in \mathbb{R}^{n \times d}$$

Going Non-Linear (but staying linear)

- To express non-linearities, just replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$,

$$\boldsymbol{\phi} : \mathbb{R}^p \to \mathbb{R}^d, \quad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \vdots \\ \phi_{d-1}(\boldsymbol{x}) \end{bmatrix} \quad \text{(typically } \phi_0(\boldsymbol{x}) = 1\text{)}$$

- The LS criterion becomes

$$\begin{aligned} \hat{\boldsymbol{w}}_{\mathsf{LS}} &= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_i))^2 \\ &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 \;=\; \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \end{aligned}$$

re where the design matrix $\boldsymbol{X}$ is now

$$\boldsymbol{X} = \begin{bmatrix} \phi_0(\boldsymbol{x}_1) & \cdots & \phi_{d-1}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_0(\boldsymbol{x}_n) & \cdots & \phi_{d-1}(\boldsymbol{x}_n) \end{bmatrix} \in \mathbb{R}^{n \times d}$$

Example: Polynomial Regression

- Order-k polynomial regression in $\mathbb{R}$:

$$\boldsymbol{\phi}(x) = [1,\ x,\ x^2,\ \ldots,\ x^k]^T$$

Example: Polynomial Regression

- Order-k polynomial regression in $\mathbb{R}$:

$$\boldsymbol{\phi}(x) = [1,\ x,\ x^2,\ \ldots,\ x^k]^T$$

- Order-k polynomial regression in $\mathbb{R}^2$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1,\ x_1,\ x_2,\ x_1^2,\ x_1 x_2,\ x_2^2,\ \ldots,\ x_1 x_2^{k-1},\ x_2^k]^T$$

...all monomials of order up to k

Example: Polynomial Regression

- Order-k polynomial regression in $\mathbb{R}$:

$$\boldsymbol{\phi}(x) = [1,\ x,\ x^2,\ \ldots,\ x^k]^T$$

- Order-k polynomial regression in $\mathbb{R}^2$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1,\ x_1,\ x_2,\ x_1^2,\ x_1 x_2,\ x_2^2,\ \ldots,\ x_1 x_2^{k-1},\ x_2^k]^T$$

...all monomials of order up to k

- Order-k polynomial regression in $\mathbb{R}^p$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \text{"vector with all monomials of degree up to } k\text{"} \in \mathbb{R}^d$$

Example: Polynomial Regression

- Order-k polynomial regression in $\mathbb{R}$:

$$\boldsymbol{\phi}(x) = [1,\ x,\ x^2,\ \ldots,\ x^k]^T$$

- Order-k polynomial regression in $\mathbb{R}^2$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [1,\ x_1,\ x_2,\ x_1^2,\ x_1 x_2,\ x_2^2,\ \ldots,\ x_1 x_2^{k-1},\ x_2^k]^T$$

...all monomials of order up to k

- Order-k polynomial regression in $\mathbb{R}^p$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \text{"vector with all monomials of degree up to } k\text{"} \in \mathbb{R}^d$$

- which has dimension

$$d = \binom{p+k}{k} = \frac{(p+k)!}{k!\,p!} \geq \left(\frac{p+k}{k}\right)^k$$

...exponential in k

Other Types of Non-Linear Regression

- Radial basis functions (RBF): $\phi_j(\boldsymbol{x}) = \psi\Big(\frac{1}{\alpha_j}\|\boldsymbol{x} - \boldsymbol{c}_j\|_2\Big)$

  ...with fixed centers $\boldsymbol{c}_j$ and widths α_j

Other Types of Non-Linear Regression

- Radial basis functions (RBF): $\phi_j(\boldsymbol{x}) = \psi\Big(\dfrac{1}{\alpha_j}\|\boldsymbol{x} - \boldsymbol{c}_j\|_2\Big)$

  ...with fixed centers $\boldsymbol{c}_j$ and widths α_j

- Typical choices:

  ✓ Gaussian RBF (GRBF): $\psi(r) = \exp(-r^2)$

  ✓ Thin plate spline RBF (TPSRBF): $\psi(r) = r^2 \log r$

- Spline regression: each ϕ_j is a piece-wise polynomial function.

Other Types of Non-Linear Regression

- Radial basis functions (RBF): $\phi_j(\boldsymbol{x}) = \psi\Big(\dfrac{1}{\alpha_j}\|\boldsymbol{x} - \boldsymbol{c}_j\|_2\Big)$
  ...with fixed centers $\boldsymbol{c}_j$ and widths α_j

- Typical choices:

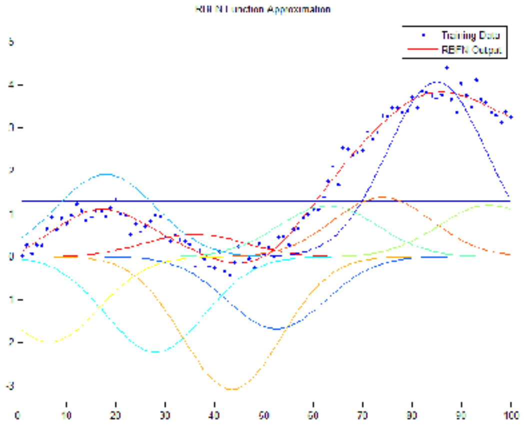
  ✓ Gaussian RBF (GRBF): $\psi(r) = \exp(-r^2)$

  ✓ Thin plate spline RBF (TPSRBF): $\psi(r) = r^2 \log r$

- Spline regression: each ϕ_j is a piece-wise polynomial function.

- Kernels: more later.

Example of Gaussian RBF Regression



Ridge Regression

- If $\texttt{rank}(\boldsymbol{X}) < p$ (for example, if $n < p$), $\hat{\boldsymbol{w}}_{\mathsf{LS}}$ cannot be computed,

$$(\boldsymbol{X}^T\boldsymbol{X}) \in \mathbb{R}^{p\times p}; \quad \texttt{rank}(\boldsymbol{X}) < p \;\Rightarrow\; (\boldsymbol{X}^T\boldsymbol{X})^{-1} \text{ does not exist}$$

Ridge Regression

- If $\mathtt{rank}(\boldsymbol{X}) < p$ (for example, if $n < p$), $\hat{\boldsymbol{w}}_{\mathsf{LS}}$ cannot be computed,

$$(\boldsymbol{X}^T\boldsymbol{X}) \in \mathbb{R}^{p\times p}; \quad \mathtt{rank}(\boldsymbol{X}) < p \;\Rightarrow\; (\boldsymbol{X}^T\boldsymbol{X})^{-1} \text{ does not exist}$$

- The classical alternative is ridge regression:

$$\begin{aligned}\hat{\boldsymbol{w}}_{\mathsf{ridge}} &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\,\|\boldsymbol{w}\|_2^2\\ &= \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

Ridge Regression

- If $\texttt{rank}(\boldsymbol{X}) < p$ (for example, if $n < p$), $\hat{\boldsymbol{w}}_{\mathsf{LS}}$ cannot be computed,

$$(\boldsymbol{X}^T\boldsymbol{X}) \in \mathbb{R}^{p\times p}; \quad \texttt{rank}(\boldsymbol{X}) < p \;\; \Rightarrow \;\; (\boldsymbol{X}^T\boldsymbol{X})^{-1} \text{ does not exist}$$

- The classical alternative is ridge regression:

$$\begin{aligned}\hat{\boldsymbol{w}}_{\mathsf{ridge}} &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\,\|\boldsymbol{w}\|_2^2\\ &= \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

- $\boldsymbol{X}^T\boldsymbol{X}$ is positive semi-definite: $\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)$ is invertible, for $\lambda > 0$

Ridge Regression

- If $\mathtt{rank}(\boldsymbol{X}) < p$ (for example, if $n < p$), $\hat{\boldsymbol{w}}_{\mathsf{LS}}$ cannot be computed,

$$(\boldsymbol{X}^T\boldsymbol{X}) \in \mathbb{R}^{p\times p}; \quad \mathtt{rank}(\boldsymbol{X}) < p \;\Rightarrow\; (\boldsymbol{X}^T\boldsymbol{X})^{-1} \text{ does not exist}$$

- The classical alternative is ridge regression:

$$\begin{aligned}\hat{\boldsymbol{w}}_{\mathsf{ridge}} &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\,\|\boldsymbol{w}\|_2^2 \\ &= \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

- $\boldsymbol{X}^T\boldsymbol{X}$ is positive semi-definite: $\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)$ is invertible, for $\lambda > 0$

- Can be seen as Bayesian estimate with Gaussian prior

$$f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w}; 0, \frac{1}{\lambda}\boldsymbol{I}\right)$$

Ridge Regression

- If $\texttt{rank}(\boldsymbol{X}) < p$ (for example, if $n < p$), $\hat{\boldsymbol{w}}_{\mathsf{LS}}$ cannot be computed,

$$(\boldsymbol{X}^T\boldsymbol{X}) \in \mathbb{R}^{p\times p}; \quad \texttt{rank}(\boldsymbol{X}) < p \;\Rightarrow\; (\boldsymbol{X}^T\boldsymbol{X})^{-1} \text{ does not exist}$$

- The classical alternative is ridge regression:

$$\begin{aligned}\hat{\boldsymbol{w}}_{\mathsf{ridge}} &= \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\,\|\boldsymbol{w}\|_2^2 \\ &= \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}\end{aligned}$$

- $\boldsymbol{X}^T\boldsymbol{X}$ is positive semi-definite: $\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)$ is invertible, for $\lambda > 0$

- Can be seen as Bayesian estimate with Gaussian prior

$$f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w}; 0, \frac{1}{\lambda}\boldsymbol{I}\right)$$

- Known by other names, in other contexts: *weight decay, penalized least squares, Tikhonov regularization,* ℓ_2 regularization,...

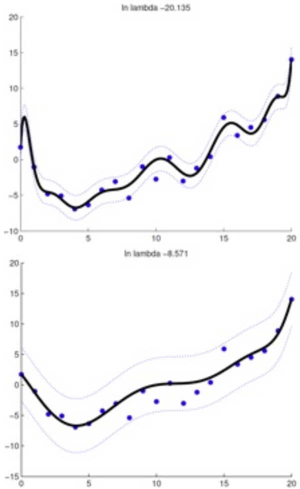
Ridge Regression: Illustration

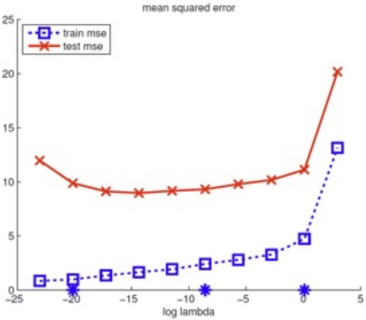
Even if $\hat{\boldsymbol{w}}_{\mathsf{LS}}$ can be computed, $\hat{\boldsymbol{w}}_{\mathsf{ridge}}$ may preferable (lower MSE)

Ridge Regression: Illustration

Even if $\hat{\boldsymbol{w}}_{\text{LS}}$ can be computed, $\hat{\boldsymbol{w}}_{\text{ridge}}$ may preferable (lower MSE)

Example: fitting an order-14 polynomial to 21 points in $\mathbb{R}$





Bias-Variance Decomposition

- Illustration with $p = 1$.

Bias-Variance Decomposition

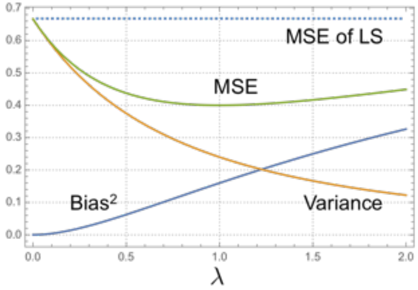
- Illustration with $p = 1$.
- The bias-variance decomposition ($\text{var}[U] = \mathbb{E}[U^2] - \mathbb{E}[U]^2$)

$$MSE = \mathbb{E}\left[(\hat{w}(\boldsymbol{Y}) - w)^2\right] = \underbrace{\text{var}\left[\hat{w}(\boldsymbol{Y})\right]}_{\text{variance}} + \underbrace{\mathbb{E}\left[\hat{w}(\boldsymbol{Y}) - w\right]^2}_{\text{squared bias}}$$

Bias-Variance Decomposition

- Illustration with $p = 1$.
- The bias-variance decomposition ($\text{var}[U] = \mathbb{E}[U^2] - \mathbb{E}[U]^2$)

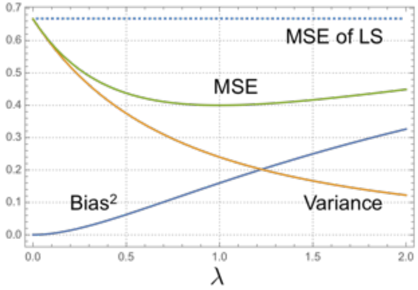
$$MSE = \mathbb{E}\left[(\hat{w}(\boldsymbol{Y}) - w)^2\right] = \underbrace{\text{var}\left[\hat{w}(\boldsymbol{Y})\right]}_{\text{variance}} + \underbrace{\mathbb{E}\left[\hat{w}(\boldsymbol{Y}) - w\right]^2}_{\text{squared bias}}$$



Bias-Variance Decomposition

- Illustration with $p = 1$.
- The bias-variance decomposition ($\text{var}[U] = \mathbb{E}[U^2] - \mathbb{E}[U]^2$)

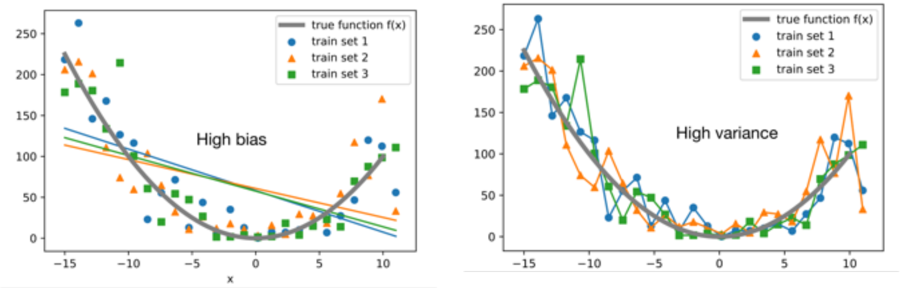
$$MSE = \mathbb{E}\left[(\hat{w}(\boldsymbol{Y}) - w)^2\right] = \underbrace{\text{var}\left[\hat{w}(\boldsymbol{Y})\right]}_{\text{variance}} + \underbrace{\mathbb{E}\left[\hat{w}(\boldsymbol{Y}) - w\right]^2}_{\text{squared bias}}$$



- Bias-variance trade-off. How to choose λ?

Bias-Variance Decomposition: Model Complexity

- Bias-variance trade-off also w.r.t. complexity



linear regression (order-1 polynomial)

Piecewise linear interpolation

Pictures by Sebastian Raschka, 2023.

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$
- Split into K disjoint subsets (folds), each with $\frac{n}{K}$ samples: S_1, ..., S_K

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$
- Split into K disjoint subsets (folds), each with $\frac{n}{K}$ samples: S_1, ..., S_K
- For each $k \in \{1, ..., K\}$, learn $\hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}$ from all the samples not in S_k.

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$
- Split into K disjoint subsets (folds), each with $\frac{n}{K}$ samples: S_1, ..., S_K
- For each $k \in \{1, ..., K\}$, learn $\hat{\boldsymbol{w}}_{\text{ridge},\lambda}^{(k)}$ from all the samples not in S_k.
- Estimate the MSE using S_k

$$\widehat{\text{MSE}}_k(\lambda) = \frac{K}{n} \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}_{\text{ridge},\lambda}^{(k)} \right)^2$$

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$
- Split into K disjoint subsets (folds), each with $\frac{n}{K}$ samples: S_1, ..., S_K
- For each $k \in \{1, ..., K\}$, learn $\hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}$ from all the samples not in S_k.
- Estimate the MSE using S_k

$$\widehat{\text{MSE}}_k(\lambda) = \frac{K}{n} \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}\right)^2$$

- Choose λ by minimizing the average MSE estimate:

$$\lambda^* = \arg\min_{\lambda} \sum_{k=1}^{K} \widehat{\text{MSE}}_k(\lambda) = \arg\min_{\lambda} \sum_{k=1}^{K} \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}\right)^2$$

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$
- Split into K disjoint subsets (folds), each with $\frac{n}{K}$ samples: S_1, ..., S_K
- For each $k \in \{1, ..., K\}$, learn $\hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}$ from all the samples not in S_k.
- Estimate the MSE using S_k

$$\widehat{\text{MSE}}_k(\lambda) = \frac{K}{n} \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}\right)^2$$

- Choose λ by minimizing the average MSE estimate:

$$\lambda^* = \arg\min_{\lambda} \sum_{k=1}^{K} \widehat{\text{MSE}}_k(\lambda) = \arg\min_{\lambda} \sum_{k=1}^{K} \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}\right)^2$$

- K-fold CV; common choices are $K = 5$ and $K = 10$.

Choosing λ via Cross Validation (CV)

- Available data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$
- Split into K disjoint subsets (folds), each with $\frac{n}{K}$ samples: S_1, ..., S_K
- For each $k \in \{1, ..., K\}$, learn $\hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}$ from all the samples not in S_k.
- Estimate the MSE using S_k

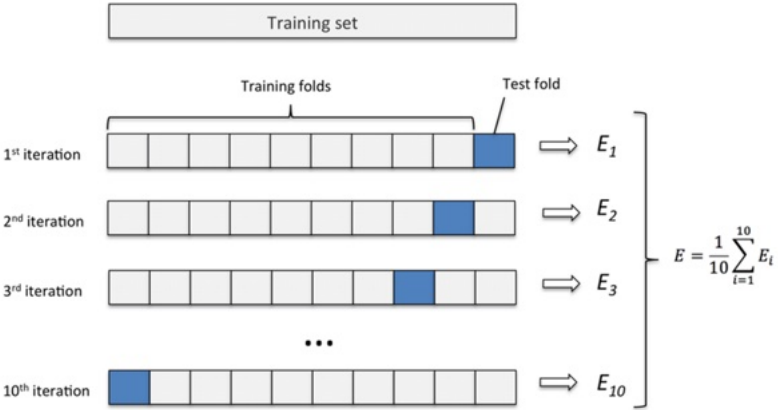
$$\widehat{\text{MSE}}_k(\lambda) = \frac{K}{n} \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}\right)^2$$

- Choose λ by minimizing the average MSE estimate:

$$\lambda^* = \arg\min_\lambda \sum_{k=1}^K \widehat{\text{MSE}}_k(\lambda) = \arg\min_\lambda \sum_{k=1}^K \sum_{i \in S_k} \left(y_i - \boldsymbol{x}_i^T \hat{\boldsymbol{w}}^{(k)}_{\text{ridge},\lambda}\right)^2$$

- K-fold CV; common choices are $K = 5$ and $K = 10$.
- Extreme case: $K = n$, leave-one-out CV (LOOCV).

Illustration od 10-fold CV



Dual Variables: Ridge Regression

- Ridge regression: $\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})$ is the solution w.r.t. $\boldsymbol{w}$ of

$$\big(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\big)\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \quad \Leftrightarrow \quad \hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \frac{1}{\lambda}\boldsymbol{X}^T\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

Dual Variables: Ridge Regression

- Ridge regression: $\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})$ is the solution w.r.t. $\boldsymbol{w}$ of

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \quad \Leftrightarrow \quad \hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \frac{1}{\lambda}\boldsymbol{X}^T\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

that is,

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T\boldsymbol{\alpha} \quad \text{with} \quad \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

Dual Variables: Ridge Regression

- Ridge regression: $\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})$ is the solution w.r.t. $\boldsymbol{w}$ of

$$\big(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\big)\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \quad \Leftrightarrow \quad \hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \frac{1}{\lambda}\boldsymbol{X}^T\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

that is,

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T\boldsymbol{\alpha} \quad \text{with} \quad \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

- $\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \displaystyle\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$, a linear combination of rows of $\boldsymbol{X}$

Dual Variables: Ridge Regression

- Ridge regression: $\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})$ is the solution w.r.t. $\boldsymbol{w}$ of

$$\big(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\big)\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \quad \Leftrightarrow \quad \hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \frac{1}{\lambda}\boldsymbol{X}^T\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

that is,

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T\boldsymbol{\alpha} \quad \text{with} \quad \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

- $\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i$, a linear combination of rows of $\boldsymbol{X}$
- Predicted value for some new point $\boldsymbol{x}$:

$$\hat{y}(\boldsymbol{x}) = \boldsymbol{x}^T\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \sum_{i=1}^{n} \alpha_i \left(\boldsymbol{x}^T\boldsymbol{x}_i\right)$$

... linear combination of the inner products of $\boldsymbol{x}$ with the $\boldsymbol{x}_i$

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T \boldsymbol{\alpha} \quad \text{with} \;\; \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T \boldsymbol{\alpha} \quad \text{with} \;\; \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

- Inserting the first equality in the second one, solving for $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{\alpha}\big) \quad \Leftrightarrow \quad \boldsymbol{\alpha} = \big(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\big)^{-1}\boldsymbol{y}$$

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T\boldsymbol{\alpha} \quad \text{with} \;\; \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

- Inserting the first equality in the second one, solving for $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \frac{1}{\lambda}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{\alpha}) \quad \Leftrightarrow \quad \boldsymbol{\alpha} = \left(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1}\boldsymbol{y}$$

thus

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T\underbrace{\left(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1}}_{n\times n \text{ inversion}}\boldsymbol{y}$$

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T \boldsymbol{\alpha} \quad \text{with} \;\; \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

- Inserting the first equality in the second one, solving for $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \frac{1}{\lambda}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{\alpha}) \quad \Leftrightarrow \quad \boldsymbol{\alpha} = \big(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\big)^{-1}\boldsymbol{y}$$

thus

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T \underbrace{\big(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\big)^{-1}}_{n\times n \text{ inversion}} \boldsymbol{y} = \underbrace{\big(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\big)^{-1}}_{p\times p \text{ inversion}} \boldsymbol{X}^T \boldsymbol{y}$$

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T \boldsymbol{\alpha} \quad \text{with} \;\; \boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y})\big)$$

- Inserting the first equality in the second one, solving for $\boldsymbol{\alpha}$

$$\boldsymbol{\alpha} = \frac{1}{\lambda}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^T\boldsymbol{\alpha}) \quad \Leftrightarrow \quad \boldsymbol{\alpha} = \big(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\big)^{-1}\boldsymbol{y}$$

thus

$$\hat{\boldsymbol{w}}_{\text{ridge}}(\boldsymbol{y}) = \boldsymbol{X}^T \underbrace{\big(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\big)^{-1}}_{n\times n \text{ inversion}} \boldsymbol{y} = \underbrace{\big(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\big)^{-1}}_{p\times p \text{ inversion}} \boldsymbol{X}^T \boldsymbol{y}$$

- $\boldsymbol{X}\boldsymbol{X}^T$ is called the Gram matrix (*i.e.*, $(\boldsymbol{X}\boldsymbol{X}^T)_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j$)

Kernel Regression

- Recall that, in dual variables,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, (\boldsymbol{x}^T \boldsymbol{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1} \boldsymbol{y}$$

- ... $\boldsymbol{X}\boldsymbol{X}^T$ is the Gram matrix, *i.e.*, $(\boldsymbol{X}\boldsymbol{X}^T)_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j$

Kernel Regression

- Recall that, in dual variables,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, (\boldsymbol{x}^T \boldsymbol{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1} \boldsymbol{y}$$

- ... $\boldsymbol{X}\boldsymbol{X}^T$ is the Gram matrix, *i.e.*, $(\boldsymbol{X}\boldsymbol{X}^T)_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Data points are only involved via inner products: $\boldsymbol{x}_i^T \boldsymbol{x}_j$ and $\boldsymbol{x}^T \boldsymbol{x}_j$

Kernel Regression

- Recall that, in dual variables,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, (\boldsymbol{x}^T \boldsymbol{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1} \boldsymbol{y}$$

- ... $\boldsymbol{X}\boldsymbol{X}^T$ is the Gram matrix, *i.e.*, $(\boldsymbol{X}\boldsymbol{X}^T)_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Data points are only involved via inner products: $\boldsymbol{x}_i^T \boldsymbol{x}_j$ and $\boldsymbol{x}^T \boldsymbol{x}_j$
- To go non-linear, use a feature map $\phi : \mathbb{R}^p \rightarrow \mathbb{R}^d$,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_i) \rangle, \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{G}\right)^{-1} \boldsymbol{y},$$

Kernel Regression

- Recall that, in dual variables,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \left(\boldsymbol{x}^T \boldsymbol{x}_i\right), \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1} \boldsymbol{y}$$

- ... $\boldsymbol{X}\boldsymbol{X}^T$ is the Gram matrix, *i.e.*, $(\boldsymbol{X}\boldsymbol{X}^T)_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Data points are only involved via inner products: $\boldsymbol{x}_i^T \boldsymbol{x}_j$ and $\boldsymbol{x}^T \boldsymbol{x}_j$
- To go non-linear, use a feature map $\phi : \mathbb{R}^p \to \mathbb{R}^d$,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_i) \right\rangle, \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{G}\right)^{-1} \boldsymbol{y},$$

- $\boldsymbol{G}$ is still the Gram matrix, that is, $\boldsymbol{G}_{ij} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$

Kernel Regression

- Recall that, in dual variables,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, (\boldsymbol{x}^T \boldsymbol{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^T\right)^{-1} \boldsymbol{y}$$

- ... $\boldsymbol{X}\boldsymbol{X}^T$ is the Gram matrix, *i.e.*, $(\boldsymbol{X}\boldsymbol{X}^T)_{ij} = \boldsymbol{x}_i^T \boldsymbol{x}_j$
- Data points are only involved via inner products: $\boldsymbol{x}_i^T \boldsymbol{x}_j$ and $\boldsymbol{x}^T \boldsymbol{x}_j$
- To go non-linear, use a feature map $\phi : \mathbb{R}^p \to \mathbb{R}^d$,

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \, \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}_i) \rangle, \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{G}\right)^{-1} \boldsymbol{y},$$

- $\boldsymbol{G}$ is still the Gram matrix, that is, $\boldsymbol{G}_{ij} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$
- Feature map moves inner products from $\mathbb{R}^p$ to $\mathbb{R}^d$. Is that bad?

Kernel Regression (2)

- Motivation example: order 2 polynomial regression in $\mathbb{R}^2$:

$$\phi(\boldsymbol{x}) = \phi([x_1,\, x_2]^T) = [1, {x_1}^2, {x_2}^2, \sqrt{2}\, x_1\, x_2]$$

Kernel Regression (2)

- Motivation example: order 2 polynomial regression in $\mathbb{R}^2$:

$$\phi(\boldsymbol{x}) = \phi([x_1,\, x_2]^T) = [1, {x_1}^2, {x_2}^2, \sqrt{2}\, x_1\, x_2]$$

- Computing the inner product in $\mathbb{R}^4$

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = 1 + {x_1}^2\, {x_1'}^2 + {x_2}^2\, {x_2'}^2 + 2\, x_1\, x_1'\, x_2\, x_2' = 1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$$

Kernel Regression (2)

- Motivation example: order 2 polynomial regression in $\mathbb{R}^2$:

$$\phi(\boldsymbol{x}) = \phi([x_1,\, x_2]^T) = [1, {x_1}^2, {x_2}^2, \sqrt{2}\, x_1\, x_2]$$

- Computing the inner product in $\mathbb{R}^4$

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = 1 + {x_1}^2\, {x_1'}^2 + {x_2}^2\, {x_2'}^2 + 2\, x_1\, x_1'\, x_2\, x_2' = 1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$$

- This inner product in $\mathbb{R}^4$ is a function of that in $\mathbb{R}^2$.

Kernel Regression (2)

- Motivation example: order 2 polynomial regression in $\mathbb{R}^2$:

$$\phi(\boldsymbol{x}) = \phi([x_1,\, x_2]^T) = [1, {x_1}^2, {x_2}^2, \sqrt{2}\, x_1\, x_2]$$

- Computing the inner product in $\mathbb{R}^4$

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = 1 + {x_1}^2\, {x_1'}^2 + {x_2}^2\, {x_2'}^2 + 2\, x_1\, x_1'\, x_2\, x_2' = 1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$$

- This inner product in $\mathbb{R}^4$ is a function of that in $\mathbb{R}^2$.

- This is called a kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$

Kernel Regression (2)

- Motivation example: order 2 polynomial regression in $\mathbb{R}^2$:

$$\phi(\boldsymbol{x}) = \phi([x_1,\, x_2]^T) = [1, {x_1}^2, {x_2}^2, \sqrt{2}\, x_1\, x_2]$$

- Computing the inner product in $\mathbb{R}^4$

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = 1 + {x_1}^2\, {x_1'}^2 + {x_2}^2\, {x_2'}^2 + 2\, x_1\, x_1'\, x_2\, x_2' = 1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$$

- This inner product in $\mathbb{R}^4$ is a function of that in $\mathbb{R}^2$.

- This is called a kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$

- Kernel least squares regression:

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i\, K(\boldsymbol{x}, \boldsymbol{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha} = \big(\lambda \boldsymbol{I} + \boldsymbol{G}\big)^{-1} \boldsymbol{y},$$

Kernel Regression (2)

- Motivation example: order 2 polynomial regression in $\mathbb{R}^2$:

$$\phi(\boldsymbol{x}) = \phi([x_1,\, x_2]^T) = [1, {x_1}^2, {x_2}^2, \sqrt{2}\, x_1\, x_2]$$

- Computing the inner product in $\mathbb{R}^4$

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = 1 + {x_1}^2\, {x_1'}^2 + {x_2}^2\, {x_2'}^2 + 2\, x_1\, x_1'\, x_2\, x_2' = 1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^2$$

- This inner product in $\mathbb{R}^4$ is a function of that in $\mathbb{R}^2$.
- This is called a kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$
- Kernel least squares regression:

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i\, K(\boldsymbol{x}, \boldsymbol{x}_i), \qquad \text{with} \qquad \boldsymbol{\alpha} = \left(\lambda \boldsymbol{I} + \boldsymbol{G}\right)^{-1} \boldsymbol{y},$$

- $\boldsymbol{G}$ is the Gram matrix, that is, $\boldsymbol{G}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Kernel Regression (3)

- No need for structure: $\boldsymbol{x} \in \mathcal{X}$, an arbitrary set.

Kernel Regression (3)

- No need for structure: $\boldsymbol{x} \in \mathcal{X}$, an arbitrary set.
- Definition: a kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle,$$

  for some $\phi : \mathcal{X} \to \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space.

Kernel Regression (3)

- No need for structure: $\boldsymbol{x} \in \mathcal{X}$, an arbitrary set.
- Definition: a kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle,$$

  for some $\phi : \mathcal{X} \to \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space.
- Hilbert space? Just a complete inner-product vector space.

Kernel Regression (3)

- No need for structure: $\boldsymbol{x} \in \mathcal{X}$, an arbitrary set.
- Definition: a kernel is a function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, such that,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle,$$

  for some $\phi : \mathcal{X} \rightarrow \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space.

- Hilbert space? Just a complete inner-product vector space.
- Mercer's theorem: a symmetric function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a kernel if and only any Gram matrix $\boldsymbol{G}$ is positive semi-definite (psd).

Kernel Regression (3)

- No need for structure: $\boldsymbol{x} \in \mathcal{X}$, an arbitrary set.
- Definition: a kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle,$$

  for some $\phi : \mathcal{X} \to \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space.
- Hilbert space? Just a complete inner-product vector space.
- Mercer's theorem: a symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only any Gram matrix $\boldsymbol{G}$ is positive semi-definite (psd).
- $\boldsymbol{G}$ is psd $\Rightarrow$ $\left(\lambda \boldsymbol{I} + \boldsymbol{G}\right)^{-1}$ exists, for $\lambda > 0$.

Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$

Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- Linear kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \langle(\boldsymbol{A}\boldsymbol{x}), (\boldsymbol{A}\boldsymbol{x}')\rangle$; mapping $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{A}\,\boldsymbol{x}$.

Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- Linear kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \langle (\boldsymbol{A}\boldsymbol{x}), (\boldsymbol{A}\boldsymbol{x}') \rangle$; mapping $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{A}\,\boldsymbol{x}$.
- Quadratic kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \big(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + A\big)^2$;

$$\boldsymbol{\phi}(\boldsymbol{x}) = [A,\, \sqrt{2A}x_1,\, \sqrt{2A}x_2,\, ...\, \sqrt{2A}x_d, x_1^2,\, x_1\, x_2, ..., x_1\, x_d,\, ...,\, x_d^2]$$

(all monomials of degree up to 2, with scaling depending on A)

Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- Linear kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \langle (\boldsymbol{A}\boldsymbol{x}), (\boldsymbol{A}\boldsymbol{x}') \rangle$; mapping $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{A}\,\boldsymbol{x}$.
- Quadratic kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \big(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + A\big)^2$;

  $\boldsymbol{\phi}(\boldsymbol{x}) = [A,\ \sqrt{2A}x_1,\ \sqrt{2A}x_2,\ ...\ \sqrt{2A}x_d, x_1^2,\ x_1\,x_2, ..., x_1\,x_d,\ ...,\ x_d^2]$

  (all monomials of degree up to 2, with scaling depending on A)
- Polynomial kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \big(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + A\big)^p$;

  $\boldsymbol{\phi}(\boldsymbol{x}) = \big[$all monomials of degree up to p, with scaling depending on $A\big]$

  $\dim \boldsymbol{\phi}(\boldsymbol{x}) = \binom{d+p}{p}$

Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$

Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- Gaussian kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}\right)$;

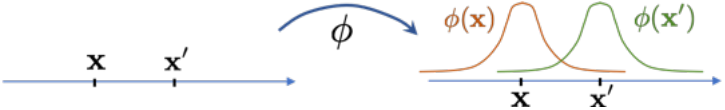
Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- Gaussian kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}\right)$;

  transformation $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathcal{F}$, where $\mathcal{F}$ has infinite dimension.

$$\boldsymbol{\phi}(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \ \cdot\ \|_2^2}{2\sigma^2}\right)$$

- Illustration for $d = 1$:



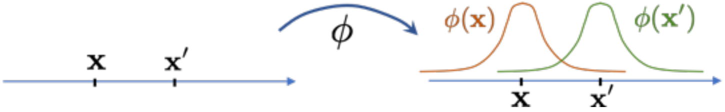
Kernels: Examples

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- Gaussian kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}\right)$;

  transformation $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathcal{F}$, where $\mathcal{F}$ has infinite dimension.

$$\boldsymbol{\phi}(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \ \cdot\ \|_2^2}{2\sigma^2}\right)$$

- Illustration for $d = 1$:



- Why?

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle = \int \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{u}\|_2^2}{2\sigma^2}\right) \exp\left(-\frac{\|\boldsymbol{x}' - \boldsymbol{u}\|_2^2}{2\sigma^2}\right) d\boldsymbol{u} = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2}\right)$$

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...

- Sets: let $\mathcal{X} = 2^{\mathcal{S}}$ (all subsets of set $\mathcal{S}$, for simplicity, assumed finite).

$$K_\cap(A, A') = |A \cap A'|, \text{ for } A, A' \in \mathcal{X}$$

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...
- Sets: let $\mathcal{X} = 2^{\mathcal{S}}$ (all subsets of set $\mathcal{S}$, for simplicity, assumed finite).

$$K_{\cap}(A, A') = |A \cap A'|, \ \ \text{for } A, A' \in \mathcal{X} \quad \text{(intersection kernel)}$$

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...
- Sets: let $\mathcal{X} = 2^{\mathcal{S}}$ (all subsets of set $\mathcal{S}$, for simplicity, assumed finite).

$$K_\cap(A, A') = |A \cap A'|, \text{ for } A, A' \in \mathcal{X} \quad \text{(intersection kernel)}$$

mapping $\phi : \mathcal{X} \to \mathcal{F}$ (space of real-valued functions in $\mathcal{S}$)

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...

- Sets: let $\mathcal{X} = 2^{\mathcal{S}}$ (all subsets of set $\mathcal{S}$, for simplicity, assumed finite).

$$K_\cap(A, A') = |A \cap A'|, \text{ for } A, A' \in \mathcal{X} \quad (\text{intersection kernel})$$

mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$ (space of real-valued functions in $\mathcal{S}$)

$$\boldsymbol{\phi}(A) = \mathbf{1}_A, \text{ that is } \mathbf{1}_A(x) = \begin{cases} 1 & \Leftarrow \quad x \in A \\ 0 & \Leftarrow \quad x \notin A \end{cases}$$

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...

- Sets: let $\mathcal{X} = 2^{\mathcal{S}}$ (all subsets of set $\mathcal{S}$, for simplicity, assumed finite).

$$K_\cap(A, A') = |A \cap A'|, \text{ for } A, A' \in \mathcal{X} \quad \text{(intersection kernel)}$$

mapping $\phi : \mathcal{X} \to \mathcal{F}$ (space of real-valued functions in $\mathcal{S}$)

$$\phi(A) = \mathbf{1}_A, \text{ that is } \mathbf{1}_A(x) = \begin{cases} 1 & \Leftarrow \quad x \in A \\ 0 & \Leftarrow \quad x \notin A \end{cases}$$

$$\langle \phi(A), \phi(A') \rangle = \sum_{x \in \mathcal{X}} \mathbf{1}_A(x) \mathbf{1}_{A'}(x) = \sum_{x \in A \cap A'} 1 = |A \cap A'| = K_\cap(A, A')$$

Kernels: Examples

- There are kernels for many other types of objects: sets, strings, images, graphs, probability density or mass functions, ...

- Sets: let $\mathcal{X} = 2^{\mathcal{S}}$ (all subsets of set $\mathcal{S}$, for simplicity, assumed finite).

$$K_\cap(A, A') = |A \cap A'|, \text{ for } A, A' \in \mathcal{X} \quad \text{(intersection kernel)}$$

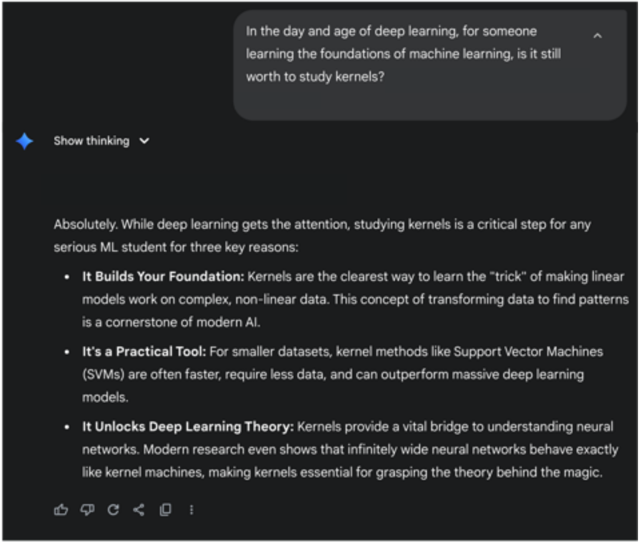
mapping $\phi : \mathcal{X} \to \mathcal{F}$ (space of real-valued functions in $\mathcal{S}$)

$$\phi(A) = \mathbf{1}_A, \text{ that is } \mathbf{1}_A(x) = \begin{cases} 1 & \Leftarrow \quad x \in A \\ 0 & \Leftarrow \quad x \notin A \end{cases}$$

$$\langle \phi(A), \phi(A') \rangle = \sum_{x \in \mathcal{X}} \mathbf{1}_A(x) \mathbf{1}_{A'}(x) = \sum_{x \in A \cap A'} 1 = |A \cap A'| = K_\cap(A, A')$$

- There are many other kernels for sets.

Kernels in 2025? Let's Ask Gemini 2.5 Pro



Minimum-Norm Linear Regression

- Consider $n < p$, with $\boldsymbol{X}$ full rank (rank$(\boldsymbol{X}) = n$)

Minimum-Norm Linear Regression

- Consider $n < p$, with $\boldsymbol{X}$ full rank ($\text{rank}(\boldsymbol{X}) = n$)
- LS regression does not have a unique solution:

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

Minimum-Norm Linear Regression

- Consider $n < p$, with $\boldsymbol{X}$ full rank ($\text{rank}(\boldsymbol{X}) = n$)
- LS regression does not have a unique solution:

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$ has infinitely many solutions.

Minimum-Norm Linear Regression

- Consider $n < p$, with $\boldsymbol{X}$ full rank ($\text{rank}(\boldsymbol{X}) = n$)
- LS regression does not have a unique solution:

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$ has infinitely many solutions.
- Minimum-norm (MN) linear regression:

$$\hat{\boldsymbol{w}}_{\mathsf{MN}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}:\ \boldsymbol{y}=\boldsymbol{X}\boldsymbol{w}} \|\boldsymbol{w}\|_2^2 = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{y}$$

Minimum-Norm Linear Regression

- Consider $n < p$, with $\boldsymbol{X}$ full rank ($\text{rank}(\boldsymbol{X}) = n$)
- LS regression does not have a unique solution:

$$\hat{\boldsymbol{w}}_{\text{LS}}(\boldsymbol{y}) \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$ has infinitely many solutions.
- Minimum-norm (MN) linear regression:

$$\hat{\boldsymbol{w}}_{\text{MN}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}:\ \boldsymbol{y}=\boldsymbol{X}\boldsymbol{w}} \|\boldsymbol{w}\|_2^2 = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{y}$$

- LS and MN: instances of the Moore-Penrose pseudo-inverse.

Minimum-Norm Linear Regression

- Consider $n < p$, with $\boldsymbol{X}$ full rank ($\text{rank}(\boldsymbol{X}) = n$)
- LS regression does not have a unique solution:

$$\hat{\boldsymbol{w}}_{\mathsf{LS}}(\boldsymbol{y}) \in \arg\min_{\boldsymbol{w}\in\mathbb{R}^p} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2$$

- $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$ has infinitely many solutions.
- Minimum-norm (MN) linear regression:

$$\hat{\boldsymbol{w}}_{\mathsf{MN}}(\boldsymbol{y}) = \arg\min_{\boldsymbol{w}:\ \boldsymbol{y}=\boldsymbol{X}\boldsymbol{w}} \|\boldsymbol{w}\|_2^2 = \boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{y}$$

- LS and MN: instances of the Moore-Penrose pseudo-inverse.
- Perfect interpolation regime: $\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}}_{\mathsf{MN}}(\boldsymbol{y}) = \boldsymbol{y}$

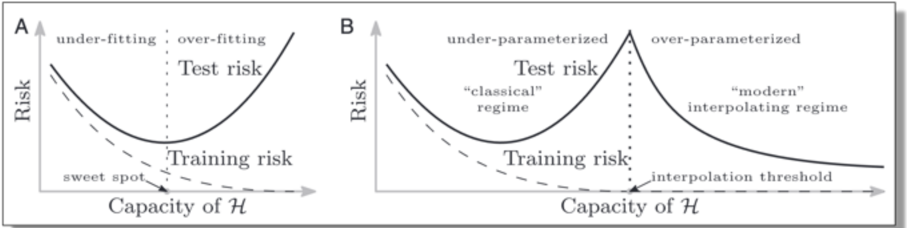
Double Descent

Reconciling modern machine-learning practice and the classical bias–variance trade-off

Mikhail Belkin[a,b,1], Daniel Hsu[c], Siyuan Ma[a], and Soumik Mandal[a]

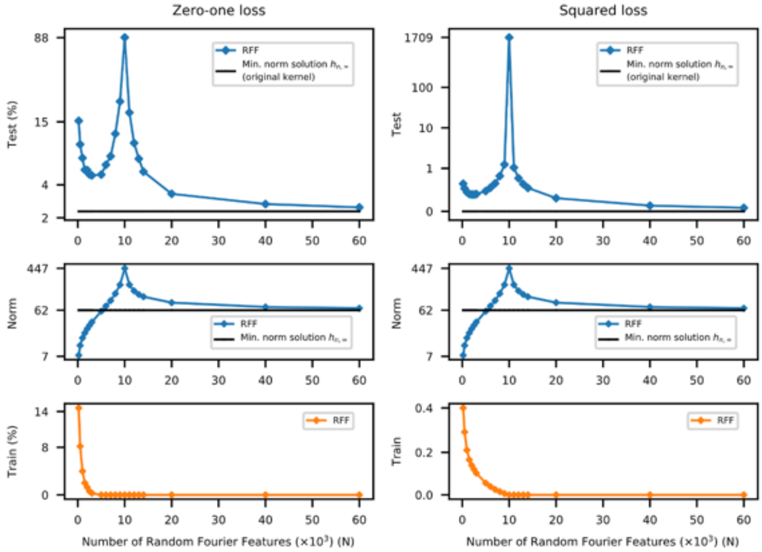
[a]Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210; [b]Department of Statistics, The Ohio State University, Columbus, OH 43210; and [c]Computer Science Department and Data Science Institute, Columbia University, New York, NY 10027

Edited by Peter J. Bickel, University of California, Berkeley, CA, and approved July 2, 2019 (received for review February 21, 2019)



Double Descent (2)

- Random Fourier features: $\phi_i(\boldsymbol{x}) = \exp(\sqrt{-1}\langle \boldsymbol{v}_i, \boldsymbol{x}\rangle), \;\; \boldsymbol{v}_i \sim \mathcal{N}(0, \boldsymbol{I})$



Overparametrization and Double Descent

- "Modern" interpolating regime: more parameters than data points.

Overparametrization and Double Descent

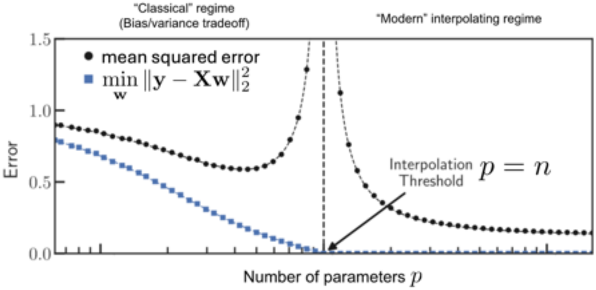
- "Modern" interpolating regime: more parameters than data points.
- For linear regression with $p \geq n$, use minimum norm solution.

Overparametrization and Double Descent

- "Modern" interpolating regime: more parameters than data points.
- For linear regression with $p \geq n$, use minimum norm solution.
- Example w/ $\phi_i(\boldsymbol{x}) = \max\{\boldsymbol{v}_i^T \boldsymbol{x}, 0\}$, where $\boldsymbol{v}_i$ are random vectors.

Overparametrization and Double Descent

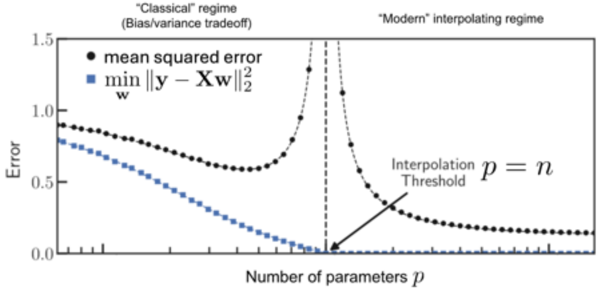
- "Modern" interpolating regime: more parameters than data points.
- For linear regression with $p \geq n$, use minimum norm solution.
- Example w/ $\phi_i(\boldsymbol{x}) = \max\{\boldsymbol{v}_i^T \boldsymbol{x}, 0\}$, where $\boldsymbol{v}_i$ are random vectors.



(Image adapted from Rocks and Mehta, 2022.)

Overparametrization and Double Descent

- "Modern" interpolating regime: more parameters than data points.
- For linear regression with $p \geq n$, use minimum norm solution.
- Example w/ $\phi_i(\boldsymbol{x}) = \max\{\boldsymbol{v}_i^T \boldsymbol{x}, 0\}$, where $\boldsymbol{v}_i$ are random vectors.

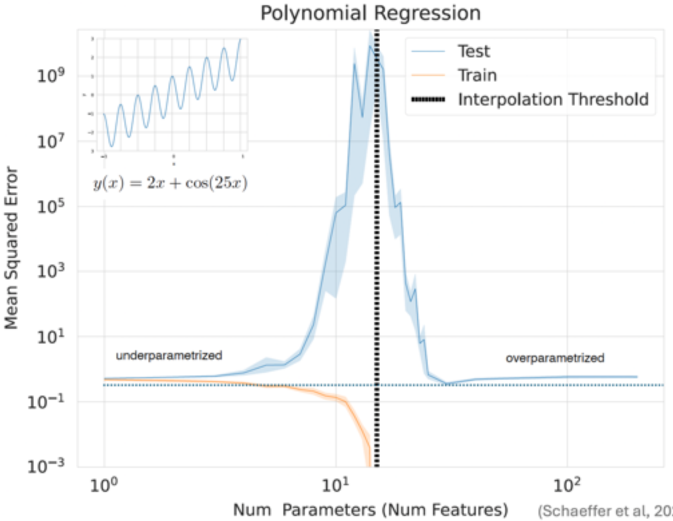


(Image adapted from Rocks and Mehta, 2022.)

- Current research topic.

Overparametrization and Double Descent (cont.)

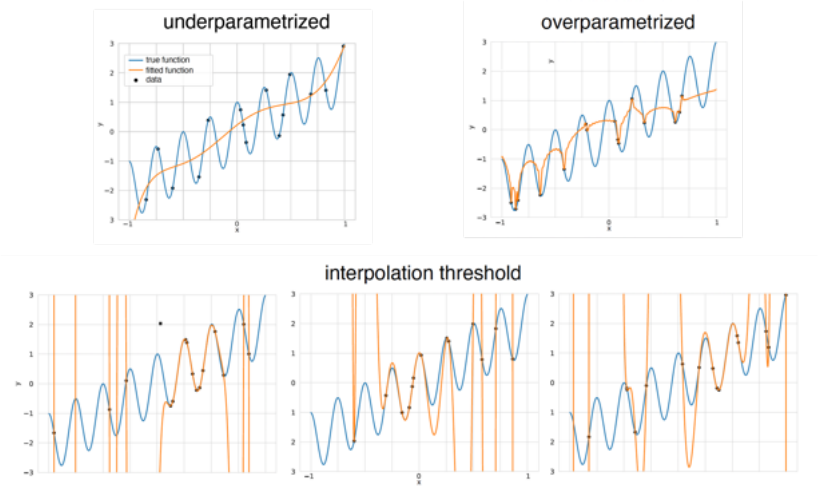
- Polynomial regression: the ϕ_i are Legendre polynomials.



(Schaeffer et al, 2023)

Overparametrization and Double Descent (cont.)

- Polynomial regression: the ϕ_i are Legendre polynomials.



Bayesian View of Ridge Regression

- Linear-Gaussian likelihood (design $\boldsymbol{D}$): $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{D}\boldsymbol{w}, \sigma^2\boldsymbol{I})$

Bayesian View of Ridge Regression

- Linear-Gaussian likelihood (design $\boldsymbol{D}$): $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{D}\boldsymbol{w}, \sigma^2\boldsymbol{I})$
- Gaussian prior: $f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{w}; 0, \boldsymbol{I}/\lambda\big)$

Bayesian View of Ridge Regression

- Linear-Gaussian likelihood (design $\boldsymbol{D}$): $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{D}\boldsymbol{w}, \sigma^2\boldsymbol{I})$
- Gaussian prior: $f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{w}; 0, \boldsymbol{I}/\lambda\big)$
- Posterior density:

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \mathcal{N}\Big(\boldsymbol{w}; \overbrace{(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}}^{\hat{\boldsymbol{w}}_{\text{ridge}}}, \sigma^2\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\Big)$$

Bayesian View of Ridge Regression

- Linear-Gaussian likelihood (design $\boldsymbol{D}$): $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{D}\boldsymbol{w}, \sigma^2 \boldsymbol{I})$
- Gaussian prior: $f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{w}; 0, \boldsymbol{I}/\lambda\big)$
- Posterior density:

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \mathcal{N}\Big(\boldsymbol{w}; \overbrace{(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}}^{\hat{\boldsymbol{w}}_{\text{ridge}}}, \sigma^2\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\Big)$$

- Prediction at new point $\boldsymbol{x}_*$ is $Y(\boldsymbol{x}_*) = \boldsymbol{x}_*^T\boldsymbol{W} + \boldsymbol{N}$ (Gaussian)

$$\begin{aligned} f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*) &= \mathcal{N}\Big(\boldsymbol{x}_*^T(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}, \sigma^2\boldsymbol{x}_*^T\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\boldsymbol{x}_* + \sigma^2\Big) \\ &= \int f_{Y|\boldsymbol{X},\boldsymbol{Y}}(y|\boldsymbol{x}_*, \boldsymbol{w}, \boldsymbol{y})\, f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y})\, d\boldsymbol{w} \end{aligned}$$

Bayesian View of Ridge Regression

- Linear-Gaussian likelihood (design $\boldsymbol{D}$): $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{D}\boldsymbol{w}, \sigma^2 \boldsymbol{I})$
- Gaussian prior: $f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{w}; 0, \boldsymbol{I}/\lambda\big)$
- Posterior density:

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \mathcal{N}\Big(\boldsymbol{w}; \overbrace{(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}}^{\hat{\boldsymbol{w}}_{\text{ridge}}}, \sigma^2\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\Big)$$

- Prediction at new point $\boldsymbol{x}_*$ is $Y(\boldsymbol{x}_*) = \boldsymbol{x}_*^T\boldsymbol{W} + \boldsymbol{N}$ (Gaussian)

$$\begin{aligned} f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*) &= \mathcal{N}\Big(\boldsymbol{x}_*^T(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}, \sigma^2\boldsymbol{x}_*^T\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\boldsymbol{x}_* + \sigma^2\Big) \\ &= \int f_{Y|\boldsymbol{X},\boldsymbol{Y}}(y|\boldsymbol{x}_*, \boldsymbol{w}, \boldsymbol{y})\, f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y})\, d\boldsymbol{w} \end{aligned}$$

...the variance/uncertainty of the prediction depends on $\boldsymbol{x}_*$

Bayesian View of Ridge Regression

- Linear-Gaussian likelihood (design $\boldsymbol{D}$): $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{D}\boldsymbol{w}, \sigma^2\boldsymbol{I})$
- Gaussian prior: $f_{\boldsymbol{W}}(\boldsymbol{w}) = \mathcal{N}\big(\boldsymbol{w}; 0, \boldsymbol{I}/\lambda\big)$
- Posterior density:

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \mathcal{N}\Big(\boldsymbol{w}; \overbrace{(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}}^{\hat{\boldsymbol{w}}_{\text{ridge}}}, \sigma^2\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\Big)$$

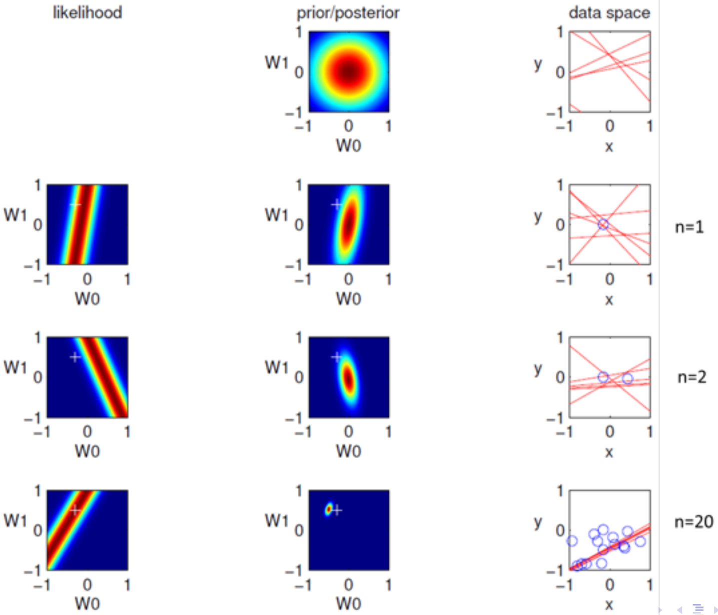
- Prediction at new point $\boldsymbol{x}_*$ is $Y(\boldsymbol{x}_*) = \boldsymbol{x}_*^T\boldsymbol{W} + \boldsymbol{N}$ (Gaussian)

$$\begin{aligned} f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*) &= \mathcal{N}\Big(\boldsymbol{x}_*^T(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I})^{-1}\boldsymbol{D}^T\boldsymbol{y}, \sigma^2\boldsymbol{x}_*^T\big(\boldsymbol{D}^T\boldsymbol{D} + \sigma^2\lambda\boldsymbol{I}\big)^{-1}\boldsymbol{x}_* + \sigma^2\Big) \\ &= \int f_{Y|\boldsymbol{X},\boldsymbol{Y}}(y|\boldsymbol{x}_*, \boldsymbol{w}, \boldsymbol{y})\, f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y})\, d\boldsymbol{w} \end{aligned}$$

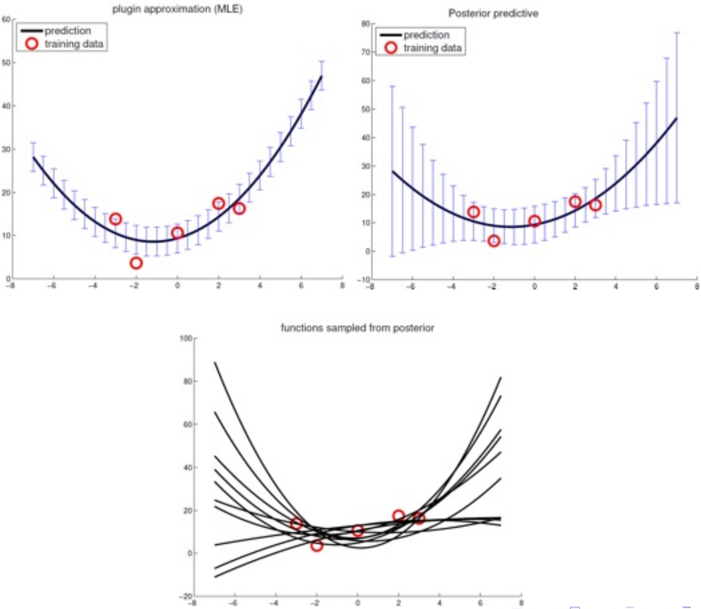
...the variance/uncertainty of the prediction depends on $\boldsymbol{x}_*$

- Example in next slide: $p = 1$, $\boldsymbol{w} = [w_0,\, w_1]^T$, $\boldsymbol{w}_{\text{true}} = [-0.3,\, 0.5]$

Bayesian View of Ridge Regression: Example 1



Bayesian View of Ridge Regression: Example 2



Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$

Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$
- Apply with $U = Y(\boldsymbol{x}')$ and $V = \boldsymbol{w}$:

$$\text{var}\left[Y(\boldsymbol{x}')\right] = \underbrace{\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right]}_{\text{aleatoric uncertainty}} + \underbrace{\text{var}_{\boldsymbol{W}}\left[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\right]}_{\text{epistemic uncertainty}}$$

Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$
- Apply with $U = Y(\boldsymbol{x}')$ and $V = \boldsymbol{w}$:

$$\text{var}\big[Y(\boldsymbol{x}')\big] = \underbrace{\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right]}_{\text{aleatoric uncertainty}} + \underbrace{\text{var}_{\boldsymbol{W}}\left[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\right]}_{\text{epistemic uncertainty}}$$

- Aleatoric uncertainty: expectation of the variability for each $\boldsymbol{w}$;

Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$
- Apply with $U = Y(\boldsymbol{x}')$ and $V = \boldsymbol{w}$:

$$\text{var}\big[Y(\boldsymbol{x}')\big] = \underbrace{\mathbb{E}_{\boldsymbol{W}}\big[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\big]}_{\text{aleatoric uncertainty}} + \underbrace{\text{var}_{\boldsymbol{W}}\big[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\big]}_{\text{epistemic uncertainty}}$$

- Aleatoric uncertainty: expectation of the variability for each $\boldsymbol{w}$;
- Epistemic uncertainty results from the variability in estimating $\boldsymbol{w}$.

Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$
- Apply with $U = Y(\boldsymbol{x}')$ and $V = \boldsymbol{w}$:

$$\text{var}\left[Y(\boldsymbol{x}')\right] = \underbrace{\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right]}_{\text{aleatoric uncertainty}} + \underbrace{\text{var}_{\boldsymbol{W}}\left[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\right]}_{\text{epistemic uncertainty}}$$

- Aleatoric uncertainty: expectation of the variability for each $\boldsymbol{w}$;
- Epistemic uncertainty results from the variability in estimating $\boldsymbol{w}$.
- For $Y(\boldsymbol{x}') = \boldsymbol{x}'^T\boldsymbol{W} + N$, with $\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ and $N \sim \mathcal{N}(0, \sigma^2)$,

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}') = \mathcal{N}\Big(y; \boldsymbol{\mu}^T\boldsymbol{x}', \boldsymbol{x}'^T\mathbf{C}\boldsymbol{x}' + \sigma^2\Big)$$

Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$
- Apply with $U = Y(\boldsymbol{x}')$ and $V = \boldsymbol{w}$:

$$\text{var}\left[Y(\boldsymbol{x}')\right] = \underbrace{\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right]}_{\text{aleatoric uncertainty}} + \underbrace{\text{var}_{\boldsymbol{W}}\left[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\right]}_{\text{epistemic uncertainty}}$$

- Aleatoric uncertainty: expectation of the variability for each $\boldsymbol{w}$;
- Epistemic uncertainty results from the variability in estimating $\boldsymbol{w}$.
- For $Y(\boldsymbol{x}') = \boldsymbol{x}'^T\boldsymbol{W} + N$, with $\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ and $N \sim \mathcal{N}(0, \sigma^2)$,

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}') = \mathcal{N}\Big(y; \boldsymbol{\mu}^T\boldsymbol{x}', \boldsymbol{x}'^T\mathbf{C}\boldsymbol{x}' + \sigma^2\Big)$$

- Aleatoric: $\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right] = \mathbb{E}_{\boldsymbol{W}}\left[\sigma^2\right] = \sigma^2$.

Epistemic and Aleatoric Uncertainty

- Law of total variance: $\text{var}[U] = \mathbb{E}_V\left[\text{var}_U[U|V]\right] + \text{var}_V\left[\mathbb{E}[U|V]\right]$
- Apply with $U = Y(\boldsymbol{x}')$ and $V = \boldsymbol{w}$:

$$\text{var}\left[Y(\boldsymbol{x}')\right] = \underbrace{\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right]}_{\text{aleatoric uncertainty}} + \underbrace{\text{var}_{\boldsymbol{W}}\left[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\right]}_{\text{epistemic uncertainty}}$$

- Aleatoric uncertainty: expectation of the variability for each $\boldsymbol{w}$;
- Epistemic uncertainty results from the variability in estimating $\boldsymbol{w}$.
- For $Y(\boldsymbol{x}') = \boldsymbol{x}'^T\boldsymbol{W} + N$, with $\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ and $N \sim \mathcal{N}(0, \sigma^2)$,

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}') = \mathcal{N}\Big(y; \boldsymbol{\mu}^T\boldsymbol{x}', \boldsymbol{x}'^T\mathbf{C}\boldsymbol{x}' + \sigma^2\Big)$$

- Aleatoric: $\mathbb{E}_{\boldsymbol{W}}\left[\text{var}[Y(\boldsymbol{x}')|\boldsymbol{W}]\right] = \mathbb{E}_{\boldsymbol{W}}\left[\sigma^2\right] = \sigma^2$.
- Epistemic: $\text{var}_{\boldsymbol{W}}\left[\mathbb{E}[Y(\boldsymbol{x}')|\boldsymbol{w}]\right] = \text{var}_{\boldsymbol{W}}\left[\boldsymbol{x}'^T\boldsymbol{W}\right] = \boldsymbol{x}'^T\mathbf{C}\boldsymbol{x}'$

LASSO regression

- Alternative to ridge regression, with built-in variable selection

$$\hat{\boldsymbol{w}}_{\text{lasso}} = \arg\min_{\boldsymbol{w}} \; \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \, \|\boldsymbol{w}\|_1$$

where $\|\boldsymbol{w}\|_1 = \; \sum_i |w_i|$, the ℓ_1 norm.

- LASSO $=$ least absolute shrinkage and selection operator

LASSO regression

- Alternative to ridge regression, with built-in variable selection

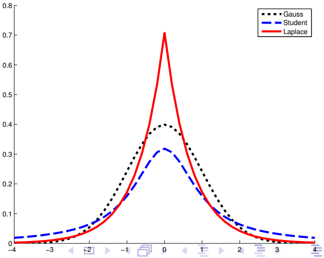
$$\hat{\boldsymbol{w}}_{\text{lasso}} = \arg\min_{\boldsymbol{w}} \; \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\, \|\boldsymbol{w}\|_1$$

where $\|\boldsymbol{w}\|_1 = \sum_i |w_i|$, the ℓ_1 norm.

- LASSO = least absolute shrinkage and selection operator

- Can be seen as MAP estimate of $\boldsymbol{w}$, under Laplacian prior

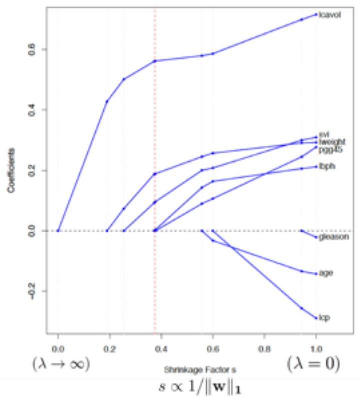
$$f_{\boldsymbol{W}}(\boldsymbol{w}) = \prod_{i=1}^{p} \frac{\lambda}{2} \exp\Big(-\lambda |w_i|\Big)$$
$$= \left(\frac{\lambda}{2}\right)^p \exp\Big(-\lambda \|\boldsymbol{w}\|_1\Big)$$



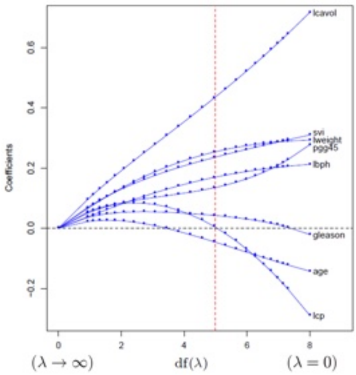
LASSO versus Ridge

- Example (prostate cancer data)

LASSO



Ridge



Solving LASSO Regression

- Ridge regression: simply a linear system:

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})\hat{\boldsymbol{w}}_{\text{ridge}} = \boldsymbol{X}^T\boldsymbol{y}$$

...may capitalize on many decades of work on numerical linear algebra.

Solving LASSO Regression

- Ridge regression: simply a linear system:

$$\big(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\big)\hat{\boldsymbol{w}}_{\text{ridge}} = \boldsymbol{X}^T\boldsymbol{y}$$

...may capitalize on many decades of work on numerical linear algebra.

- LASSO is much more challenging:

$$\hat{\boldsymbol{w}}_{\text{lasso}} = \arg\min_{\boldsymbol{w}} \ \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda \, \|\boldsymbol{w}\|_1$$

since $\|\boldsymbol{w}\|_1$ is non-differentiable (for any $w_i = 0$)

Solving LASSO Regression

- Ridge regression: simply a linear system:

$$\left(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}\right)\hat{\boldsymbol{w}}_{\text{ridge}} = \boldsymbol{X}^T\boldsymbol{y}$$

...may capitalize on many decades of work on numerical linear algebra.

- LASSO is much more challenging:

$$\hat{\boldsymbol{w}}_{\text{lasso}} = \arg\min_{\boldsymbol{w}} \; \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \lambda\,\|\boldsymbol{w}\|_1$$

since $\|\boldsymbol{w}\|_1$ is non-differentiable (for any $w_i = 0$)

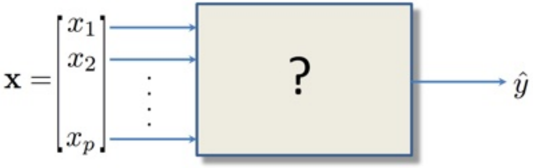
- Using gradient descent (*e.g.*, in deep learning), simply pretend that ℓ_1 is differentiable (derivative in $\{-1, 0, 1\}$), carefully adapt the step size.

Outline

1. Introduction
2. Regression
3. **Classification**
4. Optimization for Supervised Learning

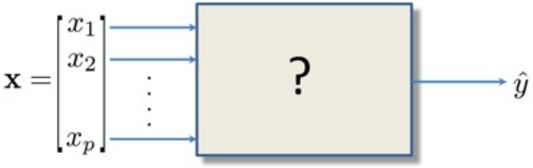
Classification (a.k.a. Pattern Recognition)

- In a nutshell: produce a "machine" that predicts/estimates/guesses a class $y \in \{1, ..., K\}$, from variables/features x_1,...,x_p



Classification (a.k.a. Pattern Recognition)

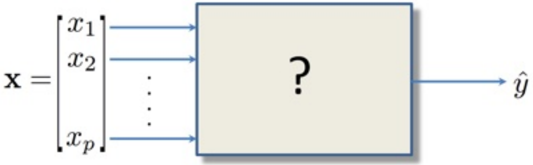
- In a nutshell: produce a "machine" that predicts/estimates/guesses a class $y \in \{1, ..., K\}$, from variables/features x_1,...,x_p



- Maybe the core machine learning problem, with countless applications.

Classification (a.k.a. Pattern Recognition)

- In a nutshell: produce a "machine" that predicts/estimates/guesses a class $y \in \{1, ..., K\}$, from variables/features x_1,...,x_p



- Maybe the core machine learning problem, with countless applications.
- Learning/training: given a collection of examples (training data)

$$\mathcal{D} = \big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big)$$

..find the "best" possible machine.

Generalized Linear Models

- Conditional probability of class y for sample $\boldsymbol{x}$:

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(y)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{\sum_{u=1}^{K} \exp\left((\boldsymbol{\eta}^{(u)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

Generalized Linear Models

- Conditional probability of class y for sample $\boldsymbol{x}$:

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(y)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{\sum_{u=1}^{K} \exp\left((\boldsymbol{\eta}^{(u)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

- Training data $\mathcal{D} = \big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big)$
  - ✓ Each y_i is a sample of $Y_i \sim f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_i)$
  - ✓ The samples are conditionally independent

Generalized Linear Models

- Conditional probability of class y for sample $\boldsymbol{x}$:

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(y)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{\sum_{u=1}^{K} \exp\left((\boldsymbol{\eta}^{(u)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

- Training data $\mathcal{D} = \big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big)$
  - ✓ Each y_i is a sample of $Y_i \sim f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_i)$
  - ✓ The samples are conditionally independent
- Parameters $\boldsymbol{\eta} = \left(\boldsymbol{\eta}^{(1)}, ..., \boldsymbol{\eta}^{(K)}\right)$, log-likelihood function:

$$\log f_{Y_1,...,Y_n}(y_1, ..., y_n; \boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{\eta}) = \sum_{i=1}^{n} \log f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i, \boldsymbol{\eta})$$

Generalized Linear Models

- Conditional probability of class y for sample $\boldsymbol{x}$:

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(y)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{\sum_{u=1}^{K} \exp\left((\boldsymbol{\eta}^{(u)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

- Training data $\mathcal{D} = \big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\big)$
  - ✓ Each y_i is a sample of $Y_i \sim f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_i)$
  - ✓ The samples are conditionally independent
- Parameters $\boldsymbol{\eta} = \left(\boldsymbol{\eta}^{(1)}, ..., \boldsymbol{\eta}^{(K)}\right)$, log-likelihood function:

$$\begin{aligned}\log f_{Y_1,...,Y_n}(y_1, ..., y_n; \boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{\eta}) &= \sum_{i=1}^{n} \log f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i, \boldsymbol{\eta}) \\ &= \sum_{i=1}^{n} \sum_{y=1}^{K} \mathbf{1}_{y=y_i} \log f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_i, \boldsymbol{\eta})\end{aligned}$$

moderly called cross-entropy loss.

The Binary Case: A Detailed Look

- Binary classification, $y \in \{1,\, 0\}$, thus

$$f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(1)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{\exp\left((\boldsymbol{\eta}^{(1)})^T \boldsymbol{\phi}(\boldsymbol{x})\right) + \exp\left((\boldsymbol{\eta}^{(0)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

The Binary Case: A Detailed Look

- Binary classification, $y \in \{1,\, 0\}$, thus

$$f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(1)})^T\boldsymbol{\phi}(\boldsymbol{x})\right)}{\exp\left((\boldsymbol{\eta}^{(1)})^T\boldsymbol{\phi}(\boldsymbol{x})\right) + \exp\left((\boldsymbol{\eta}^{(0)})^T\boldsymbol{\phi}(\boldsymbol{x})\right)}$$

- Dividing numerator and denominator by $\exp\left((\boldsymbol{\eta}^{(0)})^T\boldsymbol{\phi}(\boldsymbol{x})\right)$,

$$f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \frac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}{1 + \exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}$$

where $\boldsymbol{w} = \boldsymbol{\eta}^{(1)} - \boldsymbol{\eta}^{(0)}$.

The Binary Case: A Detailed Look

- Binary classification, $y \in \{1, 0\}$, thus

$$f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \frac{\exp\left((\boldsymbol{\eta}^{(1)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{\exp\left((\boldsymbol{\eta}^{(1)})^T \boldsymbol{\phi}(\boldsymbol{x})\right) + \exp\left((\boldsymbol{\eta}^{(0)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)}$$

- Dividing numerator and denominator by $\exp\left((\boldsymbol{\eta}^{(0)})^T \boldsymbol{\phi}(\boldsymbol{x})\right)$,

$$f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \frac{\exp\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{1 + \exp\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)} \equiv \text{sigmoid}\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)$$

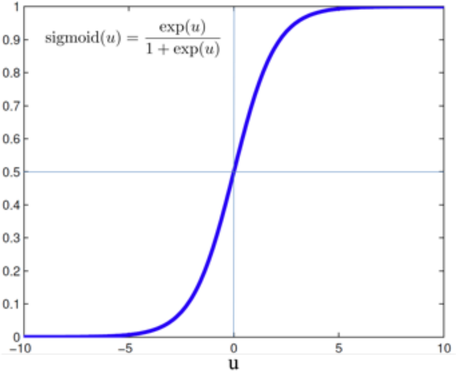
where $\boldsymbol{w} = \boldsymbol{\eta}^{(1)} - \boldsymbol{\eta}^{(0)}$.

Binary Logistic Regression

- Model: $f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \text{sigmoid}\big(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\big)$

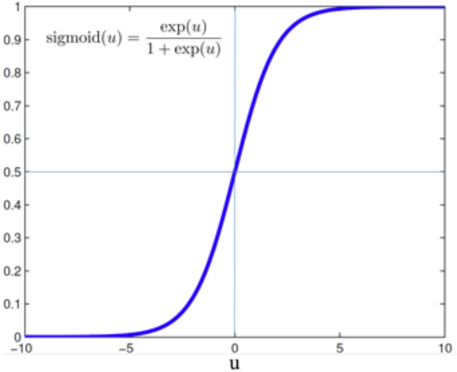
Binary Logistic Regression

- Model: $f_{Y|\boldsymbol{X}}(1|\boldsymbol{x}) = \text{sigmoid}\big(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\big)$



Binary Logistic Regression

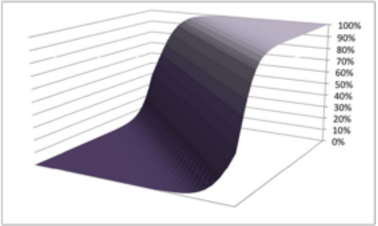
- Model: $f_{Y|\mathbf{X}}(1|\boldsymbol{x}) = \text{sigmoid}\big(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\big)$

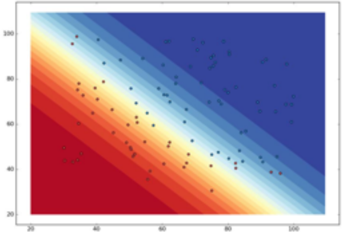


- Obviously $f_{Y|\mathbf{X}}(0|\boldsymbol{x}) = 1 - f_{Y|\mathbf{X}}(1|\boldsymbol{x})$.

Binary Logistic Regression

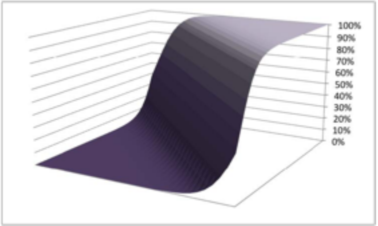
- In two dimensions ($\boldsymbol{w}, \phi(\boldsymbol{x}) \in \mathbb{R}^2$)

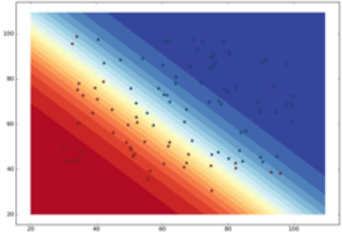




Binary Logistic Regression

- In two dimensions ($\boldsymbol{w}$, $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^2$)





- Classical decision boundary, $f_{Y|\mathbf{X}}(1|\boldsymbol{x}) = 1/2 \;\Leftrightarrow\; \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) = 0$, is linear with respect to $\boldsymbol{\phi}(\boldsymbol{x})$.

Binary Logistic Regression: Log-Likelihood

- $f_Y(y|\boldsymbol{x}) = \left(\dfrac{\exp\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)}{1 + \exp\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)} \right)^y \left(\dfrac{1}{1 + \exp\left(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})\right)} \right)^{(1-y)}$

Binary Logistic Regression: Log-Likelihood

- $f_Y(y|\boldsymbol{x}) = \left(\frac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^y \left(\frac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^{(1-y)}$

- Negative log-likelihood (NLL), given $\mathcal{D} = \Big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\Big)$,

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{i=1}^{n}\left(y_i \log \frac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)} + (1-y_i)\log\frac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}\right)$$

$$= \sum_{i=1}^{n}\left(\log\left[1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right] - y_i\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)$$

Binary Logistic Regression: Log-Likelihood

- $f_Y(y|\boldsymbol{x}) = \left(\dfrac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^y \left(\dfrac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^{(1-y)}$

- Negative log-likelihood (NLL), given $\mathcal{D} = \Big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\Big)$,

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{i=1}^{n}\left(y_i \log \frac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)} + (1-y_i)\log\frac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}\right)$$
$$= \sum_{i=1}^{n}\left(\log\left[1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right] - y_i\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)$$

- ML estimate $\hat{\boldsymbol{w}}_{\text{ML}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$

Binary Logistic Regression: Log-Likelihood

- $f_Y(y|\boldsymbol{x}) = \left(\dfrac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^y \left(\dfrac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^{(1-y)}$

- Negative log-likelihood (NLL), given $\mathcal{D} = \Big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\Big)$,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}) &= -\sum_{i=1}^{n}\left(y_i \log \frac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)} + (1-y_i)\log\frac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}\right)\\
&= \sum_{i=1}^{n}\Big(\log\big[1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)\big] - y_i\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\Big)
\end{aligned}$$

- ML estimate $\hat{\boldsymbol{w}}_{\text{ML}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$

- No closed form! We need optimization algorithms (later)

Binary Logistic Regression: Log-Likelihood

- $f_Y(y|\boldsymbol{x}) = \left(\dfrac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^{y}\left(\dfrac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})\right)}\right)^{(1-y)}$

- Negative log-likelihood (NLL), given $\mathcal{D} = \Big((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\Big)$,

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{i=1}^{n}\left(y_i \log \frac{\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)} + (1-y_i)\log\frac{1}{1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)}\right)$$
$$= \sum_{i=1}^{n}\left(\log\left[1+\exp\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)\right] - y_i\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)\right)$$

- ML estimate $\hat{\boldsymbol{w}}_{\text{ML}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w})$

- No closed form! We need optimization algorithms (later)

- $\mathcal{L}(\boldsymbol{w})$ is smooth and convex (should not be too hard to optimize)

Ridge and LASSO Logistic Regression

- Ridge logistic regression:

$$\hat{\boldsymbol{w}}_{\mathsf{ridge}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

still smooth and convex.

Ridge and LASSO Logistic Regression

- Ridge logistic regression:

$$\hat{\boldsymbol{w}}_{\text{ridge}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

still smooth and convex.

- Sparse (LASSO) logistic regression:

$$\hat{\boldsymbol{w}}_{\text{sparse}} = \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_1$$

still convex, but not smooth.

Multi-class Logistic Regression

- Recall the GLM,

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x},\boldsymbol{w}) = \frac{\exp\left(\phi(\boldsymbol{x})^T \boldsymbol{w}^{(y)}\right)}{\sum_{u=1}^{K} \exp\left(\phi(\boldsymbol{x})^T \boldsymbol{w}^{(u)}\right)}$$

... with $\boldsymbol{w} = (\boldsymbol{w}^{(1)}, ..., \boldsymbol{w}^{(K)})$.

Multi-class Logistic Regression

- Recall the GLM,

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x},\boldsymbol{w}) = \frac{\exp\left(\phi(\boldsymbol{x})^T \boldsymbol{w}^{(y)}\right)}{\sum_{u=1}^{K} \exp\left(\phi(\boldsymbol{x})^T \boldsymbol{w}^{(u)}\right)}$$

... with $\boldsymbol{w} = (\boldsymbol{w}^{(1)}, ..., \boldsymbol{w}^{(K)})$.

- This is called the multinomial/multi-class logistic, a.k.a. maximum entropy, softmax, ...

Multi-class Logistic Regression

- Recall the GLM,

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x},\boldsymbol{w}) = \frac{\exp\left(\phi(\boldsymbol{x})^T\boldsymbol{w}^{(y)}\right)}{\displaystyle\sum_{u=1}^{K}\exp\left(\phi(\boldsymbol{x})^T\boldsymbol{w}^{(u)}\right)}$$

... with $\boldsymbol{w} = (\boldsymbol{w}^{(1)}, ..., \boldsymbol{w}^{(K)})$.

- This is called the multinomial/multi-class logistic, a.k.a. maximum entropy, softmax, ...

- The negative log-likelihood function (cross-entropy loss):

$$\sum_{i=1}^{n}\log f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i,\boldsymbol{w}) = \sum_{i=1}^{n}\sum_{k=1}^{K}\mathbf{1}_{y_i=k}\log f_{Y|\boldsymbol{X}}(k|\boldsymbol{x}_i,\boldsymbol{\eta}),$$

Multi-class Logistic Regression (2)

- Using one-hot encoding: $\boldsymbol{y}_i \in \{0,1\}^K$, $y_{ik} = 1$ if $\boldsymbol{x}_i$ is in class k

Multi-class Logistic Regression (2)

- Using one-hot encoding: $\boldsymbol{y}_i \in \{0,1\}^K$, $y_{ik} = 1$ if $\boldsymbol{x}_i$ is in class k
- The negative multinomial logistic log-likelihood function

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log f_{Y|\boldsymbol{X}}(k|\boldsymbol{x}_i, \boldsymbol{w})$$

Multi-class Logistic Regression (2)

- Using one-hot encoding: $\boldsymbol{y}_i \in \{0,1\}^K$, $y_{ik} = 1$ if $\boldsymbol{x}_i$ is in class k
- The negative multinomial logistic log-likelihood function

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log f_{Y|\boldsymbol{X}}(k|\boldsymbol{x}_i, \boldsymbol{w})$$

can be written as

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \left[\log \left(\sum_{k=1}^{K} \exp(\boldsymbol{x}_i^T \boldsymbol{w}^{(k)}) \right) - \left(\sum_{k=1}^{K} y_{ik}\, \boldsymbol{x}_i^T \boldsymbol{w}^{(k)} \right) \right]$$

Multi-class Logistic Regression (2)

- Using one-hot encoding: $\boldsymbol{y}_i \in \{0,1\}^K$, $y_{ik} = 1$ if $\boldsymbol{x}_i$ is in class k
- The negative multinomial logistic log-likelihood function

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log f_{Y|\boldsymbol{X}}(k|\boldsymbol{x}_i, \boldsymbol{w})$$

can be written as

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \left[\log\left(\sum_{k=1}^{K} \exp(\boldsymbol{x}_i^T \boldsymbol{w}^{(k)}) \right) - \left(\sum_{k=1}^{K} y_{ik}\, \boldsymbol{x}_i^T \boldsymbol{w}^{(k)} \right) \right]$$

- Notice: if $\boldsymbol{x}_i$ is in class k, minimizing $\mathcal{L}(\boldsymbol{w})$ pushes $\boldsymbol{x}_i^T \boldsymbol{w}^{(k)}$ up.

Bayesian Logistic Regression

- Using some estimate $\hat{\boldsymbol{w}}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}, \hat{\boldsymbol{w}})$ ignores the randomness/uncertainty in $\hat{\boldsymbol{w}}$

Bayesian Logistic Regression

- Using some estimate $\hat{\boldsymbol{w}}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x},\hat{\boldsymbol{w}})$ ignores the randomness/uncertainty in $\hat{\boldsymbol{w}}$

- Bayesian approach: from a prior $f_{\boldsymbol{W}}(\boldsymbol{w})$, compute the posterior

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \frac{f_{\boldsymbol{W}}(\boldsymbol{w})\, f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w})}{f_{\boldsymbol{Y}}(\boldsymbol{y})}$$

where $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \prod_{i=1}^{N} f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i,\boldsymbol{w})$ (recall x_i are deterministic)

Bayesian Logistic Regression

- Using some estimate $\hat{\boldsymbol{w}}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}, \hat{\boldsymbol{w}})$ ignores the randomness/uncertainty in $\hat{\boldsymbol{w}}$

- Bayesian approach: from a prior $f_{\boldsymbol{W}}(\boldsymbol{w})$, compute the posterior

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \frac{f_{\boldsymbol{W}}(\boldsymbol{w})\, f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w})}{f_{\boldsymbol{Y}}(\boldsymbol{y})}$$

where $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \prod_{i=1}^{N} f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i, \boldsymbol{w})$ (recall $\boldsymbol{x}_i$ are deterministic)

- Given some new point $\boldsymbol{x}_*$, the predictive distribution is

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*, \boldsymbol{y}) = \int f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y})\, f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*, \boldsymbol{w})\, d\boldsymbol{w}$$

Bayesian Logistic Regression

- Using some estimate $\hat{\boldsymbol{w}}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}, \hat{\boldsymbol{w}})$ ignores the randomness/uncertainty in $\hat{\boldsymbol{w}}$

- Bayesian approach: from a prior $f_{\boldsymbol{W}}(\boldsymbol{w})$, compute the posterior

$$f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y}) = \frac{f_{\boldsymbol{W}}(\boldsymbol{w})\, f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w})}{f_{\boldsymbol{Y}}(\boldsymbol{y})}$$

where $f_{\boldsymbol{Y}|\boldsymbol{W}}(\boldsymbol{y}|\boldsymbol{w}) = \prod_{i=1}^{N} f_{Y|\boldsymbol{X}}(y_i|\boldsymbol{x}_i, \boldsymbol{w})$ (recall $\boldsymbol{x}_i$ are deterministic)

- Given some new point $\boldsymbol{x}_*$, the predictive distribution is

$$f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*, \boldsymbol{y}) = \int f_{\boldsymbol{W}|\boldsymbol{Y}}(\boldsymbol{w}|\boldsymbol{y})\, f_{Y|\boldsymbol{X}}(y|\boldsymbol{x}_*, \boldsymbol{w})\, d\boldsymbol{w}$$

- Unfortunately, none of these have closed-form expressions.

Bayesian Logistic Regression (2)

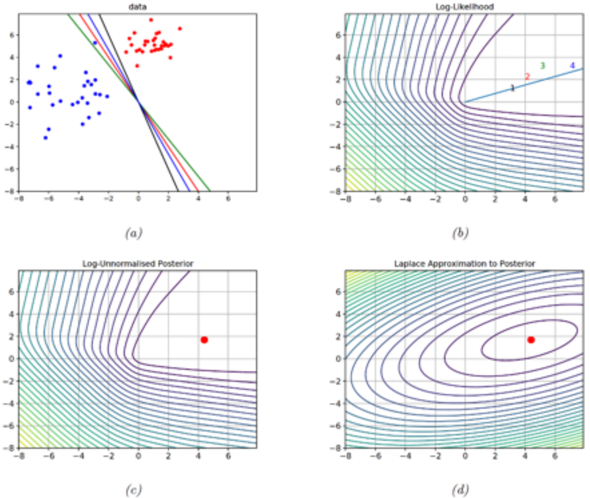


Figure 10.13: (a) Illustration of the data. (b) Log-likelihood for a logistic regression model. The line is drawn from the origin in the direction of the MLE (which is at infinity). The numbers correspond to 4 points in parameter space, corresponding to the lines in (a). (c) Unnormalized log posterior (assuming vague spherical prior). (d) Laplace approximation to posterior. Adapted from a figure by Mark Girolami. Generated by code at figures.probml.ai/book1/10.13.

Bayesian Logistic Regression (3)

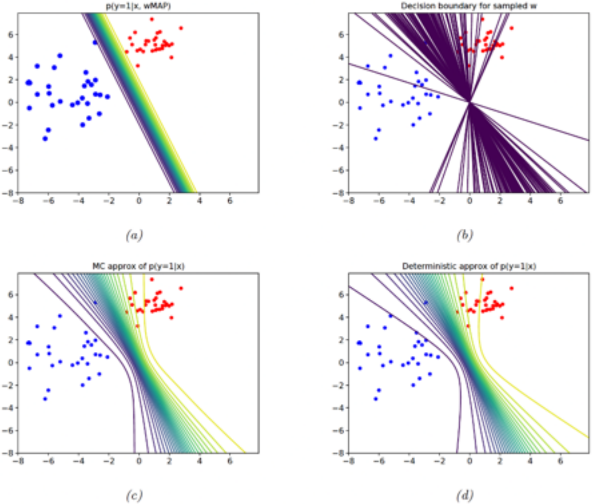
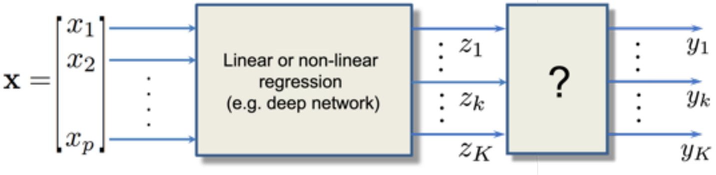
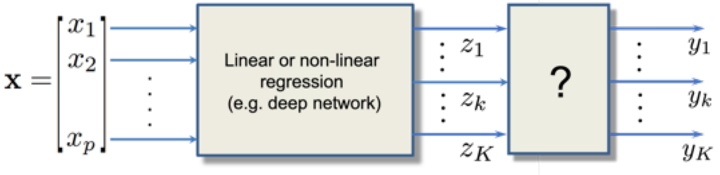


Figure 10.14: Posterior predictive distribution for a logistic regression model in 2d. (a): contours of $p(y = 1|\boldsymbol{x}, \hat{\boldsymbol{w}}_{map})$*. (b): samples from the posterior predictive distribution. (c): Averaging over these samples. (d): moderated output (probit approximation). Adapted from a figure by Mark Girolami. Generated by code at* *figures.probml.ai/book1/10.14.*

Another View of (and Beyond) Softmax

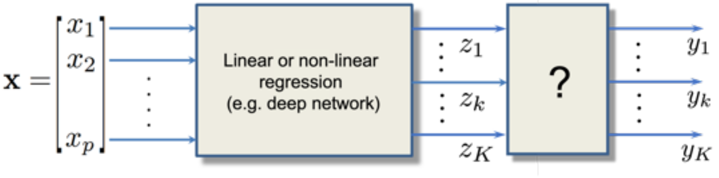


Another View of (and Beyond) Softmax



- Scores: $\boldsymbol{z} \in \mathbb{R}^K$, without constraints/restrictions.

Another View of (and Beyond) Softmax

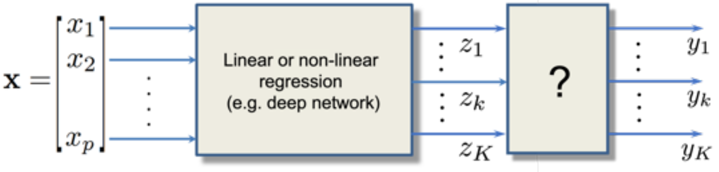


- Scores: $\boldsymbol{z} \in \mathbb{R}^K$, without constraints/restrictions.

- Probabilities: $y_k = \mathbb{P}[\text{class } k|\boldsymbol{x}]$, thus $\boldsymbol{y} \in \Delta_{K-1}$, where

$$\Delta_{K-1} = \left\{\boldsymbol{y} \in \mathbb{R}^K, \text{ s.t. } y_1, ..., y_K \geq 0 \text{ and } \sum_{k=1}^{K} y_i = 1\right\} \qquad \text{(simplex)}$$

Another View of (and Beyond) Softmax



- Scores: $\boldsymbol{z} \in \mathbb{R}^K$, without constraints/restrictions.

- Probabilities: $y_k = \mathbb{P}[\text{class } k|\boldsymbol{x}]$, thus $\boldsymbol{y} \in \Delta_{K-1}$, where

$$\Delta_{K-1} = \left\{ \boldsymbol{y} \in \mathbb{R}^K, \text{ s.t. } y_1, ..., y_K \geq 0 \text{ and } \sum_{k=1}^{K} y_i = 1 \right\} \qquad \text{(simplex)}$$

- How to map from $\boldsymbol{z} \in \mathbb{R}^K$ to $\boldsymbol{y} \in \Delta_{K-1}$, such that

$$z_i = z_j \;\Rightarrow\; y_i = y_j \quad \text{and} \quad z_i > z_j \Rightarrow y_i \geq y_j$$

Argmax and Softmax

- First possibility: probability vector "most aligned" with $\boldsymbol{z}$:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \boldsymbol{p}^T \boldsymbol{z} \quad \Longrightarrow \quad y_k \neq 0 \Leftrightarrow k \in \arg\max_j \{z_j,\ j = 1, ..., K\}$$

.

Argmax and Softmax

- First possibility: probability vector "most aligned" with $\boldsymbol{z}$:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \boldsymbol{p}^T \boldsymbol{z} \quad \Longrightarrow \quad y_k \neq 0 \Leftrightarrow k \in \arg\max_j \{z_j,\ j = 1, ..., K\}$$

called the $\color{red}{\text{argmax}}$ operator/mapping.

Argmax and Softmax

- First possibility: probability vector "most aligned" with $\boldsymbol{z}$:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T \boldsymbol{z} \quad \Longrightarrow \quad y_k \neq 0 \Leftrightarrow k \in \arg\max_j \{z_j,\ j = 1, ..., K\}$$

called the argmax operator/mapping.

- Second possibility: encourage more uniform probability distribution:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T \boldsymbol{z} + H(\boldsymbol{p})$$

where $H(\boldsymbol{p})$ is Shannon's entropy,

$$H(\boldsymbol{p}) = -\sum_{k=1}^{K} p_i \log p_i$$

Argmax and Softmax

- First possibility: probability vector "most aligned" with $\boldsymbol{z}$:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \boldsymbol{p}^T \boldsymbol{z} \quad \Longrightarrow \quad y_k \neq 0 \Leftrightarrow k \in \arg\max_j \{z_j,\ j = 1, ..., K\}$$

called the argmax operator/mapping.

- Second possibility: encourage more uniform probability distribution:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \boldsymbol{p}^T \boldsymbol{z} + H(\boldsymbol{p})$$

where $H(\boldsymbol{p})$ is Shannon's entropy,

$$H(\boldsymbol{p}) = -\sum_{k=1}^{K} p_i \log p_i$$

- H satisfies: $H(\boldsymbol{p}) \geq 0$ and $H(\boldsymbol{p}) \leq \log K$ (attained for $p_i = 1/K$).

Argmax and Softmax

- First possibility: probability vector "most aligned" with $\boldsymbol{z}$:

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T\boldsymbol{z} \quad \Longrightarrow \quad y_k \neq 0 \Leftrightarrow k \in \arg\max_j\{z_j,\ j=1,...,K\}$$

called the argmax operator/mapping.

- Second possibility: encourage more uniform probability distribution:

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T\boldsymbol{z} + H(\boldsymbol{p}) \quad \Rightarrow \quad \boldsymbol{y} = \textbf{softmax}(\boldsymbol{z}),\ \text{i.e. } y_k \propto \exp(z_k)$$

where $H(\boldsymbol{p})$ is Shannon's entropy,

$$H(\boldsymbol{p}) = -\sum_{k=1}^{K} p_i \log p_i$$

- H satisfies: $H(\boldsymbol{p}) \geq 0$ and $H(\boldsymbol{p}) \leq \log K$ (attained for $p_i = 1/K$).

Softmax as Maximum Entropy

- Encouraging high entropy (with weight $1/\beta$):

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \;+\; H(\boldsymbol{p})$$

Softmax as Maximum Entropy

- Encouraging high entropy (with weight $1/\beta$):

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}\in\Delta_{K-1}} \beta\,\boldsymbol{p}^T\boldsymbol{z} \;+\; H(\boldsymbol{p})$$

- Add Lagrangian for the simplex constraint:

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}} \beta\,\boldsymbol{p}^T\boldsymbol{z} \;+\; H(\boldsymbol{p}) \;+\lambda\,(\mathbf{1}^T\boldsymbol{p} - 1)$$

Softmax as Maximum Entropy

- Encouraging high entropy (with weight $1/\beta$):

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \; + \; H(\boldsymbol{p})$$

- Add Lagrangian for the simplex constraint:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \; + \; H(\boldsymbol{p}) \; + \lambda\, (\boldsymbol{1}^T \boldsymbol{p} - 1)$$

- Taking derivatives (gradient) w.r.t. $p_1, ..., p_K$ and equating to zero:

$$\beta\, z_i - 1 - \log p_i + \lambda = 0$$

Softmax as Maximum Entropy

- Encouraging high entropy (with weight $1/\beta$):

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \; + \; H(\boldsymbol{p})$$

- Add Lagrangian for the simplex constraint:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \; + \; H(\boldsymbol{p}) \; + \lambda\, (\mathbf{1}^T \boldsymbol{p} - 1)$$

- Taking derivatives (gradient) w.r.t. $p_1, ..., p_K$ and equating to zero:

$$\beta\, z_i - 1 - \log p_i + \lambda = 0 \quad \Leftrightarrow \quad p_i = \exp\big[\beta\, z_i + \lambda - 1\big]$$

Softmax as Maximum Entropy

- Encouraging high entropy (with weight $1/\beta$):

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p} \in \Delta_{K-1}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \; + \; H(\boldsymbol{p})$$

- Add Lagrangian for the simplex constraint:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \; + \; H(\boldsymbol{p}) \; + \lambda \, (\boldsymbol{1}^T \boldsymbol{p} - 1)$$

- Taking derivatives (gradient) w.r.t. $p_1, ..., p_K$ and equating to zero:

$$\beta\, z_i - 1 - \log p_i + \lambda = 0 \quad \Leftrightarrow \quad p_i = \exp\big[\beta\, z_i + \lambda - 1\big] = \frac{e^{\beta\, z_i}}{Z(\beta, \lambda)}$$

Softmax as Maximum Entropy

- Encouraging high entropy (with weight $1/\beta$):

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}\in\Delta_{K-1}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \;+\; H(\boldsymbol{p})$$

- Add Lagrangian for the simplex constraint:

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}} \beta\, \boldsymbol{p}^T \boldsymbol{z} \;+\; H(\boldsymbol{p}) \;+ \lambda\, (\mathbf{1}^T \boldsymbol{p} - 1)$$

- Taking derivatives (gradient) w.r.t. $p_1, ..., p_K$ and equating to zero:

$$\beta\, z_i - 1 - \log p_i + \lambda = 0 \;\;\Leftrightarrow\;\; p_i = \exp\big[\beta\, z_i + \lambda - 1\big] = \frac{e^{\beta\, z_i}}{Z(\beta, \lambda)}$$

- Choosing λ to satisfy the constraint $\mathbf{1}^T \boldsymbol{p} = 1$ determines $Z(\beta, \lambda)$

$$y_i = \frac{e^{\beta\, z_i}}{\sum_{j=1}^K e^{\beta\, z_j}} = \big[\mathbf{softmax}(\beta\, \boldsymbol{z})\big]_i$$

Beyond Softmax: Sparsemax

- A third possibility[1]: simply project $\boldsymbol{z}$ onto Δ_{K-1}

$$\boldsymbol{y} = \arg\min_{\boldsymbol{p}\in\Delta_{K-1}} \|\boldsymbol{p} - \boldsymbol{z}\|_2^2 \quad \Longrightarrow \quad \boldsymbol{y} = \textbf{sparsemax}(\boldsymbol{z})$$

[1] A. Martins and R. Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification", ICML, 2016.

Beyond Softmax: Sparsemax

- A third possibility[1]: simply project $\boldsymbol{z}$ onto Δ_{K-1}

$$\boldsymbol{y} = \arg\min_{\boldsymbol{p}\in\Delta_{K-1}} \|\boldsymbol{p}-\boldsymbol{z}\|_2^2 \quad \Longrightarrow \quad \boldsymbol{y} = \textbf{sparsemax}(\boldsymbol{z})$$

- It can also be written as

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T\boldsymbol{z} - \frac{1}{2}\|\boldsymbol{p}\|_2^2$$

[1] A. Martins and R. Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification", ICML, 2016.

Beyond Softmax: Sparsemax

- A third possibility[1]: simply project $\boldsymbol{z}$ onto Δ_{K-1}

$$\boldsymbol{y} = \arg \min_{\boldsymbol{p}\in\Delta_{K-1}} \|\boldsymbol{p} - \boldsymbol{z}\|_2^2 \quad \Longrightarrow \quad \boldsymbol{y} = \mathbf{sparsemax}(\boldsymbol{z})$$

- It can also be written as

$$\boldsymbol{y} = \arg \max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T\boldsymbol{z} - \frac{1}{2}\|\boldsymbol{p}\|_2^2$$

- $-\|\boldsymbol{p}\|_2^2$ is (up to a constant) a Tsallis entropy.

[1] A. Martins and R. Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification", ICML, 2016.

Beyond Softmax: Sparsemax

- A third possibility[1]: simply project $\boldsymbol{z}$ onto Δ_{K-1}

$$\boldsymbol{y} = \arg\min_{\boldsymbol{p}\in\Delta_{K-1}} \|\boldsymbol{p} - \boldsymbol{z}\|_2^2 \quad \Longrightarrow \quad \boldsymbol{y} = \textbf{sparsemax}(\boldsymbol{z})$$

- It can also be written as

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}\in\Delta_{K-1}} \boldsymbol{p}^T\boldsymbol{z} - \frac{1}{2}\|\boldsymbol{p}\|_2^2$$

- $-\|\boldsymbol{p}\|_2^2$ is (up to a constant) a Tsallis entropy.

- General family, where Ω is some entropy,

$$\boldsymbol{y} = \arg\max_{\boldsymbol{p}\in\Delta_{K-1}} \beta\,\boldsymbol{p}^T\boldsymbol{z} + \Omega(\boldsymbol{p})$$

[1] A. Martins and R. Astudillo. "From softmax to sparsemax: A sparse model of attention and multi-label classification", ICML, 2016.

Argmax, Softmax, and Sparsemax

- All these mappings satisfy: $\boldsymbol{z}' = \boldsymbol{z} + \alpha \mathbf{1} \;\Rightarrow\; \boldsymbol{y}' = \boldsymbol{y}$

Argmax, Softmax, and Sparsemax

- All these mappings satisfy: $\boldsymbol{z}' = \boldsymbol{z} + \alpha\mathbf{1} \;\Rightarrow\; \boldsymbol{y}' = \boldsymbol{y}$
- They are also permutation equivariant: if R is a permutation,

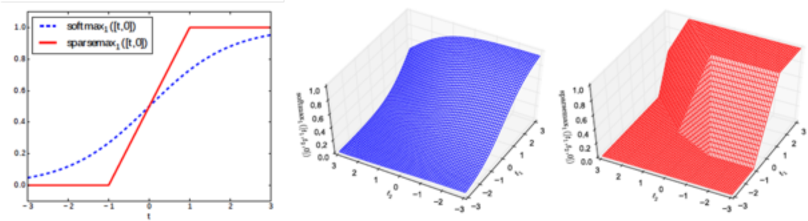
$$\boldsymbol{z}' = R(\boldsymbol{z}) \;\Rightarrow\; \boldsymbol{y}' = R(\boldsymbol{y})$$

Argmax, Softmax, and Sparsemax

- All these mappings satisfy: $\boldsymbol{z}' = \boldsymbol{z} + \alpha \mathbf{1} \;\Rightarrow\; \boldsymbol{y}' = \boldsymbol{y}$
- They are also permutation equivariant: if R is a permutation,

$$\boldsymbol{z}' = R(\boldsymbol{z}) \;\Rightarrow\; \boldsymbol{y}' = R(\boldsymbol{y})$$

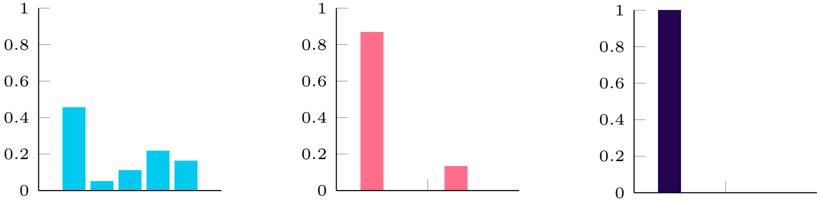
- Sparsemax versus softmax:



Argmax, Softmax, and Sparsemax

- Sparsemax is in-between softmax and argmax
- For $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$

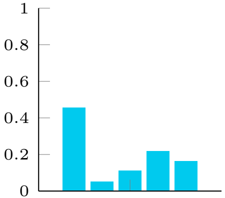
$\mathbf{softmax}(\boldsymbol{z})$ $\mathbf{sparsemax}(\boldsymbol{z})$ $\mathrm{argmax}(\boldsymbol{z})$



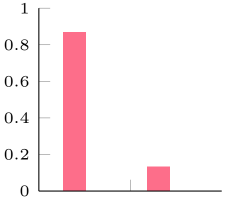
Argmax, Softmax, and Sparsemax

- Sparsemax is in-between softmax and argmax
- For $\boldsymbol{z} = [1.0716, -1.1221, -0.3288, 0.3368, 0.0425]$

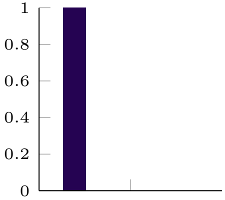
$\mathbf{softmax}(\boldsymbol{z})$



$\mathbf{sparsemax}(\boldsymbol{z})$



$\mathrm{argmax}(\boldsymbol{z})$



- Sparsemax, unlike softmax, may yield exact zeros.

Temperature

- Softmax and sparsemax may include a "temperature" parameter T,
- Scale the argument by $1/T$: $\mathbf{softmax}(\boldsymbol{z}/T)$ and $\mathbf{sparsemax}(\boldsymbol{z}/T)$

Temperature

- Softmax and sparsemax may include a "temperature" parameter T,
- Scale the argument by $1/T$: $\mathbf{softmax}(\boldsymbol{z}/T)$ and $\mathbf{sparsemax}(\boldsymbol{z}/T)$
- Zero temperature limit:

$$\lim_{T\to 0} \mathbf{softmax}(\boldsymbol{z}/T) = \lim_{T\to 0} \mathbf{sparsemax}(\boldsymbol{z}/T) = \operatorname{argmax}(\boldsymbol{z})$$

Temperature

- Softmax and sparsemax may include a "temperature" parameter T,
- Scale the argument by $1/T$: $\mathbf{softmax}(\boldsymbol{z}/T)$ and $\mathbf{sparsemax}(\boldsymbol{z}/T)$
- Zero temperature limit:

$$\lim_{T\to 0} \mathbf{softmax}(\boldsymbol{z}/T) = \lim_{T\to 0} \mathbf{sparsemax}(\boldsymbol{z}/T) = \operatorname{argmax}(\boldsymbol{z})$$

- High temperature limit:

$$\lim_{T\to \infty} \mathbf{softmax}(\boldsymbol{z}/T) = \lim_{T\to \infty} \mathbf{sparsemax}(\boldsymbol{z}/T) = \left(\frac{1}{K}, ..., \frac{1}{K}\right)$$

Temperature

- Softmax and sparsemax may include a "temperature" parameter T,
- Scale the argument by $1/T$: $\mathbf{softmax}(\boldsymbol{z}/T)$ and $\mathbf{sparsemax}(\boldsymbol{z}/T)$
- Zero temperature limit:

$$\lim_{T\to 0} \mathbf{softmax}(\boldsymbol{z}/T) = \lim_{T\to 0} \mathbf{sparsemax}(\boldsymbol{z}/T) = \operatorname{argmax}(\boldsymbol{z})$$

- High temperature limit:

$$\lim_{T\to \infty} \mathbf{softmax}(\boldsymbol{z}/T) = \lim_{T\to \infty} \mathbf{sparsemax}(\boldsymbol{z}/T) = \left(\frac{1}{K}, ..., \frac{1}{K}\right)$$

- The temperature controls how peaked the softmax is and how sparse the sparsemax is.

Classification: The Loss Function Perspective

- Consider binary classifiers of the form $\hat{y}(\boldsymbol{x}) = \text{sign}\big(f(\boldsymbol{x};\boldsymbol{\theta})\big)$

Classification: The Loss Function Perspective

- Consider binary classifiers of the form $\hat{y}(\boldsymbol{x}) = \text{sign}\big(f(\boldsymbol{x};\boldsymbol{\theta})\big)$
- In the linear case, $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{x}$

Classification: The Loss Function Perspective

- Consider binary classifiers of the form $\hat{y}(\boldsymbol{x}) = \text{sign}\big(f(\boldsymbol{x};\boldsymbol{\theta})\big)$

- In the linear case, $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{x}$

- Both logistic regression and SVM can be seen as minimizing a regularized loss:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} \quad + \quad \frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}$$

Classification: The Loss Function Perspective

- Consider binary classifiers of the form $\hat{y}(\boldsymbol{x}) = \text{sign}\big(f(\boldsymbol{x};\boldsymbol{\theta})\big)$

- In the linear case, $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{x}$

- Both logistic regression and SVM can be seen as minimizing a regularized loss:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} \quad + \quad \frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}$$

- Logistic loss: $L_{\text{logistic}}(f, y) \propto \log\big(1 + \exp(-y\, f)\big)$

Classification: The Loss Function Perspective

- Consider binary classifiers of the form $\hat{y}(\boldsymbol{x}) = \text{sign}\big(f(\boldsymbol{x};\boldsymbol{\theta})\big)$
- In the linear case, $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{x}$
- Both logistic regression and SVM can be seen as minimizing a regularized loss:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} \quad + \quad \frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}$$

- Logistic loss: $L_{\text{logistic}}(f, y) \propto \log\big(1 + \exp(-y\, f)\big)$
- Hinge loss: $L_{\text{hinge}}(f, y) \propto \max\{0, 1 - y\, f\}$
  ... underlies support vector machines (SVM)

Classification: The Loss Function Perspective (2)

- Both the hinge and the logistic loss can be seen as convex replacements for the error loss (or misclassification loss)

$$L_{\text{error}}(f, y) \propto \mathbf{1}_{y\,f<0} = \begin{cases} 1 & \Leftarrow \quad \text{sign}(f) \neq y \\ 0 & \Leftarrow \quad \text{sign}(f) = y \end{cases}$$

Classification: The Loss Function Perspective (2)

- Both the hinge and the logistic loss can be seen as convex replacements for the error loss (or misclassification loss)

$$L_{\text{error}}(f, y) \propto \mathbf{1}_{y\,f<0} = \begin{cases} 1 & \Leftarrow \quad \text{sign}(f) \neq y \\ 0 & \Leftarrow \quad \text{sign}(f) = y \end{cases}$$

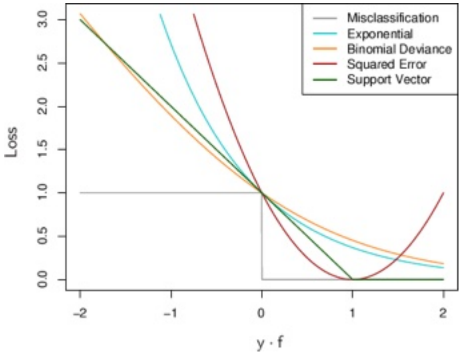
- Naturally, other losses can be used (binomial deviance = logistic):

Classification: The Loss Function Perspective (2)

- Both the hinge and the logistic loss can be seen as convex replacements for the error loss (or misclassification loss)

$$L_{\text{error}}(f, y) \propto \mathbf{1}_{y\,f<0} = \begin{cases} 1 & \Leftarrow \quad \text{sign}(f) \neq y \\ 0 & \Leftarrow \quad \text{sign}(f) = y \end{cases}$$

- Naturally, other losses can be used (binomial deviance = logistic):



Classification: Empirical and Expected Risk

- The quantity

$$\frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

Classification: Empirical and Expected Risk

- The quantity

$$\frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

is a sample-based (empirical) estimate of the expected loss (the risk)

$$\mathbb{E}\left[L(f(\boldsymbol{X};\boldsymbol{\theta}), Y)\right] = \mathcal{R}[f(\cdot;\boldsymbol{\theta})]$$

Classification: Empirical and Expected Risk

- The quantity (empirical risk)

$$\frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) = \mathcal{R}_{\text{emp}}[f(\cdot;\boldsymbol{\theta})]$$

is a sample-based (empirical) estimate of the expected loss (the risk)

$$\mathbb{E}\left[L(f(\boldsymbol{X};\boldsymbol{\theta}), Y)\right] = \mathcal{R}[f(\cdot;\boldsymbol{\theta})]$$

Classification: Empirical and Expected Risk

- The quantity (empirical risk)

$$\frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) = \mathcal{R}_{\mathsf{emp}}[f(\cdot;\boldsymbol{\theta})]$$

  is a sample-based (empirical) estimate of the expected loss (the risk)

$$\mathbb{E}\left[L(f(\boldsymbol{X};\boldsymbol{\theta}), Y)\right] = \mathcal{R}[f(\cdot;\boldsymbol{\theta})]$$

- Of course, $\mathcal{R}[f(\cdot;\boldsymbol{\theta})]$ cannot be computed: $f_{\boldsymbol{X},Y}$ is unknown. Instead, we have training data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n) \sim f_{\boldsymbol{X},Y}$, i.i.d.

Classification: Empirical and Expected Risk

- The quantity (empirical risk)

$$\frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) = \mathcal{R}_{\text{emp}}[f(\cdot;\boldsymbol{\theta})]$$

  is a sample-based (empirical) estimate of the expected loss (the risk)

$$\mathbb{E}\left[L(f(\boldsymbol{X};\boldsymbol{\theta}), Y)\right] = \mathcal{R}[f(\cdot;\boldsymbol{\theta})]$$

- Of course, $\mathcal{R}[f(\cdot;\boldsymbol{\theta})]$ cannot be computed: $f_{\boldsymbol{X},Y}$ is unknown. Instead, we have training data $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n) \sim f_{\boldsymbol{X},Y}$, i.i.d.

- Logistic regression and SVMs solve regularized ERM problems, with convex surrogates of the error loss

Outline

1 Introduction

2 Regression

3 Classification

4 Optimization for Supervised Learning

Classification: The Loss Function Perspective

- Recall that supervised learning can be formulated as regularized empirical risk minimization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} + \overbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}}^{\text{empirical risk}}$$

Classification: The Loss Function Perspective

- Recall that supervised learning can be formulated as regularized empirical risk minimization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} \quad + \quad \overbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}}^{\text{empirical risk}}$$

- Quadratic loss: $L_{\text{quadratic}}(f, y) \propto (f - y)^2$

Classification: The Loss Function Perspective

- Recall that supervised learning can be formulated as regularized empirical risk minimization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} + \overbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}}^{\text{empirical risk}}$$

- Quadratic loss: $L_{\text{quadratic}}(f, y) \propto (f - y)^2$

- Logistic loss: $L_{\text{logistic}}(f, y) \propto \log\big(1 + \exp(-y\, f)\big)$

Classification: The Loss Function Perspective

- Recall that supervised learning can be formulated as regularized empirical risk minimization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} \quad + \quad \overbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}}^{\text{empirical risk}}$$

- Quadratic loss: $L_{\text{quadratic}}(f, y) \propto (f - y)^2$
- Logistic loss: $L_{\text{logistic}}(f, y) \propto \log\big(1 + \exp(-y\, f)\big)$
- Hinge loss: $L_{\text{hinge}}(f, y) \propto \max\{0, 1 - y\, f\}$

Classification: The Loss Function Perspective

- Recall that supervised learning can be formulated as regularized empirical risk minimization:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \quad \underbrace{R(\boldsymbol{\theta})}_{\text{regularizer}} \quad + \quad \overbrace{\frac{1}{n}\sum_{i=1}^{n} \underbrace{L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)}_{\text{loss}}}^{\text{empirical risk}}$$

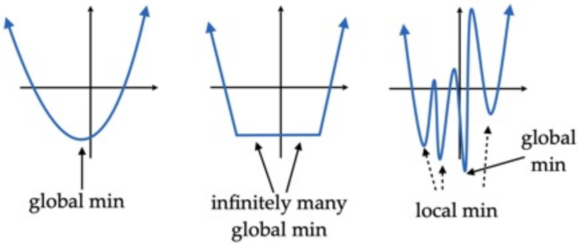
- Quadratic loss: $L_{\text{quadratic}}(f, y) \propto (f - y)^2$
- Logistic loss: $L_{\text{logistic}}(f, y) \propto \log\big(1 + \exp(-y\, f)\big)$
- Hinge loss: $L_{\text{hinge}}(f, y) \propto \max\{0, 1 - y\, f\}$
- Absolute error loss: $L_{\text{abs}}(f, y) \propto |f - y|$ (not covered today)

Minimizers

- Goal: find $\boldsymbol{\theta}^*$, a minimizer of $F(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} \in \mathbb{R}^d$

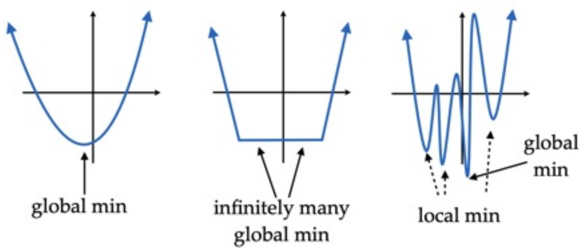
Minimizers

- Goal: find $\boldsymbol{\theta}^*$, a minimizer of $F(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} \in \mathbb{R}^d$
- Types of minimizers:
  - ✓ global, if $F(\boldsymbol{\theta}^*) \leq F(\boldsymbol{\theta})$, for any $\boldsymbol{\theta} \in \mathbb{R}^d$
  - ✓ local, if $F(\boldsymbol{\theta}^*) \leq F(\boldsymbol{\theta})$, for any $\boldsymbol{\theta} \in \mathbb{R}^d$ s.t. $\|\boldsymbol{\theta} - \boldsymbol{\theta}\| \leq \varepsilon$, for some ε.



Minimizers

- Goal: find $\boldsymbol{\theta}^*$, a minimizer of $F(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta} \in \mathbb{R}^d$
- Types of minimizers:
  - ✓ global, if $F(\boldsymbol{\theta}^*) \leq F(\boldsymbol{\theta})$, for any $\boldsymbol{\theta} \in \mathbb{R}^d$
  - ✓ local, if $F(\boldsymbol{\theta}^*) \leq F(\boldsymbol{\theta})$, for any $\boldsymbol{\theta} \in \mathbb{R}^d$ s.t. $\|\boldsymbol{\theta} - \boldsymbol{\theta}\| \leq \varepsilon$, for some ε.



- Minimizers: global $\Rightarrow$ local; local $\not\Rightarrow$ global.

Convexity

- F is a convex function if, for all $\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in [0,\, 1] \;\Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) \leq \lambda F(\boldsymbol{\theta}_1) + (1-\lambda) F(\boldsymbol{\theta}_2)$$

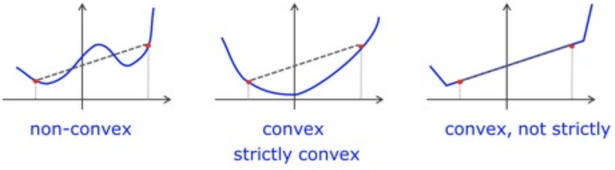
Convexity

- F is a convex function if, for all $\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in [0,\, 1] \;\Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) \leq \lambda F(\boldsymbol{\theta}_1) + (1-\lambda)F(\boldsymbol{\theta}_2)$$

- F is a strictly convex function if, for all $\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in \,]\,0,\, 1\,[\; \Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) < \lambda F(\boldsymbol{\theta}_1) + (1-\lambda)F(\boldsymbol{\theta}_2)$$



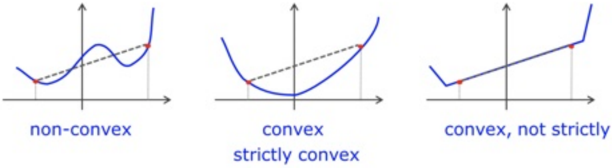
Convexity

- F is a convex function if, for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in [0,\, 1] \;\Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) \leq \lambda F(\boldsymbol{\theta}_1) + (1-\lambda)F(\boldsymbol{\theta}_2)$$

- F is a strictly convex function if, for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in \,]\,0,\, 1\,[\; \Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) < \lambda F(\boldsymbol{\theta}_1) + (1-\lambda)F(\boldsymbol{\theta}_2)$$



- Convexity $\Rightarrow$ all local minima are global minima.

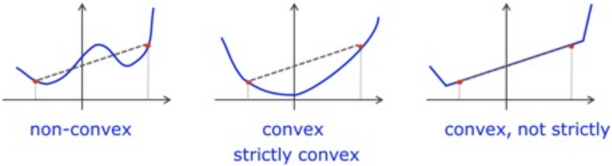
Convexity

- F is a convex function if, for all $\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in [0,\, 1] \;\Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) \leq \lambda F(\boldsymbol{\theta}_1) + (1-\lambda) F(\boldsymbol{\theta}_2)$$

- F is a strictly convex function if, for all $\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \in \mathbb{R}^d$,

$$\lambda \in \,]0,\, 1[\; \Rightarrow\; F(\lambda\boldsymbol{\theta}_1 + (1-\lambda)\boldsymbol{\theta}_2) < \lambda F(\boldsymbol{\theta}_1) + (1-\lambda) F(\boldsymbol{\theta}_2)$$



- Convexity $\Rightarrow$ all local minima are global minima.

- Convexity $\Rightarrow$ continuity.

Hessian

- For F twice differentiable, the Hessian is

$$H(\boldsymbol{\theta}) = \nabla^2 F(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial^2 F}{\partial \theta_1^2} & \frac{\partial^2 F}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 F}{\partial \theta_1 \partial \theta_d} \\ \frac{\partial^2 F}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 F}{\partial \theta_2^2} & \cdots & \frac{\partial^2 F}{\partial \theta_2 \partial \theta_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial \theta_d \partial \theta_1} & \frac{\partial^2 F}{\partial \theta_d \partial \theta_2} & \cdots & \frac{\partial^2 F}{\partial \theta_d^2} \end{bmatrix} \in \mathbb{R}^{d \times d}$$

Hessian

- For F twice differentiable, the Hessian is

$$H(\boldsymbol{\theta}) = \nabla^2 F(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial^2 F}{\partial \theta_1^2} & \frac{\partial^2 F}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 F}{\partial \theta_1 \partial \theta_d} \\ \frac{\partial^2 F}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 F}{\partial \theta_2^2} & \cdots & \frac{\partial^2 F}{\partial \theta_2 \partial \theta_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial \theta_d \partial \theta_1} & \frac{\partial^2 F}{\partial \theta_d \partial \theta_2} & \cdots & \frac{\partial^2 F}{\partial \theta_d^2} \end{bmatrix} \in \mathbb{R}^{d \times d}$$

- F convex $\Leftrightarrow$ $H(\boldsymbol{\theta}) \succeq 0$ (positive semi-definite — psd)

Hessian

- For F twice differentiable, the Hessian is

$$H(\boldsymbol{\theta}) = \nabla^2 F(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial^2 F}{\partial \theta_1^2} & \frac{\partial^2 F}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 F}{\partial \theta_1 \partial \theta_d} \\ \frac{\partial^2 F}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 F}{\partial \theta_2^2} & \cdots & \frac{\partial^2 F}{\partial \theta_2 \partial \theta_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial \theta_d \partial \theta_1} & \frac{\partial^2 F}{\partial \theta_d \partial \theta_2} & \cdots & \frac{\partial^2 F}{\partial \theta_d^2} \end{bmatrix} \in \mathbb{R}^{d \times d}$$

- F convex $\Leftrightarrow$ $H(\boldsymbol{\theta}) \succeq 0$ (positive semi-definite — psd)

- F strictly convex $\Leftrightarrow$ $H(\boldsymbol{\theta}) \succ 0$ (positive definite — pd)

Coercivity

- F is a coercive function if: $\lim_{\|\boldsymbol{\theta}\| \to +\infty} F(\boldsymbol{\theta}) = +\infty$

Coercivity

- F is a coercive function if: $\displaystyle\lim_{\|\boldsymbol{\theta}\| \to +\infty} F(\boldsymbol{\theta}) = +\infty$
- Let $G = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$, the set of global minimizers.

Coercivity

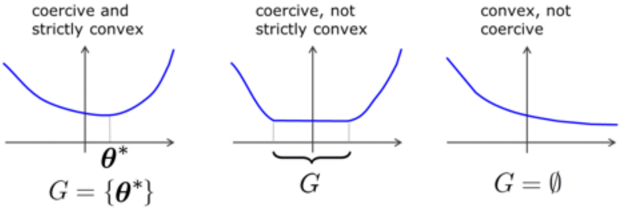
- F is a coercive function if: $\displaystyle\lim_{\|\boldsymbol{\theta}\|\rightarrow+\infty} F(\boldsymbol{\theta}) = +\infty$
- Let $G = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$, the set of global minimizers.
- F is coercive $\overset{\not\Leftarrow}{\Rightarrow}$ $G \neq \emptyset$ (example?)

Coercivity

- F is a coercive function if: $\displaystyle\lim_{\|\boldsymbol{\theta}\|\rightarrow+\infty} F(\boldsymbol{\theta}) = +\infty$
- Let $G = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$, the set of global minimizers.
- F is coercive $\overset{\not\Leftarrow}{\Rightarrow}$ $G \neq \emptyset$ (example?)
- F is strictly convex $\overset{\not\Leftarrow}{\Rightarrow}$ G has at most one element (example?)

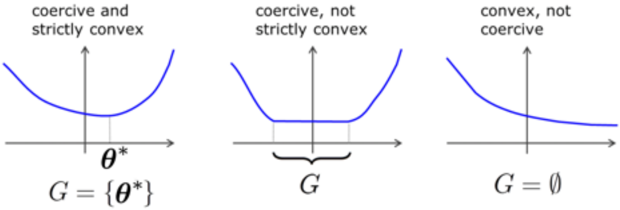
Coercivity

- F is a coercive function if: $\displaystyle\lim_{\|\boldsymbol{\theta}\|\to+\infty} F(\boldsymbol{\theta}) = +\infty$
- Let $G = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$, the set of global minimizers.
- F is coercive $\overset{\not\Leftarrow}{\Rightarrow}$ $G \neq \emptyset$ (example?)
- F is strictly convex $\overset{\not\Leftarrow}{\Rightarrow}$ G has at most one element (example?)



Coercivity

- F is a coercive function if: $\displaystyle\lim_{\|\boldsymbol{\theta}\| \to +\infty} F(\boldsymbol{\theta}) = +\infty$
- Let $G = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$, the set of global minimizers.
- F is coercive $\overset{\not\Leftarrow}{\Rightarrow}$ $G \neq \emptyset$ (example?)
- F is strictly convex $\overset{\not\Leftarrow}{\Rightarrow}$ G has at most one element (example?)



- Non-coercivity examples: logistic regression on separable data; linear regression for $n < p$.

Descent Directions

- Definition: $\boldsymbol{\eta}$ is a descent direction at $\boldsymbol{\theta}_0$ if

$$F(\boldsymbol{\theta}_0 + \alpha\,\boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \quad \text{for some} \quad \alpha > 0.$$

Descent Directions

- Definition: $\boldsymbol{\eta}$ is a descent direction at $\boldsymbol{\theta}_0$ if

$$F(\boldsymbol{\theta}_0 + \alpha\,\boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \ \ \text{for some} \ \ \alpha > 0.$$

- For differentiable F,

$$\boldsymbol{\eta}^T \nabla F(\boldsymbol{\theta}_0) < 0 \ \ \Leftrightarrow \ \ \boldsymbol{\eta} \text{ is a descent direction.}$$

Descent Directions

- Definition: $\boldsymbol{\eta}$ is a descent direction at $\boldsymbol{\theta}_0$ if

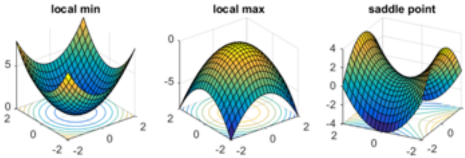
$$F(\boldsymbol{\theta}_0 + \alpha\, \boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \quad \text{for some} \quad \alpha > 0.$$

- For differentiable F,

$$\boldsymbol{\eta}^T \nabla F(\boldsymbol{\theta}_0) < 0 \quad \Leftrightarrow \quad \boldsymbol{\eta} \text{ is a descent direction.}$$

- Thus, for differentiable F,

$$\boldsymbol{\theta}^* \text{ is a local minimizer} \quad \substack{\not\Leftarrow \\ \Rightarrow} \quad \nabla F(\boldsymbol{\theta}^*) = 0$$



Descent Directions

- Definition: $\boldsymbol{\eta}$ is a descent direction at $\boldsymbol{\theta}_0$ if

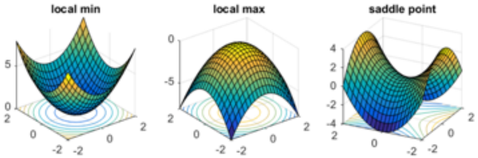
$$F(\boldsymbol{\theta}_0 + \alpha\,\boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \text{ for some } \alpha > 0.$$

- For differentiable F,

$$\boldsymbol{\eta}^T \nabla F(\boldsymbol{\theta}_0) < 0 \;\Leftrightarrow\; \boldsymbol{\eta} \text{ is a descent direction.}$$

- Thus, for differentiable F,

$$\boldsymbol{\theta}^* \text{ is a local minimizer } \underset{\Rightarrow}{\not\Leftarrow} \nabla F(\boldsymbol{\theta}^*) = 0$$



- If F is convex, $\boldsymbol{\theta}^*$ is a global minimizer $\Leftrightarrow \nabla F(\boldsymbol{\theta}^*) = 0$

Gradient Descent

- Key idea: if not at a minimizer, take a step in a descent direction.

Gradient Descent

- Key idea: if not at a minimizer, take a step in a descent direction.
- Gradient descent algorithm:
  - ✓ Start at some initial point $\boldsymbol{\theta}_0 \in \mathbb{R}^d$
  - ✓ For $t = 1, 2, ...,$
    - ▷ choose step-size α_t,
    - ▷ take a step of size α_t in the direction of the negative gradient:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \nabla F(\boldsymbol{\theta}_{t-1})$$

Gradient Descent

- Key idea: if not at a minimizer, take a step in a descent direction.
- Gradient descent algorithm:
  - ✓ Start at some initial point $\boldsymbol{\theta}_0 \in \mathbb{R}^d$
  - ✓ For $t = 1, 2, ...,$
    - ▷ choose step-size α_t,
    - ▷ take a step of size α_t in the direction of the negative gradient:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \nabla F(\boldsymbol{\theta}_{t-1})$$

- Several (many) ways to choose α_t; big research topic.

Gradient Descent

- Key idea: if not at a minimizer, take a step in a descent direction.
- Gradient descent algorithm:
  - ✓ Start at some initial point $\boldsymbol{\theta}_0 \in \mathbb{R}^d$
  - ✓ For $t = 1, 2, \ldots,$
    - ▷ choose step-size α_t,
    - ▷ take a step of size α_t in the direction of the negative gradient:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \nabla F(\boldsymbol{\theta}_{t-1})$$

- Several (many) ways to choose α_t; big research topic.
- Some stopping criterion is used; *e.g.*, $\|\nabla F(\boldsymbol{\theta}_t)\| \leq \delta$

Convex Case

Convex Case

- L-smoothness,

$$\|\nabla F(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$$

Convex Case

- L-smoothness,

$$\|\nabla F(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$$

- If F is twice differentiable, L-smoothness $\Leftrightarrow$ $H(\boldsymbol{\theta}) \preceq L\mathbf{I}$.

Convex Case

- L-smoothness,

$$\|\nabla F(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$$

- If F is twice differentiable, L-smoothness $\Leftrightarrow$ $H(\boldsymbol{\theta}) \preceq L\mathbf{I}$.

- μ-strong convexity,

$$F(\boldsymbol{\theta}) \geq F(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \nabla F(\boldsymbol{\theta}') + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$$

Convex Case

- L-smoothness,

$$\|\nabla F(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$$

- If F is twice differentiable, L-smoothness $\Leftrightarrow$ $H(\boldsymbol{\theta}) \preceq L\mathbf{I}$.

- μ-strong convexity,

$$F(\boldsymbol{\theta}) \geq F(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \nabla F(\boldsymbol{\theta}') + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$$

- If F is twice differentiable, μ-strong convexity $\Leftrightarrow$ $H(\boldsymbol{\theta}) \succeq \mu\mathbf{I}$.

Convex Case

- L-smoothness,

$$\|\nabla F(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$$

- If F is twice differentiable, L-smoothness $\Leftrightarrow$ $H(\boldsymbol{\theta}) \preceq L\mathbf{I}$.

- μ-strong convexity,

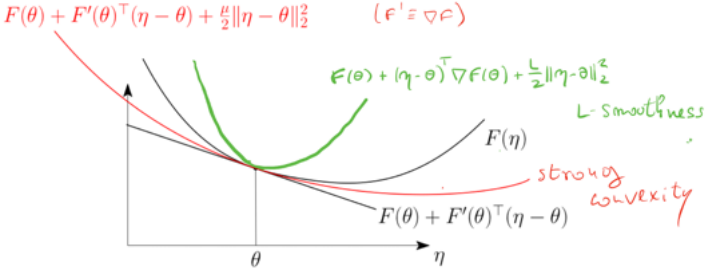
$$F(\boldsymbol{\theta}) \geq F(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \nabla F(\boldsymbol{\theta}') + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$$

- If F is twice differentiable, μ-strong convexity $\Leftrightarrow$ $H(\boldsymbol{\theta}) \succeq \mu\mathbf{I}$.

- Condition number $\kappa = \dfrac{L}{\mu}$.

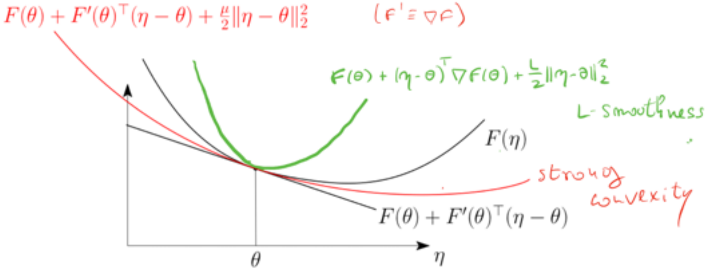
L-smoothness and $\mu-$Strongly Convex

- L-smooth and $\mu-$strongly convex function: upper and lower bounded by quadratics.



L-smoothness and $\mu-$Strongly Convex

- L-smooth and $\mu-$strongly convex function: upper and lower bounded by quadratics.



- Regularization: if $F(\boldsymbol{\theta})$ is convex, $F(\boldsymbol{\theta}) + \frac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$ is μ-strongly convex.

Gradient Descent for Convex Functions

- Gradient descent with step-size $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa}\right)^t \big(F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*)\big)$$

called linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \leq \gamma < 1$, with $\Delta_t = F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*)$).

Gradient Descent for Convex Functions

- Gradient descent with step-size $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa}\right)^t \big(F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*)\big)$$

  called linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \leq \gamma < 1$, with $\Delta_t = F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*)$).

- If $\mu = 0$ (not strongly convex),

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \frac{L}{2\,t}\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2$$

  called sub-linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \to 1$)

Gradient Descent for Convex Functions

- Gradient descent with step-size $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa}\right)^t \big(F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*)\big)$$

called linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \leq \gamma < 1$, with $\Delta_t = F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*)$).

- If $\mu = 0$ (not strongly convex),

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \frac{L}{2\,t} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2$$

called sub-linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \rightarrow 1$)

- In practice, these are very different (next slide).

Gradient Descent for Convex Functions

- Gradient descent with step-size $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa}\right)^t \big(F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*)\big)$$

  called linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \leq \gamma < 1$, with $\Delta_t = F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*)$).

- If $\mu = 0$ (not strongly convex),

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \frac{L}{2\,t} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2$$

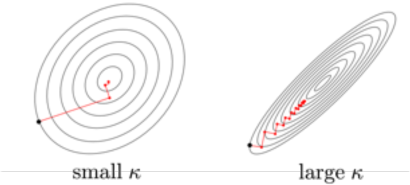
  called sub-linear convergence ($\frac{\Delta_t}{\Delta_{t-1}} \rightarrow 1$)

- In practice, these are very different (next slide).

- Proofs: see recommended reading (F. Bach).

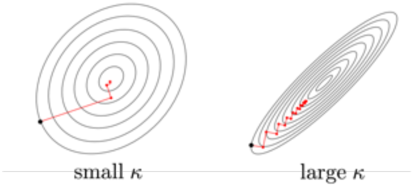
Gradient Descent: Strongly Convex Case

- The condition number κ expresses the problem difficulty.

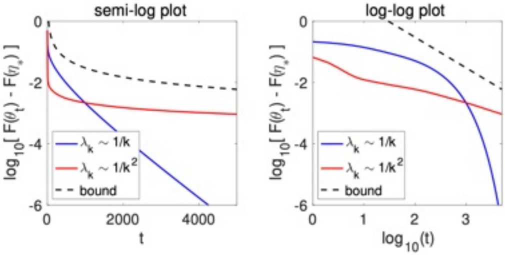


Gradient Descent: Strongly Convex Case

- The condition number κ expresses the problem difficulty.

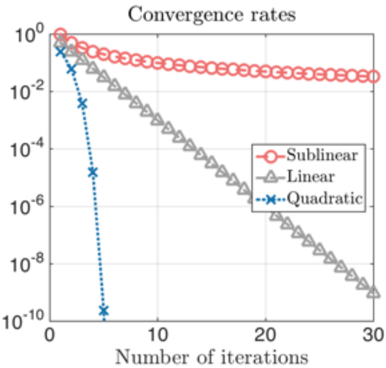


- Convergence for different distributions of eigenvalues.



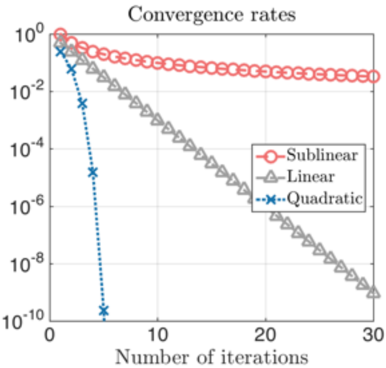
(pictures from F. Bach).

Linear vs Sublinear Convergence



- Quadratic ($\frac{\Delta_t}{\Delta_{t-1}^2} \to \beta < \infty$) and super-linear ($\frac{\Delta_t}{\Delta_{t-1}} \to 0$) convergence: not achievable using only gradient information.

Linear vs Sublinear Convergence



- Quadratic ($\frac{\Delta_t}{\Delta_{t-1}^2} \rightarrow \beta < \infty$) and super-linear ($\frac{\Delta_t}{\Delta_{t-1}} \rightarrow 0$) convergence: not achievable using only gradient information.
- Optimization is a central tool in machine learning; it is a huge field.

Stochastic Gradient "Descent"

- Back to empirical risk minimization: $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) \quad \big(\text{maybe } + R(\boldsymbol{\theta})\big)$$

Stochastic Gradient "Descent"

- Back to empirical risk minimization: $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) \quad \big(\text{maybe } + R(\boldsymbol{\theta})\big)$$

- For large n, computing $\nabla F(\boldsymbol{\theta})$ is expensive:

$$\nabla F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

Stochastic Gradient "Descent"

- Back to empirical risk minimization: $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$

$$F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) \quad \big(\text{maybe } + R(\boldsymbol{\theta})\big)$$

- For large n, computing $\nabla F(\boldsymbol{\theta})$ is expensive:

$$\nabla F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \nabla L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$$

- Alternative: stochastic gradient "descent" (SGD):
  - ✓ Start at some initial point $\boldsymbol{\theta}_0 \in \mathbb{R}^d$
  - ✓ For $t = 1, 2, ...,$
    - ▷ sample $i \in \{1, ..., n\}$ at random and choose step-size α_t,
    - ▷ take a step of size α_t in the direction of the negative gradient:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \nabla L(f(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}), y_i)$$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{X},Y}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$.

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{X},Y}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$.
- To do gradient descent, we need

$$\nabla\mathcal{R}(\boldsymbol{\theta}) = \nabla\mathbb{E}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)] = \mathbb{E}[\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{X},Y}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$.
- To do gradient descent, we need

$$\nabla\mathcal{R}(\boldsymbol{\theta}) = \nabla\mathbb{E}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)] = \mathbb{E}[\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$$

- Thus, $\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)$ is an unbiased estimate of $\nabla\mathcal{R}(\boldsymbol{\theta})$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{X},Y}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$.
- To do gradient descent, we need

$$\nabla\mathcal{R}(\boldsymbol{\theta}) = \nabla\mathbb{E}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)] = \mathbb{E}[\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$$

- Thus, $\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)$ is an unbiased estimate of $\nabla\mathcal{R}(\boldsymbol{\theta})$
- SGD with samples from $f_{\boldsymbol{X},Y}$ is a sequence of random variables,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla L(f(\boldsymbol{X};\boldsymbol{\theta}_t),Y)$$

that is, in expectation,

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}_{t+1}] &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \mathbb{E}[\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}_t),Y)]\\ &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \nabla\mathcal{R}(\boldsymbol{\theta}_t)\end{aligned}$$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{X},Y}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$.
- To do gradient descent, we need

$$\nabla\mathcal{R}(\boldsymbol{\theta}) = \nabla\mathbb{E}[L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)] = \mathbb{E}[\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)]$$

- Thus, $\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}),Y)$ is an unbiased estimate of $\nabla\mathcal{R}(\boldsymbol{\theta})$
- SGD with samples from $f_{\boldsymbol{X},Y}$ is a sequence of random variables,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla L(f(\boldsymbol{X};\boldsymbol{\theta}_t),Y)$$

that is, in expectation,

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}_{t+1}] &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \mathbb{E}[\nabla L(f(\boldsymbol{X};\boldsymbol{\theta}_t),Y)] \\ &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \nabla\mathcal{R}(\boldsymbol{\theta}_t)\end{aligned}$$

- In expectation, SGD by sampling $f_{\boldsymbol{X},Y}$ is gradient descent on $\mathcal{R}(\boldsymbol{\theta})$.

Convergence of Stochastic Gradient Descent

- SGD uses noisy gradients: $\boldsymbol{G}(\boldsymbol{\theta})$, such that $\mathbb{E}[\boldsymbol{G}(\boldsymbol{\theta})] = \nabla F(\boldsymbol{\theta})$
- True for $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta})$ and for $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$.

Convergence of Stochastic Gradient Descent

- SGD uses noisy gradients: $\boldsymbol{G}(\boldsymbol{\theta})$, such that $\mathbb{E}[\boldsymbol{G}(\boldsymbol{\theta})] = \nabla F(\boldsymbol{\theta})$
- True for $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta})$ and for $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$.
- Assumptions: F is convex; $\|\boldsymbol{G}(\boldsymbol{\theta})\|_2^2 \leq B^2$; $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \leq D$.

Convergence of Stochastic Gradient Descent

- SGD uses noisy gradients: $\boldsymbol{G}(\boldsymbol{\theta})$, such that $\mathbb{E}[\boldsymbol{G}(\boldsymbol{\theta})] = \nabla F(\boldsymbol{\theta})$
- True for $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta})$ and for $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$.
- Assumptions: F is convex; $\|\boldsymbol{G}(\boldsymbol{\theta})\|_2^2 \leq B^2$; $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \leq D$.
- Step size: $\alpha_t = \frac{D}{B\sqrt{t}}$.

Convergence of Stochastic Gradient Descent

- SGD uses noisy gradients: $\boldsymbol{G}(\boldsymbol{\theta})$, such that $\mathbb{E}[\boldsymbol{G}(\boldsymbol{\theta})] = \nabla F(\boldsymbol{\theta})$
- True for $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta})$ and for $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$.
- Assumptions: F is convex; $\|\boldsymbol{G}(\boldsymbol{\theta})\|_2^2 \leq B^2$; $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \leq D$.
- Step size: $\alpha_t = \frac{D}{B\sqrt{t}}$.
- Average iterates: $\bar{\boldsymbol{\theta}}_t = \dfrac{\sum_{s=1}^{t} \alpha_s \boldsymbol{\theta}_{s-1}}{\sum_{s=1}^{t} \alpha_s}$

Convergence of Stochastic Gradient Descent

- SGD uses noisy gradients: $\boldsymbol{G}(\boldsymbol{\theta})$, such that $\mathbb{E}[\boldsymbol{G}(\boldsymbol{\theta})] = \nabla F(\boldsymbol{\theta})$
- True for $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta})$ and for $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$.
- Assumptions: F is convex; $\|\boldsymbol{G}(\boldsymbol{\theta})\|_2^2 \leq B^2$; $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \leq D$.
- Step size: $\alpha_t = \frac{D}{B\sqrt{t}}$.
- Average iterates: $\bar{\boldsymbol{\theta}}_t = \dfrac{\sum_{s=1}^{t} \alpha_s \boldsymbol{\theta}_{s-1}}{\sum_{s=1}^{t} \alpha_s}$
- Then,

$$\mathbb{E}\left[F(\bar{\boldsymbol{\theta}}_t) - F(\boldsymbol{\theta}^*)\right] \leq \frac{D\,B\,(2+\log t)}{2\,\sqrt{t}}$$

Convergence of Stochastic Gradient Descent

- SGD uses noisy gradients: $\boldsymbol{G}(\boldsymbol{\theta})$, such that $\mathbb{E}[\boldsymbol{G}(\boldsymbol{\theta})] = \nabla F(\boldsymbol{\theta})$
- True for $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta})$ and for $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)$.
- Assumptions: F is convex; $\|\boldsymbol{G}(\boldsymbol{\theta})\|_2^2 \leq B^2$; $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2 \leq D$.
- Step size: $\alpha_t = \frac{D}{B\sqrt{t}}$.
- Average iterates: $\bar{\boldsymbol{\theta}}_t = \dfrac{\sum_{s=1}^{t} \alpha_s \boldsymbol{\theta}_{s-1}}{\sum_{s=1}^{t} \alpha_s}$
- Then,

$$\mathbb{E}\left[F(\bar{\boldsymbol{\theta}}_t) - F(\boldsymbol{\theta}^*)\right] \leq \frac{D\, B\, (2 + \log t)}{2\sqrt{t}}$$

- Important: not practical to compute $F(\boldsymbol{\theta}_t)$. Selecting the best iterate is thus impractical and would beat the purpose of SGD.

Convergence of SGD: Strongly Convex Case

- Regularization: $F(\boldsymbol{\theta}) = \dfrac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \dfrac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$

Convergence of SGD: Strongly Convex Case

- Regularization: $F(\boldsymbol{\theta}) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \dfrac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$
- Consequence: F is μ-strongly convex;

Convergence of SGD: Strongly Convex Case

- Regularization: $F(\boldsymbol{\theta}) = \dfrac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \dfrac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$
- Consequence: F is μ-strongly convex;
- Step size: $\alpha_t = \dfrac{1}{\mu\, t}$

Convergence of SGD: Strongly Convex Case

- Regularization: $F(\boldsymbol{\theta}) = \dfrac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \dfrac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$

- Consequence: F is μ-strongly convex;

- Step size: $\alpha_t = \dfrac{1}{\mu\, t}$

- Average iterates: $\bar{\boldsymbol{\theta}}_t = \dfrac{1}{t}\sum_{s=1}^{t} \boldsymbol{\theta}_{s-1}$

Convergence of SGD: Strongly Convex Case

- Regularization: $F(\boldsymbol{\theta}) = \dfrac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \dfrac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$

- Consequence: F is μ-strongly convex;

- Step size: $\alpha_t = \dfrac{1}{\mu\, t}$

- Average iterates: $\bar{\boldsymbol{\theta}}_t = \dfrac{1}{t}\sum_{s=1}^{t} \boldsymbol{\theta}_{s-1}$

- Then,

$$\mathbb{E}\left[F(\bar{\boldsymbol{\theta}}_t) - F(\boldsymbol{\theta}^*)\right] \leq \frac{2\,B^2\,(1+\log t)}{\mu\, t}$$

Convergence of SGD: Strongly Convex Case

- Regularization: $F(\boldsymbol{\theta}) = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \dfrac{\mu}{2}\|\boldsymbol{\theta}\|_2^2$
- Consequence: F is μ-strongly convex;
- Step size: $\alpha_t = \dfrac{1}{\mu\, t}$
- Average iterates: $\bar{\boldsymbol{\theta}}_t = \dfrac{1}{t}\displaystyle\sum_{s=1}^{t} \boldsymbol{\theta}_{s-1}$
- Then,

$$\mathbb{E}\left[F(\bar{\boldsymbol{\theta}}_t) - F(\boldsymbol{\theta}^*)\right] \leq \frac{2\,B^2\,(1+\log t)}{\mu\, t}$$

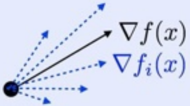
- Strong convexity speeds up convergence from $O(1/\sqrt{t})$ to $O(1/t)$

Visual Summary

Finite sums

$$f(x) \overset{\text{def.}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

$$\nabla f(x) = \frac{1}{n}\sum_i \nabla f_i(x)$$



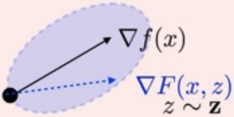
Draw $i \in \{1, \ldots, n\}$ uniformly.

$$x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$$

Expectation

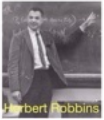
$$f(x) \overset{\text{def.}}{=} \mathbb{E}_{\mathbf{z}}(f(x, \mathbf{z}))$$

$$\nabla f(x) = \mathbb{E}_{\mathbf{z}}(\nabla F(x, \mathbf{z}))$$



Draw $z \sim \mathbf{z}$

$$x_{k+1} = x_k - \tau_k \nabla F(x, z)$$



Theorem: If f is strongly convex and $\tau_k \sim 1/k$,

$$\mathbb{E}(\|x_k - x^\star\|^2) = O(1/k)$$

(Picture by Gabriel Peyré)

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i\boldsymbol{\theta}^T\boldsymbol{x}_i)$

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i)$
- Several choices (all convex):
  - ✓ hinge loss (SVM): $\ell(u) = \max\{0, 1-u\}$
  - ✓ logistic loss: $\ell(u) = \log(1 + \exp(-u))$
  - ✓ squared loss: $\ell(u) = (1-u)^2$

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i)$

- Several choices (all convex):

  ✓ hinge loss (SVM): $\ell(u) = \max\{0, 1-u\}$

  ✓ logistic loss: $\ell(u) = \log(1+\exp(-u))$

  ✓ squared loss: $\ell(u) = (1-u)^2$

- From the gradient of the composite function,

$$\nabla \ell(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i) = \left.\frac{d\,\ell(u)}{d\,u}\right|_{u = y_i \boldsymbol{\theta}^T \boldsymbol{x}_i} \nabla (y_i \boldsymbol{\theta}^T \boldsymbol{x}_i)$$

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\boldsymbol{x}_i; \boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i)$
- Several choices (all convex):
  - ✓ hinge loss (SVM): $\ell(u) = \max\{0, 1-u\}$
  - ✓ logistic loss: $\ell(u) = \log(1 + \exp(-u))$
  - ✓ squared loss: $\ell(u) = (1-u)^2$
- From the gradient of the composite function,

$$\nabla \ell(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i) = \left. \frac{d\,\ell(u)}{d\,u} \right|_{u = y_i \boldsymbol{\theta}^T \boldsymbol{x}_i} \nabla (y_i \boldsymbol{\theta}^T \boldsymbol{x}_i) = \left(\left. \frac{d\,\ell(u)}{d\,u} \right|_{u = y_i \boldsymbol{\theta}^T \boldsymbol{x}_i} y_i \right) \boldsymbol{x}_i$$

showing that $\nabla \ell(y_i \boldsymbol{\theta}^T \boldsymbol{x}_i)$ is co-linear with $\boldsymbol{x}_i$.

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\boldsymbol{x}_i;\boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i\boldsymbol{\theta}^T\boldsymbol{x}_i)$

- Several choices (all convex):

  ✓ hinge loss (SVM): $\ell(u) = \max\{0, 1-u\}$

  ✓ logistic loss: $\ell(u) = \log(1+\exp(-u))$

  ✓ squared loss: $\ell(u) = (1-u)^2$

- From the gradient of the composite function,

$$\nabla\ell(y_i\boldsymbol{\theta}^T\boldsymbol{x}_i) = \left.\frac{d\,\ell(u)}{d\,u}\right|_{u=y_i\boldsymbol{\theta}^T\boldsymbol{x}_i}\nabla(y_i\boldsymbol{\theta}^T\boldsymbol{x}_i) = \left(\left.\frac{d\,\ell(u)}{d\,u}\right|_{u=y_i\boldsymbol{\theta}^T\boldsymbol{x}_i} y_i\right)\boldsymbol{x}_i$$

  showing that $\nabla\ell(y_i\boldsymbol{\theta}^T\boldsymbol{x}_i)$ is co-linear with $\boldsymbol{x}_i$.

- Each SGD update moves $\boldsymbol{\theta}_t$ in a direction parallel to sample $\boldsymbol{x}_i$.

The Perceptron Algorithm

- Hinge loss: $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\,\ell(u)}{d\,u} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

ignoring the non-differentiability at $u = \tau$.

The Perceptron Algorithm

- Hinge loss: $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\,\ell(u)}{d\,u} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

  ignoring the non-differentiability at $u = \tau$.

- Each iteration of SGD, with constant step size α, choose sample i,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \begin{cases} y_i \boldsymbol{x}_i & \text{if } y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

The Perceptron Algorithm

- Hinge loss: $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\,\ell(u)}{d\,u} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

ignoring the non-differentiability at $u = \tau$.

- Each iteration of SGD, with constant step size α, choose sample i,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \begin{cases} y_i \boldsymbol{x}_i & \text{if } y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

- Points with wrong classification ($y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i < 0$) or insufficient margin ($y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i \leq \tau$) move $\boldsymbol{\theta}_t$ towards/away from $\boldsymbol{x}_i$ depending on y_i

The Perceptron Algorithm

- Hinge loss: $\ell(u) = \max\{0, 1-\tau\}$, thus

$$\frac{d\,\ell(u)}{d\,u} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

ignoring the non-differentiability at $u = \tau$.

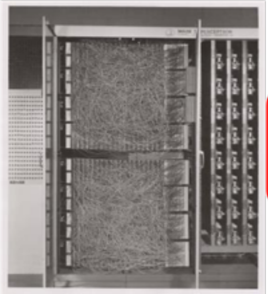
- Each iteration of SGD, with constant step size α, choose sample i,

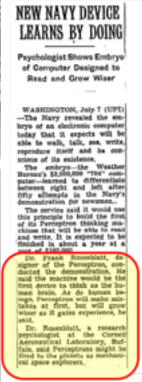
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \begin{cases} y_i \boldsymbol{x}_i & \text{if } y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

- Points with wrong classification $(y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i < 0)$ or insufficient margin $(y_i \boldsymbol{\theta}_t^T \boldsymbol{x}_i \leq \tau)$ move $\boldsymbol{\theta}_t$ towards/away from $\boldsymbol{x}_i$ depending on y_i

- This is the famous Perceptron algorithm, proposed in 1957 by Frank Rosenblatt (with $\tau = 0$), the percursor of modern neural networks.

A Bit of History: The Perceptron





NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

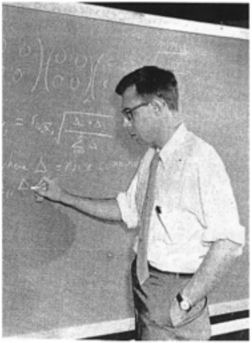
WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.



The New York Times, 1958

Minsky and Pappert, 1969

Perceptron Mistake Bound

- Definitions:
  - ✓ The training data is linearly separable with margin $\gamma > 0$ iff there is a weight vector $\boldsymbol{u}$, with $\|\boldsymbol{u}\| = 1$, such that

$$y_n \, \boldsymbol{u}^T \boldsymbol{x}_n \geq \gamma, \quad \forall n.$$

[2] A. Novikoff, "On convergence proofs for perceptrons", *Symposium on the Mathematical Theory of Automata*, 1962.

Perceptron Mistake Bound

- Definitions:
  - ✓ The training data is linearly separable with margin $\gamma > 0$ iff there is a weight vector $\boldsymbol{u}$, with $\|\boldsymbol{u}\| = 1$, such that
  $$y_n \, \boldsymbol{u}^T \boldsymbol{x}_n \geq \gamma, \quad \forall n.$$
  - ✓ Radius of the data: $R = \max_n \|\boldsymbol{x}_n\|$.

[2] A. Novikoff, "On convergence proofs for perceptrons", *Symposium on the Mathematical Theory of Automata*, 1962.

Perceptron Mistake Bound

- Definitions:
  - ✓ The training data is linearly separable with margin $\gamma > 0$ iff there is a weight vector $\boldsymbol{u}$, with $\|\boldsymbol{u}\| = 1$, such that
  $$y_n \, \boldsymbol{u}^T \boldsymbol{x}_n \geq \gamma, \quad \forall n.$$
  - ✓ Radius of the data: $R = \max_n \|\boldsymbol{x}_n\|$.
- Then, the following bound of the number of mistakes holds[2]

Theorem

The perceptron algorithm is guaranteed to find a separating hyperplane after at most $\frac{R^2}{\gamma^2}$ mistakes (non-zero updates).

[2] A. Novikoff, "On convergence proofs for perceptrons", *Symposium on the Mathematical Theory of Automata*, 1962.

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + y_i\, \boldsymbol{x}_i$.

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + y_i\, \boldsymbol{x}_i$.
- Lower bound on $\|\boldsymbol{\theta}_t\|$, after M mistakes:

$$\begin{aligned} \boldsymbol{u}^T \boldsymbol{\theta}_t &= \boldsymbol{u}^T \boldsymbol{\theta}_{t-1} + y_i\, \boldsymbol{u}^T \boldsymbol{x}_i \\ &\geq \boldsymbol{u}^T \boldsymbol{\theta}_{t-1} + \gamma \\ &\geq \boldsymbol{u}^T \boldsymbol{\theta}_0 + M\,\gamma \;=\; M\,\gamma \qquad (\text{recall } \boldsymbol{\theta}_0 = 0) \end{aligned}$$

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + y_i\, \boldsymbol{x}_i$.
- Lower bound on $\|\boldsymbol{\theta}_t\|$, after M mistakes:

$$\begin{aligned} \boldsymbol{u}^T \boldsymbol{\theta}_t &= \boldsymbol{u}^T \boldsymbol{\theta}_{t-1} + y_i\, \boldsymbol{u}^T \boldsymbol{x}_i \\ &\geq \boldsymbol{u}^T \boldsymbol{\theta}_{t-1} + \gamma \\ &\geq \boldsymbol{u}^T \boldsymbol{\theta}_0 + M\,\gamma \;=\; M\,\gamma \qquad \text{(recall } \boldsymbol{\theta}_0 = 0\text{)} \end{aligned}$$

Thus, $\|\boldsymbol{\theta}_t\| = \underbrace{\|\boldsymbol{u}\|}_{1}\, \|\boldsymbol{\theta}_t\| \geq \boldsymbol{u}^T \boldsymbol{\theta}_t \geq M\,\gamma$ (Cauchy-Schwarz)

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + y_i\, \boldsymbol{x}_i$.
- Lower bound on $\|\boldsymbol{\theta}_t\|$, after M mistakes:

$$\begin{aligned} \boldsymbol{u}^T\boldsymbol{\theta}_t &= \boldsymbol{u}^T\boldsymbol{\theta}_{t-1} + y_i\,\boldsymbol{u}^T\boldsymbol{x}_i \\ &\geq \boldsymbol{u}^T\boldsymbol{\theta}_{t-1} + \gamma \\ &\geq \boldsymbol{u}^T\boldsymbol{\theta}_0 + M\,\gamma \;=\; M\,\gamma \qquad \text{(recall } \boldsymbol{\theta}_0 = 0\text{)} \end{aligned}$$

Thus, $\|\boldsymbol{\theta}_t\| = \underbrace{\|\boldsymbol{u}\|}_{1}\,\|\boldsymbol{\theta}_t\| \geq \boldsymbol{u}^T\boldsymbol{\theta}_t \geq M\,\gamma$ (Cauchy-Schwarz)

- Upper bound on $\|\boldsymbol{\theta}_t\|$:

$$\begin{aligned} \|\boldsymbol{\theta}_t\|^2 &= \|\boldsymbol{\theta}_{t-1}\|^2 + \|\boldsymbol{x}_i\|^2 + 2\,\overbrace{y_i\,\boldsymbol{\theta}_{t-1}^T\boldsymbol{x}_i}^{\leq 0,\ \text{if mistake}} \\ &\leq \|\boldsymbol{\theta}_{t-1}\|^2 + R^2 \\ &\leq M\,R^2 \end{aligned}$$

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + y_i\, \boldsymbol{x}_i$.
- Lower bound on $\|\boldsymbol{\theta}_t\|$, after M mistakes:

$$\begin{aligned} \boldsymbol{u}^T \boldsymbol{\theta}_t &= \boldsymbol{u}^T \boldsymbol{\theta}_{t-1} + y_i\, \boldsymbol{u}^T \boldsymbol{x}_i \\ &\geq \boldsymbol{u}^T \boldsymbol{\theta}_{t-1} + \gamma \\ &\geq \boldsymbol{u}^T \boldsymbol{\theta}_0 + M\,\gamma \;=\; M\,\gamma \qquad \text{(recall } \boldsymbol{\theta}_0 = 0) \end{aligned}$$

Thus, $\|\boldsymbol{\theta}_t\| = \underbrace{\|\boldsymbol{u}\|}_{1}\, \|\boldsymbol{\theta}_t\| \geq \boldsymbol{u}^T \boldsymbol{\theta}_t \geq M\,\gamma$ (Cauchy-Schwarz)

- Upper bound on $\|\boldsymbol{\theta}_t\|$:

$$\begin{aligned} \|\boldsymbol{\theta}_t\|^2 &= \|\boldsymbol{\theta}_{t-1}\|^2 + \|\boldsymbol{x}_i\|^2 + 2\; \overbrace{y_i\, \boldsymbol{\theta}_{t-1}^T \boldsymbol{x}_i}^{\leq 0,\ \text{if mistake}} \\ &\leq \|\boldsymbol{\theta}_{t-1}\|^2 + R^2 \\ &\leq M\,R^2 \end{aligned}$$

- Equating both sides, $(M\gamma)^2 \leq \|\boldsymbol{\theta}_t\|^2 \leq M\,R^2 \;\Rightarrow\; M \leq R^2/\gamma^2$ ■

Implicit Regularization

- SGD in linear prediction, with i_t denoting the sample at iteration t,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \, e_{i_t} \, \boldsymbol{x}_{i_t}$$

where e_{i_t} depends on the loss gradient and label y_{i_t}.

Implicit Regularization

- SGD in linear prediction, with i_t denoting the sample at iteration t,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \, e_{i_t} \, \boldsymbol{x}_{i_t}$$

  where e_{i_t} depends on the loss gradient and label y_{i_t}.

- Minibatch or full batch gradient descent:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \, \boldsymbol{x}_j$$

Implicit Regularization

- SGD in linear prediction, with i_t denoting the sample at iteration t,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \, e_{i_t} \, \boldsymbol{x}_{i_t}$$

  where e_{i_t} depends on the loss gradient and label y_{i_t}.

- Minibatch or full batch gradient descent:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \, \boldsymbol{x}_j$$

- Initializing at $\boldsymbol{\theta}_0 = 0 \quad \Rightarrow \quad \boldsymbol{\theta}_t \in \text{span}(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$.

Implicit Regularization

- SGD in linear prediction, with i_t denoting the sample at iteration t,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \, e_{i_t} \, \boldsymbol{x}_{i_t}$$

  where e_{i_t} depends on the loss gradient and label y_{i_t}.

- Minibatch or full batch gradient descent:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \, \boldsymbol{x}_j$$

- Initializing at $\boldsymbol{\theta}_0 = 0 \quad \Rightarrow \quad \boldsymbol{\theta}_t \in \text{span}(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$.

- If there are multiple $\boldsymbol{\theta}^*$ with $F(\boldsymbol{\theta}^*) = 0$, and the predictions only depend on $\boldsymbol{\theta}^T \boldsymbol{x}_i$, this corresponds to solving

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2, \quad \text{such that } \; L(\boldsymbol{\theta}^T \boldsymbol{x}_i, y_i) = 0, \;\; \text{for } i = 1, ..., n.$$

Implicit Regularization

- SGD in linear prediction, with i_t denoting the sample at iteration t,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \, e_{i_t} \, \boldsymbol{x}_{i_t}$$

where e_{i_t} depends on the loss gradient and label y_{i_t}.

- Minibatch or full batch gradient descent:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \, \boldsymbol{x}_j$$

- Initializing at $\boldsymbol{\theta}_0 = 0 \quad \Rightarrow \quad \boldsymbol{\theta}_t \in \text{span}(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$.
- If there are multiple $\boldsymbol{\theta}^*$ with $F(\boldsymbol{\theta}^*) = 0$, and the predictions only depend on $\boldsymbol{\theta}^T \boldsymbol{x}_i$, this corresponds to solving

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2, \quad \text{such that} \;\; L(\boldsymbol{\theta}^T \boldsymbol{x}_i, y_i) = 0, \;\; \text{for } i = 1, ..., n.$$

- This is sometimes called the overparametrized or interpolating regime and is a central tool in the understanding of modern deep learning.

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

- Let $\boldsymbol{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$
- Let $\boldsymbol{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk
- Gradient of the regularizer: $\lambda\,\boldsymbol{\theta}$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

- Let $\boldsymbol{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk

- Gradient of the regularizer: $\lambda\,\boldsymbol{\theta}$

- Gradient descent (batch or stochastic):

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t\left(\boldsymbol{g}(\boldsymbol{\theta}_{t-1}) + \lambda\boldsymbol{\theta}_{t-1}\right)$$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

- Let $\boldsymbol{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk

- Gradient of the regularizer: $\lambda\,\boldsymbol{\theta}$

- Gradient descent (batch or stochastic):

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha_t\left(\boldsymbol{g}(\boldsymbol{\theta}_{t-1}) + \lambda\boldsymbol{\theta}_{t-1}\right)\\ &= (1-\lambda\,\alpha_t)\boldsymbol{\theta}_{t-1} - \alpha_t\,\boldsymbol{g}(\boldsymbol{\theta}_{t-1})\end{aligned}$$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$

- Let $\boldsymbol{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk

- Gradient of the regularizer: $\lambda\,\boldsymbol{\theta}$

- Gradient descent (batch or stochastic):

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha_t\left(\boldsymbol{g}(\boldsymbol{\theta}_{t-1}) + \lambda\boldsymbol{\theta}_{t-1}\right)\\ &= (1-\lambda\,\alpha_t)\boldsymbol{\theta}_{t-1} - \alpha_t\,\boldsymbol{g}(\boldsymbol{\theta}_{t-1})\end{aligned}$$

- For α_t and λ small enough, $0 < (1-\lambda\,\alpha_t) < 1$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$
- Let $\boldsymbol{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk
- Gradient of the regularizer: $\lambda\,\boldsymbol{\theta}$
- Gradient descent (batch or stochastic):

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha_t\left(\boldsymbol{g}(\boldsymbol{\theta}_{t-1}) + \lambda\boldsymbol{\theta}_{t-1}\right)\\ &= (1-\lambda\,\alpha_t)\boldsymbol{\theta}_{t-1} - \alpha_t\,\boldsymbol{g}(\boldsymbol{\theta}_{t-1})\end{aligned}$$

- For α_t and λ small enough, $0 < (1-\lambda\,\alpha_t) < 1$
- $\boldsymbol{\theta}_{t-1}$ is shrunk/decayed before being updated: weight decay

Tricks of the Trade

- Choosing the step size is critical: active research area.

Tricks of the Trade

- Choosing the step size is critical: active research area.
- Decay the step size: either continuously, or after each epoch (a single pass through some set of samples, *e.g.*, the whole training set) .

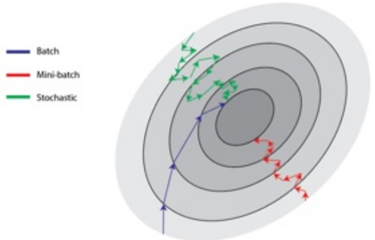
Tricks of the Trade

- Choosing the step size is critical: active research area.
- Decay the step size: either continuously, or after each epoch (a single pass through some set of samples, *e.g.*, the whole training set) .
- Shuffling the data after each epoch.

Tricks of the Trade

- Choosing the step size is critical: active research area.
- Decay the step size: either continuously, or after each epoch (a single pass through some set of samples, *e.g.*, the whole training set) .
- Shuffling the data after each epoch.
- Minibatching: instead of a single sample, use minibatches (size m)

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\alpha_t}{m} \sum_{j \in \text{minibatch } t} \nabla L(f(\boldsymbol{x}_j; \boldsymbol{\theta}_{t-1}), y_j)$$



Momentum

- Momentum: remember the previous step, combine it in the update:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \boldsymbol{g}(\boldsymbol{\theta}_{t-1}) + \gamma_t(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_{t-2});$$

$\boldsymbol{g}(\boldsymbol{\theta}_t)$ is the gradient estimate (batch, single sample, minibatch).

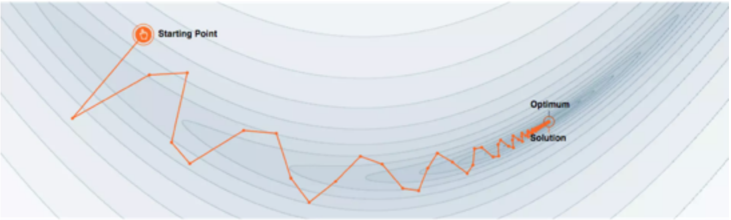
Momentum

- Momentum: remember the previous step, combine it in the update:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \boldsymbol{g}(\boldsymbol{\theta}_{t-1}) + \gamma_t(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_{t-2});$$

  $\boldsymbol{g}(\boldsymbol{\theta}_t)$ is the gradient estimate (batch, single sample, minibatch).

- Advantage: reduces the update in directions with changing gradients; increases the update in directions with stable gradient.



Adaptive Gradient (AdaGrad)

- AdaGrad[3]: use separate step sizes for each component of $\boldsymbol{\theta}_t$.

[3] J. Duchi, E. Hazan, Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Jour. of Machine Learning Research, vo. 12, 2011

Adaptive Gradient (AdaGrad)

- AdaGrad[3]: use separate step sizes for each component of $\boldsymbol{\theta}_t$.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \sum_{t'=1}^{t} \big(g_j(\boldsymbol{\theta}_{t'})\big)^2 = G_{j,t-1} + \big(g_j(\boldsymbol{\theta}_t)\big)^2$$

[3] J. Duchi, E. Hazan, Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Jour. of Machine Learning Research, vo. 12, 2011

Adaptive Gradient (AdaGrad)

- AdaGrad[3]: use separate step sizes for each component of $\boldsymbol{\theta}_t$.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \sum_{t'=1}^{t} \big(g_j(\boldsymbol{\theta}_{t'})\big)^2 = G_{j,t-1} + \big(g_j(\boldsymbol{\theta}_t)\big)^2$$

- Scale the update of each component (ε for numerical stability)

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}}\, g_j(\boldsymbol{\theta}_{t-1})$$

[3] J. Duchi, E. Hazan, Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Jour. of Machine Learning Research, vo. 12, 2011

Adaptive Gradient (AdaGrad)

- AdaGrad[3]: use separate step sizes for each component of $\boldsymbol{\theta}_t$.

- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \sum_{t'=1}^{t} \big(g_j(\boldsymbol{\theta}_{t'})\big)^2 = G_{j,t-1} + \big(g_j(\boldsymbol{\theta}_t)\big)^2$$

- Scale the update of each component (ε for numerical stability)

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}}\, g_j(\boldsymbol{\theta}_{t-1})$$

- Advantages: robust to choice of α; robust to different parameter scaling.

[3] J. Duchi, E. Hazan, Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Jour. of Machine Learning Research, vo. 12, 2011

Adaptive Gradient (AdaGrad)

- AdaGrad[3]: use separate step sizes for each component of $\boldsymbol{\theta}_t$.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \sum_{t'=1}^{t} \big(g_j(\boldsymbol{\theta}_{t'})\big)^2 = G_{j,t-1} + \big(g_j(\boldsymbol{\theta}_t)\big)^2$$

- Scale the update of each component (ε for numerical stability)

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}}\, g_j(\boldsymbol{\theta}_{t-1})$$

- Advantages: robust to choice of α; robust to different parameter scaling.
- Drawbacks: updated step size (learning rate) vanishes, since $G_{j,t} \geq G_{j,t-1}$.

[3] J. Duchi, E. Hazan, Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", Jour. of Machine Learning Research, vo. 12, 2011

Root Mean Square Propagation (RMSProp)

- RMSProp[4] addresses the vanishing learning issue.

[4] Presented by G. Hinton in a Coursera lecture.

Root Mean Square Propagation (RMSProp)

- RMSProp[4] addresses the vanishing learning issue.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \gamma\, G_{j,t-1} + (1-\gamma)\big(g_j(\boldsymbol{\theta}_t)\big)^2$$

[4]Presented by G. Hinton in a Coursera lecture.

Root Mean Square Propagation (RMSProp)

- RMSProp[4] addresses the vanishing learning issue.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \gamma \, G_{j,t-1} + (1-\gamma)\big(g_j(\boldsymbol{\theta}_t)\big)^2$$

- Forgetting factor γ (typically 0.9): $G_{j,t}$ may be smaller than $G_{j,t-1}$.

[4] Presented by G. Hinton in a Coursera lecture.

Root Mean Square Propagation (RMSProp)

- RMSProp[4] addresses the vanishing learning issue.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \gamma\, G_{j,t-1} + (1-\gamma)\big(g_j(\boldsymbol{\theta}_t)\big)^2$$

- Forgetting factor γ (typically 0.9): $G_{j,t}$ may be smaller than $G_{j,t-1}$.
- Scale the update of each component

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}}\, g_j(\boldsymbol{\theta}_{t-1})$$

[4]Presented by G. Hinton in a Coursera lecture.

Root Mean Square Propagation (RMSProp)

- RMSProp[4] addresses the vanishing learning issue.
- For component j of $\boldsymbol{\theta}_t$,

$$G_{j,t} = \gamma\, G_{j,t-1} + (1-\gamma)\big(g_j(\boldsymbol{\theta}_t)\big)^2$$

- Forgetting factor γ (typically 0.9): $G_{j,t}$ may be smaller than $G_{j,t-1}$.
- Scale the update of each component

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}}\, g_j(\boldsymbol{\theta}_{t-1})$$

- Advantages: robust to choice of α (typically 0.01 or 0.001); robust to different parameter scaling.

[4] Presented by G. Hinton in a Coursera lecture.

Adam Algorithm: Adaptive Moment Estimation

- Adam[5]: combines aspects of AdaGrad and RMSProp.

[5] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference for Learning Representations*, 2015. (more than 220K citations)

Adam Algorithm: Adaptive Moment Estimation

- Adam[5]: combines aspects of AdaGrad and RMSProp.
- Separate moving averages of gradient and squared gradient.

[5] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference for Learning Representations*, 2015. (more than 220K citations)

Adam Algorithm: Adaptive Moment Estimation

- Adam[5]: combines aspects of AdaGrad and RMSProp.
- Separate moving averages of gradient and squared gradient.
- Initial: $\boldsymbol{m}_t = 0$, $\boldsymbol{v}_t = 0$ (typical $\beta_1 = 0.9, \beta_2 = 0.999, \alpha = 10^{-3}$):

$$
\begin{aligned}
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\
\boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \\
\hat{\boldsymbol{m}}_t &= \boldsymbol{m}_t/(1-\beta_1^t) && \text{(bias correction due to } {}_0 = 0) \\
\hat{\boldsymbol{v}}_t &= \boldsymbol{v}_t/(1-\beta_2^t) && \text{(bias correction due to } \boldsymbol{v}_0 = 0) \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} && \text{(component-wise)}
\end{aligned}
$$

[5] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference for Learning Representations*, 2015. (more than 220K citations)

Adam Algorithm: Adaptive Moment Estimation

- Adam[5]: combines aspects of AdaGrad and RMSProp.
- Separate moving averages of gradient and squared gradient.
- Initial: $\boldsymbol{m}_t = 0$, $\boldsymbol{v}_t = 0$ (typical $\beta_1 = 0.9, \beta_2 = 0.999, \alpha = 10^{-3}$):

$$
\begin{aligned}
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\
\boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \\
\hat{\boldsymbol{m}}_t &= \boldsymbol{m}_t/(1-\beta_1^t) && \text{(bias correction due to } {}_0 = 0) \\
\hat{\boldsymbol{v}}_t &= \boldsymbol{v}_t/(1-\beta_2^t) && \text{(bias correction due to } \boldsymbol{v}_0 = 0) \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} && \text{(component-wise)}
\end{aligned}
$$

- Advantages: Computationally efficient, low memory usage, suitable for large datasets and many parameters.

[5] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference for Learning Representations*, 2015. (more than 220K citations)

Adam Algorithm: Adaptive Moment Estimation

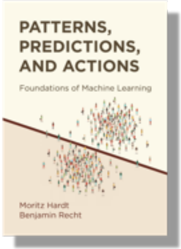
- Adam[5]: combines aspects of AdaGrad and RMSProp.
- Separate moving averages of gradient and squared gradient.
- Initial: $\boldsymbol{m}_t = 0$, $\boldsymbol{v}_t = 0$ (typical $\beta_1 = 0.9, \beta_2 = 0.999, \alpha = 10^{-3}$):

$$
\begin{aligned}
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t \\
\boldsymbol{v}_t &= \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2 \\
\hat{\boldsymbol{m}}_t &= \boldsymbol{m}_t / (1 - \beta_1^t) && \text{(bias correction due to } {}_0 = 0) \\
\hat{\boldsymbol{v}}_t &= \boldsymbol{v}_t / (1 - \beta_2^t) && \text{(bias correction due to } \boldsymbol{v}_0 = 0) \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} && \text{(component-wise)}
\end{aligned}
$$

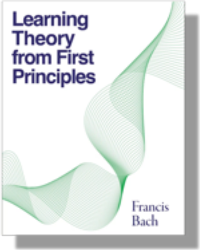
- Advantages: Computationally efficient, low memory usage, suitable for large datasets and many parameters.
- Drawbacks: Possible convergence issues and noisy gradient estimates.

[5] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference for Learning Representations*, 2015. (more than 220K citations)

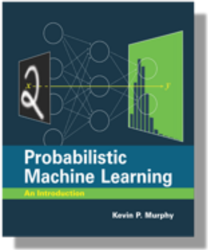
Recommended Books



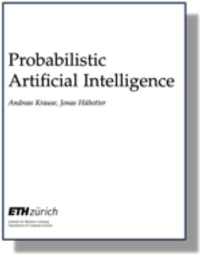
https://mlstory.org/
2022



www.di.ens.fr/~fbach/ltfp_book.pdf
2024

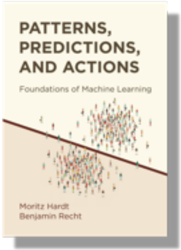


probml.github.io/pml-book
2022

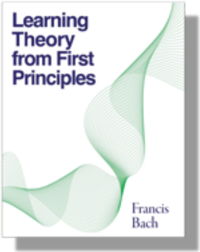


arxiv.org/abs/2502.05244
2025

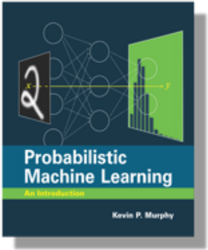
Recommended Books



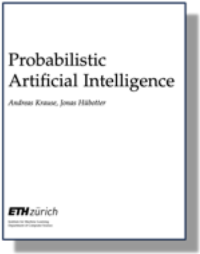
https://mlstory.org/
2022



www.di.ens.fr/~fbach/ltfp_book.pdf
2024



probml.github.io/pml-book
2022



arxiv.org/abs/2502.05244
2025

Thank you!   Questions?