

Learning with Linear Models: Foundations of Machine Learning

Mário A. T. Figueiredo



16th Lisbon Machine Learning Summer School, LxMLS 2026

Outline

- ➊ Introduction
- ➋ Regression
- ➌ Classification
- ➍ Optimization for Supervised Learning

CLASSICAL MACHINE LEARNING

Data is pre-categorized
or numerical

SUPERVISED

Predict
a category

CLASSIFICATION

«Divide the socks by color»



Predict
a number

REGRESSION

«Divide the ties by length»



Data is not labeled
in any way

UNSUPERVISED

Divide
by similarity

CLUSTERING

«Split up similar clothing
into stacks»



Identify sequences

Find hidden
dependencies

ASSOCIATION

«Find what clothes I often
wear together»



DIMENSION REDUCTION (generalization)

«Make the best outfits from the given clothes»

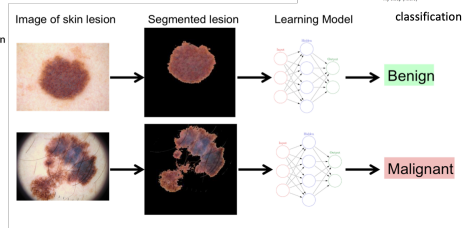
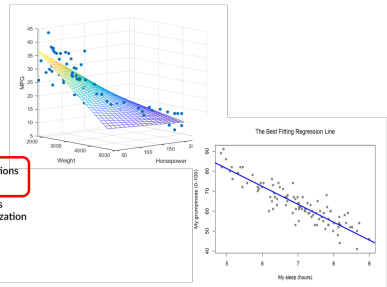
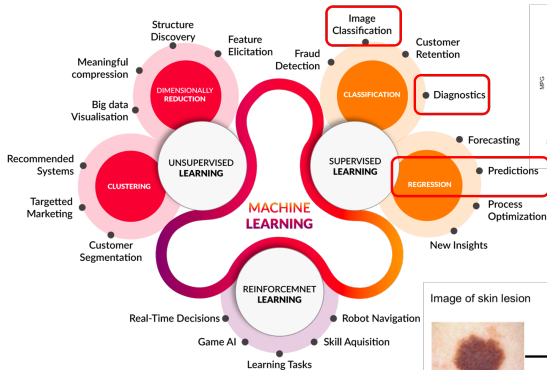


vas3k.com

Supervised Learning



Types of Machine Learning



Linear Models, Why?

- In 2026, **deep neural networks** (DNNs) are ubiquitous!

Linear Models, Why?

- In 2026, **deep neural networks** (DNNs) are ubiquitous!
- Why a lecture on **linear models**?
 - ✓ Underlying **machine learning** (ML) core concepts are the same.

Linear Models, Why?

- In 2026, **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 simpler** to understand.

Linear Models, Why?

- In 2026, **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 simpler** to understand.
 - ✓ Still **widely used**, specially if data is scarce.

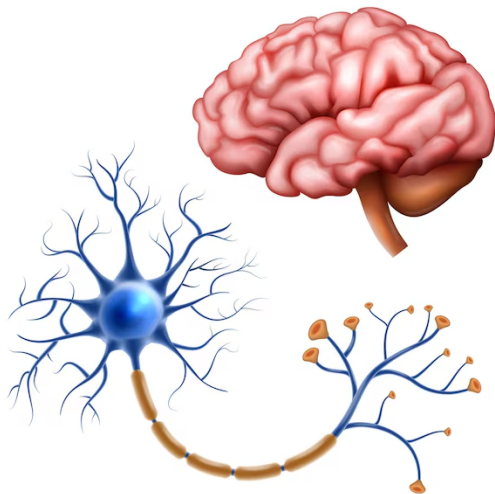
Linear Models, Why?

- In 2026, **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 simpler** to understand.
 - ✓ Still **widely used**, specially if data is scarce.
 - ✓ They are a **component of DNNs**.

Linear Models, Why?

- In 2026, **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 simpler** to understand.
 - ✓ Still **widely used**, specially if data is scarce.
 - ✓ They are a **component of DNNs**.
 - ✓ Natural **starting point** for studying ML.

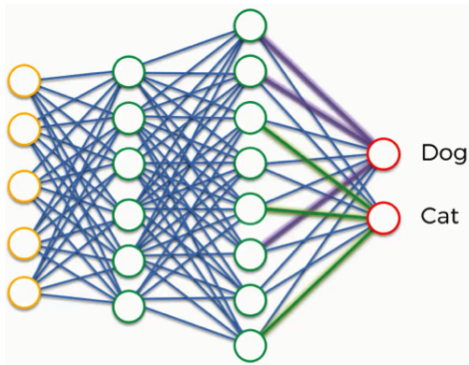
Brains and Neurons



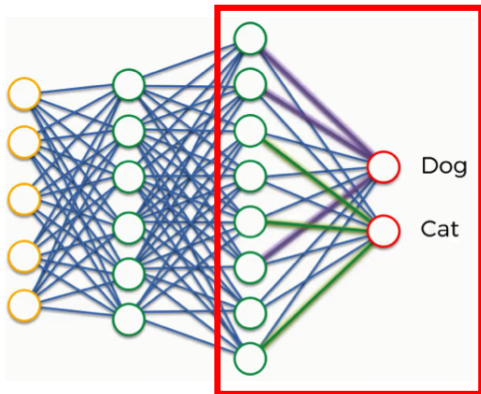
Spherical Cow



Linear Classifiers and Neural Networks

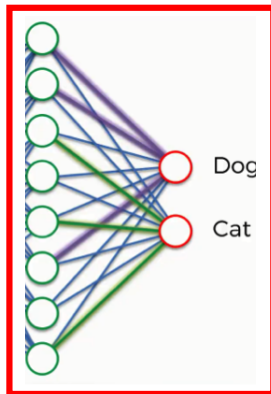


Linear Classifiers and Neural Networks



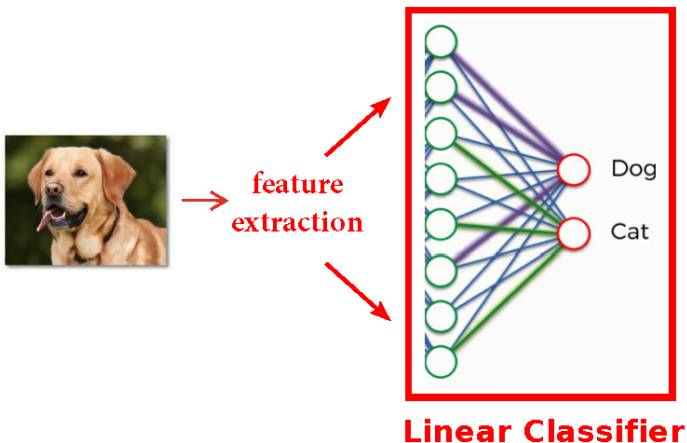
Linear Classifier

Linear Classifiers and Neural Networks

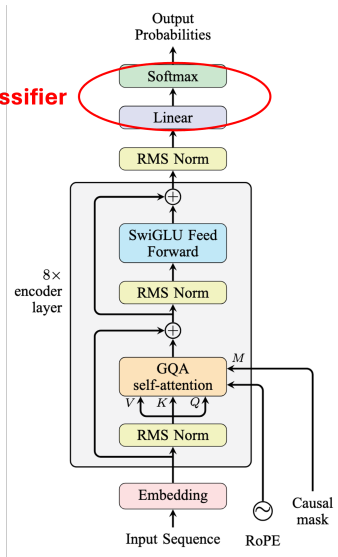
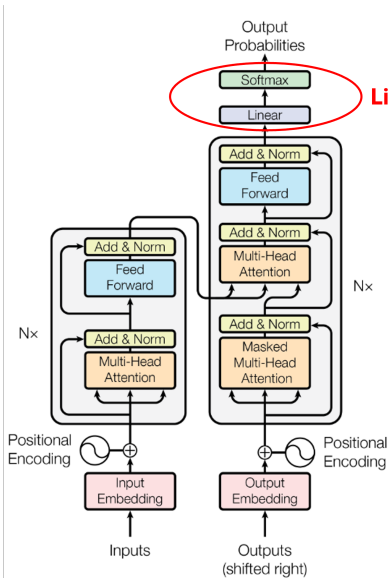


Linear Classifier

Linear Classifiers and Neural Networks



... even transformers!



Notation: Inputs and Outputs

- **Input** $x \in \mathcal{X}$
 - ✓ **Examples:** a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, a sequence of words, ...

Notation: Inputs and Outputs

- **Input** $x \in \mathcal{X}$
 - ✓ **Examples:** a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, a sequence of words, ...
- **Output** $y \in \mathcal{Y}$
 - ✓ **Examples:** fake/true, spam/legitimate, an identity, a diagnostic, fraud/legitimate, next word, ...

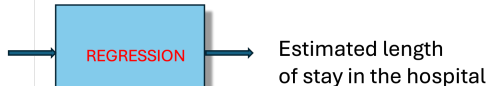
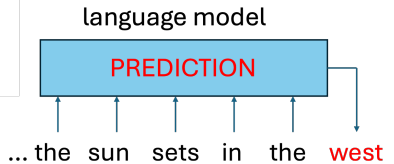
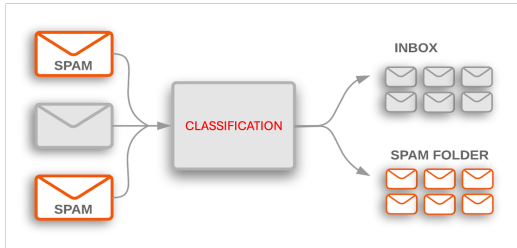
Notation: Inputs and Outputs

- **Input** $x \in \mathcal{X}$
 - ✓ **Examples:** a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, a sequence of words, ...
- **Output** $y \in \mathcal{Y}$
 - ✓ **Examples:** fake/true, spam/legitimate, an identity, a diagnostic, fraud/legitimate, next word, ...
- **Input/output** pair: $(x, y) \in \mathcal{X} \times \mathcal{Y}$

Notation: Inputs and Outputs

- **Input** $x \in \mathcal{X}$
 - ✓ **Examples:** a news article, an email message, a face image, a collection of laboratory test results, features of a credit card transaction, a sequence of words, ...
- **Output** $y \in \mathcal{Y}$
 - ✓ **Examples:** fake/true, spam/legitimate, an identity, a diagnostic, fraud/legitimate, next word, ...
- **Input/output** pair: $(x, y) \in \mathcal{X} \times \mathcal{Y}$
 - ✓ 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



Optimal Decisions

- **Goal:** find a “good” decision function: $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict

$$\hat{y} = h(x)$$

Optimal Decisions

- **Goal:** find a “good” decision function: $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict

$$\hat{y} = h(x)$$

- **Ideal situation:** joint distribution $f_{\mathbf{X}, \mathbf{Y}}(\mathbf{x}, \mathbf{y})$ is known.

Optimal Decisions

- **Goal**: find a “good” decision function: $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict

$$\hat{y} = h(x)$$

- **Ideal situation**: joint distribution $f_{\mathbf{X}, \mathbf{Y}}(\mathbf{x}, \mathbf{y})$ is known.
- **Assessing decisions** (how good/bad?): **loss function**:

$L(\mathbf{y}, \hat{\mathbf{y}})$ = loss caused by deciding $\hat{\mathbf{y}}$ if the truth is $\mathbf{y}$

Optimal Decisions

- **Goal**: find a “good” decision function: $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict

$$\hat{\mathbf{y}} = h(\mathbf{x})$$

- **Ideal situation**: joint distribution $f_{\mathbf{X}, \mathbf{Y}}(\mathbf{x}, \mathbf{y})$ is known.
- **Assessing decisions** (how good/bad?): **loss function**:

$L(\mathbf{y}, \hat{\mathbf{y}})$ = loss caused by deciding $\hat{\mathbf{y}}$ if the truth is $\mathbf{y}$

- **Optimal decision functions**: minimize **expected loss** or **risk**:

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

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

Optimal Decisions

- **Goal**: find a “good” decision function: $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict

$$\hat{\mathbf{y}} = h(\mathbf{x})$$

- **Ideal situation**: joint distribution $f_{\mathbf{X}, \mathbf{Y}}(\mathbf{x}, \mathbf{y})$ is known.
- **Assessing decisions** (how good/bad?): **loss function**:

$L(\mathbf{y}, \hat{\mathbf{y}})$ = loss caused by deciding $\hat{\mathbf{y}}$ if the truth is $\mathbf{y}$

- **Optimal decision functions**: minimize **expected loss** or **risk**:

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

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

- However, $f_{\mathbf{X}, \mathbf{Y}}(\mathbf{x}, \mathbf{y})$ is **unknown**: use **supervised learning**

Supervised Learning

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

$$\mathcal{D} = (\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y} \quad (\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})$$

Supervised Learning

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

$$\mathcal{D} = (\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y} \quad (\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})$$

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

Supervised Learning

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

$$\mathcal{D} = (\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y} \quad (\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})$$

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

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

Supervised Learning

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

$$\mathcal{D} = (\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y} \quad (\mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y})$$

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

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

- This lecture focuses on **discriminative supervised learning**.

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**: $\mathcal{Y} = \{1, 2, \dots, K\}$ (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**: $\mathcal{Y} = \{1, 2, \dots, K\}$ (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

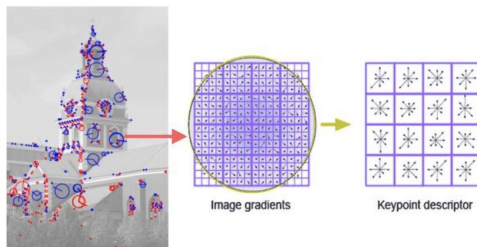
- Feature engineering is (was?) an important step for linear models:

Feature Representations

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

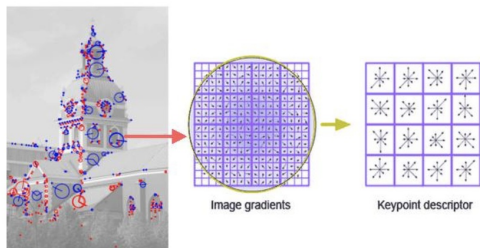
Feature Representations

- 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

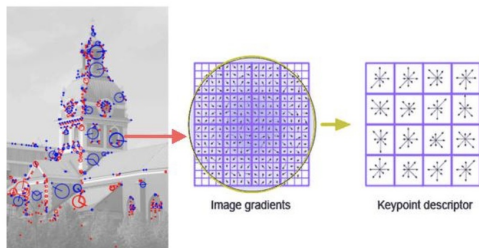
- **Feature engineering** is (was?) an important step for linear models:
 - ✓ Bag-of-words features for text, parts-of-speech, ...
 - ✓ SIFT **features** and wavelet representations in computer vision



- ✓ 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” x
- Typical approach: a **feature map** $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$

Feature Representations

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

Feature Representations

- Feature represent information about an “object” x
- Typical approach: a **feature map** $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$
- $\phi(x)$ is a (maybe high-dimensional) **feature vector**
- Categorical features (e.g., words/tokens in a sentence) may be represented by vectors: **embeddings**.

Feature Representations

- Feature represent information about an “object” x
- Typical approach: a **feature map** $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$
- $\phi(x)$ is a (maybe high-dimensional) **feature vector**
- Categorical features (e.g., words/tokens in a sentence) may be represented by vectors: **embeddings**.
- **Embeddings** play a central role in modern machine 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



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**



Tomorrow's lecture by **Bhiksha Raj**



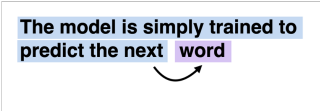
What about self-supervised learning?

What about self-supervised learning?

- Essentially, it's just supervised learning with programmatically defined training outputs.

What about self-supervised learning?

- Essentially, it's just supervised learning with programmatically defined training outputs.
- Classical examples: next word prediction.



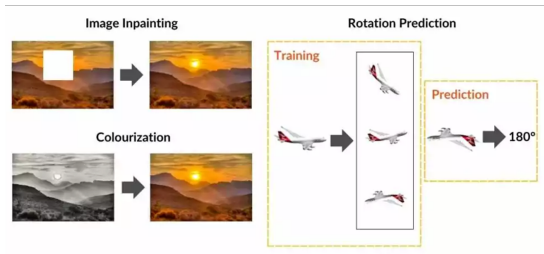
The model is simply trained to
predict the next word

What about self-supervised learning?

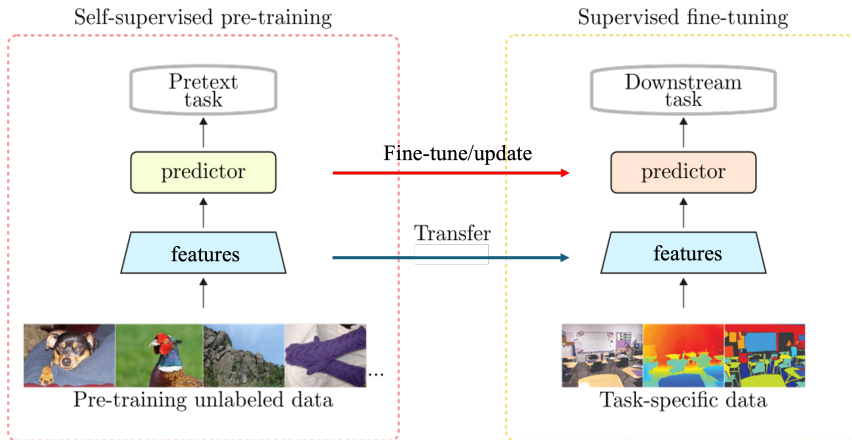
- Essentially, it's just supervised learning with programmatically defined training outputs.
- Classical examples: next word prediction.

The model is simply trained to predict the next word

- Many other possible pretext tasks:

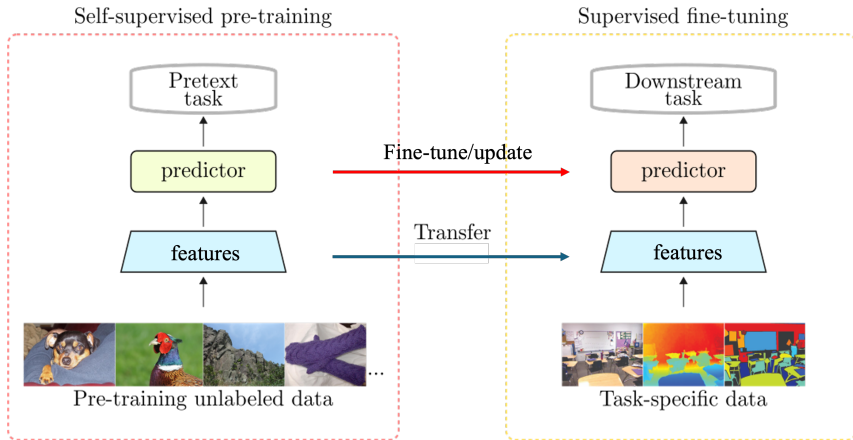


Self-supervised Learning (SSL)



(picture by Thalles Silva)

Self-supervised Learning (SSL)



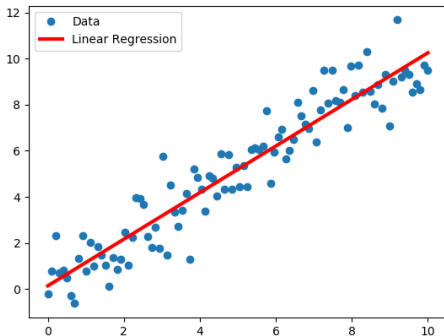
(picture by Thalles Silva)

- Both phases of **SSL** are essentially **supervised learning**.

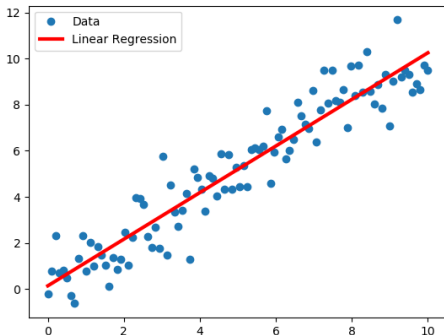
Outline

- ① Introduction
- ② Regression
- ③ Classification
- ④ Optimization for Supervised Learning

Linear Regression: A Picture



Linear Regression: A Picture



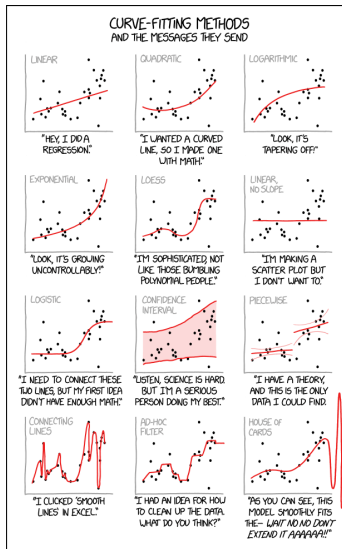
“When you’re fundraising, it’s **AI**.

When you’re hiring, it’s **ML**.

When you’re implementing, it’s just **linear regression**” (B. Schwartz)

Linear (Nonlinear) Regression

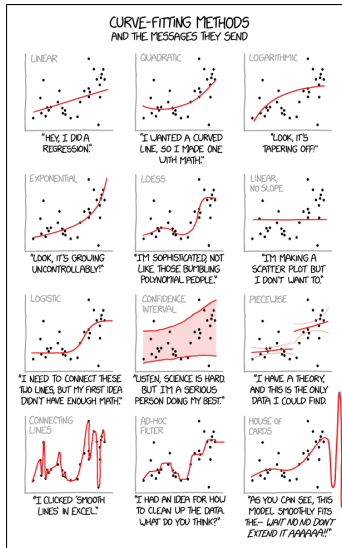
- “Linear” regression may be nonlinear (more later)



xkcd.com

Linear (Nonlinear) Regression

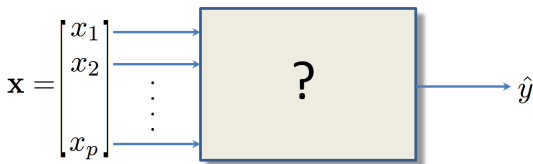
- “**Linear**” regression may be **nonlinear** (more later)
- Beware the **inductive bias**



xkcd.com

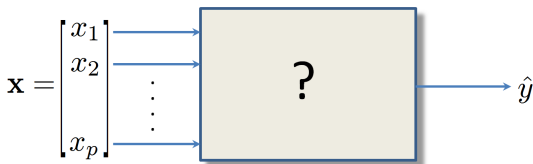
Regression

- In a nutshell: build a “machine” that predicts/estimates/guesses a quantity y from other “quantities” / “features” $x_1, \dots, x_p$



Regression

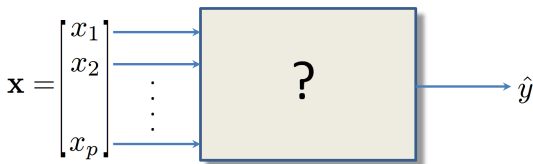
- In a nutshell: build a “machine” that predicts/estimates/guesses a quantity y from of other “quantities” / “features” $x_1, \dots, x_p$



- 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” $x_1, \dots, x_p$



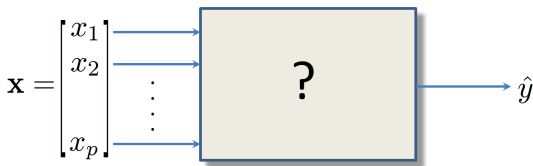
- Fundamental tool in data analysis, thus in much of science (biology, social sciences, economics, physics, ...) and engineering.
- Learning/training: given a collection of examples (training data)

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

..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” $x_1, \dots, x_p$



- 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)

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

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

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

Linear Regression

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

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

Linear Regression

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

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

Linear Regression

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

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

Linear Regression

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

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

Linear Regression

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

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

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

Linear Regression

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

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

- Standard loss: $L(y, \hat{y}) = (y - \hat{y})^2$ (why? what assumptions?)
- Empirical risk and residual sum of squares (RSS)

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

Linear Regression

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

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

- Standard loss: $L(y, \hat{y}) = (y - \hat{y})^2$ (why? what assumptions?)
- Empirical risk and residual sum of squares (RSS)

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

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

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

Linear Regression: Statistical Assumptions

- The inputs $\mathbf{x}_1, \dots, \mathbf{x}_n$ are seen as **deterministic, fixed, given**.

Linear Regression: Statistical Assumptions

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

Linear Regression: Statistical Assumptions

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

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

Linear Regression: Statistical Assumptions

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

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

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

Linear Regression: Statistical Assumptions

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

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

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

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

Linear Regression: Statistical Assumptions

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

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

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

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

Linear Regression: Statistical Assumptions

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

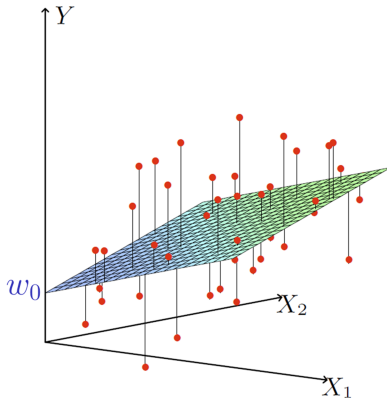
$$Y_i = w_0 + \mathbf{w}^T \mathbf{x}_i + N_i, \quad \text{where } N_i \sim \mathcal{N}(0, \sigma^2)$$

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

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

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

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 $\mathbf{x}_i$ with $\mathbf{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \in \mathbb{R}^{p+1}$

Linear Regression: Getting Rid of w_0

- Replace each original $\mathbf{x}_i$ with $\mathbf{x}_i = \begin{bmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \in \mathbb{R}^{p+1}$
- Let $\mathbf{w}$ now denote a $(p+1)$ -dimensional vector: $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1}$

Linear Regression: Getting Rid of w_0

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

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

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

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

Linear Regression: Getting Rid of w_0

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

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

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

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

- From now on, we will mostly **ignore** w_0 .

Linear Regression: Vector Notation

- Least squares (LS), a.k.a. ordinary LS (OLS) regression,

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

Linear Regression: Vector Notation

- Least squares (LS), a.k.a. ordinary LS (OLS) regression,

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

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

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

Linear Regression: Vector Notation

- Least squares (LS), a.k.a. ordinary LS (OLS) regression,

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

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

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

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

Linear Regression: Vector Notation

- Least squares (LS), a.k.a. ordinary LS (OLS) regression,

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

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

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

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

$$\hat{\mathbf{w}}_{\text{LS}}(\mathbf{y}) = \text{solution of } (\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0)$$

Linear Regression: Vector Notation

- Least squares (LS), a.k.a. ordinary LS (OLS) regression,

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

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

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

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

$$\hat{\mathbf{w}}_{\text{LS}}(\mathbf{y}) = \text{solution of } (\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0) = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Linear Regression: Vector Notation

- Least squares (LS), a.k.a. ordinary LS (OLS) regression,

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

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

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

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

$$\hat{\mathbf{w}}_{\text{LS}}(\mathbf{y}) = \text{solution of } (\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

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

A Classic: Coefficient of Determination R^2

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

A Classic: Coefficient of Determination R^2

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

A Classic: Coefficient of Determination R^2

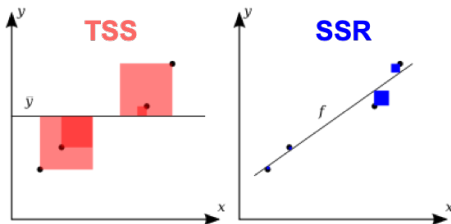
- Total sum of squares: $TSS = \sum_{i=1}^n (y_i - \bar{y})^2$ (variance $\times n$)
- Sum of squared residuals: $SSR = \sum_{i=1}^n (y_i - \hat{w}^T x_i)^2$
- Coefficient of determination:

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

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: $SSR = \sum_{i=1}^n (y_i - \hat{w}^T x_i)^2$
- Coefficient of determination:

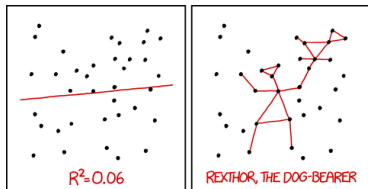
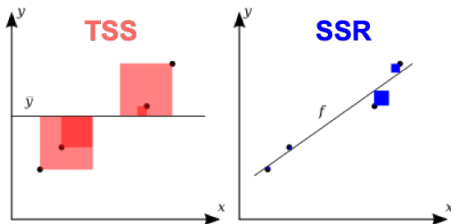
$$R^2 = 1 - \frac{SSR}{TSS} = 1 - FVU \quad (1 - \text{fraction of variance unexplained})$$



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: $SSR = \sum_{i=1}^n (y_i - \hat{w}^T x_i)^2$
- Coefficient of determination:

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



I DON'T TRUST LINEAR REGRESSIONS WHEN IT'S HARDER TO GUESS THE DIRECTION OF THE CORRELATION FROM THE SCATTER PLOT THAN TO FIND NEW CONSTELLATIONS ON IT.

The Geometry of Linear Regression

- Predicted values at the sampled points:

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

The Geometry of Linear Regression

- Predicted values at the sampled points:

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

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

The Geometry of Linear Regression

- Predicted values at the sampled points:

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

- Matrix $\mathbf{P}$ is a **projection matrix**; it is idempotent, $\mathbf{P}\mathbf{P} = \mathbf{P}$:
- Clearly, $\hat{\mathbf{y}} \in \text{range}(\mathbf{X})$ (span of the columns of $\mathbf{X}$); in fact,

$$\mathbf{P}\mathbf{y} = \mathbf{X} \underbrace{\left(\arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \right)}_{\hat{\mathbf{w}}_{\text{LS}}(\mathbf{y})}$$

The Geometry of Linear Regression

- Predicted values at the sampled points:

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

- Matrix $\mathbf{P}$ is a **projection matrix**; it is idempotent, $\mathbf{P}\mathbf{P} = \mathbf{P}$:
- Clearly, $\hat{\mathbf{y}} \in \text{range}(\mathbf{X})$ (span of the columns of $\mathbf{X}$); in fact,

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

The Geometry of Linear Regression

- Predicted values at the sampled points:

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

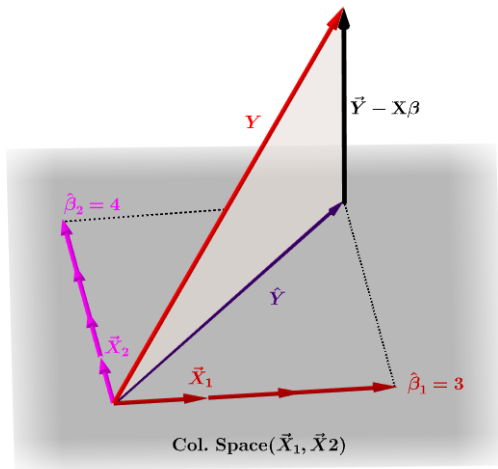
- Matrix $\mathbf{P}$ is a **projection matrix**; it is idempotent, $\mathbf{P}\mathbf{P} = \mathbf{P}$:
- Clearly, $\hat{\mathbf{y}} \in \text{range}(\mathbf{X})$ (span of the columns of $\mathbf{X}$); in fact,

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

i.e., the **orthogonal projection** onto $\text{range}(\mathbf{X})$.

Geometry of Linear Regression: Euclidean Projection

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



Going Non-Linear

- To express **non-linearities**, just replace x with $\phi(x)$,

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

Going Non-Linear

- To express **non-linearities**, just replace x with $\phi(x)$,

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

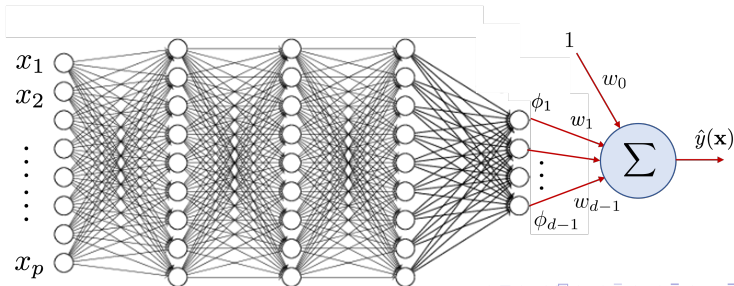
- Components of ϕ often called **features**, and ϕ a **feature map**.

Going Non-Linear

- To express **non-linearities**, just replace x with $\phi(x)$,

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

- 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 x with $\phi(x)$,

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

Going Non-Linear (but staying linear)

- To express **non-linearities**, just replace x with $\phi(x)$,

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

- The **LS criterion** becomes

$$\begin{aligned} \hat{w}_{\text{LS}} &= \arg \min_w \sum_{i=1}^n (y_i - w^T \phi(x_i))^2 \\ &= \arg \min_w \|y - Xw\|_2^2 \end{aligned}$$

where the **design matrix** X is now

$$X = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_{d-1}(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_n) & \cdots & \phi_{d-1}(x_n) \end{bmatrix} \in \mathbb{R}^{n \times d}$$

Going Non-Linear (but staying linear)

- To express **non-linearities**, just replace x with $\phi(x)$,

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

- The **LS criterion** becomes

$$\begin{aligned} \hat{w}_{\text{LS}} &= \arg \min_w \sum_{i=1}^n (y_i - w^T \phi(x_i))^2 \\ &= \arg \min_w \|y - Xw\|_2^2 = (X^T X)^{-1} X^T y \end{aligned}$$

where the **design matrix** X is now

$$X = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_{d-1}(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_n) & \cdots & \phi_{d-1}(x_n) \end{bmatrix} \in \mathbb{R}^{n \times d}$$

Example: Polynomial Regression

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

$$\phi(x) = [1, x, x^2, \dots, x^k]^T$$

Example: Polynomial Regression

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

$$\phi(x) = [1, x, x^2, \dots, x^k]^T$$

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

$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, \dots, x_1x_2^{k-1}, x_2^k]^T$$

...all monomials of order up to k

Example: Polynomial Regression

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

$$\phi(x) = [1, x, x^2, \dots, x^k]^T$$

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

$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, \dots, x_1x_2^{k-1}, x_2^k]^T$$

...all monomials of order up to k

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

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

Other Types of Non-Linear Regression

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

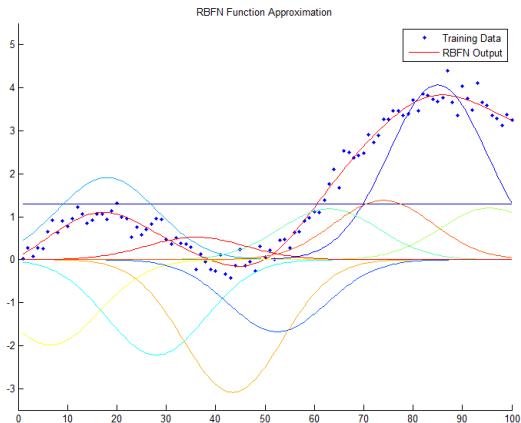
Other Types of Non-Linear Regression

- **Radial basis functions** (RBF): $\phi_j(\mathbf{x}) = \psi\left(\frac{1}{\alpha_j}\|\mathbf{x} - \mathbf{c}_j\|_2\right)$
...with fixed **centers** $\mathbf{c}_j$ 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): $\phi_j(\mathbf{x}) = \psi\left(\frac{1}{\alpha_j}\|\mathbf{x} - \mathbf{c}_j\|_2\right)$
...with fixed **centers** $\mathbf{c}_j$ 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 $\text{rank}(\mathbf{X}) < p$ (for example, if $n < p$), $\hat{\mathbf{w}}_{LS}$ cannot be computed,

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

Ridge Regression

- If $\text{rank}(\mathbf{X}) < p$ (for example, if $n < p$), $\hat{\mathbf{w}}_{\text{LS}}$ cannot be computed,
 $(\mathbf{X}^T \mathbf{X}) \in \mathbb{R}^{p \times p}$; $\text{rank}(\mathbf{X}) < p \Rightarrow (\mathbf{X}^T \mathbf{X})^{-1}$ **does not exist**
- The classical alternative is **ridge regression**:

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

Ridge Regression

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

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

- The classical alternative is **ridge regression**:

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

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

Ridge Regression

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

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

- The classical alternative is **ridge regression**:

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

- $\mathbf{X}^T \mathbf{X}$ is **positive semi-definite**: $\left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I} \right)$ is invertible, for $\lambda > 0$
- Can be seen as Bayesian estimate with Gaussian prior

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

Ridge Regression

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

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

- The classical alternative is **ridge regression**:

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

- $\mathbf{X}^T \mathbf{X}$ is **positive semi-definite**: $\left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I} \right)$ is invertible, for $\lambda > 0$
- Can be seen as Bayesian estimate with Gaussian prior

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

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

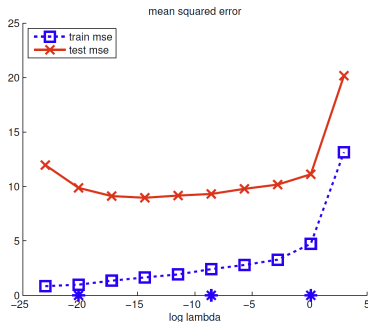
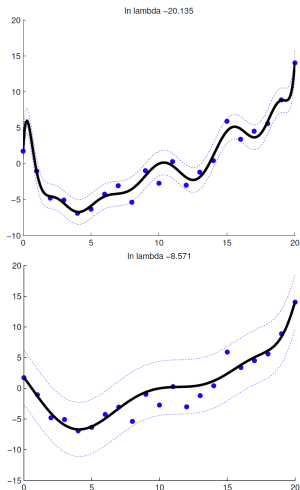
Ridge Regression: Illustration

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

Ridge Regression: Illustration

Even if $\hat{w}_{LS}$ can be computed, $\hat{w}_{ridge}$ may be 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

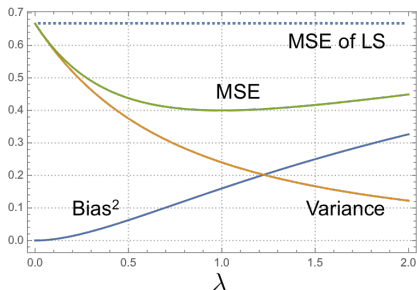
- Illustration with $p = 1$.
- The **bias-variance decomposition** ($\text{var}[U] = \mathbb{E}[U^2] - \mathbb{E}[U]^2$)

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

Bias-Variance Decomposition

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

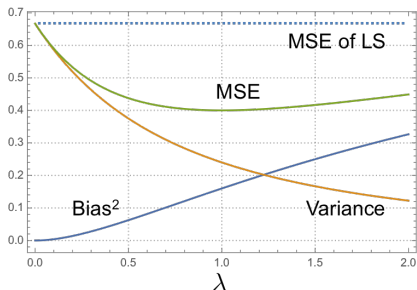
$$MSE = \mathbb{E}[(\hat{w}(\mathbf{Y}) - w)^2] = \underbrace{\text{var}[\hat{w}(\mathbf{Y})]}_{\text{variance}} + \underbrace{\mathbb{E}[\hat{w}(\mathbf{Y}) - w]^2}_{\text{squared bias}}$$



Bias-Variance Decomposition

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

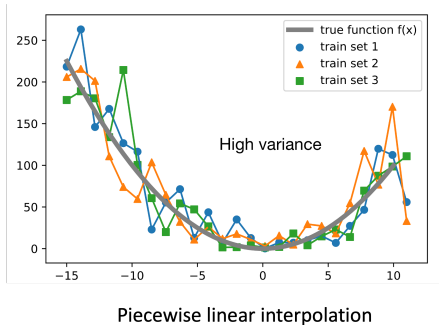
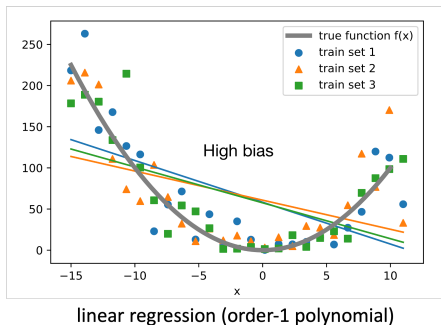
$$MSE = \mathbb{E}[(\hat{w}(\mathbf{Y}) - w)^2] = \underbrace{\text{var}[\hat{w}(\mathbf{Y})]}_{\text{variance}} + \underbrace{\mathbb{E}[\hat{w}(\mathbf{Y}) - w]^2}_{\text{squared bias}}$$



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

Bias-Variance Decomposition: Model Complexity

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



Pictures by Sebastian Raschka, 2023.

Choosing λ via Cross Validation (CV)

- Available data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$

Choosing λ via Cross Validation (CV)

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

Choosing λ via Cross Validation (CV)

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

Choosing λ via Cross Validation (CV)

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

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

Choosing λ via Cross Validation (CV)

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

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

- Choose λ by minimizing the average MSE estimate:

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

Choosing λ via Cross Validation (CV)

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

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

- Choose λ by minimizing the average MSE estimate:

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

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

Choosing λ via Cross Validation (CV)

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

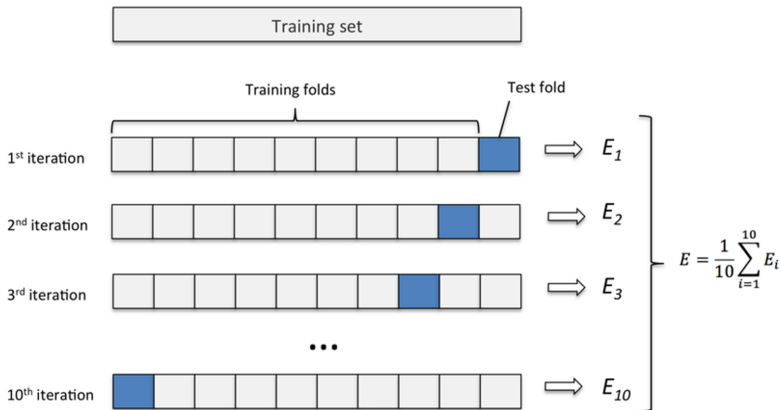
$$\widehat{\text{MSE}}_k(\lambda) = \frac{K}{n} \sum_{i \in S_k} (y_i - \mathbf{x}_i^T \hat{\mathbf{w}}_{\text{ridge}, \lambda}^{(k)})^2$$

- Choose λ by minimizing the average MSE estimate:

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

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

Illustration of 10-fold CV



Dual Variables: Ridge Regression

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

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} = \mathbf{X}^T \mathbf{y} \quad \Leftrightarrow \quad \mathbf{w} = \frac{1}{\lambda} \mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w})$$

Dual Variables: Ridge Regression

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

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} = \mathbf{X}^T \mathbf{y} \quad \Leftrightarrow \quad \mathbf{w} = \frac{1}{\lambda} \mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w})$$

that is,

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

Dual Variables: Ridge Regression

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

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} = \mathbf{X}^T \mathbf{y} \quad \Leftrightarrow \quad \mathbf{w} = \frac{1}{\lambda} \mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w})$$

that is,

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

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

Dual Variables: Ridge Regression

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

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} = \mathbf{X}^T \mathbf{y} \quad \Leftrightarrow \quad \mathbf{w} = \frac{1}{\lambda} \mathbf{X}^T (\mathbf{y} - \mathbf{X} \mathbf{w})$$

that is,

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

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

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

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

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

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

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

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

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

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

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

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

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

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

thus

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

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

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

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

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

thus

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

Dual Variables: Ridge Regression (2)

- Ridge regression in dual variables:

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

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

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

thus

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

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

The Representer Theorem

- For ridge regression, we found $\hat{\mathbf{w}} = \mathbf{X}^T \boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$;
is this specific to ridge, or a general fact?

The Representer Theorem

- For ridge regression, we found $\hat{\mathbf{w}} = \mathbf{X}^T \boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$;
is this specific to ridge, or a general fact?
- Representer theorem: consider any problem of the form

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w} \in \mathbb{R}^p} \sum_{i=1}^n L(\mathbf{w}^T \mathbf{x}_i, y_i) + \Omega(\|\mathbf{w}\|_2),$$

with $\Omega : [0, \infty) \rightarrow \mathbb{R}$ non-decreasing. Then, solution satisfies

$$\hat{\mathbf{w}} = \sum_{i=1}^n \alpha_i \mathbf{x}_i \in \text{span}\{\mathbf{x}_1, \dots, \mathbf{x}_n\}, \quad \text{for some } \boldsymbol{\alpha} \in \mathbb{R}^n.$$

The Representer Theorem

- For ridge regression, we found $\hat{\mathbf{w}} = \mathbf{X}^T \boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$;
is this specific to ridge, or a general fact?
- **Representer theorem**: consider any problem of the form

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w} \in \mathbb{R}^p} \sum_{i=1}^n L(\mathbf{w}^T \mathbf{x}_i, y_i) + \Omega(\|\mathbf{w}\|_2),$$

with $\Omega : [0, \infty) \rightarrow \mathbb{R}$ **non-decreasing**. Then, solution satisfies

$$\hat{\mathbf{w}} = \sum_{i=1}^n \alpha_i \mathbf{x}_i \in \text{span}\{\mathbf{x}_1, \dots, \mathbf{x}_n\}, \quad \text{for some } \boldsymbol{\alpha} \in \mathbb{R}^n.$$

- **Why it matters**: the optimal $\hat{\mathbf{w}}$ lives in an (at most) n -dimensional subspace, *regardless of* p (even if $p = \infty$).

The Representer Theorem

- For ridge regression, we found $\hat{\mathbf{w}} = \mathbf{X}^T \boldsymbol{\alpha} = \sum_{i=1}^n \alpha_i \mathbf{x}_i$;
is this specific to ridge, or a general fact?
- Representer theorem: consider any problem of the form

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w} \in \mathbb{R}^p} \sum_{i=1}^n L(\mathbf{w}^T \mathbf{x}_i, y_i) + \Omega(\|\mathbf{w}\|_2),$$

with $\Omega : [0, \infty) \rightarrow \mathbb{R}$ non-decreasing. Then, solution satisfies

$$\hat{\mathbf{w}} = \sum_{i=1}^n \alpha_i \mathbf{x}_i \in \text{span}\{\mathbf{x}_1, \dots, \mathbf{x}_n\}, \quad \text{for some } \boldsymbol{\alpha} \in \mathbb{R}^n.$$

- Why it matters: the optimal $\hat{\mathbf{w}}$ lives in an (at most) n -dimensional subspace, *regardless of* p (even if $p = \infty$).
- The loss L is *arbitrary* (needs not be squared error); ridge is one instance, with $\Omega(u) = \lambda u^2$.

The Representer Theorem: Why It Holds

- **Proof sketch:** split any w into components in and orthogonal to $\text{span}\{x_1, \dots, x_n\}$,

$$w = \underbrace{w_{\parallel}}_{\in \text{span}\{x_i\}} + w_{\perp}, \quad \text{with } w_{\perp}^T x_i = 0, \quad \forall i.$$

The Representer Theorem: Why It Holds

- **Proof sketch**: split any w into components in and orthogonal to $\text{span}\{x_1, \dots, x_n\}$,

$$w = \underbrace{w_{\parallel}}_{\in \text{span}\{x_i\}} + w_{\perp}, \quad \text{with } w_{\perp}^T x_i = 0, \quad \forall i.$$

- The **loss** only sees w through $w^T x_i = w_{\parallel}^T x_i$; the $w_{\perp}$ part is **invisible**

The Representer Theorem: Why It Holds

- **Proof sketch**: split any w into components in and orthogonal to $\text{span}\{x_1, \dots, x_n\}$,

$$w = \underbrace{w_{\parallel}}_{\in \text{span}\{x_i\}} + w_{\perp}, \quad \text{with } w_{\perp}^T x_i = 0, \quad \forall i.$$

- The **loss** only sees w through $w^T x_i = w_{\parallel}^T x_i$; the $w_{\perp}$ part is **invisible**
- The **regularizer** can only grow with $w_{\perp}$, since

$$\|w\|_2 = \sqrt{\|w_{\parallel}\|_2^2 + \|w_{\perp}\|_2^2} \geq \|w_{\parallel}\|_2,$$

and Ω is non-decreasing, thus setting $w_{\perp} = \mathbf{0}$ is optimal. $\square$

The Representer Theorem: Why It Holds

- **Proof sketch**: split any w into components in and orthogonal to $\text{span}\{x_1, \dots, x_n\}$,

$$w = \underbrace{w_{\parallel}}_{\in \text{span}\{x_i\}} + w_{\perp}, \quad \text{with } w_{\perp}^T x_i = 0, \quad \forall i.$$

- The **loss** only sees w through $w^T x_i = w_{\parallel}^T x_i$; the $w_{\perp}$ part is **invisible**
- The **regularizer** can only grow with $w_{\perp}$, since

$$\|w\|_2 = \sqrt{\|w_{\parallel}\|_2^2 + \|w_{\perp}\|_2^2} \geq \|w_{\parallel}\|_2,$$

and Ω is non-decreasing, thus setting $w_{\perp} = \mathbf{0}$ is optimal. □

- **Consequence**: substitute $w = \sum_i \alpha_i x_i$ and optimize over $\alpha \in \mathbb{R}^n$ instead of $w \in \mathbb{R}^p$.

The Representer Theorem: Why It Holds

- **Proof sketch**: split any w into components in and orthogonal to $\text{span}\{x_1, \dots, x_n\}$,

$$w = \underbrace{w_{\parallel}}_{\in \text{span}\{x_i\}} + w_{\perp}, \quad \text{with } w_{\perp}^T x_i = 0, \quad \forall i.$$

- The **loss** only sees w through $w^T x_i = w_{\parallel}^T x_i$; the $w_{\perp}$ part is **invisible**
- The **regularizer** can only grow with $w_{\perp}$, since

$$\|w\|_2 = \sqrt{\|w_{\parallel}\|_2^2 + \|w_{\perp}\|_2^2} \geq \|w_{\parallel}\|_2,$$

and Ω is non-decreasing, thus setting $w_{\perp} = \mathbf{0}$ is optimal. □

- **Consequence**: substitute $w = \sum_i \alpha_i x_i$ and optimize over $\alpha \in \mathbb{R}^n$ instead of $w \in \mathbb{R}^p$.
- Everything depends on the data only through inner products $x_i^T x_j$: **ready to be kernelized**.

Kernel Regression

- Recall that, in dual variables,

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

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

Kernel Regression

- Recall that, in dual variables,

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

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

Kernel Regression

- Recall that, in dual variables,

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

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

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

Kernel Regression

- Recall that, in dual variables,

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

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

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

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

Kernel Regression

- Recall that, in dual variables,

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

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

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

- $\mathbf{G}$ is still the **Gram matrix**, that is, $\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$
- 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 $\mathbb{R}^2$:

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

Kernel Regression (2)

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

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

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

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

Kernel Regression (2)

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

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

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

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

- 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 $\mathbb{R}^2$:

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

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

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

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

Kernel Regression (2)

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

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

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

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

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

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

Kernel Regression (2)

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

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

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

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

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

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

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

Kernel Regression (3)

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

Kernel Regression (3)

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

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

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

Kernel Regression (3)

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

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

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

- Hilbert space? Just a complete inner-product vector space.

Kernel Regression (3)

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

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

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** G is **positive semi-definite** (psd).

Kernel Regression (3)

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

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

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** $\mathbf{G}$ is **positive semi-definite** (psd).
- $\mathbf{G}$ is psd $\Rightarrow (\lambda \mathbf{I} + \mathbf{G})^{-1}$ exists, for $\lambda > 0$.

Kernels: Examples (1)

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

Kernels: Examples (1)

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

Kernels: Examples (1)

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

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

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

Kernels: Examples (1)

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

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

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

- **Polynomial** kernel: $K(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + A)^p$;

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

Kernels: Examples (1)

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

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

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

- **Polynomial** kernel: $K(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + A)^p$;

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

- There are kernels for many types of **objects**: sets, strings, images, graphs, probability distributions, point sets, ...

Kernels: Examples (2)

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

Kernels: Examples (2)

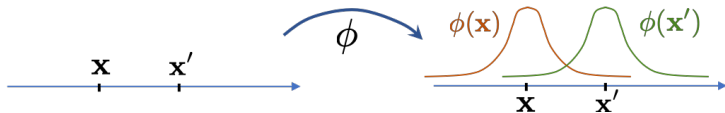
- In this slide, $\mathcal{X} = \mathbb{R}^d$
- **Gaussian** kernel: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right);$

Kernels: Examples (2)

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- **Gaussian** kernel: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right)$;
transformation $\phi : \mathbb{R}^d \rightarrow \mathcal{F}$, where $\mathcal{F}$ has infinite dimension.

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

- Illustration for $d = 1$:

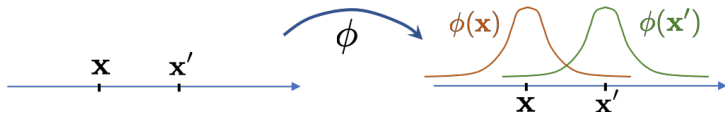


Kernels: Examples (2)

- In this slide, $\mathcal{X} = \mathbb{R}^d$
- **Gaussian** kernel: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right)$;
transformation $\phi : \mathbb{R}^d \rightarrow \mathcal{F}$, where $\mathcal{F}$ has infinite dimension.

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

- Illustration for $d = 1$:



- Why?

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

Kernels in 2026? Let's ask Claude

In the day and age of deep learning, transformers, LLMs, AI, when learning the foundations of machine learning, is it still relevant to study kernels and kernel methods? Please provide a short answer.

Yes. Kernel methods remain worth studying for several reasons:

They teach core concepts that generalize everywhere: the kernel trick illustrates how nonlinear problems become linear in higher-dimensional feature spaces, and they're the cleanest entry point to understanding regularization, the bias-variance tradeoff, and margin-based learning.

They connect directly to deep learning theory. The Neural Tangent Kernel shows that infinitely wide neural networks behave like kernel machines, and Gaussian processes (kernel methods in disguise) underpin much of Bayesian ML and uncertainty quantification.

They're still practically useful. On small-to-medium tabular datasets, SVMs and kernel ridge regression often match or beat neural nets with far less tuning and compute.

They build mathematical maturity. Reproducing kernel Hilbert spaces, representer theorems, and the associated optimization sharpen the linear-algebra and functional-analysis intuition that makes the rest of ML easier to reason about.

So while you won't reach for an SVM to build a chatbot, kernels are foundational scaffolding rather than a historical footnote.



Why Linear Models Still Matter

- A single-layer linear transformer has forward pass

$$\text{Output}(\mathbf{X}) = \frac{1}{n} \underbrace{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T(\mathbf{X}\mathbf{W}_V)}_{\text{linear attention}} \cdot \mathbf{W}_{\text{FFN}} \cdot \mathbf{W}_P$$

¹X. Tan, Y. Zhao, arXiv, 2026; P. Goulet Coulombe, SSRN, 2025.

Why Linear Models Still Matter

- A **single-layer linear transformer** has forward pass

$$\text{Output}(\mathbf{X}) = \frac{1}{n} \underbrace{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T(\mathbf{X}\mathbf{W}_V)}_{\text{linear attention}} \cdot \mathbf{W}_{\text{FFN}} \cdot \mathbf{W}_P$$

- $(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T$ is a **Gram matrix** of **inner products** between *transformed* inputs: compare $\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.

¹X. Tan, Y. Zhao, arXiv, 2026; P. Goulet Coulombe, SSRN, 2025.

Why Linear Models Still Matter

- A **single-layer linear transformer** has forward pass

$$\text{Output}(\mathbf{X}) = \frac{1}{n} \underbrace{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T(\mathbf{X}\mathbf{W}_V)}_{\text{linear attention}} \cdot \mathbf{W}_{\text{FFN}} \cdot \mathbf{W}_P$$

- $(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T$ is a **Gram matrix** of **inner products** between *transformed* inputs: compare $\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.
- Attention output = **similarity-weighted combination** of the rows of $\mathbf{X}\mathbf{W}_V$; kernel regression = similarity-weighted combination of the y_i .

¹X. Tan, Y. Zhao, arXiv, 2026; P. Goulet Coulombe, SSRN, 2025.

Why Linear Models Still Matter

- A **single-layer linear transformer** has forward pass

$$\text{Output}(\mathbf{X}) = \frac{1}{n} \underbrace{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T(\mathbf{X}\mathbf{W}_V)}_{\text{linear attention}} \cdot \mathbf{W}_{\text{FFN}} \cdot \mathbf{W}_P$$

- $(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T$ is a **Gram matrix** of **inner products** between *transformed* inputs: compare $\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.
- Attention output = **similarity-weighted combination** of the rows of $\mathbf{X}\mathbf{W}_V$; kernel regression = similarity-weighted combination of the y_i .
- **Claim**¹: for some $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_{\text{FFN}}, \mathbf{W}_P\}$, this is *exactly* the OLS projection $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$ (recall the **hat matrix**).

¹X. Tan, Y. Zhao, arXiv, 2026; P. Goulet Coulombe, SSRN, 2025.

Why Linear Models Still Matter

- A **single-layer linear transformer** has forward pass

$$\text{Output}(\mathbf{X}) = \frac{1}{n} \underbrace{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T(\mathbf{X}\mathbf{W}_V)}_{\text{linear attention}} \cdot \mathbf{W}_{\text{FFN}} \cdot \mathbf{W}_P$$

- $(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T$ is a **Gram matrix** of **inner products** between *transformed* inputs: compare $\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.
- Attention output = **similarity-weighted combination** of the rows of $\mathbf{X}\mathbf{W}_V$; kernel regression = similarity-weighted combination of the y_i .
- **Claim**¹: for some $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_{\text{FFN}}, \mathbf{W}_P\}$, this is *exactly* the OLS projection $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$ (recall the **hat matrix**).
- **Caveat**: this is *linear* attention (more later).

¹X. Tan, Y. Zhao, arXiv, 2026; P. Goulet Coulombe, SSRN, 2025. 

Why Linear Models Still Matter

- A **single-layer linear transformer** has forward pass

$$\text{Output}(\mathbf{X}) = \frac{1}{n} \underbrace{(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T(\mathbf{X}\mathbf{W}_V)}_{\text{linear attention}} \cdot \mathbf{W}_{\text{FFN}} \cdot \mathbf{W}_P$$

- $(\mathbf{X}\mathbf{W}_Q)(\mathbf{X}\mathbf{W}_K)^T$ is a **Gram matrix** of **inner products** between *transformed* inputs: compare $\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$.
- Attention output = **similarity-weighted combination** of the rows of $\mathbf{X}\mathbf{W}_V$; kernel regression = similarity-weighted combination of the y_i .
- **Claim**¹: for some $\{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_{\text{FFN}}, \mathbf{W}_P\}$, this is *exactly* the OLS projection $\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$ (recall the **hat matrix**).
- **Caveat**: this is *linear* attention (more later).
- Projections, Gram matrices, inner products: the tools to understand **attention** are **exactly** the ones in this lecture.

¹X. Tan, Y. Zhao, arXiv, 2026; P. Goulet Coulombe, SSRN, 2025. 

Minimum-Norm Linear Regression

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

Minimum-Norm Linear Regression

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

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

Minimum-Norm Linear Regression

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

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

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

Minimum-Norm Linear Regression

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

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

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

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

Minimum-Norm Linear Regression

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

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

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

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

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

Minimum-Norm Linear Regression

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

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

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

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

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

Minimum-Norm Linear Regression

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

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

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

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

- LS and MN: instances of the **Moore-Penrose pseudo-inverse**.
- **Perfect interpolation** regime: $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}_{\text{MN}}(\mathbf{y}) = \mathbf{y}$
- **Preview**: this reappears later as the **implicit bias** of gradient descent initialized at $\mathbf{0}$, even *without* regularization.

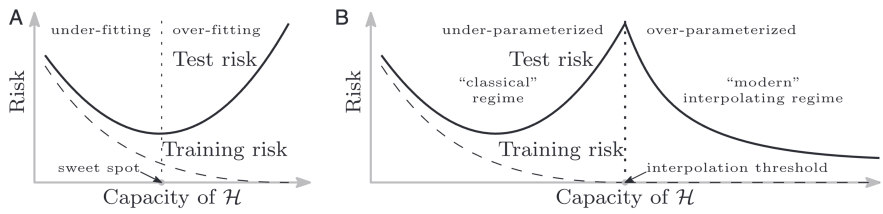
Double Descent

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

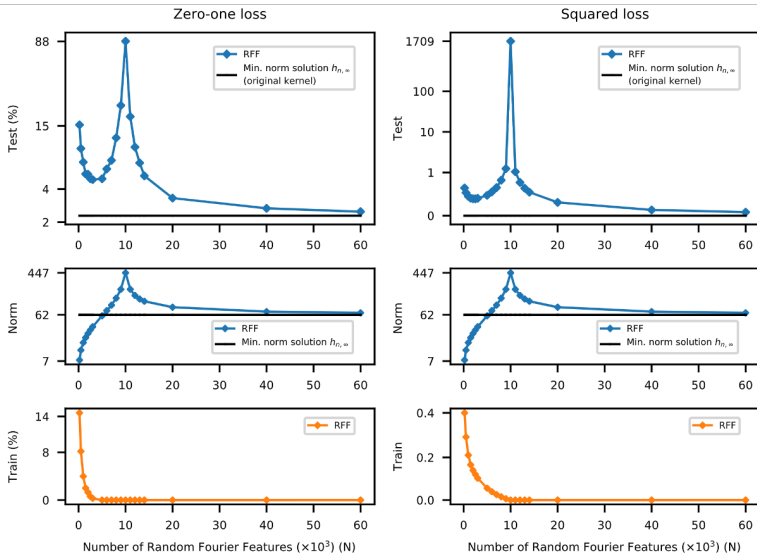
^aDepartment of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210; ^bDepartment of Statistics, The Ohio State University, Columbus, OH 43210; and ^cComputer Science Department and Data Science Institute, Columbia University, New York, NY 10027

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



Overparametrization and Double Descent

- “Modern” interpolating regime: more parameters than data points.

Overparametrization and Double Descent

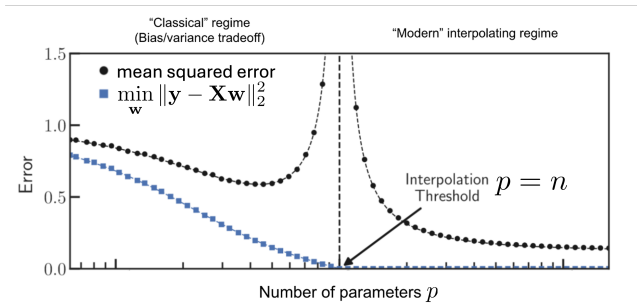
- “Modern” interpolating regime: more parameters than data points.
- For linear regression with $p \geq n$, use **minimum norm solution**.

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/ $\phi_i(\mathbf{x}) = \max\{\mathbf{v}_i^T \mathbf{x}, 0\}$, where $\mathbf{v}_i$ are random vectors.

Overparametrization and Double Descent

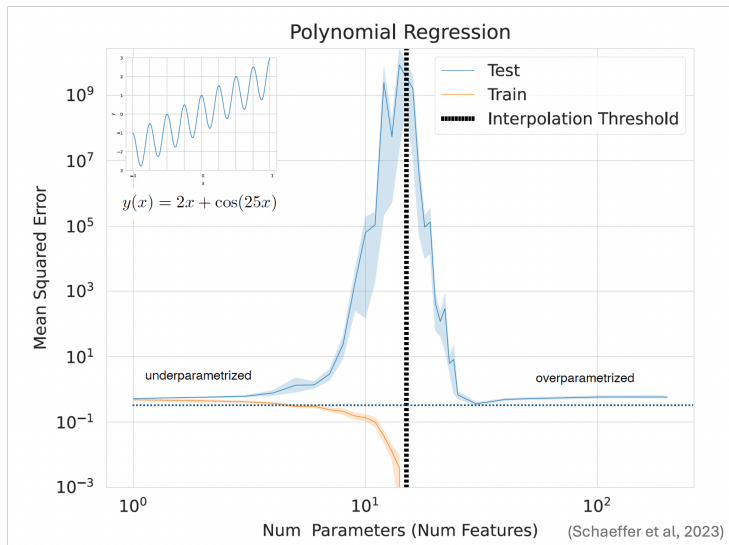
- “Modern” interpolating regime: more parameters than data points.
- For linear regression with $p \geq n$, use **minimum norm solution**.
- Example w/ $\phi_i(\mathbf{x}) = \max\{\mathbf{v}_i^T \mathbf{x}, 0\}$, where $\mathbf{v}_i$ are random vectors.



(Image adapted from Rocks and Mehta, 2022.)

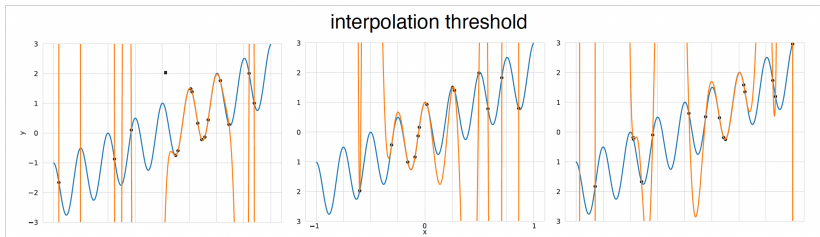
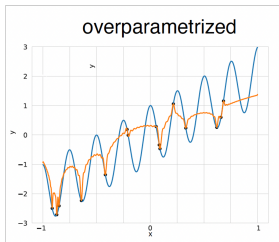
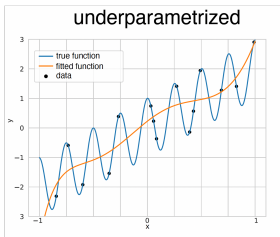
Overparametrization and Double Descent (cont.)

- Polynomial regression: the ϕ_i are Legendre polynomials.



Overparametrization and Double Descent (cont.)

- Polynomial regression: the ϕ_i are Legendre polynomials.



Bayesian View of Ridge Regression

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

Bayesian View of Ridge Regression

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

Bayesian View of Ridge Regression

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

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

Bayesian View of Ridge Regression

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

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

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

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

Bayesian View of Ridge Regression

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

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

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

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

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

Bayesian View of Ridge Regression

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

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

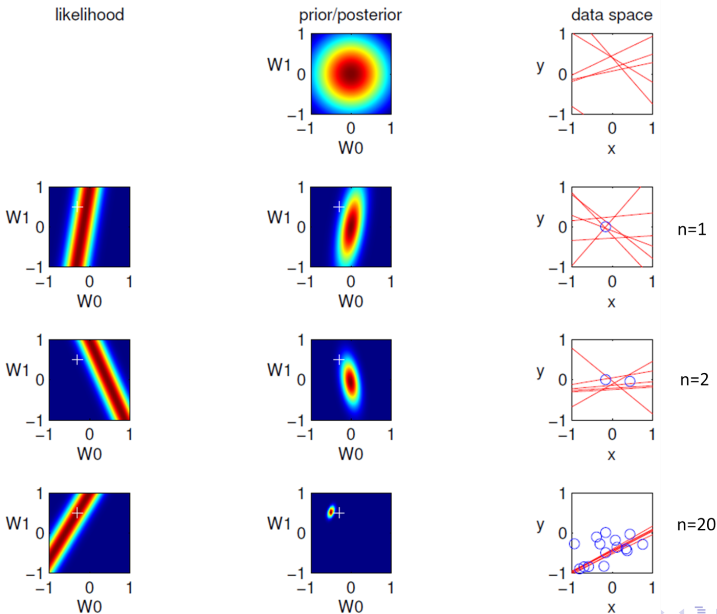
- **Prediction** at new point $\mathbf{x}_*$ is $Y(\mathbf{x}_*) = \mathbf{x}_*^T \mathbf{W} + N$ (Gaussian)

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

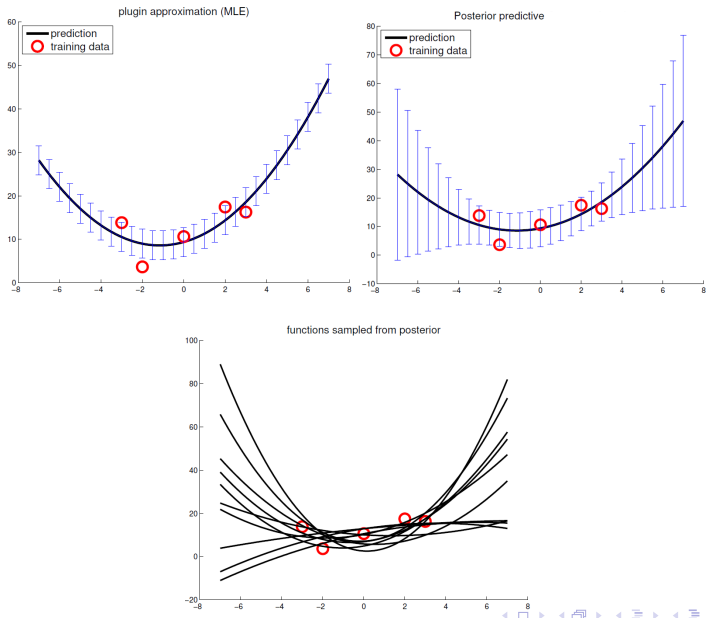
...the **variance/uncertainty** of the prediction depends on $\mathbf{x}_*$

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

Bayesian View of Ridge Regression: Example 1



Bayesian View of Ridge Regression: Example 2



Epistemic and Aleatoric Uncertainty

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

Epistemic and Aleatoric Uncertainty

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

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

Epistemic and Aleatoric Uncertainty

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

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

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

Epistemic and Aleatoric Uncertainty

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

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

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

Epistemic and Aleatoric Uncertainty

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

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

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

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

Epistemic and Aleatoric Uncertainty

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

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

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

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

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

Epistemic and Aleatoric Uncertainty

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

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

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

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

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

LASSO regression

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

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

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

- LASSO = least absolute shrinkage and selection operator

LASSO regression

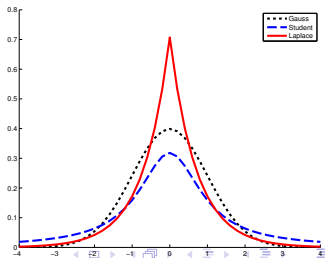
- Alternative to ridge regression, with built-in variable selection

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

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

- LASSO = least absolute shrinkage and selection operator
- Can be seen as MAP estimate of $\mathbf{w}$, under Laplacian prior

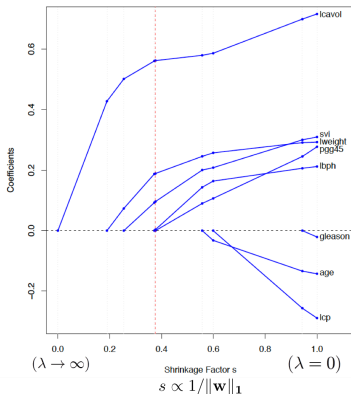
$$\begin{aligned} f_{\mathbf{w}}(\mathbf{w}) &= \prod_{i=1}^p \frac{\lambda}{2} \exp(-\lambda |w_i|) \\ &= \left(\frac{\lambda}{2}\right)^p \exp(-\lambda \|\mathbf{w}\|_1) \end{aligned}$$



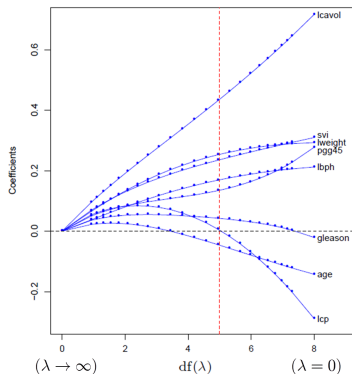
LASSO versus Ridge

- Example (prostate cancer data)

LASSO



Ridge



Solving LASSO Regression

- Ridge regression: simply a linear system:

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

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

Solving LASSO Regression

- Ridge regression: simply a linear system:

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

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

- LASSO is much more challenging:

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

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

Solving LASSO Regression

- Ridge regression: simply a linear system:

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

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

- LASSO is much more challenging:

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

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

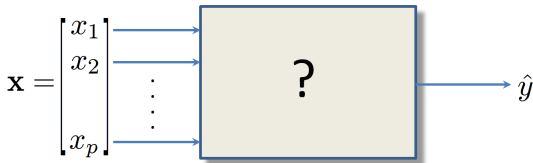
- Using gradient descent (e.g., in deep learning), simply pretend that ℓ_1 is differentiable (derivative in $\{-1, 0, 1\}$), carefully adapt the step size.

Outline

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

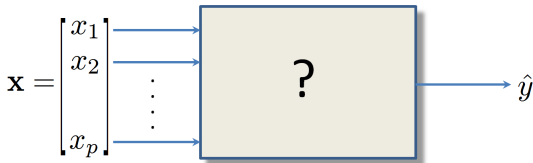
Classification (a.k.a. Pattern Recognition)

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



Classification (a.k.a. Pattern Recognition)

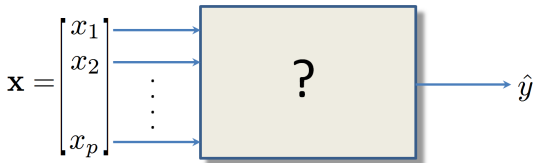
- In a nutshell: produce a “machine” that predicts/estimates/guesses a class $y \in \{1, \dots, K\}$, from variables/features $x_1, \dots, x_p$



- 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 $y \in \{1, \dots, K\}$, from variables/features $x_1, \dots, x_p$



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

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

..find the “best” possible machine.

Binary Classification: The Simplest Case

- Start with **two classes**: $y \in \{0, 1\}$ (e.g., spam/legit, sick/healthy, ...)

Binary Classification: The Simplest Case

- Start with **two classes**: $y \in \{0, 1\}$ (e.g., spam/legit, sick/healthy, ...)
- Goal: a **probability**, not just a decision. How **confident** that $\hat{y} = 1$?

Binary Classification: The Simplest Case

- Start with **two classes**: $y \in \{0, 1\}$ (e.g., spam/legit, sick/healthy, ...)
- Goal: a **probability**, not just a decision. How **confident** that $\hat{y} = 1$?
- As in regression, predictions depend on x through **linear** function

$$\mathbf{w}^T \phi(\mathbf{x}), \quad (\phi = \text{feature map, as before; } \phi(\mathbf{x}) = \mathbf{x} \text{ is allowed})$$

Binary Classification: The Simplest Case

- Start with **two classes**: $y \in \{0, 1\}$ (e.g., spam/legit, sick/healthy, ...)
- Goal: a **probability**, not just a decision. How **confident** that $\hat{y} = 1$?
- As in regression, predictions depend on x through **linear** function

$$w^T \phi(x), \quad (\phi = \text{feature map, as before; } \phi(x) = x \text{ is allowed})$$

- **Problem**: $w^T \phi(x) \in \mathbb{R}$, but a probability must lie in $[0, 1]$.

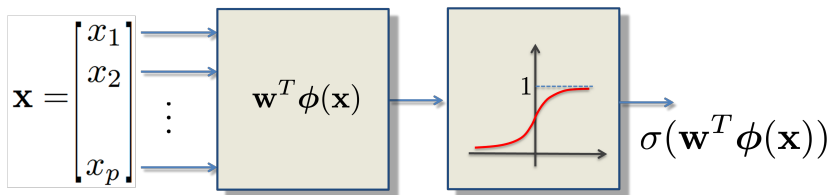
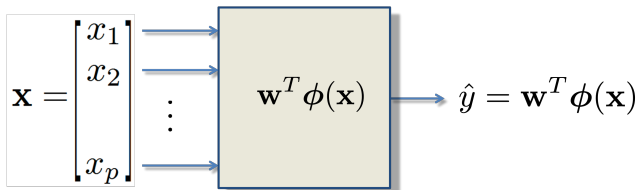
Binary Classification: The Simplest Case

- Start with **two classes**: $y \in \{0, 1\}$ (e.g., spam/legit, sick/healthy, ...)
- Goal: a **probability**, not just a decision. How **confident** that $\hat{y} = 1$?
- As in regression, predictions depend on x through **linear** function

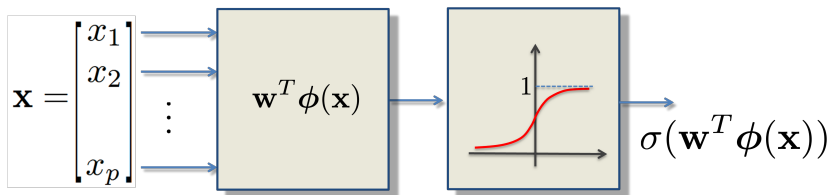
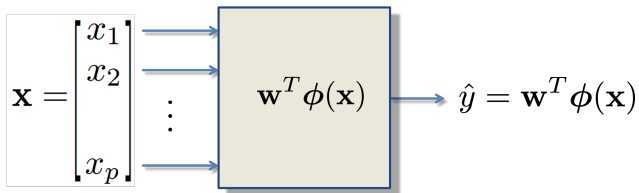
$$w^T \phi(x), \quad (\phi = \text{feature map, as before; } \phi(x) = x \text{ is allowed})$$

- **Problem**: $w^T \phi(x) \in \mathbb{R}$, but a probability must lie in $[0, 1]$.
- **Solution**: squash it through a function $\sigma : \mathbb{R} \rightarrow [0, 1]$.

From Linear Regression to Binary Classification



From Linear Regression to Binary Classification



- Since $\sigma \in [0, 1]$, we can take

$$\mathbb{P}[Y = 1 | \mathbf{X} = \mathbf{x}] = f_{Y|\mathbf{X}}(1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$$

Binary Logistic Regression

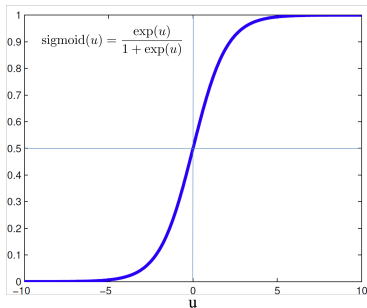
- Model:

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

Binary Logistic Regression

- Model:

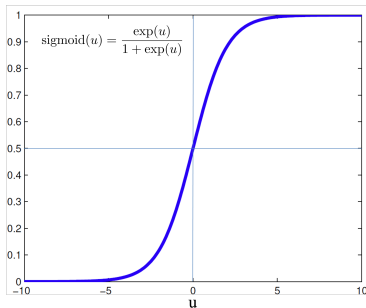
$$f_{Y|X}(1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x})) = \text{sigmoid}(\mathbf{w}^T \phi(\mathbf{x})) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}))}{1 + \exp(\mathbf{w}^T \phi(\mathbf{x}))}$$



Binary Logistic Regression

- Model:

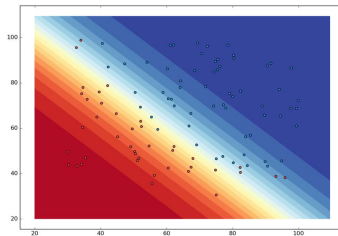
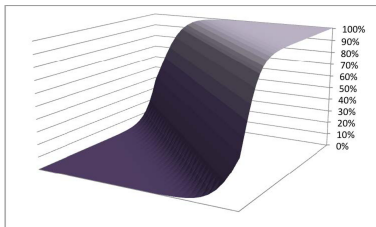
$$f_{Y|X}(1|\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x})) = \text{sigmoid}(\mathbf{w}^T \phi(\mathbf{x})) = \frac{\exp(\mathbf{w}^T \phi(\mathbf{x}))}{1 + \exp(\mathbf{w}^T \phi(\mathbf{x}))}$$



- Obviously, $f_{Y|X}(0|\mathbf{x}) = 1 - f_{Y|X}(1|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T \phi(\mathbf{x}))}$.

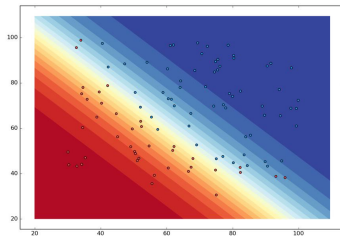
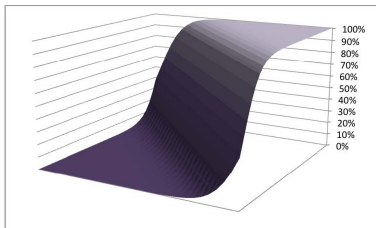
Binary Logistic Regression

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



Binary Logistic Regression

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



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

Binary Logistic Regression: Log-Likelihood

- As before, the y_i are **conditionally independent** given $\mathbf{x}_1, \dots, \mathbf{x}_n$, with

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

Binary Logistic Regression: Log-Likelihood

- As before, the y_i are **conditionally independent** given $\mathbf{x}_1, \dots, \mathbf{x}_n$, with

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

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

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

Binary Logistic Regression: Log-Likelihood

- As before, the y_i are **conditionally independent** given $\mathbf{x}_1, \dots, \mathbf{x}_n$, with

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

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

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

- ML estimate:** $\hat{\mathbf{w}}_{\text{ML}} = \arg \min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ ($\mathcal{L}$ = **cross-entropy loss**)

Binary Logistic Regression: Log-Likelihood

- As before, the y_i are **conditionally independent** given $\mathbf{x}_1, \dots, \mathbf{x}_n$, with

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

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

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

- ML estimate:** $\hat{\mathbf{w}}_{\text{ML}} = \arg \min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ ($\mathcal{L}$ = **cross-entropy loss**)
- No closed form!** We need **optimization algorithms** (later)

Binary Logistic Regression: Log-Likelihood

- As before, the y_i are **conditionally independent** given $\mathbf{x}_1, \dots, \mathbf{x}_n$, with

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

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

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

- ML estimate:** $\hat{\mathbf{w}}_{\text{ML}} = \arg \min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ ($\mathcal{L}$ = **cross-entropy loss**)
- No closed form!** We need **optimization algorithms** (later)
- $\mathcal{L}(\mathbf{w})$ is **smooth and convex** (should not be too hard to optimize)

From 2 to K Classes: Generalized Linear Models

- For K classes, give *each* class its own parameter vector $\mathbf{w}^{(y)}$, and normalize:

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

From 2 to K Classes: Generalized Linear Models

- For K classes, give *each* class its own parameter vector $\mathbf{w}^{(y)}$, and normalize:

$$f_{Y|\mathbf{X}}(y|\mathbf{x}) = \frac{\exp((\mathbf{w}^{(y)})^T \phi(\mathbf{x}))}{\sum_{u=1}^K \exp((\mathbf{w}^{(u)})^T \phi(\mathbf{x}))} = \text{softmax}(\phi(\mathbf{x})^T \mathbf{w})_y$$

From 2 to K Classes: Generalized Linear Models

- For K classes, give *each* class its own parameter vector $\mathbf{w}^{(y)}$, and normalize:

$$f_{Y|\mathbf{X}}(y|\mathbf{x}) = \frac{\exp((\mathbf{w}^{(y)})^T \phi(\mathbf{x}))}{\sum_{u=1}^K \exp((\mathbf{w}^{(u)})^T \phi(\mathbf{x}))} = \text{softmax}(\phi(\mathbf{x})^T \mathbf{w})_y$$

- This is a **generalized linear model** (GLM); also called **multinomial logistic regression**, **maximum entropy**, **softmax** classifier.

From 2 to K Classes: Generalized Linear Models

- For K classes, give *each* class its own parameter vector $\mathbf{w}^{(y)}$, and normalize:

$$f_{Y|X}(y|\mathbf{x}) = \frac{\exp((\mathbf{w}^{(y)})^T \phi(\mathbf{x}))}{\sum_{u=1}^K \exp((\mathbf{w}^{(u)})^T \phi(\mathbf{x}))} = \text{softmax}(\phi(\mathbf{x})^T \mathbf{w})_y$$

- This is a **generalized linear model** (GLM); also called **multinomial logistic regression**, **maximum entropy**, **softmax** classifier.
- Consistency check**: for $K = 2$, dividing numerator and denominator by $\exp((\mathbf{w}^{(0)})^T \phi(\mathbf{x}))$ recovers

$$f_{Y|X}(1|\mathbf{x}) = \text{sigmoid}(\mathbf{w}^T \phi(\mathbf{x})), \quad \text{with } \mathbf{w} = \mathbf{w}^{(1)} - \mathbf{w}^{(0)}.$$

From 2 to K Classes: Generalized Linear Models

- For K classes, give *each* class its own parameter vector $\mathbf{w}^{(y)}$, and normalize:

$$f_{Y|X}(y|\mathbf{x}) = \frac{\exp((\mathbf{w}^{(y)})^T \phi(\mathbf{x}))}{\sum_{u=1}^K \exp((\mathbf{w}^{(u)})^T \phi(\mathbf{x}))} = \text{softmax}(\phi(\mathbf{x})^T \mathbf{w})_y$$

- This is a **generalized linear model** (GLM); also called **multinomial logistic regression**, **maximum entropy**, **softmax** classifier.
- Consistency check**: for $K = 2$, dividing numerator and denominator by $\exp((\mathbf{w}^{(0)})^T \phi(\mathbf{x}))$ recovers

$$f_{Y|X}(1|\mathbf{x}) = \text{sigmoid}(\mathbf{w}^T \phi(\mathbf{x})), \quad \text{with } \mathbf{w} = \mathbf{w}^{(1)} - \mathbf{w}^{(0)}.$$

- Only **differences** between the $\mathbf{w}^{(y)}$ matter: the model is **over-parametrized**.

Ridge and LASSO Logistic Regression

- Ridge logistic regression:

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

still smooth and convex.

Ridge and LASSO Logistic Regression

- Ridge logistic regression:

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

still smooth and convex.

- Sparse (LASSO) logistic regression:

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

still convex, but not smooth.

Multi-class Logistic Regression

- Recall the GLM,

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

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

Multi-class Logistic Regression

- Recall the GLM,

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

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

- A.k.a. multinomial/multi-class logistic, maximum entropy, softmax,

Multi-class Logistic Regression

- Recall the GLM,

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

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

- A.k.a. multinomial/multi-class logistic, maximum entropy, softmax,
- The negative log-likelihood function (cross-entropy loss):

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

Multi-class Logistic Regression (2)

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

Multi-class Logistic Regression (2)

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

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

Multi-class Logistic Regression (2)

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

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

can be written as

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

Multi-class Logistic Regression (2)

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

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

can be written as

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

- **Notice:** if $\mathbf{x}_i$ is in class k , **minimize** $\mathcal{L}(\mathbf{w}) \Rightarrow$ **maximize** $\mathbf{x}_i^T \mathbf{w}^{(k)}$

Bayesian Logistic Regression

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

Bayesian Logistic Regression

- Using some estimate $\hat{w}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|X}(y|x, \hat{w})$ ignores the **randomness/uncertainty** in $\hat{w}$
- **Bayesian approach**: from a **prior** $f_W(w)$, compute the **posterior**

$$f_{W|Y}(w|y) = \frac{f_W(w) f_{Y|W}(y|w)}{f_Y(y)}$$

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

Bayesian Logistic Regression

- Using some estimate $\hat{w}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|X}(y|x, \hat{w})$ ignores the **randomness/uncertainty** in $\hat{w}$
- **Bayesian approach**: from a **prior** $f_W(w)$, compute the **posterior**

$$f_{W|Y}(w|y) = \frac{f_W(w) f_{Y|W}(y|w)}{f_Y(y)}$$

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

- Given some new point x_* , the **predictive distribution** is

$$f_{Y|X}(y|x_*, y) = \int f_{W|Y}(w|y) f_{Y|X}(y|x_*, w) dw$$

Bayesian Logistic Regression

- Using some estimate $\hat{w}$, obtained from data $\mathcal{D}$, and plugging it into $f_{Y|X}(y|x, \hat{w})$ ignores the **randomness/uncertainty** in $\hat{w}$
- Bayesian approach**: from a **prior** $f_W(w)$, compute the **posterior**

$$f_{W|Y}(w|y) = \frac{f_W(w) f_{Y|W}(y|w)}{f_Y(y)}$$

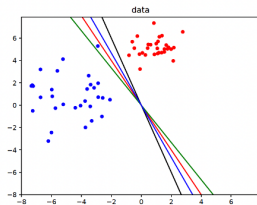
where $f_{Y|W}(y|w) = \prod_{i=1}^N f_{Y|X}(y_i|x_i, w)$ (recall x_i are deterministic)

- Given some new point x_* , the **predictive distribution** is

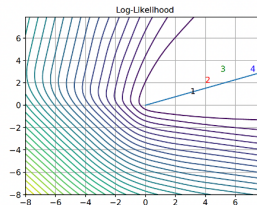
$$f_{Y|X}(y|x_*, y) = \int f_{W|Y}(w|y) f_{Y|X}(y|x_*, w) dw$$

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

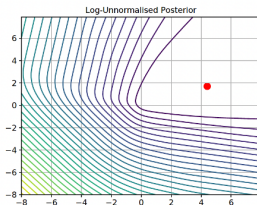
Bayesian Logistic Regression (2)



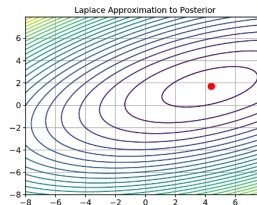
(a)



(b)



(c)



(d)

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.pyro.ai/book1/10.13.

Bayesian Logistic Regression (3)

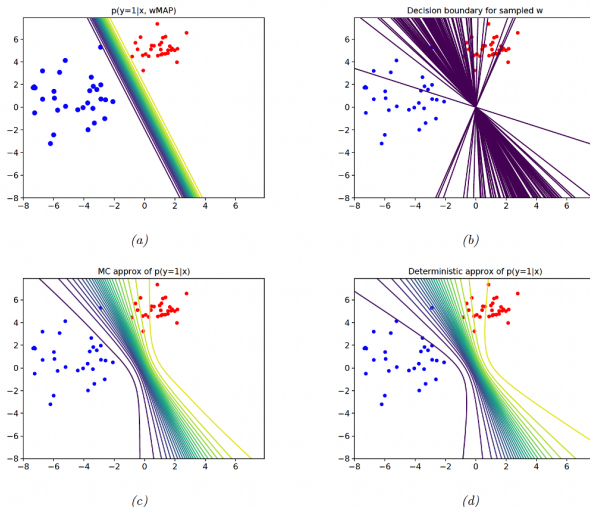
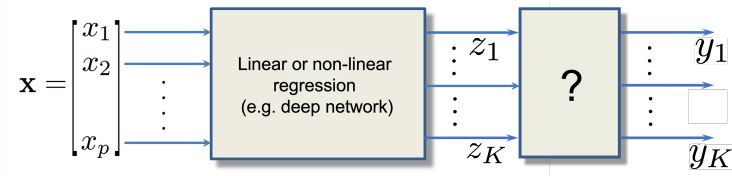
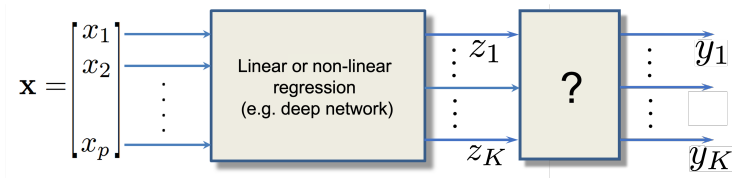


Figure 10.14: Posterior predictive distribution for a logistic regression model in 2d. (a): contours of $p(y=1|x, \hat{w}_{MAP})$. (b): samples from the posterior predictive distribution. (c): Averaging over these samples. (d): moderated output (probit approximation). Adapted from a figure by Mark Girolami.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

Another View of (and Beyond) Softmax

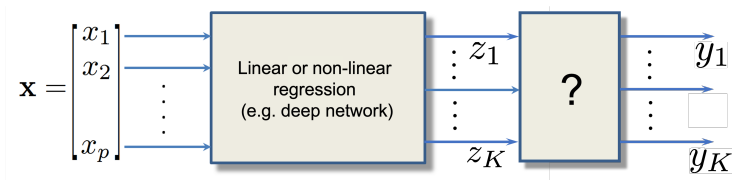


Another View of (and Beyond) Softmax



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

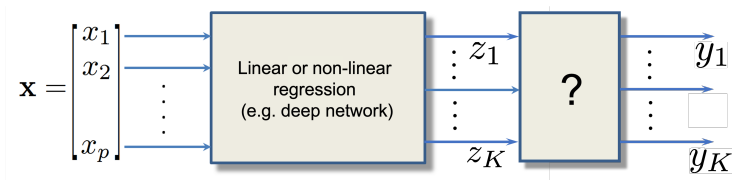
Another View of (and Beyond) Softmax



- **Scores:** $\mathbf{z} \in \mathbb{R}^K$, without constraints/restrictions.
- **Probabilities:** $y_k = \mathbb{P}[\text{class } k | \mathbf{x}]$, thus $\mathbf{y} \in \Delta_{K-1}$, where

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

Another View of (and Beyond) Softmax



- **Scores:** $\mathbf{z} \in \mathbb{R}^K$, without constraints/restrictions.
- **Probabilities:** $y_k = \mathbb{P}[\text{class } k | \mathbf{x}]$, thus $\mathbf{y} \in \Delta_{K-1}$, where

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

- How to map from $\mathbf{z} \in \mathbb{R}^K$ to $\mathbf{y} \in \Delta_{K-1}$, 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 $\mathbf{z}$:

$$\mathbf{y} = \arg \max_{\mathbf{p} \in \Delta_{K-1}} \mathbf{p}^T \mathbf{z} \implies y_k \neq 0 \Leftrightarrow k \in \arg \max_j \{z_j, j = 1, \dots, K\}$$

.

Argmax and Softmax

- First possibility: probability vector “most aligned” with $\mathbf{z}$:

$$\mathbf{y} = \arg \max_{\mathbf{p} \in \Delta_{K-1}} \mathbf{p}^T \mathbf{z} \implies y_k \neq 0 \Leftrightarrow k \in \arg \max_j \{z_j, j = 1, \dots, K\}$$

called the **argmax** operator/mapping.

Argmax and Softmax

- First possibility: probability vector “most aligned” with $\mathbf{z}$:

$$\mathbf{y} = \arg \max_{\mathbf{p} \in \Delta_{K-1}} \mathbf{p}^T \mathbf{z} \implies y_k \neq 0 \Leftrightarrow k \in \arg \max_j \{z_j, j = 1, \dots, K\}$$

called the **argmax** operator/mapping.

- Second possibility: encourage **more uniform** probability distribution:

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

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

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

Argmax and Softmax

- First possibility: probability vector “most aligned” with $\mathbf{z}$:

$$\mathbf{y} = \arg \max_{\mathbf{p} \in \Delta_{K-1}} \mathbf{p}^T \mathbf{z} \implies y_k \neq 0 \Leftrightarrow k \in \arg \max_j \{z_j, j = 1, \dots, K\}$$

called the **argmax** operator/mapping.

- Second possibility: encourage **more uniform** probability distribution:

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

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

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

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

Argmax and Softmax

- First possibility: probability vector “most aligned” with $\mathbf{z}$:

$$\mathbf{y} = \arg \max_{\mathbf{p} \in \Delta_{K-1}} \mathbf{p}^T \mathbf{z} \implies y_k \neq 0 \Leftrightarrow k \in \arg \max_j \{z_j, j = 1, \dots, K\}$$

called the **argmax** operator/mapping.

- Second possibility: encourage **more uniform** probability distribution:

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

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

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

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

Softmax as Maximum Entropy

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

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

Softmax as Maximum Entropy

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

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

- Add **Lagrangian** for the simplex constraint:

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

Softmax as Maximum Entropy

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

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

- Add **Lagrangian** for the simplex constraint:

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

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

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

Softmax as Maximum Entropy

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

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

- Add **Lagrangian** for the simplex constraint:

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

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

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

Softmax as Maximum Entropy

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

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

- Add **Lagrangian** for the simplex constraint:

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

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

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

Softmax as Maximum Entropy

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

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

- Add **Lagrangian** for the simplex constraint:

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

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

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

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

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

Beyond Softmax: Sparsemax

- A third possibility²: simply project z onto Δ_{K-1}

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

²A. Martins and R. Astudillo. “From softmax to sparsemax: A sparse model of attention and multi-label classification”, ICML, 2016.

Beyond Softmax: Sparsemax

- A third possibility²: simply project z onto Δ_{K-1}

$$y = \arg \min_{p \in \Delta_{K-1}} \|p - z\|_2^2 \implies y = \text{sparsemax}(z)$$

- It can also be written as

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

²A. Martins and R. Astudillo. “From softmax to sparsemax: A sparse model of attention and multi-label classification”, ICML, 2016.

Beyond Softmax: Sparsemax

- A third possibility²: simply project z onto Δ_{K-1}

$$y = \arg \min_{p \in \Delta_{K-1}} \|p - z\|_2^2 \implies y = \text{sparsemax}(z)$$

- It can also be written as

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

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

²A. Martins and R. Astudillo. “From softmax to sparsemax: A sparse model of attention and multi-label classification”, ICML, 2016.

Beyond Softmax: Sparsemax

- A third possibility²: simply project z onto Δ_{K-1}

$$y = \arg \min_{p \in \Delta_{K-1}} \|p - z\|_2^2 \implies y = \text{sparsemax}(z)$$

- It can also be written as

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

- $-\|p\|_2^2$ is (up to a constant) a Tsallis entropy.
- General family, where Ω is some entropy,

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

²A. 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: $\mathbf{z}' = \mathbf{z} + \alpha \mathbf{1} \Rightarrow \mathbf{y}' = \mathbf{y}$

Argmax, Softmax, and Sparsemax

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

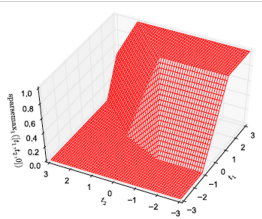
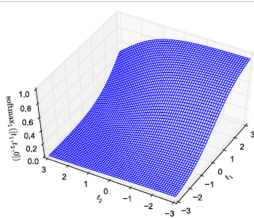
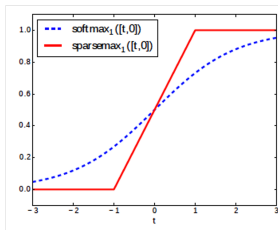
$$\mathbf{z}' = R(\mathbf{z}) \Rightarrow \mathbf{y}' = R(\mathbf{y})$$

Argmax, Softmax, and Sparsemax

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

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

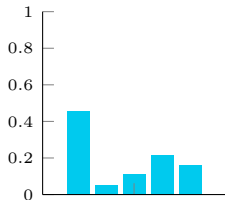
- Sparsemax versus softmax:



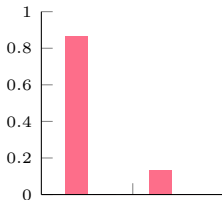
Argmax, Softmax, and Sparsemax

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

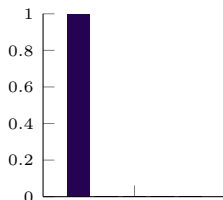
$\text{softmax}(z)$



$\text{sparsemax}(z)$



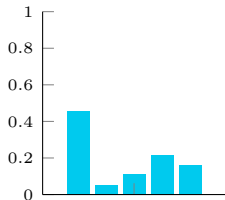
$\text{argmax}(z)$



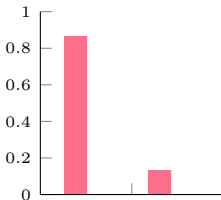
Argmax, Softmax, and Sparsemax

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

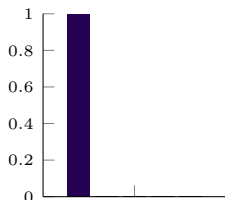
$\text{softmax}(z)$



$\text{sparsemax}(z)$



$\text{argmax}(z)$



- Sparsemax, unlike softmax, may yield exact zeros.

Temperature

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

Temperature

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

$$\lim_{T \rightarrow 0} \text{softmax}(z/T) = \lim_{T \rightarrow 0} \text{sparsemax}(z/T) = \text{argmax}(z)$$

Temperature

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

$$\lim_{T \rightarrow 0} \text{softmax}(z/T) = \lim_{T \rightarrow 0} \text{sparsemax}(z/T) = \text{argmax}(z)$$

- High temperature limit:

$$\lim_{T \rightarrow \infty} \text{softmax}(z/T) = \lim_{T \rightarrow \infty} \text{sparsemax}(z/T) = \left(\frac{1}{K}, \dots, \frac{1}{K} \right)$$

Temperature

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

$$\lim_{T \rightarrow 0} \text{softmax}(z/T) = \lim_{T \rightarrow 0} \text{sparsemax}(z/T) = \text{argmax}(z)$$

- High temperature limit:

$$\lim_{T \rightarrow \infty} \text{softmax}(z/T) = \lim_{T \rightarrow \infty} \text{sparsemax}(z/T) = \left(\frac{1}{K}, \dots, \frac{1}{K} \right)$$

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

Back to Attention: Now With Softmax

- Earlier: **linear** attention can be seen as OLS projection Py .

³A. Martins, R. Astudillo, “From softmax to sparsemax”, ICML, 2016.

Back to Attention: Now With Softmax

- Earlier: **linear** attention can be seen as OLS projection Py .
- **Actual** (scaled dot-product) attention applies a **softmax** to each row:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$

with $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$.

³A. Martins, R. Astudillo, “From softmax to sparsemax”, ICML, 2016.

Back to Attention: Now With Softmax

- Earlier: **linear** attention can be seen as OLS projection Py .
- **Actual** (scaled dot-product) attention applies a **softmax** to each row:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$

with $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$.

- The $\sqrt{d_k}$ is precisely a **temperature**: $\text{softmax}(z/T)$ with $T = \sqrt{d_k}$.

³A. Martins, R. Astudillo, "From softmax to sparsemax", ICML, 2016.

Back to Attention: Now With Softmax

- Earlier: **linear** attention can be seen as OLS projection Py .
- **Actual** (scaled dot-product) attention applies a **softmax** to each row:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$

with $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$.

- The $\sqrt{d_k}$ is precisely a **temperature**: $\text{softmax}(z/T)$ with $T = \sqrt{d_k}$.
- Each output is a **convex combination** of rows of $\mathbf{V}$; low $T \Rightarrow$ peaked, nearly argmax; high $T \Rightarrow$ nearly uniform averaging.

³A. Martins, R. Astudillo, "From softmax to sparsemax", ICML, 2016.

Back to Attention: Now With Softmax

- Earlier: **linear** attention can be seen as OLS projection Py .
- **Actual** (scaled dot-product) attention applies a **softmax** to each row:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$

with $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$.

- The $\sqrt{d_k}$ is precisely a **temperature**: $\text{softmax}(z/T)$ with $T = \sqrt{d_k}$.
- Each output is a **convex combination** of rows of $\mathbf{V}$; low $T \Rightarrow$ peaked, nearly argmax; high $T \Rightarrow$ nearly uniform averaging.
- Replacing **softmax** by **sparsemax**: **sparse attention**: exactly zero weight on some positions³.

³A. Martins, R. Astudillo, "From softmax to sparsemax", ICML, 2016.

Back to Attention: Now With Softmax

- Earlier: **linear** attention can be seen as OLS projection Py .
- **Actual** (scaled dot-product) attention applies a **softmax** to each row:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V},$$

with $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$.

- The $\sqrt{d_k}$ is precisely a **temperature**: $\text{softmax}(z/T)$ with $T = \sqrt{d_k}$.
- Each output is a **convex combination** of rows of $\mathbf{V}$; low $T \Rightarrow$ peaked, nearly argmax; high $T \Rightarrow$ nearly uniform averaging.
- Replacing **softmax** by **sparsemax**: **sparse attention**: exactly zero weight on some positions³.
- Much more in André Martins' lecture on Thursday!



³A. Martins, R. Astudillo, "From softmax to sparsemax", ICML, 2016.

Classification: The Loss Function Perspective

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

Classification: The Loss Function Perspective

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

Classification: The Loss Function Perspective

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

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

Classification: The Loss Function Perspective

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

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

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

Classification: The Loss Function Perspective

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

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

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

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

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**)

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

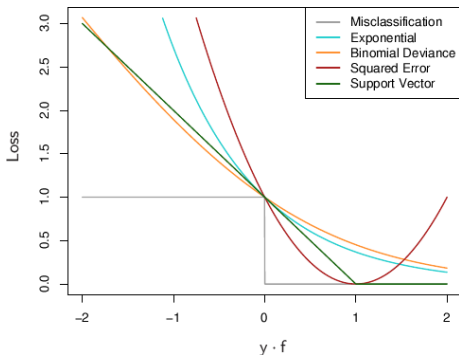
- Naturally, other losses can be used (binomial deviance = logistic):

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**)

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

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



Classification: Empirical and Expected Risk

- The quantity

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

Classification: Empirical and Expected Risk

- The quantity

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

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

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

Classification: Empirical and Expected Risk

- The quantity (empirical risk)

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

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

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

Classification: Empirical and Expected Risk

- The quantity (empirical risk)

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

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

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

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

Classification: Empirical and Expected Risk

- The quantity (empirical risk)

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

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

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

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

The Loss Function Perspective

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

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

loss

The Loss Function Perspective

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

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

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

The Loss Function Perspective

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

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

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

The Loss Function Perspective

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

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

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

The Loss Function Perspective

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

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

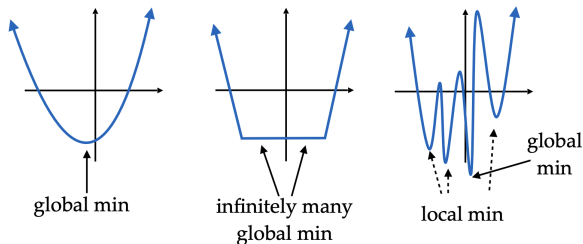
- Quadratic loss**: $L_{\text{quadratic}}(f, y) \propto (f - y)^2$
- Logistic loss**: $L_{\text{logistic}}(f, y) \propto \log(1 + \exp(-y f))$
- Hinge loss**: $L_{\text{hinge}}(f, y) \propto \max\{0, 1 - y f\}$
- Absolute error loss**: $L_{\text{abs}}(f, y) \propto |f - y|$ (not covered today)

Minimizers

- **Goal:** find θ^* , a minimizer of $F(\theta)$ with respect to $\theta \in \mathbb{R}^d$

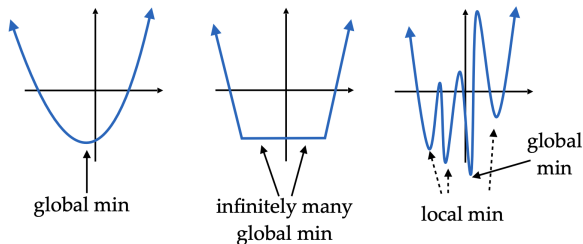
Minimizers

- **Goal:** find θ^* , a minimizer of $F(\theta)$ with respect to $\theta \in \mathbb{R}^d$
- Types of **minimizers**:
 - ✓ **global**, if $F(\theta^*) \leq F(\theta)$, for any $\theta \in \mathbb{R}^d$
 - ✓ **local**, if $F(\theta^*) \leq F(\theta)$, for any $\theta \in \mathbb{R}^d$ s.t. $\|\theta - \theta^*\| \leq \varepsilon$, for some ε .



Minimizers

- **Goal:** find θ^* , a minimizer of $F(\theta)$ with respect to $\theta \in \mathbb{R}^d$
- Types of **minimizers**:
 - ✓ **global**, if $F(\theta^*) \leq F(\theta)$, for any $\theta \in \mathbb{R}^d$
 - ✓ **local**, if $F(\theta^*) \leq F(\theta)$, for any $\theta \in \mathbb{R}^d$ s.t. $\|\theta - \theta^*\| \leq \varepsilon$, for some ε .



- **Minimizers:** global $\Rightarrow$ local; local $\nRightarrow$ global.

Convexity

- F is a **convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in [0, 1] \Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) \leq \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$

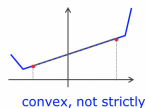
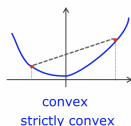
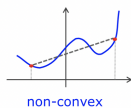
Convexity

- F is a **convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in [0, 1] \Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) \leq \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$

- F is a **strictly convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in]0, 1[\Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) < \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$



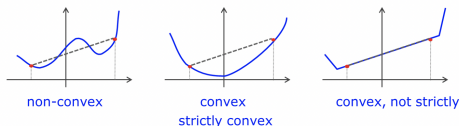
Convexity

- F is a **convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in [0, 1] \Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) \leq \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$

- F is a **strictly convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in]0, 1[\Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) < \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$



- Convexity $\Rightarrow$ local minima = global minima; continuity.

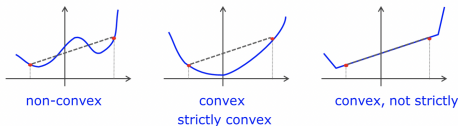
Convexity

- F is a **convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in [0, 1] \Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) \leq \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$

- F is a **strictly convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in]0, 1[\Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) < \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$



- Convexity $\Rightarrow$ local minima = global minima; continuity.

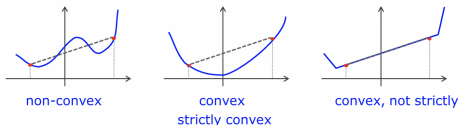
Convexity

- F is a **convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in [0, 1] \Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) \leq \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$

- F is a **strictly convex function** if, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\lambda \in]0, 1[\Rightarrow F(\lambda\theta_1 + (1 - \lambda)\theta_2) < \lambda F(\theta_1) + (1 - \lambda)F(\theta_2)$$



- Convexity $\Rightarrow$ local minima = global minima; continuity.
- Second-order conditions:** if F is twice differentiable, with **Hessian** $H(\theta) = \nabla^2 F(\theta) \in \mathbb{R}^{d \times d}$, $(H(\theta))_{ij} = \frac{\partial^2 F}{\partial \theta_i \partial \theta_j}$,

$$F \text{ convex} \Leftrightarrow H(\theta) \succeq 0 \text{ (psd);}$$

$$F \text{ strictly convex} \Leftrightarrow H(\theta) \succ 0 \text{ (pd)}$$

Descent Directions

- Definition: $\boldsymbol{\eta}$ is a **descent direction** at $\boldsymbol{\theta}_0$ if

$$F(\boldsymbol{\theta}_0 + \alpha \boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \text{ for some } \alpha > 0.$$

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

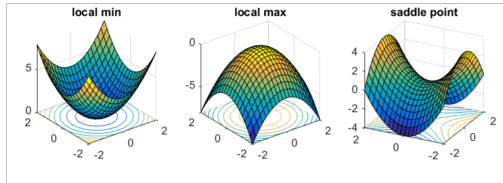
$$F(\boldsymbol{\theta}_0 + \alpha \boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \text{ for some } \alpha > 0.$$

- For differentiable F ,

$$\boldsymbol{\eta}^T \nabla F(\boldsymbol{\theta}_0) < 0 \Leftrightarrow \boldsymbol{\eta} \text{ is a descent direction.}$$

- Thus, for differentiable F ,

$$\boldsymbol{\theta}^* \text{ is a local minimizer} \begin{matrix} \nRightarrow \\ \Rightarrow \end{matrix} \nabla F(\boldsymbol{\theta}^*) = 0$$



Descent Directions

- Definition: $\boldsymbol{\eta}$ is a **descent direction** at $\boldsymbol{\theta}_0$ if

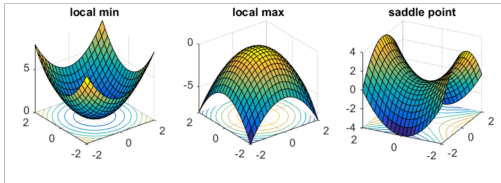
$$F(\boldsymbol{\theta}_0 + \alpha \boldsymbol{\eta}) < F(\boldsymbol{\theta}_0), \text{ for some } \alpha > 0.$$

- For differentiable F ,

$$\boldsymbol{\eta}^T \nabla F(\boldsymbol{\theta}_0) < 0 \Leftrightarrow \boldsymbol{\eta} \text{ is a descent direction.}$$

- Thus, for differentiable F ,

$$\boldsymbol{\theta}^* \text{ is a local minimizer} \begin{matrix} \nRightarrow \\ \Rightarrow \end{matrix} \nabla F(\boldsymbol{\theta}^*) = 0$$



- 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 $\theta_0 \in \mathbb{R}^d$
 - ✓ For $t = 1, 2, \dots$,
 - ▷ choose step-size α_t ,
 - ▷ take a step of size α_t in the direction of the negative gradient:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla F(\theta_{t-1})$$

Gradient Descent

- Key idea: if not at a minimizer, take a step in a descent direction.
- **Gradient descent** algorithm:
 - ✓ Start at some initial point $\theta_0 \in \mathbb{R}^d$
 - ✓ For $t = 1, 2, \dots$,
 - ▷ choose **step-size** α_t ,
 - ▷ take a step of size α_t in the direction of the **negative gradient**:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla F(\theta_{t-1})$$

- 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 $\theta_0 \in \mathbb{R}^d$
 - ✓ For $t = 1, 2, \dots$,
 - ▷ choose **step-size** α_t ,
 - ▷ take a step of size α_t in the direction of the **negative gradient**:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla F(\theta_{t-1})$$

- Several (many) ways to choose α_t ; big research topic.
- Some **stopping criterion** is used; e.g., $\|\nabla F(\theta_t)\| \leq \delta$

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,

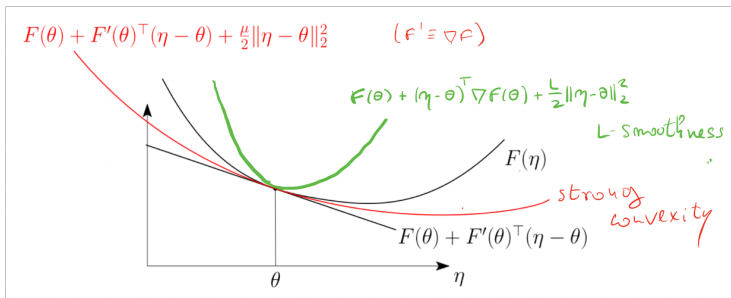
$$F(\boldsymbol{\theta}) \geq F(\boldsymbol{\theta}') + (\boldsymbol{\theta} - \boldsymbol{\theta}')^T \nabla F(\boldsymbol{\theta}') + \frac{\mu}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$$

- If F is twice differentiable, μ -strong convexity $\Leftrightarrow H(\boldsymbol{\theta}) \succeq \mu\mathbf{I}$.

- Condition number $\kappa = \frac{L}{\mu}$.

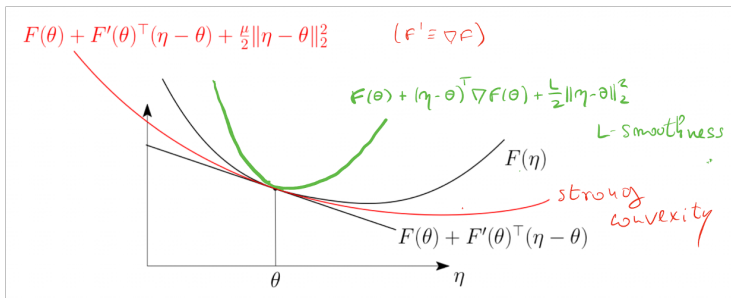
L -smoothness and μ -Strongly Convex

- L -smooth and μ -strongly convex function: upper and lower bounded by quadratics.



L -smoothness and μ -Strongly Convex

- L -smooth and μ -strongly convex function: upper and lower bounded by quadratics.



- Regularization:** if $F(\theta)$ is convex, $F(\theta) + \frac{\mu}{2}\|\theta\|_2^2$ is μ -strongly convex.

Gradient Descent for Convex Functions

- Gradient descent with step-size $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa} \right)^t (F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*))$$

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 $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa} \right)^t (F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*))$$

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),

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \frac{L}{2t} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2$$

called **sub-linear convergence** ($\frac{\Delta_t}{\Delta_{t-1}} \rightarrow 1$)

Gradient Descent for Convex Functions

- Gradient descent with step-size $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa} \right)^t (F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*))$$

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),

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \frac{L}{2t} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2$$

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 $\alpha = 1/L$,

$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \left(\frac{\kappa - 1}{\kappa} \right)^t (F(\boldsymbol{\theta}_0) - F(\boldsymbol{\theta}^*))$$

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),

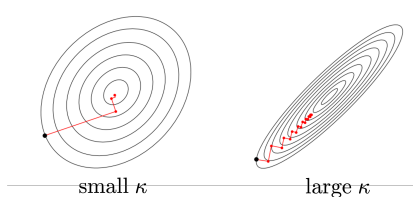
$$F(\boldsymbol{\theta}_t) - F(\boldsymbol{\theta}^*) \leq \frac{L}{2t} \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2$$

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).

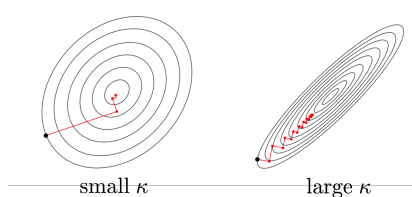
Gradient Descent: Strongly Convex Case

- The **condition number** κ expresses the problem difficulty.

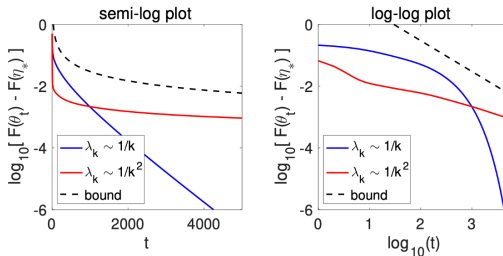


Gradient Descent: Strongly Convex Case

- The **condition number** κ expresses the problem difficulty.

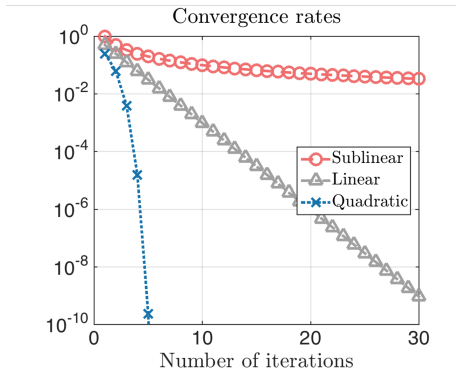


- Convergence for different distributions of eigenvalues.



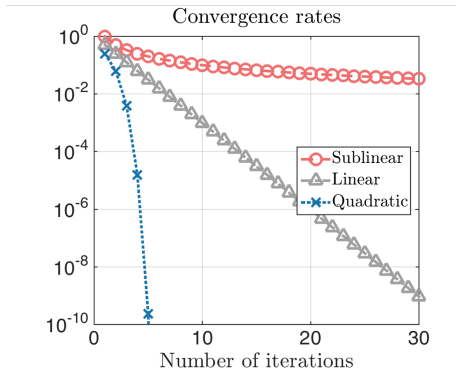
(pictures from F. Bach)

Linear vs Sublinear Convergence



- Quadratic ($\frac{\Delta_t}{\Delta_{t-1}^2} \rightarrow \beta < \infty$) and super-linear ($\frac{\Delta_t}{\Delta_{t-1}} \rightarrow 0$) not achievable using only gradient information.

Linear vs Sublinear Convergence



- Quadratic ($\frac{\Delta_t}{\Delta_{t-1}^2} \rightarrow \beta < \infty$) and super-linear ($\frac{\Delta_t}{\Delta_{t-1}} \rightarrow 0$) 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: $\hat{\theta} = \arg \min_{\theta} F(\theta)$

$$F(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \theta), y_i) \quad (\text{maybe} + R(\theta))$$

Stochastic Gradient “Descent”

- Back to empirical risk minimization: $\hat{\theta} = \arg \min_{\theta} F(\theta)$

$$F(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \theta), y_i) \quad (\text{maybe} + R(\theta))$$

- For large n , computing $\nabla F(\theta)$ is expensive:

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

Stochastic Gradient “Descent”

- Back to empirical risk minimization: $\hat{\theta} = \arg \min_{\theta} F(\theta)$

$$F(\theta) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \theta), y_i) \quad (\text{maybe} + R(\theta))$$

- For large n , computing $\nabla F(\theta)$ is expensive:

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

- Alternative: stochastic gradient “descent” (SGD):

✓ Start at some initial point $\theta_0 \in \mathbb{R}^d$

✓ For $t = 1, 2, \dots$,

▷ sample $i \in \{1, \dots, n\}$ at random and choose step-size α_t ,

▷ take a step of size α_t in the direction of the negative gradient:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla L(f(\mathbf{x}_i; \theta_{t-1}), y_i)$$

Motivation for SGD: Computing a Mean

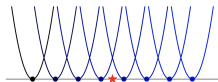
- Consider the goal of computing a mean: $\mu = \frac{1}{n} \sum_{i=1}^n x_i$.

Motivation for SGD: Computing a Mean

- Consider the goal of computing a mean: $\mu = \frac{1}{n} \sum_{i=1}^n x_i$.

- It is well known (prove it) that the mean is the solution of

$$\mu = \arg \min_{\theta \in \mathbb{R}^d} \frac{1}{2} \sum_{i=1}^n \|\theta - x_i\|_2^2$$

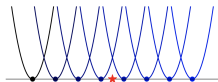


Motivation for SGD: Computing a Mean

- Consider the goal of computing a mean: $\mu = \frac{1}{n} \sum_{i=1}^n x_i$.

- It is well known (prove it) that the mean is the solution of

$$\mu = \arg \min_{\theta \in \mathbb{R}^d} \frac{1}{2} \sum_{i=1}^n \|\theta - x_i\|_2^2$$



- Let's use "SGD": $L(x_i, \theta) = \frac{1}{2} \|\theta - x_i\|_2^2$, thus $\nabla L(x_i, \theta) = \theta - x_i$

✓ Set initial point $\theta_0 = 0$

✓ For $t = 1, 2, \dots$,

▷ take $i \in \{1, \dots, n\}$ **sequentially** ($i = t$) and use **step-size** $\alpha_t = 1/t$,

▷ take a step of size α_t in the direction of the **negative gradient**:

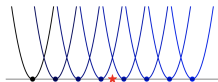
$$\theta_t = \theta_{t-1} - \frac{1}{t} (\theta_{t-1} - x_t) = \frac{t-1}{t} \theta_{t-1} + \frac{1}{t} x_t$$

Motivation for SGD: Computing a Mean

- Consider the goal of computing a mean: $\mu = \frac{1}{n} \sum_{i=1}^n x_i$.

- It is well known (prove it) that the mean is the solution of

$$\mu = \arg \min_{\theta \in \mathbb{R}^d} \frac{1}{2} \sum_{i=1}^n \|\theta - x_i\|_2^2$$



- Let's use "SGD": $L(x_i, \theta) = \frac{1}{2} \|\theta - x_i\|_2^2$, thus $\nabla L(x_i, \theta) = \theta - x_i$

✓ Set initial point $\theta_0 = 0$

✓ For $t = 1, 2, \dots$,

▷ take $i \in \{1, \dots, n\}$ **sequentially** ($i = t$) and use **step-size** $\alpha_t = 1/t$,

▷ take a step of size α_t in the direction of the **negative gradient**:

$$\theta_t = \theta_{t-1} - \frac{1}{t}(\theta_{t-1} - x_t) = \frac{t-1}{t} \theta_{t-1} + \frac{1}{t} x_t$$

- Notice that $(t-1)\theta_{t-1} = \sum_{i=1}^{t-1} x_i$, thus $\theta_t = \frac{1}{t} \sum_{i=1}^t x_i$

Motivation: Computing an Expected Value

- Goal: computing an **expectation**,

$$\mu = \mathbb{E}[X] = \arg \min_w \frac{1}{2} \overbrace{\mathbb{E}[(w - X)^2]}^{R(w)}$$

Motivation: Computing an Expected Value

- Goal: computing an **expectation**,

$$\mu = \mathbb{E}[X] = \arg \min_w \overbrace{\frac{1}{2} \mathbb{E}[(w - X)^2]}^{R(w)}$$

- SGD** with i.i.d. samples X_i , for $i = 1, 2, \dots, n$, and **step-size** $\alpha_t = \frac{1}{t}$,

$$W_n = W_{n-1} + \alpha_t(W_{t-1} - X_t) = \frac{1}{n} \sum_{i=1}^n X_i \quad (\text{random sequence})$$

Motivation: Computing an Expected Value

- Goal: computing an **expectation**,

$$\mu = \mathbb{E}[X] = \arg \min_w \overbrace{\frac{1}{2} \mathbb{E}[(w - X)^2]}^{R(w)}$$

- SGD** with i.i.d. samples X_i , for $i = 1, 2, \dots, n$, and **step-size** $\alpha_t = \frac{1}{t}$,

$$W_n = W_{n-1} + \alpha_t(W_{t-1} - X_t) = \frac{1}{n} \sum_{i=1}^n X_i \quad (\text{random sequence})$$

- Expected cost** (assuming variance σ^2),

$$\mathbb{E}[R(W_n)] = \frac{1}{2} \mathbb{E} \left[\left(\frac{1}{n} \sum_{i=1}^n X_i - X \right)^2 \right] = \frac{\sigma^2}{2} \left(\frac{n+1}{n} \right)$$

Motivation: Computing an Expected Value

- Goal: computing an **expectation**,

$$\mu = \mathbb{E}[X] = \arg \min_w \overbrace{\frac{1}{2} \mathbb{E}[(w - X)^2]}^{R(w)}$$

- SGD** with i.i.d. samples X_i , for $i = 1, 2, \dots, n$, and **step-size** $\alpha_t = \frac{1}{t}$,

$$W_n = W_{n-1} + \alpha_t(W_{t-1} - X_t) = \frac{1}{n} \sum_{i=1}^n X_i \quad (\text{random sequence})$$

- Expected cost** (assuming variance σ^2),

$$\mathbb{E}[R(W_n)] = \frac{1}{2} \mathbb{E} \left[\left(\frac{1}{n} \sum_{i=1}^n X_i - X \right)^2 \right] = \frac{\sigma^2}{2} \left(\frac{n+1}{n} \right)$$

- Optimal cost**, for $w^* = \mu$, is $R(\mu) = \frac{1}{2} \mathbb{E}[(\mu - X)^2] = \frac{\sigma^2}{2}$

Motivation: Computing an Expected Value

- Goal: computing an **expectation**,

$$\mu = \mathbb{E}[X] = \arg \min_w \overbrace{\frac{1}{2} \mathbb{E}[(w - X)^2]}^{R(w)}$$

- SGD** with i.i.d. samples X_i , for $i = 1, 2, \dots, n$, and **step-size** $\alpha_t = \frac{1}{t}$,

$$W_n = W_{n-1} + \alpha_t(W_{t-1} - X_t) = \frac{1}{n} \sum_{i=1}^n X_i \quad (\text{random sequence})$$

- Expected cost** (assuming variance σ^2),

$$\mathbb{E}[R(W_n)] = \frac{1}{2} \mathbb{E} \left[\left(\frac{1}{n} \sum_{i=1}^n X_i - X \right)^2 \right] = \frac{\sigma^2}{2} \left(\frac{n+1}{n} \right)$$

- Optimal cost**, for $w^* = \mu$, is $R(\mu) = \frac{1}{2} \mathbb{E}[(\mu - X)^2] = \frac{\sigma^2}{2}$

- Optimality gap**: $\mathbb{E}[R(W_n) - R(\mu)] = \frac{\sigma^2}{2n}$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, Y}[L(f(\mathbf{X}; \boldsymbol{\theta}), Y)]$.

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, Y}[L(f(\mathbf{X}; \boldsymbol{\theta}), Y)]$.
- To do gradient descent, we need

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

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, Y}[L(f(\mathbf{X}; \boldsymbol{\theta}), Y)]$.
- To do gradient descent, we need

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

- Thus, $\nabla L(f(\mathbf{X}; \boldsymbol{\theta}), Y)$ is an unbiased estimate of $\nabla \mathcal{R}(\boldsymbol{\theta})$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, Y}[L(f(\mathbf{X}; \boldsymbol{\theta}), Y)]$.
- To do gradient descent, we need

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

- Thus, $\nabla L(f(\mathbf{X}; \boldsymbol{\theta}), Y)$ is an **unbiased estimate** of $\nabla \mathcal{R}(\boldsymbol{\theta})$
- SGD with samples from $f_{\mathbf{X}, Y}$ is a sequence of **random variables**,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla L(f(\mathbf{X}; \boldsymbol{\theta}_t), Y)$$

that is, **in expectation**,

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}_{t+1}] &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \mathbb{E}[\nabla L(f(\mathbf{X}; \boldsymbol{\theta}_t), Y)] \\ &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \nabla \mathcal{R}(\boldsymbol{\theta}_t)\end{aligned}$$

Stochastic Gradient Descent

- Expected loss (risk): $F(\boldsymbol{\theta}) = \mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{X}, Y}[L(f(\mathbf{X}; \boldsymbol{\theta}), Y)]$.
- To do gradient descent, we need

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

- Thus, $\nabla L(f(\mathbf{X}; \boldsymbol{\theta}), Y)$ is an **unbiased estimate** of $\nabla \mathcal{R}(\boldsymbol{\theta})$
- SGD with samples from $f_{\mathbf{X}, Y}$ is a sequence of **random variables**,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla L(f(\mathbf{X}; \boldsymbol{\theta}_t), Y)$$

that is, **in expectation**,

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}_{t+1}] &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \mathbb{E}[\nabla L(f(\mathbf{X}; \boldsymbol{\theta}_t), Y)] \\ &= \mathbb{E}[\boldsymbol{\theta}_t] - \alpha_t \nabla \mathcal{R}(\boldsymbol{\theta}_t)\end{aligned}$$

- In expectation**, SGD by sampling $f_{\mathbf{X}, Y}$ is **gradient descent** on $\mathcal{R}(\boldsymbol{\theta})$.

Convergence of SGD: Convex vs Strongly Convex

- Assume bounded noisy gradients, $\|G(\theta)\|_2^2 \leq B^2$; guarantees are on the **averaged iterates** $\bar{\theta}_t$, *not* on the last (or best) one.

Convex

$$\|\theta_0 - \theta^*\|_2 \leq D$$

$$\text{Step size: } \alpha_t = \frac{D}{B\sqrt{t}}$$

$$\bar{\theta}_t = \frac{\sum_s \alpha_s \theta_{s-1}}{\sum_s \alpha_s}$$

$$\frac{\mathbb{E}[F(\bar{\theta}_t) - F(\theta^*)]}{2\sqrt{t}} \leq$$

$$O(1/\sqrt{t})$$

μ -strongly convex

$$\text{e.g., add } \frac{\mu}{2} \|\theta\|_2^2$$

$$\text{Step size: } \alpha_t = \frac{1}{\mu t}$$

$$\bar{\theta}_t = \frac{1}{t} \sum_s \theta_{s-1}$$

$$\frac{\mathbb{E}[F(\bar{\theta}_t) - F(\theta^*)]}{\mu t} \leq$$

$$O(1/t)$$

Convergence of SGD: Convex vs Strongly Convex

- Assume bounded noisy gradients, $\|G(\theta)\|_2^2 \leq B^2$; guarantees are on the **averaged iterates** $\bar{\theta}_t$, *not* on the last (or best) one.

Convex

$$\|\theta_0 - \theta^*\|_2 \leq D$$

$$\text{Step size: } \alpha_t = \frac{D}{B\sqrt{t}}$$

$$\bar{\theta}_t = \frac{\sum_s \alpha_s \theta_{s-1}}{\sum_s \alpha_s}$$

$$\frac{\mathbb{E}[F(\bar{\theta}_t) - F(\theta^*)]}{2\sqrt{t}} \leq$$

$$O(1/\sqrt{t})$$

μ -strongly convex

$$\text{e.g., add } \frac{\mu}{2} \|\theta\|_2^2$$

$$\text{Step size: } \alpha_t = \frac{1}{\mu t}$$

$$\bar{\theta}_t = \frac{1}{t} \sum_s \theta_{s-1}$$

$$\frac{\mathbb{E}[F(\bar{\theta}_t) - F(\theta^*)]}{\mu t} \leq$$

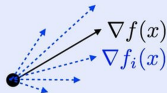
$$O(1/t)$$

- Note:** computing $F(\theta_t)$ is impractical; picking the best iterate would defeat the purpose of SGD.

SGD: From Robbins–Monro to Modern ML

Finite sums

$$f(x) \stackrel{\text{def.}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$
$$\nabla f(x) = \frac{1}{n} \sum_i \nabla f_i(x)$$



Draw $i \in \{1, \dots, n\}$ uniformly.

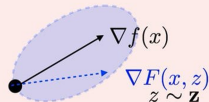
$$x_{k+1} = x_k - \tau_k \nabla f_i(x_k)$$



Herbert Robbins

Expectation

$$f(x) \stackrel{\text{def.}}{=} \mathbb{E}_{\mathbf{z}}(f(x, \mathbf{z}))$$
$$\nabla f(x) = \mathbb{E}_{\mathbf{z}}(\nabla F(x, \mathbf{z}))$$



Draw $z \sim \mathbf{z}$

$$x_{k+1} = x_k - \tau_k \nabla F(x, z)$$

Theorem: If f is strongly convex and $\tau_k \sim 1/k$,
$$\mathbb{E}(\|x_k - x^*\|^2) = O(1/k)$$

(picture by Gabriel Peyré)

Robbins & Monro, “A stochastic approximation method”, *Annals of Mathematical Statistics*, 1951.

Bottou & Bousquet, “The tradeoffs of large-scale learning”, *NeurIPS*, 2007

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) .

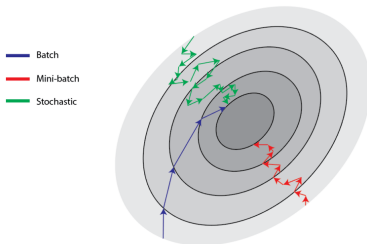
Tricks of the Trade

- 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)

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\alpha_t}{m} \sum_{j \in \text{minibatch } t} \nabla L(f(\mathbf{x}_j; \boldsymbol{\theta}_{t-1}), y_j)$$



Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$
- 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: $L(f(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$
- 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,

$$\nabla \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) = \left. \frac{d\ell(u)}{du} \right|_{u=y_i \boldsymbol{\theta}^T \mathbf{x}_i} \nabla (y_i \boldsymbol{\theta}^T \mathbf{x}_i)$$

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$
- 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,

$$\nabla \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) = \left. \frac{d\ell(u)}{du} \right|_{u=y_i \boldsymbol{\theta}^T \mathbf{x}_i} \nabla(y_i \boldsymbol{\theta}^T \mathbf{x}_i) = \left(\left. \frac{d\ell(u)}{du} \right|_{u=y_i \boldsymbol{\theta}^T \mathbf{x}_i} y_i \right) \mathbf{x}_i$$

showing that $\nabla \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$ is co-linear with $\mathbf{x}_i$.

Stochastic Gradient Descent: Linear Classification

- Linear predictor with margin loss: $L(f(\mathbf{x}_i; \boldsymbol{\theta}_{t-1}), y_i) = \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$
- 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,

$$\nabla \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i) = \left. \frac{d\ell(u)}{du} \right|_{u=y_i \boldsymbol{\theta}^T \mathbf{x}_i} \nabla (y_i \boldsymbol{\theta}^T \mathbf{x}_i) = \left(\left. \frac{d\ell(u)}{du} \right|_{u=y_i \boldsymbol{\theta}^T \mathbf{x}_i} y_i \right) \mathbf{x}_i$$

showing that $\nabla \ell(y_i \boldsymbol{\theta}^T \mathbf{x}_i)$ is co-linear with $\mathbf{x}_i$.

- Each SGD update moves $\boldsymbol{\theta}_t$ in a direction parallel to sample $\mathbf{x}_i$.

The Perceptron Algorithm

- Hinge loss: $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\ell(u)}{du} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

ignoring the non-differentiability at $u = \tau$.

The Perceptron Algorithm

- **Hinge loss:** $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\ell(u)}{du} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

ignoring the non-differentiability at $u = \tau$.

- Each iteration of SGD, with constant step size α , choose sample i ,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \begin{cases} y_i \mathbf{x}_i & \text{if } y_i \boldsymbol{\theta}_t^T \mathbf{x}_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

The Perceptron Algorithm

- **Hinge loss**: $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\ell(u)}{du} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

ignoring the non-differentiability at $u = \tau$.

- Each iteration of SGD, with constant step size α , choose sample i ,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \begin{cases} y_i \mathbf{x}_i & \text{if } y_i \boldsymbol{\theta}_t^T \mathbf{x}_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

- Points with **wrong classification** ($y_i \boldsymbol{\theta}_t^T \mathbf{x}_i < 0$) or **insufficient margin** ($y_i \boldsymbol{\theta}_t^T \mathbf{x}_i \leq \tau$) move $\boldsymbol{\theta}_t$ **towards/away** from $\mathbf{x}_i$ depending on y_i

The Perceptron Algorithm

- **Hinge loss**: $\ell(u) = \max\{0, 1 - \tau\}$, thus

$$\frac{d\ell(u)}{du} = \begin{cases} -1, & \text{if } u \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

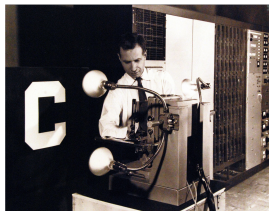
ignoring the non-differentiability at $u = \tau$.

- Each iteration of SGD, with constant step size α , choose sample i ,

$$\theta_{t+1} = \theta_t + \alpha \begin{cases} y_i x_i & \text{if } y_i \theta_t^T x_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

- Points with **wrong classification** ($y_i \theta_t^T x_i < 0$) or **insufficient margin** ($y_i \theta_t^T x_i \leq \tau$) move θ_t **towards/away** from x_i depending on y_i
- This is the famous **Perceptron algorithm**, proposed in 1957 by Frank Rosenblatt (with $\tau = 0$), the precursor 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

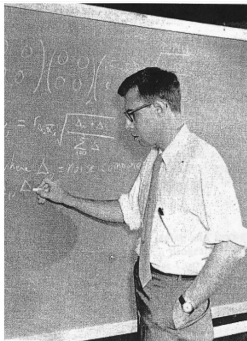
WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

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.

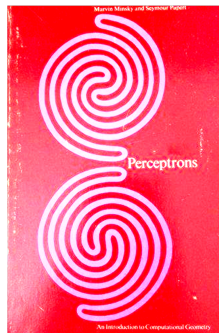
The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

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.

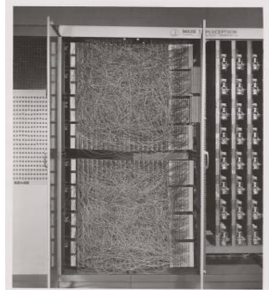
Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fitted to the planets as mechanical space explorers.



The New York Times, 1958



Minsky and Papert, 1969



Perceptron Mistake Bound

- Definitions:

- ✓ The training data is **linearly separable** with **margin** $\gamma > 0$ iff there is a weight vector $\mathbf{u}$, with $\|\mathbf{u}\| = 1$, such that

$$y_n \mathbf{u}^T \mathbf{x}_n \geq \gamma, \quad \forall n.$$

⁴A. 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 $\mathbf{u}$, with $\|\mathbf{u}\| = 1$, such that

$$y_n \mathbf{u}^T \mathbf{x}_n \geq \gamma, \quad \forall n.$$

- ✓ **Radius** of the data: $R = \max_n \|\mathbf{x}_n\|$.

⁴A. 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 $\mathbf{u}$, with $\|\mathbf{u}\| = 1$, such that

$$y_n \mathbf{u}^T \mathbf{x}_n \geq \gamma, \quad \forall n.$$

- ✓ **Radius** of the data: $R = \max_n \|\mathbf{x}_n\|$.

- Then, the following bound of the **number of mistakes** holds⁴

Theorem

The perceptron algorithm is guaranteed to find a separating hyperplane after at most $\frac{R^2}{\gamma^2}$ mistakes (non-zero updates).

⁴A. 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: $\theta_{t+1} = \theta_t + y_i x_i$.

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\theta_{t+1} = \theta_t + y_i x_i$.
- Lower bound on $\|\theta_t\|$, after M mistakes:

$$\begin{aligned} \mathbf{u}^T \theta_t &= \mathbf{u}^T \theta_{t-1} + y_i \mathbf{u}^T x_i \\ &\geq \mathbf{u}^T \theta_{t-1} + \gamma \\ &\geq \mathbf{u}^T \theta_0 + M \gamma = M \gamma \quad (\text{recall } \theta_0 = 0) \end{aligned}$$

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\theta_{t+1} = \theta_t + y_i x_i$.
- Lower bound on $\|\theta_t\|$, after M mistakes:

$$\begin{aligned} u^T \theta_t &= u^T \theta_{t-1} + y_i u^T x_i \\ &\geq u^T \theta_{t-1} + \gamma \\ &\geq u^T \theta_0 + M \gamma = M \gamma \quad (\text{recall } \theta_0 = 0) \end{aligned}$$

$$\text{Thus, } \|\theta_t\| = \underbrace{\|u\|}_1 \|\theta_t\| \geq u^T \theta_t \geq M \gamma \quad (\text{Cauchy-Schwarz})$$

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\theta_{t+1} = \theta_t + y_i x_i$.
- Lower bound on $\|\theta_t\|$, after M mistakes:

$$\begin{aligned} u^T \theta_t &= u^T \theta_{t-1} + y_i u^T x_i \\ &\geq u^T \theta_{t-1} + \gamma \\ &\geq u^T \theta_0 + M \gamma = M \gamma \quad (\text{recall } \theta_0 = 0) \end{aligned}$$

Thus, $\|\theta_t\| = \underbrace{\|u\|}_1 \|\theta_t\| \geq u^T \theta_t \geq M \gamma$ (Cauchy-Schwarz)

- Upper bound on $\|\theta_t\|$:

$$\begin{aligned} \|\theta_t\|^2 &= \|\theta_{t-1}\|^2 + \|x_i\|^2 + 2 \overbrace{y_i \theta_{t-1}^T x_i}^{\leq 0, \text{ if mistake}} \\ &\leq \|\theta_{t-1}\|^2 + R^2 \\ &\leq M R^2 \end{aligned}$$

Novikoff's Theorem: One-Slide Proof

- Recall that non-zero updates (mistakes) are: $\theta_{t+1} = \theta_t + y_i x_i$.
- Lower bound on $\|\theta_t\|$, after M mistakes:

$$\begin{aligned} u^T \theta_t &= u^T \theta_{t-1} + y_i u^T x_i \\ &\geq u^T \theta_{t-1} + \gamma \\ &\geq u^T \theta_0 + M \gamma = M \gamma \quad (\text{recall } \theta_0 = 0) \end{aligned}$$

Thus, $\|\theta_t\| = \underbrace{\|u\|}_1 \|\theta_t\| \geq u^T \theta_t \geq M \gamma$ (Cauchy-Schwarz)

- Upper bound on $\|\theta_t\|$:

$$\begin{aligned} \|\theta_t\|^2 &= \|\theta_{t-1}\|^2 + \|x_i\|^2 + 2 \overbrace{y_i \theta_{t-1}^T x_i}^{\leq 0, \text{ if mistake}} \\ &\leq \|\theta_{t-1}\|^2 + R^2 \\ &\leq M R^2 \end{aligned}$$

- Equating both sides, $(M\gamma)^2 \leq \|\theta_t\|^2 \leq M R^2 \Rightarrow M \leq R^2 / \gamma^2$ 

Implicit Regularization

- SGD in **linear prediction**, with i_t denoting the sample at iteration t ,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t e_{i_t} \mathbf{x}_{i_t}$$

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 ,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t e_{i_t} \mathbf{x}_{i_t}$$

where e_{i_t} depends on the loss gradient and label y_{i_t} .

- Minibatch or full batch **gradient descent**:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \mathbf{x}_j$$

Implicit Regularization

- SGD in **linear prediction**, with i_t denoting the sample at iteration t ,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t e_{i_t} \mathbf{x}_{i_t}$$

where e_{i_t} depends on the loss gradient and label y_{i_t} .

- Minibatch or full batch **gradient descent**:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \mathbf{x}_j$$

- Initializing at $\boldsymbol{\theta}_0 = 0 \Rightarrow \boldsymbol{\theta}_t \in \text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$.

Implicit Regularization

- SGD in **linear prediction**, with i_t denoting the sample at iteration t ,

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t e_{i_t} \mathbf{x}_{i_t}$$

where e_{i_t} depends on the loss gradient and label y_{i_t} .

- Minibatch or full batch **gradient descent**:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \sum_{j \in B_t} e_j \mathbf{x}_j$$

- Initializing at $\boldsymbol{\theta}_0 = \mathbf{0}$** $\Rightarrow \boldsymbol{\theta}_t \in \text{span}(\mathbf{x}_1, \dots, \mathbf{x}_n)$.
- In the **overparametrized/interpolating** regime, $F(\boldsymbol{\theta}^*) = 0$ has many solutions; the predictions only depend on $\boldsymbol{\theta}^T \mathbf{x}_i$, thus we are solving

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2, \quad \text{such that } L(\boldsymbol{\theta}^T \mathbf{x}_i, y_i) = 0, \quad \text{for } i = 1, \dots, n.$$

- Recall representer theorem**: there $\boldsymbol{\theta} \in \text{span}\{\mathbf{x}_i\}$ came from an *explicit* penalty; here it comes **implicitly** from GD started at $\mathbf{0}$.

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 $F(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2$
- Let $\mathbf{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk
- Gradient of the regularizer: $\lambda \boldsymbol{\theta}$
- Gradient descent (batch or stochastic):

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t (\mathbf{g}(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1})$$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2$
- Let $\mathbf{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk
- Gradient of the regularizer: $\lambda \boldsymbol{\theta}$
- Gradient descent (batch or stochastic):

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha_t (\mathbf{g}(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}) \\ &= (1 - \lambda \alpha_t) \boldsymbol{\theta}_{t-1} - \alpha_t \mathbf{g}(\boldsymbol{\theta}_{t-1})\end{aligned}$$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2$
- Let $\mathbf{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk
- Gradient of the regularizer: $\lambda \boldsymbol{\theta}$
- Gradient descent (batch or stochastic):

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha_t (\mathbf{g}(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}) \\ &= (1 - \lambda \alpha_t) \boldsymbol{\theta}_{t-1} - \alpha_t \mathbf{g}(\boldsymbol{\theta}_{t-1})\end{aligned}$$

- For α_t and λ small enough, $0 < (1 - \lambda \alpha_t) < 1$

Explicit Regularization: Weight Decay

- Objective function $F(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2$
- Let $\mathbf{g}(\boldsymbol{\theta})$ be a (batch or stochastic) gradient of the empirical risk
- Gradient of the regularizer: $\lambda \boldsymbol{\theta}$
- Gradient descent (batch or stochastic):

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \alpha_t (\mathbf{g}(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}) \\ &= (1 - \lambda \alpha_t) \boldsymbol{\theta}_{t-1} - \alpha_t \mathbf{g}(\boldsymbol{\theta}_{t-1})\end{aligned}$$

- 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

Momentum

- **Momentum**: remember the previous step, combine it in the update:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \mathbf{g}(\boldsymbol{\theta}_{t-1}) + \gamma_t (\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_{t-2});$$

$\mathbf{g}(\boldsymbol{\theta}_t)$ is the gradient estimate (batch, single sample, minibatch).

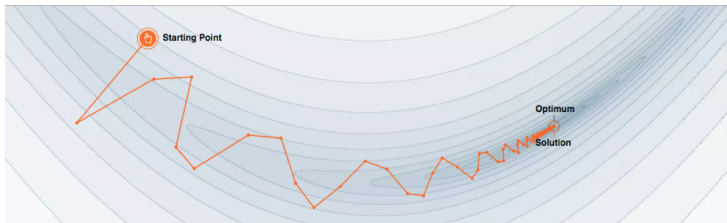
Momentum

- **Momentum**: remember the previous step, combine it in the update:

$$\theta_t = \theta_{t-1} - \alpha_t g(\theta_{t-1}) + \gamma_t (\theta_{t-1} - \theta_{t-2});$$

$g(\theta_t)$ 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 Step Sizes: AdaGrad, RMSProp, Adam

- **Common idea:** a *separate* step size per component of θ , obtained by dividing α by a running estimate of the gradient magnitude:

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}} g_j(\theta_{t-1})$$

Adaptive Step Sizes: AdaGrad, RMSProp, Adam

- **Common idea:** a *separate* step size per component of θ , obtained by dividing α by a running estimate of the gradient magnitude:

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}} g_j(\theta_{t-1})$$

- They differ *only* in how $G_{j,t}$ accumulates:

Method	Accumulator $G_{j,t}$	Behaviour
AdaGrad	$G_{j,t-1} + (g_j(\theta_t))^2$	step size vanishes
RMSProp	$\gamma G_{j,t-1} + (1 - \gamma)(g_j(\theta_t))^2$	forgets ($\gamma \approx 0.9$)
Adam	RMSProp, plus momentum	averages g and g^2

Adaptive Step Sizes: AdaGrad, RMSProp, Adam

- **Common idea**: a *separate* step size per component of θ , obtained by dividing α by a running estimate of the gradient magnitude:

$$\theta_{j,t} = \theta_{j,t-1} - \frac{\alpha}{\sqrt{G_{j,t-1} + \varepsilon}} g_j(\theta_{t-1})$$

- They differ *only* in how $G_{j,t}$ accumulates:

Method	Accumulator $G_{j,t}$	Behaviour
AdaGrad	$G_{j,t-1} + (g_j(\theta_t))^2$	step size vanishes
RMSProp	$\gamma G_{j,t-1} + (1 - \gamma)(g_j(\theta_t))^2$	forgets ($\gamma \approx 0.9$)
Adam	RMSProp, plus momentum	averages g and g^2

- **Adam** also *bias-corrects* both averages (they start at **0**); by far the most used in **deep learning**. (Kingma & Ba, 2015; ~250K citations)

Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.

Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.
- ... yet the **algorithms are the same**: SGD, momentum, Adam are applied *unchanged*. There is no separate “non-convex optimizer”.

Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.
- ... yet the **algorithms are the same**: SGD, momentum, Adam are applied *unchanged*. There is no separate “non-convex optimizer”.

Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.
- ... yet the **algorithms are the same**: SGD, momentum, Adam are applied *unchanged*. There is no separate “non-convex optimizer”.
- **Practice inherits from theory**: step-size decay, averaging, momentum, preconditioning: all born in the convex setting, all still standard.

Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.
- ... yet the **algorithms are the same**: SGD, momentum, Adam are applied *unchanged*. There is no separate “non-convex optimizer”.
- **Practice inherits from theory**: step-size decay, averaging, momentum, preconditioning: all born in the convex setting, all still standard.
- The **last layer is convex**: with the representation fixed, training the final linear+softmax layer is *exactly* the problem in this lecture.

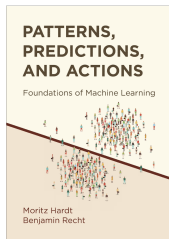
Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.
- ... yet the **algorithms are the same**: SGD, momentum, Adam are applied *unchanged*. There is no separate “non-convex optimizer”.
- **Practice inherits from theory**: step-size decay, averaging, momentum, preconditioning: all born in the convex setting, all still standard.
- The **last layer is convex**: with the representation fixed, training the final linear+**softmax** layer is *exactly* the problem in this lecture.
- **Diagnostics**: when training fails, the theory says *which* assumption broke (smoothness? bounded gradients? conditioning?).

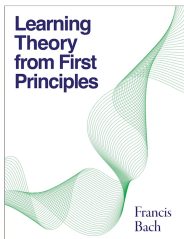
Convex Optimization in 2026: Why Bother?

- **Objection**: deep learning objectives are **severely non-convex**; almost nothing above applies! **True**.
- ... yet the **algorithms are the same**: SGD, momentum, Adam are applied *unchanged*. There is no separate “non-convex optimizer”.
- **Practice inherits from theory**: step-size decay, averaging, momentum, preconditioning: all born in the convex setting, all still standard.
- The **last layer is convex**: with the representation fixed, training the final linear+**softmax** layer is *exactly* the problem in this lecture.
- **Diagnostics**: when training fails, the theory says *which* assumption broke (smoothness? bounded gradients? conditioning?).
- **Finally**: the implicit regularization of GD and SGM provide insight on why deep learning works.

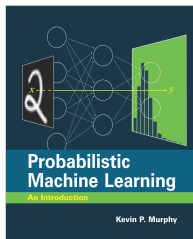
Recommended Books



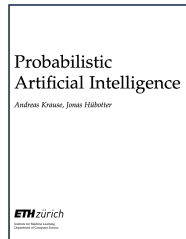
<https://mlstory.org/>
2022



www.di.ens.fr/~fbach/lftp_book.pdf
2024

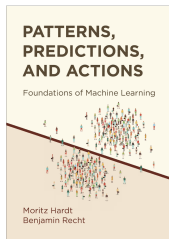


probml.github.io/pml-book
2022

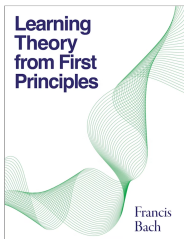


arxiv.org/abs/2502.05244
2025

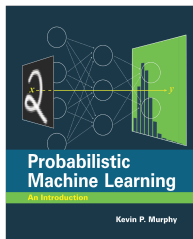
Recommended Books



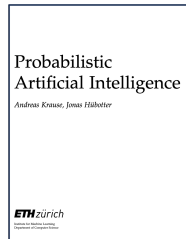
<https://mlstory.org/>
2022



www.di.ens.fr/~fbach/lftp_book.pdf
2024



probml.github.io/pml-book
2022



arxiv.org/abs/2502.05244
2025

Thank you! Questions?