COMP0197
Applied Deep Learning

Andre Altmann

Department of Medical Physics and Biomedical Engineering

Centre for Medical Image Computing (CMIC)

a.altmann@ucl.ac.uk

London, January 11th 2024

My Background

Master's in **Computer Science**

- Speech and Language technologies
- Machine learning

PhD in **Computational Biology**

- Machine Learning … applied to biological problems

Associate Professor in **Medical Physics Department**

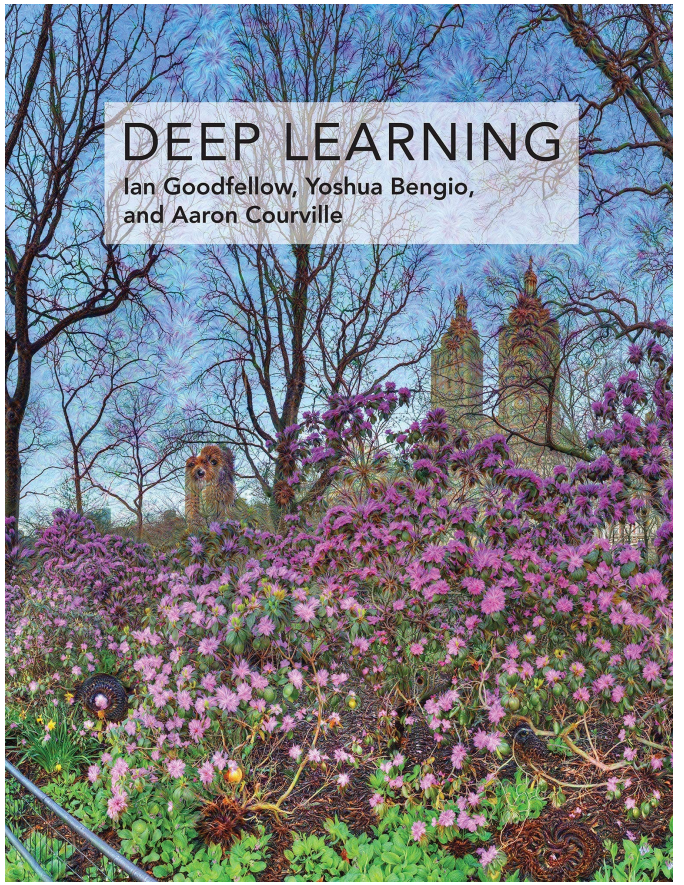
- Linking medical imaging data to genetics etc. …
- … using Machine Learning

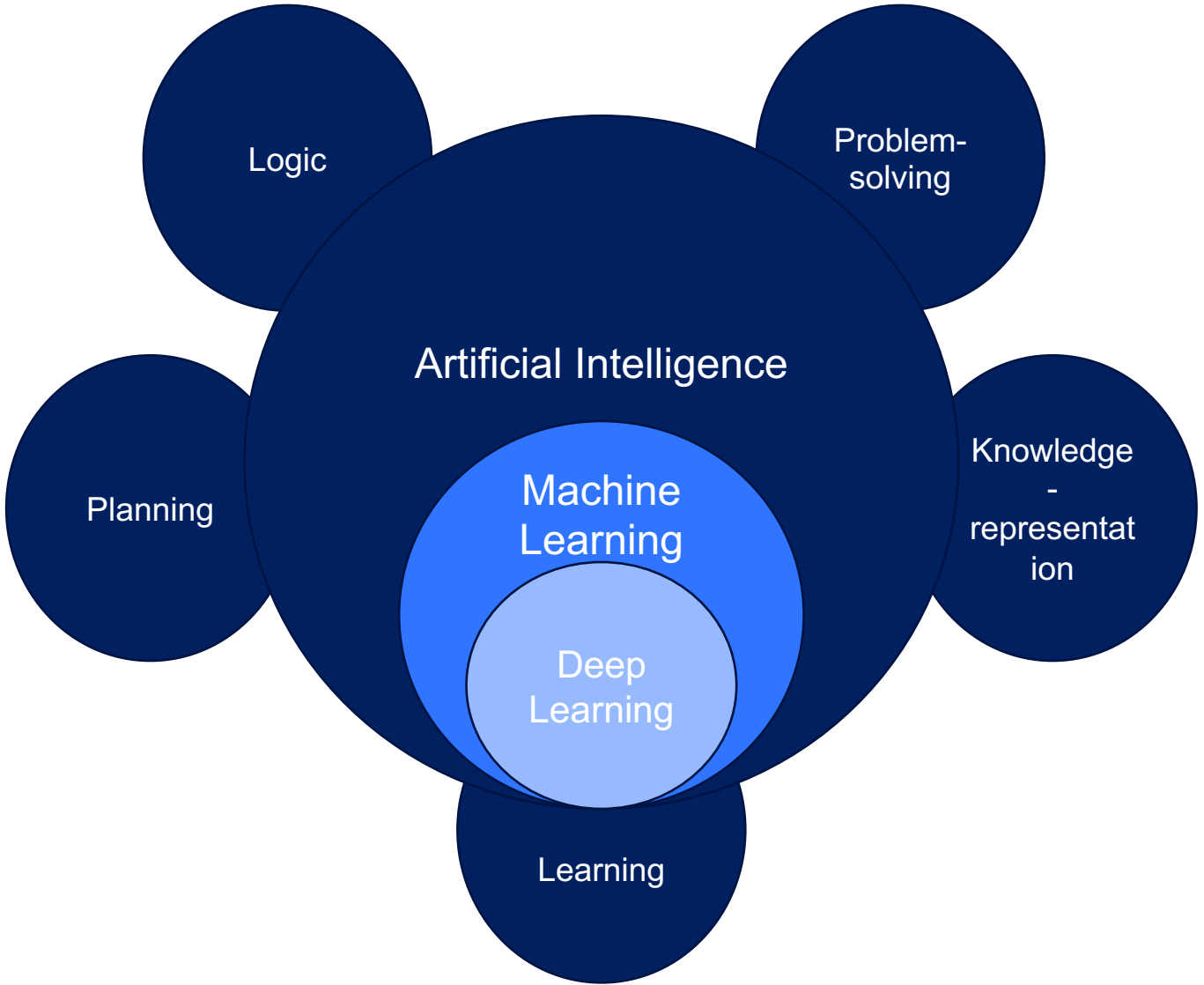
Overview

A refresher on linear algebra and Probability Theory can be found in Chapters 2 and 3

Today

- T, E, P
- Linear regression
- Overfitting/underfitting/capacity
- Hyperparameters
- Bias-Variance
- Making decisions




Logic
Problem-solving
Artificial Intelligence
Planning
Machine Learning
Knowledge - representation
Deep Learning
Learning

***T*ask**
***E*xperience**
***P*erformance**

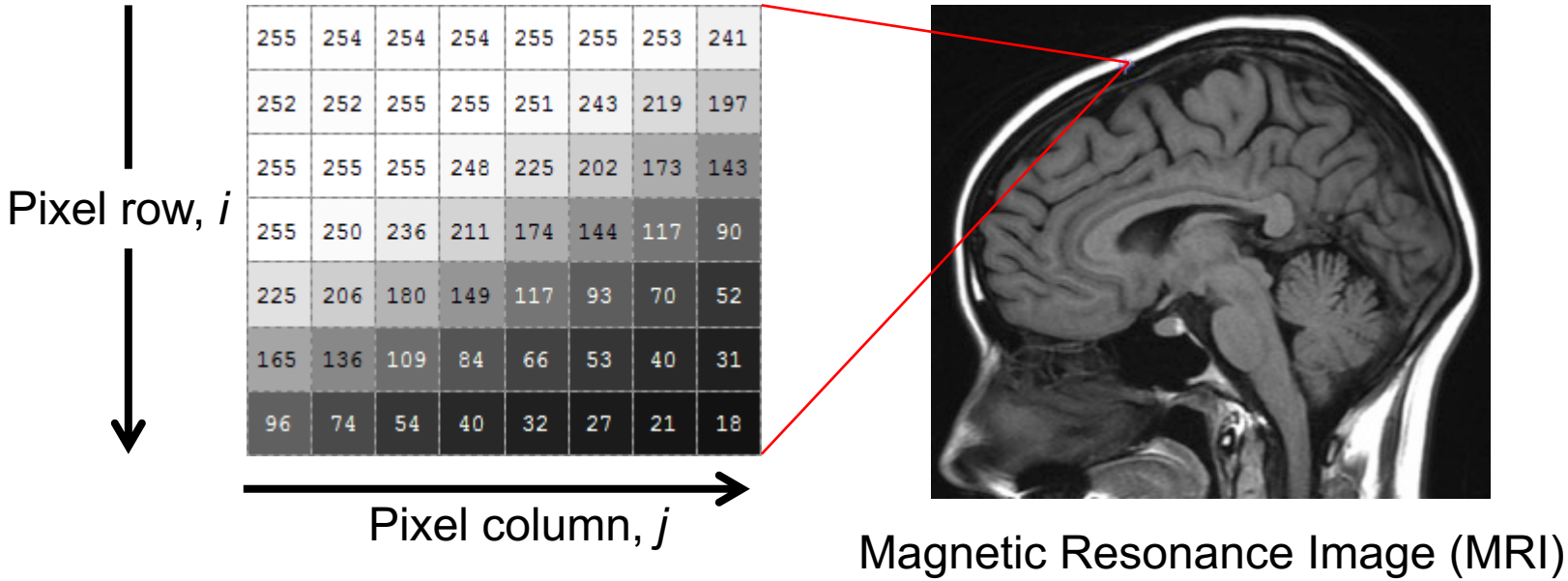
A definition of (machine) learning

"A computer program is said to learning from experience *E* with respect to some class of task *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."

- Mitchell (1997)

Some definitions

- *E* (experience) comes in the form of **examples**
- Examples are collections of **features**



Magnetic Resonance Image (MRI)

- Formally: $\boldsymbol{x} \in \mathbb{R}^n$
  - Each entry of the vector x_i is a **feature**

Different groups of Tasks

Classification

- Specify to which of *k* categories an input belongs to

$$f\colon \mathbb{R}^n \to \{1, \ldots, k\}$$

Many tasks are ‘binary’ problems (yes vs. no; *k*=2)

- One-hot encoding: $(0,0,1,0) \leftrightarrow 3$

Regression

- Predict a numerical value from the input:

$$f\colon \mathbb{R}^n \to \mathbb{R}$$

Further groups of tasks

Transcription: produce text describing a picture, transcribe spoken words to written text, …

Machine translation: translate text from one language to another

Structured output: Output is a vector rather than a scalar

Density estimation: Learn $p_{model}: \mathbb{R}^n \rightarrow \mathbb{R}$, where p_{model} is viewed as a probability density function

…

The performance measure *P*

- The performance measure *P* is specific to the task *T*

- Often used:
  - **Accuracy:** “percentage correct”
  - **Error rate:** “percentage incorrect” (0-1-Loss)
  - …

- *P* can sometimes be hard to ‘quantify’:
  - Translation: need to consider word ordering etc.
  - Density estimation

The performance measure *P*

- The performance measure can massively influence what the algorithm learns!

- A regression problem, typically we optimize the “mean squared error”

$$\mathrm{MSE} = \frac{1}{m}\sum_{i}(\hat{y}_i - y_i)^2$$

- MSE is sensitive to ‘outliers’

  Better mean absolute error: $\mathrm{MAE} = \frac{1}{m}\sum_i |\hat{y}_i - y_i|$

The performance measure *P*

Interested in how well the algorithm performs on new, unseen data:

generalization

= good performance *P* on an independent **test set** of data, i.e., not seen during training time

The experience *E*

Typically, the *Experience* is a **dataset** comprising many examples aka **data points**

Two board categories

Unsupervised

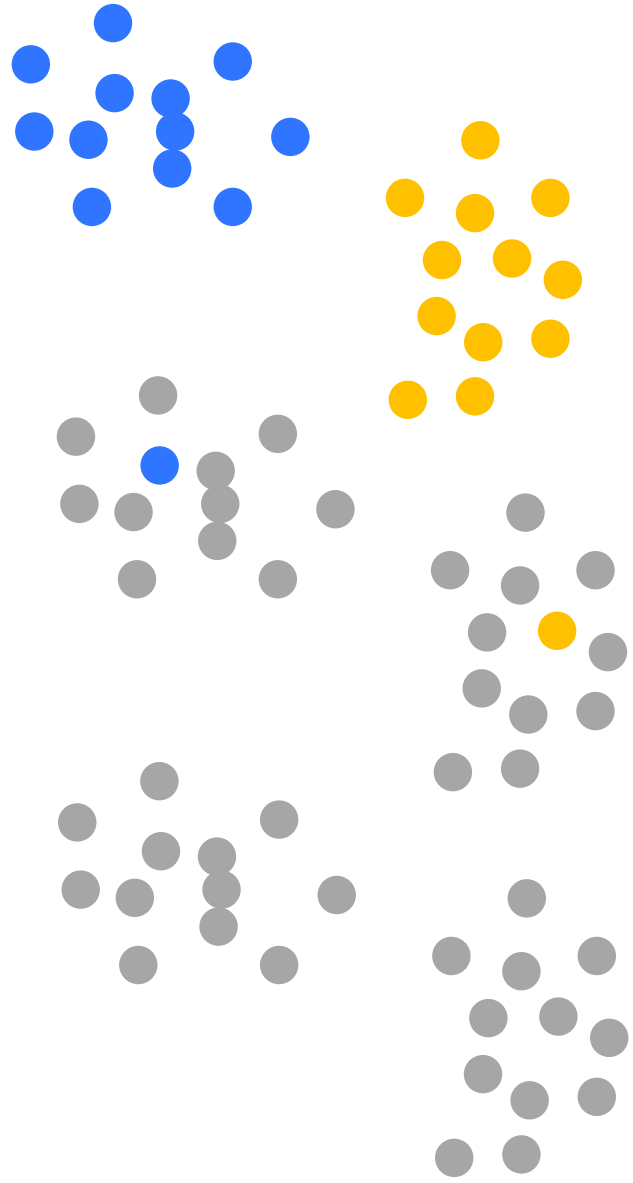
- contains many **features** and the task is to extract interesting properties

Supervised

- contains **features** as well as an **output label/target**

Broad categories of ML

- Supervised Learning
  - We are provided a label/target value
  - E.g. support vector machines
- Semi-supervised learning
  - Exploit distributions etc. of unlabeled data to improve supervised learning
- Unsupervised Learning
  - No labels used, aim to find a pattern
  - E.g., clustering



The experience *E*

- Often our examples are vectors of the same length

- **Design matrix** $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, m: samples, n: features

- More general: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
  - Each x can have a different dimension (e.g., images, sentences)

Linear regression

Linear regression

- Input $\boldsymbol{x} \in \mathbb{R}^n$
- Output $y \in \mathbb{R}$

- The model assumes that y is a linear combination of the values in $\boldsymbol{x}$:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x}.$$

- $\boldsymbol{w} \in \mathbb{R}^n$: model parameters learnt from the training data

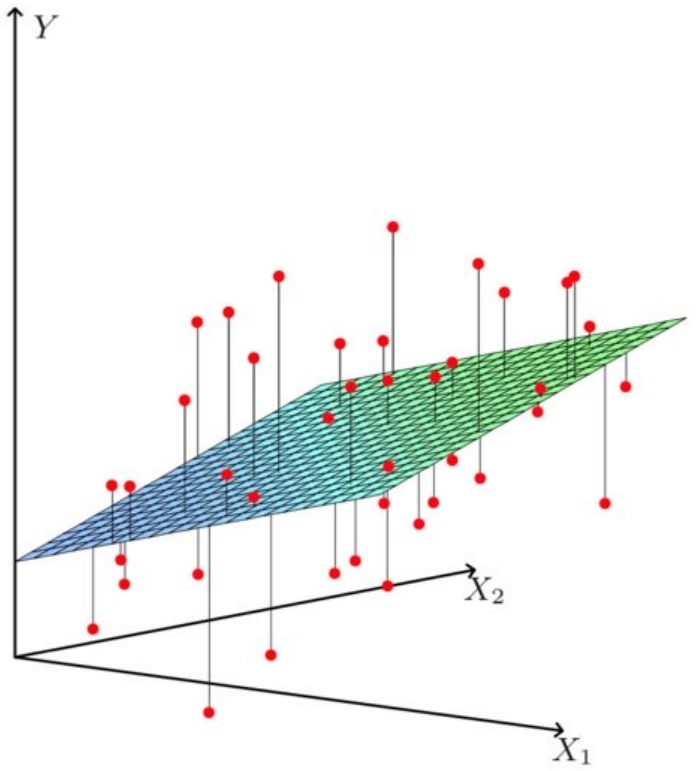


FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*

Linear regression

- *T:* predict y from $\boldsymbol{x}$ by computing $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$.
- *P: "distance from observation y to prediction $\hat{y}$."*
  - Assume we have a test set of *m* samples $\boldsymbol{X}^{(\text{test})}$ along with the targets $\boldsymbol{y}^{(\text{test})}$.
  - Mean squared error:

$$\text{MSE}_{\text{test}} = \frac{1}{\text{m}} \sum_i \left(\hat{\boldsymbol{y}}^{(\text{test})} - \boldsymbol{y}^{(\text{test})} \right)_i^2$$

$$= \frac{1}{m} \left\| \hat{\boldsymbol{y}}^{(\text{test})} - \boldsymbol{y}^{(\text{test})} \right\|_2^2$$

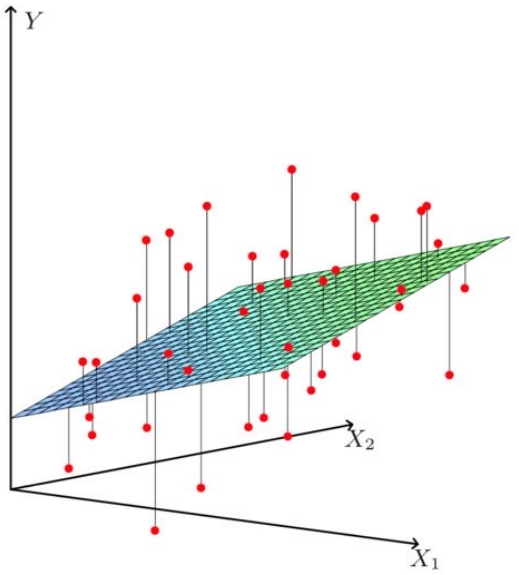


FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y.*

Find $\boldsymbol{w}^T$ that minimize MSE_{test}

Linear regression

During training we have the training data $\{\boldsymbol{X}^{(\text{train})}, \boldsymbol{y}^{(\text{train})}\}$.

We devise an ML algorithm by minimizing $\text{MSE}_{\text{train}}$

- Has a '*closed form*' solution:

$$\nabla_{\boldsymbol{w}} \text{MSE}_{\text{train}} = 0.$$

('set the derivative of $\text{MSE}_{\text{train}}$ to 0')

Linear regression

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} \left\| \hat{\boldsymbol{y}}^{(\mathrm{train})} - \boldsymbol{y}^{(\mathrm{train})} \right\|_2^2 = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} \left\| \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right\|_2^2 = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right)^T \left(\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right) = 0$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^T \boldsymbol{X}^{(\mathrm{train})T} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X}^{(\mathrm{train})T} \boldsymbol{y}^{(\mathrm{train})} + \boldsymbol{y}^{(\mathrm{train})T} \boldsymbol{y}^{(\mathrm{train})} \right) = 0$$

$$\Rightarrow 2\boldsymbol{X}^{(\mathrm{train})T} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2\boldsymbol{X}^{(\mathrm{train})T} \boldsymbol{y}^{(\mathrm{train})} = 0$$

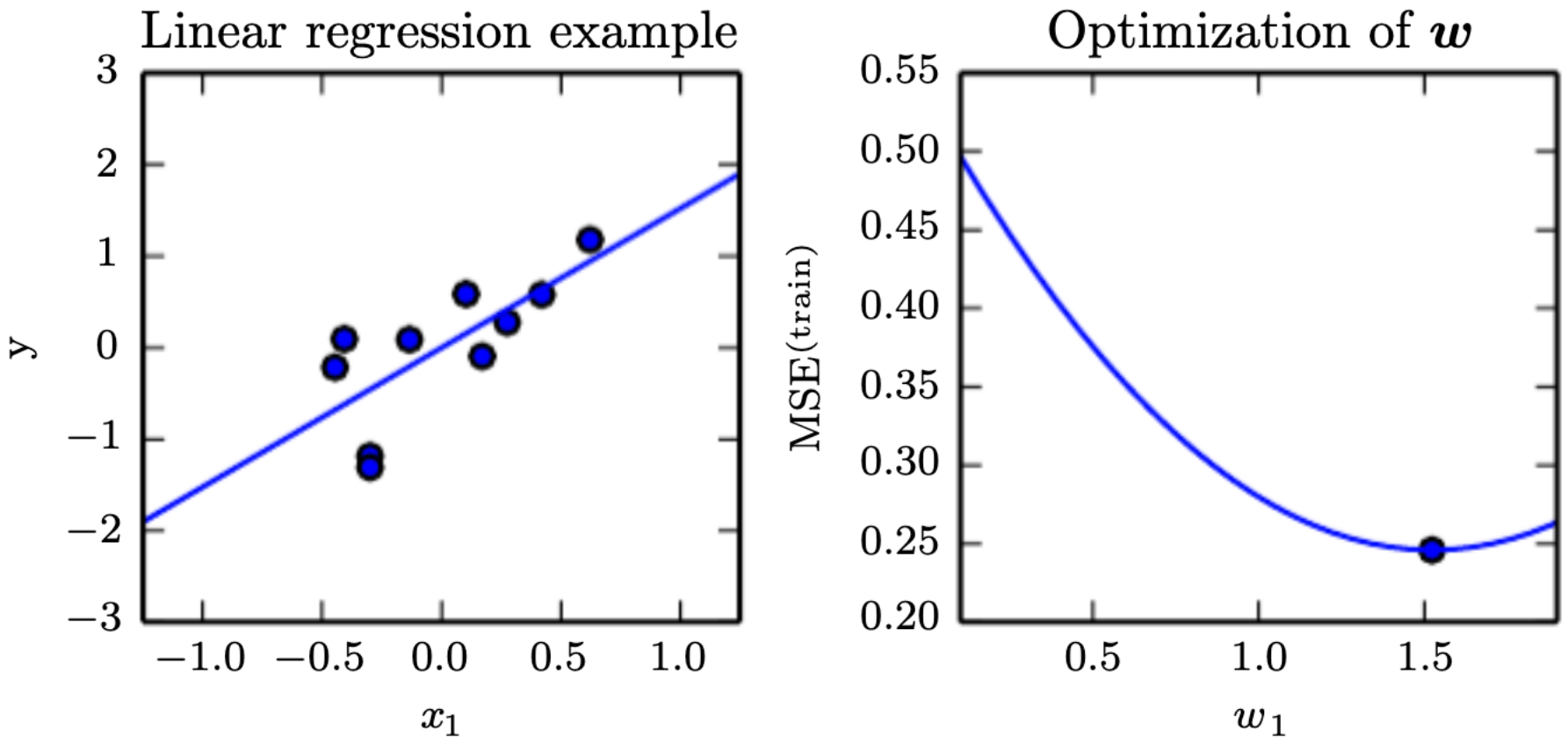
$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\mathrm{train})T} \boldsymbol{X}^{(\mathrm{train})} \right)^{-1} \boldsymbol{X}^{(\mathrm{train})T} \boldsymbol{y}^{(\mathrm{train})}$$

Normal equation

Linear regression:

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b.$$

Linear regression



https://setosa.io/ev/ordinary-least-squares-regression/

Linear regression

Essence of (supervised) machine learning

- We have a model that represents types of functions and is parameterized by $\boldsymbol{\theta}$

$$f(\boldsymbol{x}; \boldsymbol{\theta})$$

- We find the setting θ^* that minimizes the **Loss** (makes the fewest errors) on our training data:

$$\boldsymbol{\theta}^* = \text{argmin}_{\theta}\, J(\boldsymbol{\theta}) = \text{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{m} L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), y_i\right)$$

Overfitting, Underfitting & Capacity

Overfitting/Underfitting

- In practice we optimize the **training error**
- However, we aim to minimize the **test error** or **generalization error**
- Measured on a test set $\left\{\boldsymbol{X}^{(\text{test})}, \boldsymbol{y}^{(\text{test})}\right\}$:

$$\frac{1}{m^{\text{test}}} \left\| \boldsymbol{X}^{(\text{test})} \boldsymbol{w} - \boldsymbol{y}^{(\text{test})} \right\|_2^2$$

- Works: if training and test data were generated by the same data-generating process: $p(\boldsymbol{x}, y)$.
- **Realistic assumption?**

Test set size

- Test sets need to be larg*ish*
- Accuracy $\frac{1}{\mathrm{m}}\sum_i^m \delta_{y_i=\widehat{y_i}}$ is essentially a 'mean'
- Standard error of the mean:

$$\mathrm{SE} = \sigma/\sqrt{m}$$

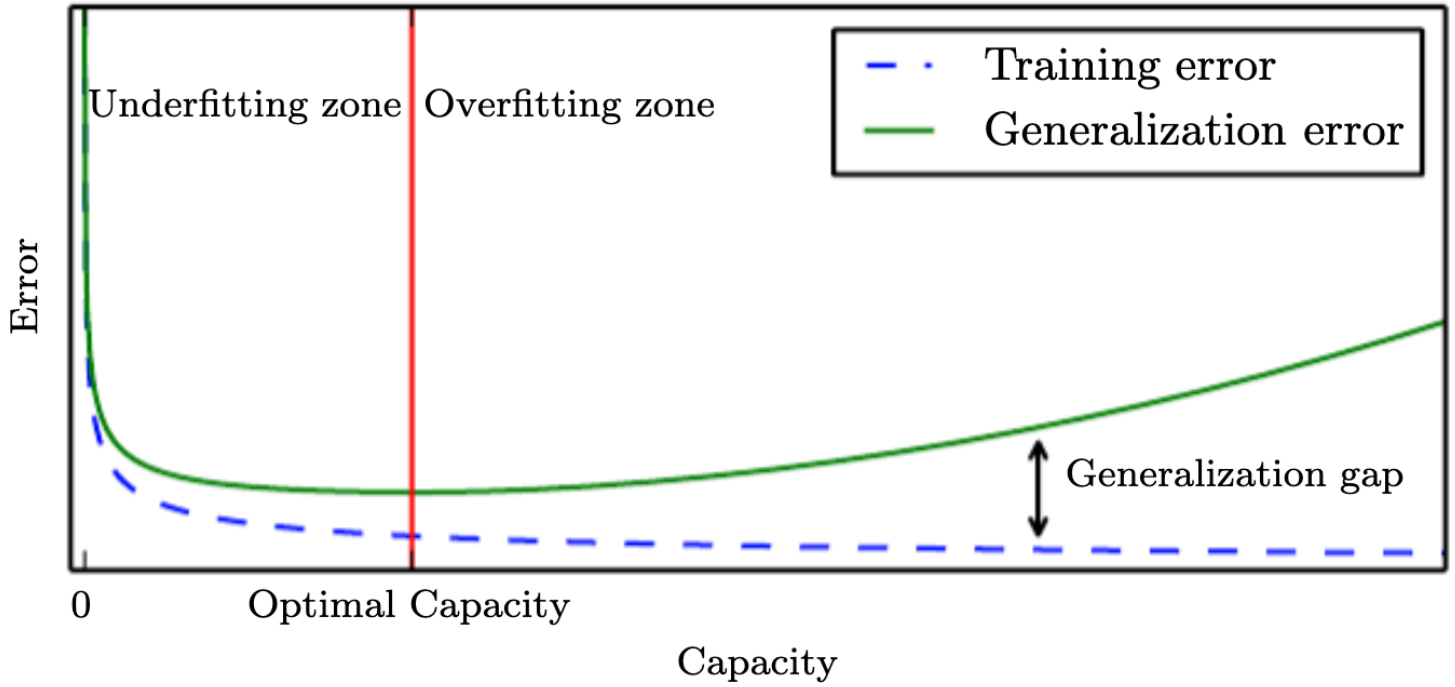
- Thus, accuracy estimate (with 95% CI):

$$[\mathrm{Acc} - 1.96 * \mathrm{SE}, \mathrm{Acc} + 1.96 * \mathrm{SE}]$$

Thus, if your test set is small, you have high uncertainty on your Acc (or other performance measure)

Overfitting/Underfitting

- Gap between training and test error
  - Overfitting: model too tuned to training data and performance on test data declines
  - Underfitting: model does not use full potential of the training data



Capacity

- Model "*flexibility*" as in ability to fit a wide variety of functions

- Linear model rather 'inflexible':

$$\hat{y} = b + wx$$

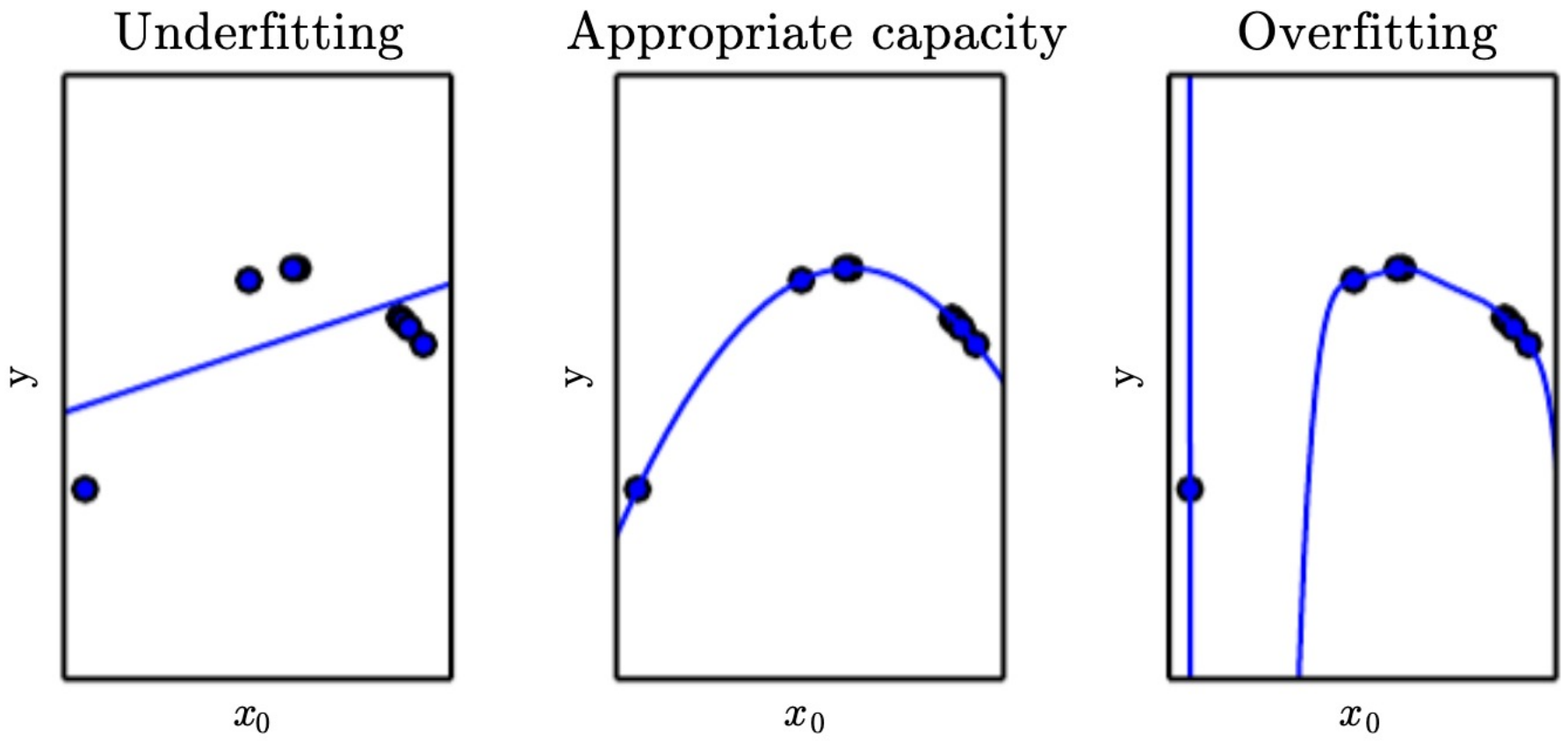
- Adding 'powers' adds flexibility:

$$\hat{y} = b + w_1 x + w_2 x^2$$

$$\hat{y} = b + \sum_i w_i x^i$$

- The power *i* becomes a model parameter

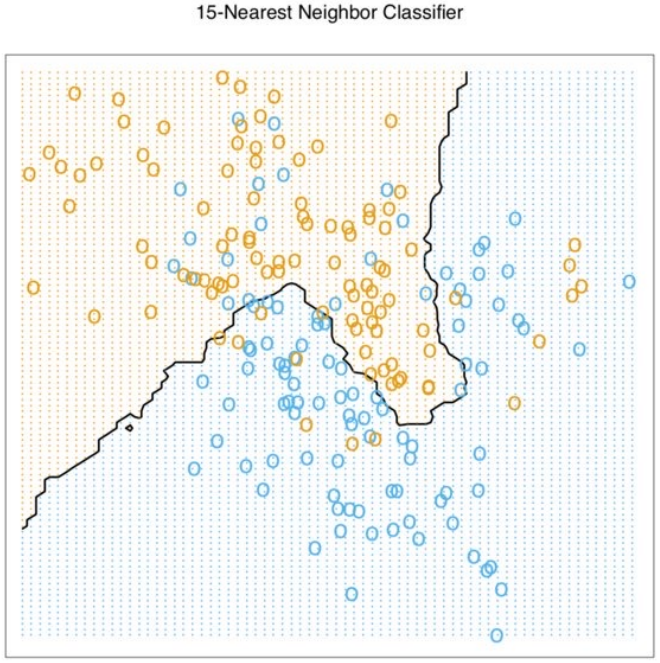
Capacity

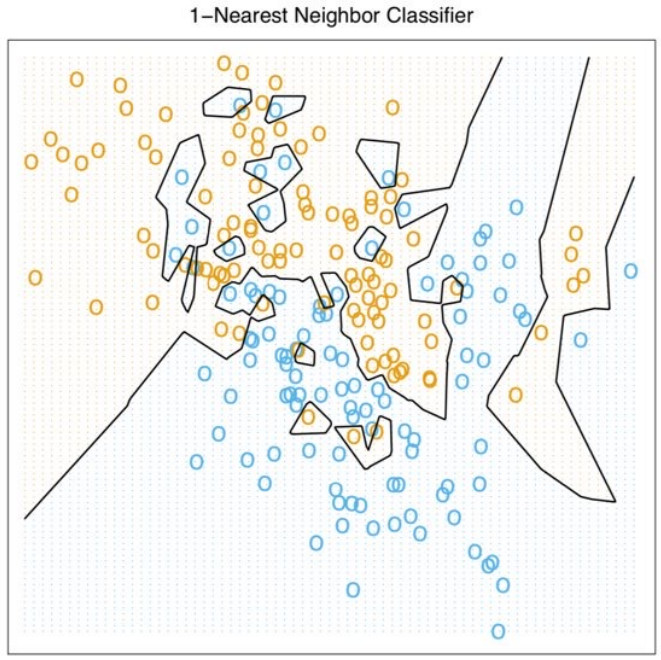


i=1 i=2 i=9

Representational capacity **Occam's razor**

Capacity





Nearest Neighbor Classifier

$$\hat{y} = y_i$$
$$i = \text{argmin}\|\boldsymbol{X}_i - \boldsymbol{x}\|_2^2$$

Regularization & Hyperparameter tuning

No Free Lunch Theorem

"Averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points."

- Wolpert (1996)

Regularization

- Using background knowledge in algorithm design improves the model
  - E.g., we know the data follows a quadratic function etc.
  - Dependencies in real life complex
  - In practice we constrain the possible solutions

- Previously we optimized only *P:*

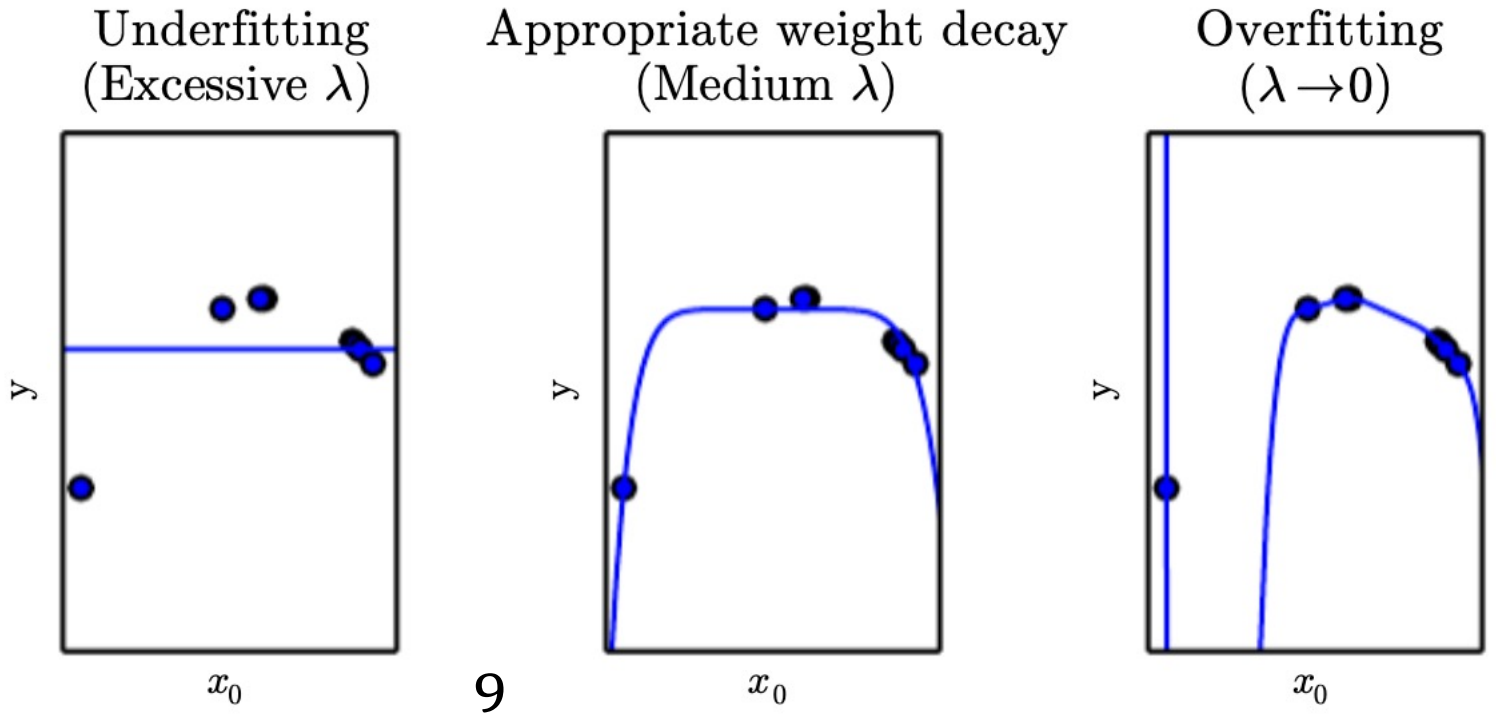
$$J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}}$$

- Now we '*punish*' model complexity:

$$J(\boldsymbol{w}; \lambda) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

L^2 –norm
weight decay

Regularization

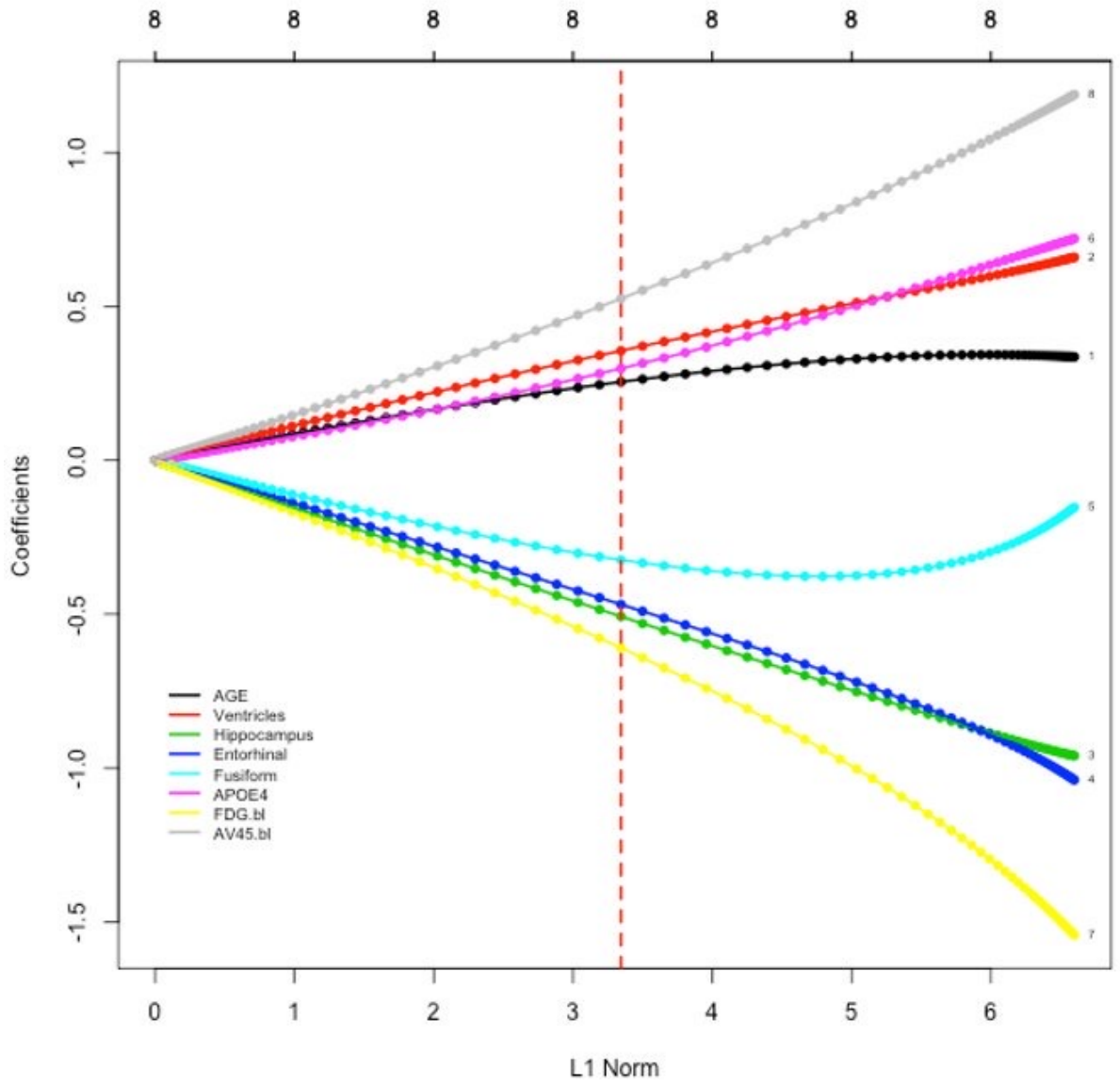


$$\hat{y} = b + \sum_{i}^{9} w_i x^i$$

$$\Omega(\boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{w}$$

Regularizer

Regularization – Ridge regression



Tuning hyperparamters

- *Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.*

- In the example before λ is a parameter that controls **capacity: hyperparameter**

- How to tune the hyperparameter?

Tuning hyperparameter

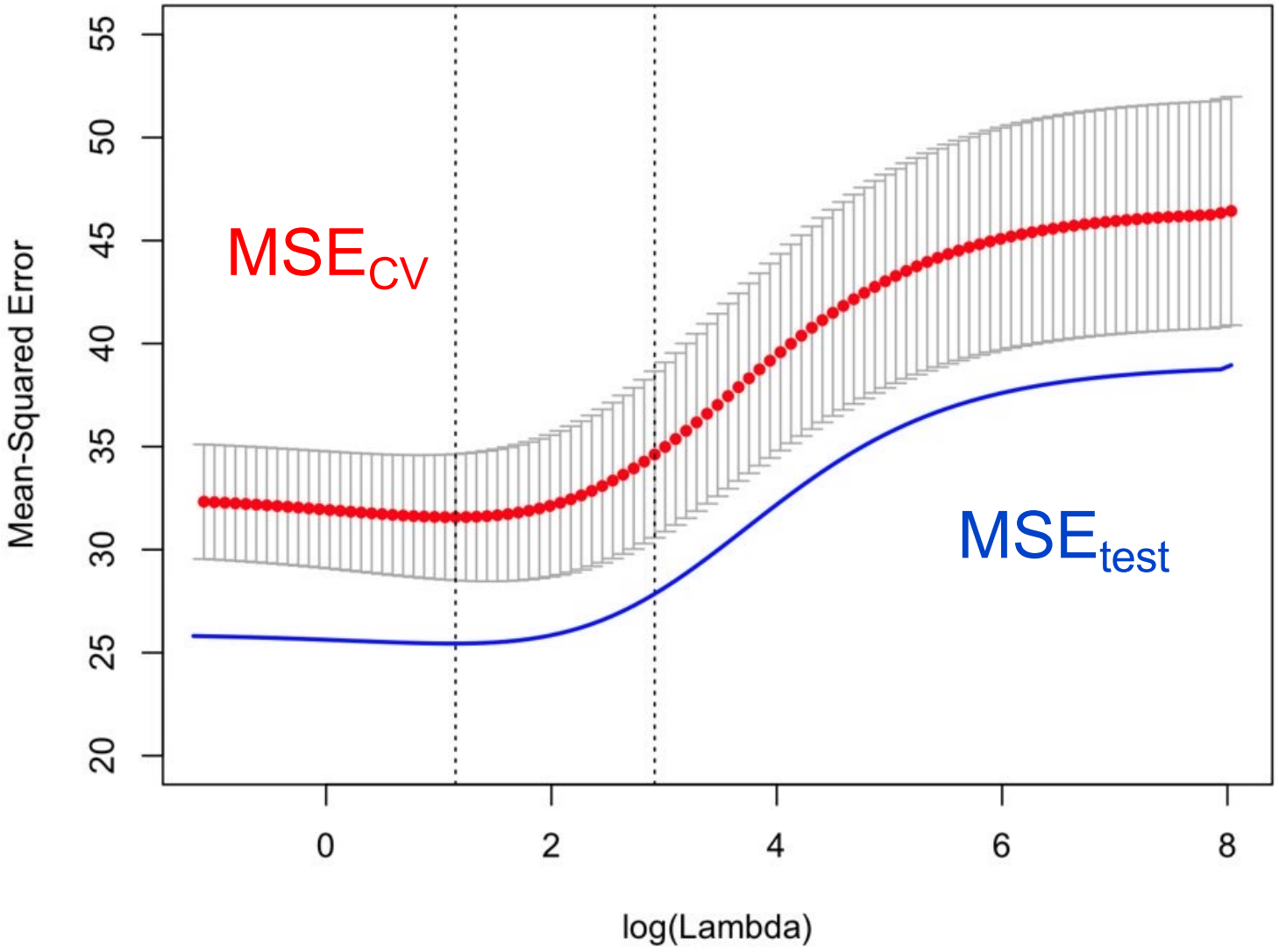
- An ‘independent’ dataset: **validation set**
- “training set for hyper parameters”

Cross-validation (CV)

- Train-validation split works well when we are ‘data rich’
- For small data set: split repeatedly
- Split into *k* non-overlapping sets (‘*k*-fold’)
  - Rotate role of validation set

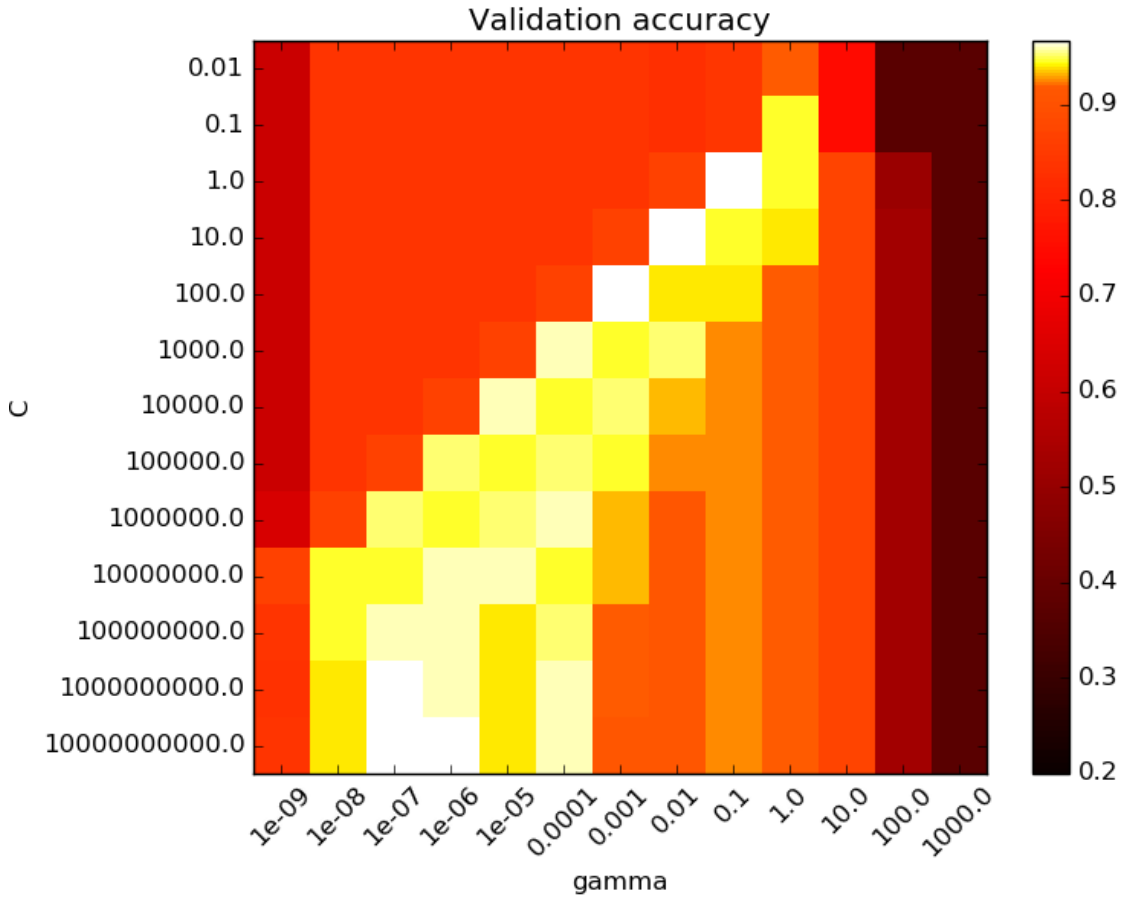
5-fold cross-validation

Cross-validation (CV)



Grid search

- More than one hyperparameter?



Bias Variance Trade-off

Bias-variance-tradeoff

- One can decompose MSE into Bias and variance

$$\mathrm{MSE} = \mathbb{E}\left[\left(\hat{\theta}_m - \theta\right)^2\right]$$
$$= \mathrm{Bias}\left(\hat{\theta}_m\right)^2 + \mathrm{Var}\left(\hat{\theta}_m\right)$$

- Bias: “Error made by assumption.“
- Variance: “Error made b/c adopting to data.“

Point Estimates

- True value: $\boldsymbol{\theta}$
- Estimate from data: $\widehat{\boldsymbol{\theta}}$

- Let $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ be a set of *m* i.i.d. points
- Then: $\widehat{\boldsymbol{\theta}}_m = g\left(x^{(1)}, \ldots, x^{(m)}\right)$
- $\widehat{\boldsymbol{\theta}}_m$ is a point estimate of $\boldsymbol{\theta}$, which generated the data
- Example: $\mathcal{N}(x^{(i)}; \mu, \sigma^2)$ generates data; $\boldsymbol{\theta} = (\mu, \sigma)$
  by analyzing *m* datapoints we get $\widehat{\boldsymbol{\theta}}_m = (\hat{\mu}_m, \hat{\sigma}_m)$

Bias

- Estimates deviate from the ‘ground truth’: bias

$$\text{bias}\left(\widehat{\boldsymbol{\theta}}_m\right) = \mathbb{E}\left(\widehat{\boldsymbol{\theta}}_m\right) - \boldsymbol{\theta}$$

- **Unbiased:** if $\text{bias}\left(\widehat{\boldsymbol{\theta}}_m\right) = \mathbf{0}$

- **Asymptotically unbiased:** $\lim_{\mathrm{m}\to\infty} \text{bias}\left(\widehat{\boldsymbol{\theta}}_m\right) = \mathbf{0}$

Example: Gaussian distrib.

$$p\left(x^{(i)}; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{\left(x^{(i)} - \mu\right)^2}{\sigma^2}\right)$$

- Sample mean: $\hat{\mu}_m = \frac{1}{m}\sum_i x^{(i)}$
- Bias: $\text{bias}(\hat{\mu}_m) = \mathbb{E}[\hat{\mu}_m] - \mu = \mathbb{E}\left[\frac{1}{m}\sum_i x^{(i)}\right] - \mu$

$$= \left(\frac{1}{m}\sum_i \mathbb{E}\left[x^{(i)}\right]\right) - \mu$$

$$= \left(\frac{1}{m}\sum_i \mu\right) - \mu = \mu - \mu = 0$$

Example: Gaussian distrib.

$$p\left(x^{(i)};\mu,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2}\frac{\left(x^{(i)}-\mu\right)^2}{\sigma^2}\right)$$

- Sample variance: $\hat{\sigma}^2{}_m=\frac{1}{m}\sum_i(x^{(i)}-\hat{\mu}_m)^2$
- Bias: $\text{bias}(\hat{\sigma}^2{}_m)=\mathbb{E}[\hat{\sigma}^2{}_m]-\sigma^2$

- $\mathbb{E}[\hat{\sigma}^2{}_m]=\mathbb{E}\left[\frac{1}{m}\sum_i(x^{(i)}-\hat{\mu}_m)^2\right]=\cdots=\frac{m-1}{m}\sigma^2$

Sample variance is biased. That is why we use $\frac{1}{m-1}\sum_i(x^{(i)}-\hat{\mu}_m)^2$ instead.

Variance of an Estimator

- *"How much does the estimator vary as a function of the data sample?"*

$$\mathrm{Var}(\hat{\theta})$$

- E.g., an estimator $g\left(x^{(1)}, x^{(2)}, \ldots, x^{(m)}\right) = c$ returning a constant has variance 0

Bias-variance-tradeoff

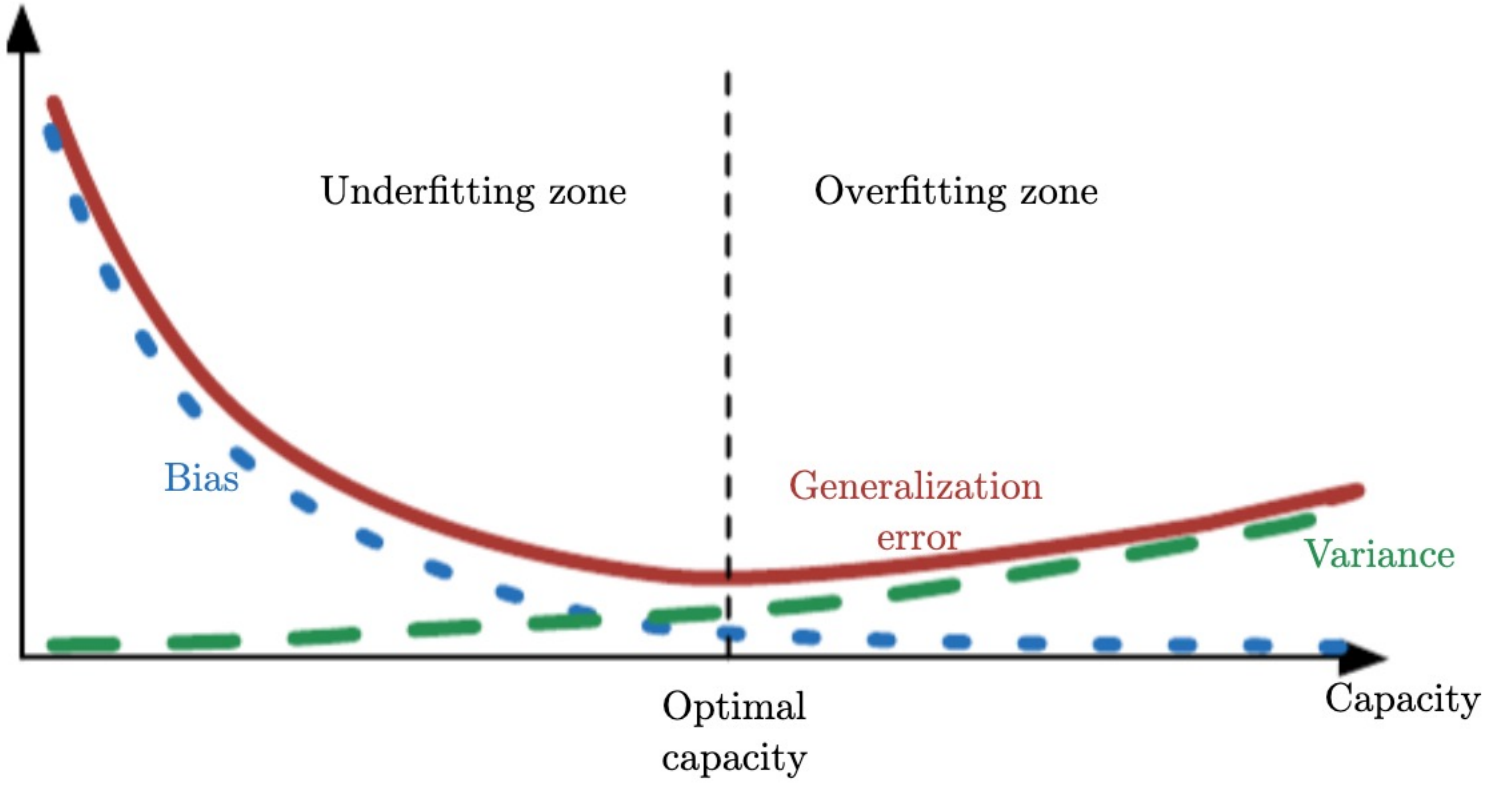
- One can decompose MSE into Bias and variance

$$\mathrm{MSE} = \mathbb{E}\left[\left(\hat{\theta}_m - \theta\right)^2\right]$$
$$= \mathrm{Bias}\left(\hat{\theta}_m\right)^2 + \mathrm{Var}\left(\hat{\theta}_m\right)$$

- Bias: "Error made by assumption."
- Variance: "Error made b/c adopting to data."

$$J(\boldsymbol{w}) = \frac{1}{m}\sum_i \left(\mathbf{X}^{(\mathrm{train})}\boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}\right)_i^2 + \lambda \boldsymbol{w}^T\boldsymbol{w}$$

Bias-variance-tradeoff



$$J(\boldsymbol{w}) = \frac{1}{m}\sum_{i}\left(\mathbf{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})}\right)_i^2 + \lambda \boldsymbol{w}^T\boldsymbol{w}$$

Making decisions

Discriminant functions

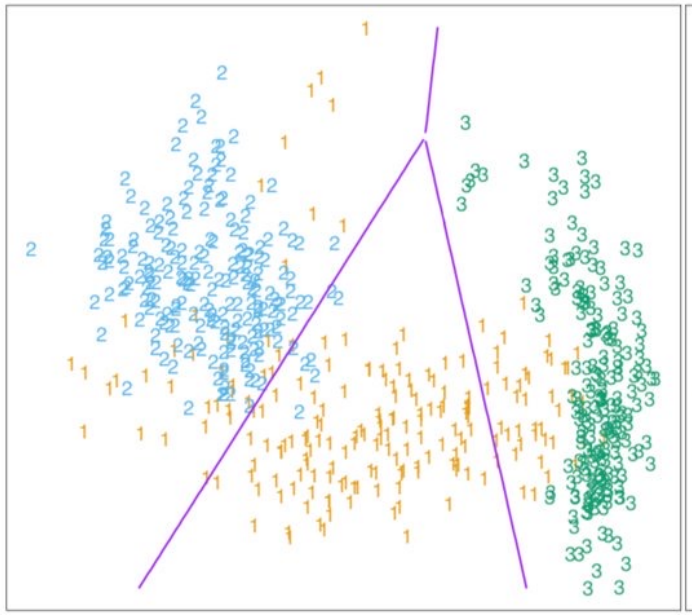
- In **Regression** we have functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- In **Classification** our target is $\in \{1, \ldots, k\}$

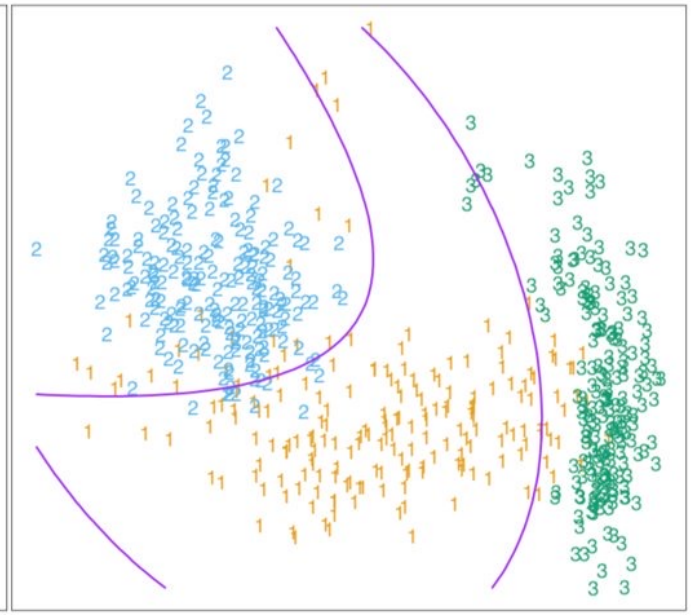
- Methods typically model continuous *discriminant function* for each class: $\delta_k(x)$
- Predicted class is the one with the largest value for $\delta_k(x)$

- E.g., $P(y = k|x)$ is a discriminant function

Decision boundaries

- Where the support for class *k* and *k'* is the same
- $\delta_k(\boldsymbol{x}) = \delta_{k'\ell}(x)$





Bayes Theorem

- Very important and useful:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

- Applied to classification:

likelihood Class prior

$$P(y = k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y = k) * P(y = k)}{P(\boldsymbol{x})}$$

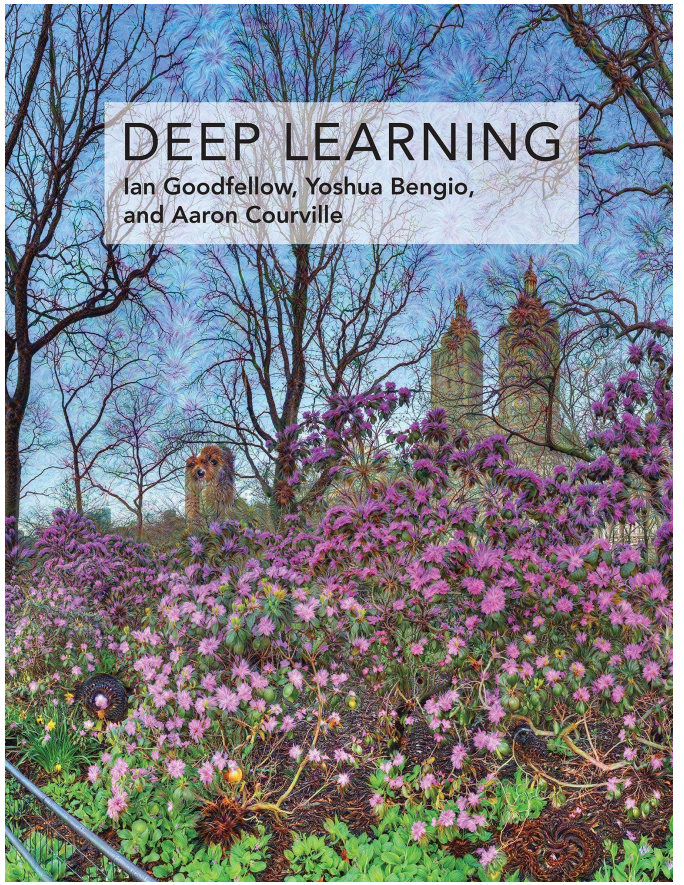
support

Overview

A refresher on linear algebra and Probability Theory can be found in Chapters 2 and 3

Today

- T, E, P
- Linear regression
- Overfitting/underfitting/capacity
- Hyperparameters
- Bias-Variance
- Making decisions



Thank you!