Machine Learning

Last Time

- Bayesian Networks (AIMA Ch 13 & 14)

- Basic premise — a Bayesian Network is a directed acyclic graph over random variables, together with conditional probability tables describing each random variable given all of its incoming edges (often referred to as parents).

- A BN can represent a joint probability distribution over N variables with fewer than 2^N parameters (subject to the distribution).

- Bayesian networks allow easy specification of conditional independence relations between random variables.

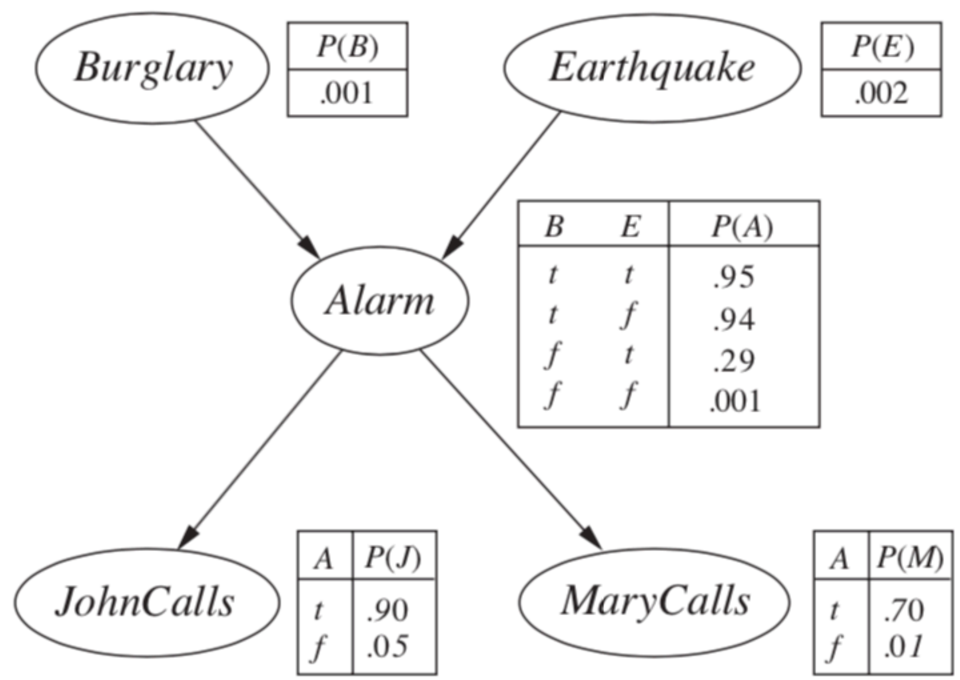
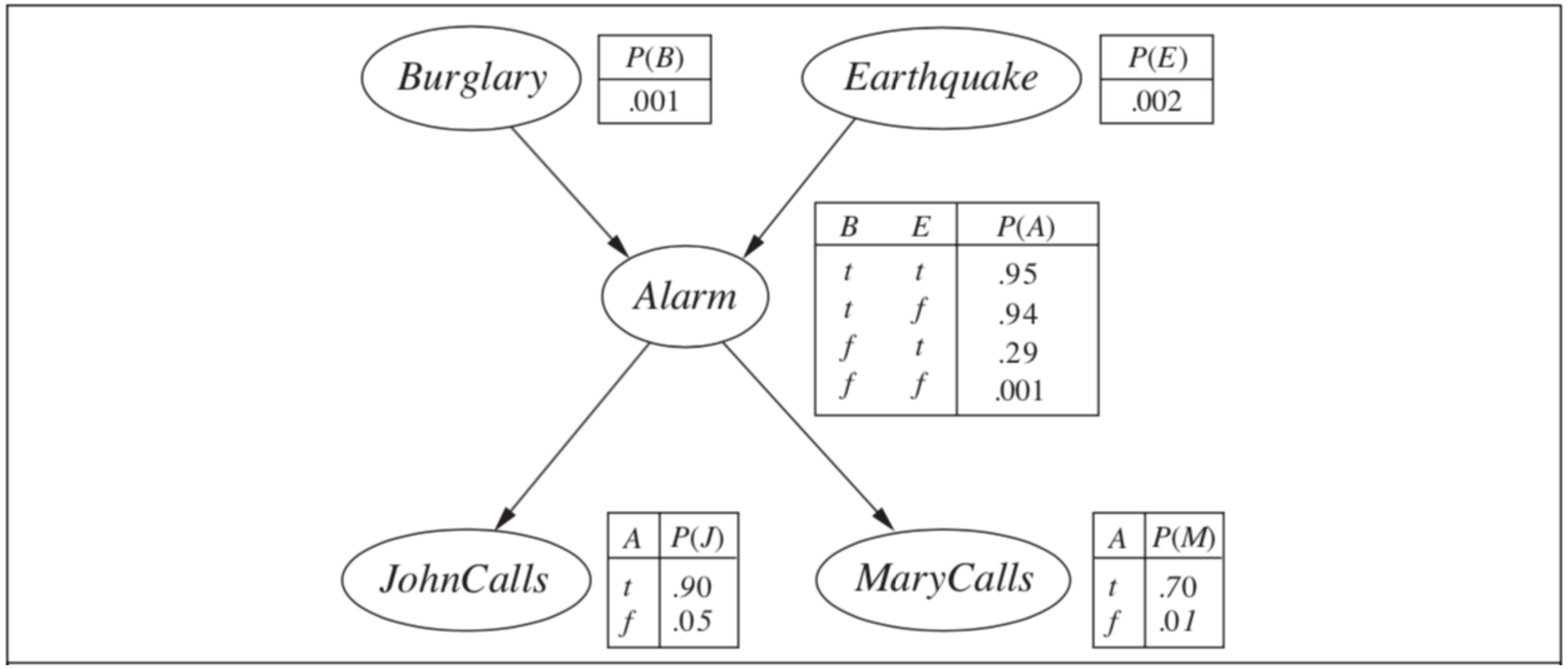


Figure 14.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.



$$\mathbf{P}(MaryCalls \mid JohnCalls, Alarm, Earthquake, Burglary) = \mathbf{P}(MaryCalls \mid Alarm)$$

$$\mathbf{P}(Burglary \mid Alarm, JohnCalls, MaryCalls) = \mathbf{P}(Burglary \mid Alarm)$$

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j \mid a)P(m \mid a)P(a \mid \neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$

Inference

- Pr(Query | Evidence)

- Pr(Query ^ Evidence) / Pr(Evidence)

- Pr(Q ^ E) = sum_x Pr(Q ^ E ^ X)

    - Monstrous exponential sum.

    - Exact algorithms — n*2^n, variable ordering gets it to 2^n

    - Approximate algorithms… next time.

Approximate Inference

- What is sampling?
- Generate several random samples, count each of the outcomes for the query, in the long run this approximates the expectation which is the real value.
- How to do sampling?
  - Rejection — basic, top-down, reject inconsistent, can reject too much
  - Likelihood weighting — no rejection, but still individual contributions of samples can be very small
  - Gibbs sampling — conditions on markov blanket, walks state space, ergodicity, markov chains

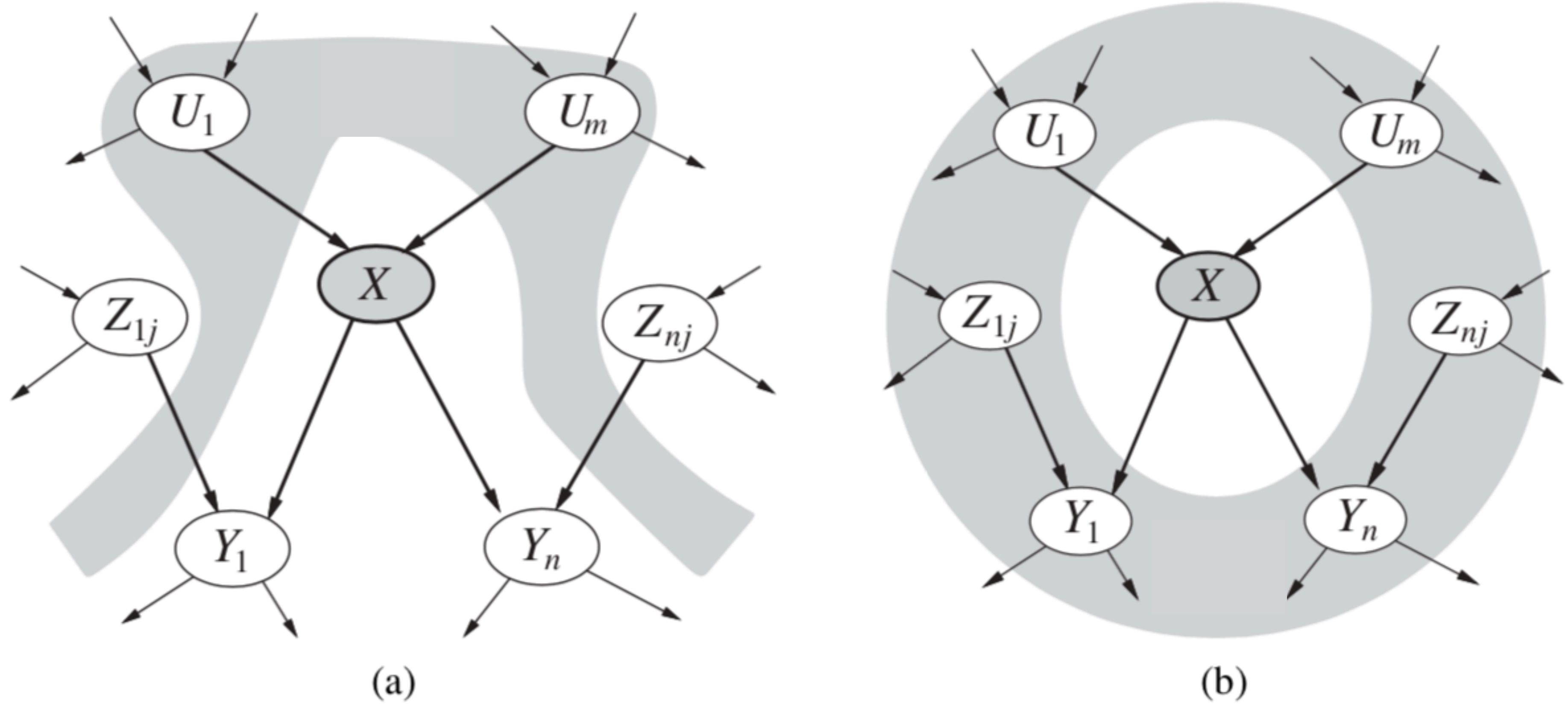


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

Up Next

- Machine Learning (AIMA Ch 18)

Paradigms

- Most machine learning problems fall into one of three broad categories, based on the kind of **feedback** or **supervision** presented to the method.
- No Feedback — Clustering
- Rewards / Penalties — Reinforcement Learning
- Examples / Labels — Supervised Learning

+ Semi-Supervised methods, too.

Examples

Clustering	
Reinforcement Learning	
Supervised Learning	

Examples

Clustering	deciphering words in a new language
Reinforcement Learning	
Supervised Learning	

Examples

Clustering	deciphering words in a new language
Reinforcement Learning	playing a game of chess
Supervised Learning	

Examples

Clustering	deciphering words in a new language
Reinforcement Learning	playing a game of chess
Supervised Learning	predicting weather based on observations

Examples

Clustering	???
Reinforcement Learning	???
Supervised Learning	???

Our Focus

- Let’s focus on **supervised learning** for now.
- Today we will understand the general paradigm, later we will look at specific instances based on decision trees, linear models, logistic regression, and neural networks.

Supervised Learning

- Supervised learning problems can be viewed as learning a functional mapping from inputs to outputs.

Regression vs Classification

- Regression — mapping from inputs to real numbers
  - Examples — estimating stock prices, forecasting temperature, predicting probability of state transitions for actions, enhancing photos with super resolution, …

Regression vs Classification

- Classification — mapping from inputs to categories
    - Examples — identifying animals based on images, identifying political party based on social media posts, diagnosing diseases, road sign reader, friend or foe, …

Supervised Learning

- We must select a **hypothesis space** of candidate mappings. Often our hypothesis space will be associated with a set of numerical **parameters** and **hyper-parameters.**

Example:

Learn y = f(**X**) given some training data.

Hypothesis space	
Parameters	
Hyperparameters	

Example:

Learn y = f(**X**) given some training data.

Hypothesis space	the set of all polynomial functions in a single variable. $\hat{y} = a_n x^n + \cdots + a_1 x + a_0$
Parameters	
Hyperparameters	

Example:

Learn y = f(**X**) given some training data.

Hypothesis space	the set of all polynomial functions in a single variable. $\hat{y} = a_n x^n + \cdots + a_1 x + a_0$
Parameters	the coefficients $a_n, a_{n-1}, \cdots, a_1, a_0$
Hyperparameters	

Example:

Learn y = f(**X**) given some training data.

Hypothesis space	the set of all polynomial functions in a single variable. $\hat{y} = a_n x^n + \cdots + a_1 x + a_0$
Parameters	the coefficients $a_n, a_{n-1}, \cdots, a_1, a_0$
Hyperparameters	the highest order of the polynomials n

Example:

Learn y = f(**X**) given some training data.

Hypothesis space	the set of all polynomial functions in a single variable. $\hat{y} = a_n x^n + \cdots + a_1 x + a_0$
Parameters	the coefficients $a_n, a_{n-1}, \cdots, a_1, a_0$
Hyperparameters	the highest order of the polynomials n

What if the true function is not a polynomial?

Interpolation and Extrapolation

- Machine learning algorithms are *great* at **interpolation**, which is a kind-of filling-in-the-gaps between data points.

- "It was snowing when I went to bed, and it was snowing when I woke up, so it was probably snowing throughout the night."

Interpolation and Extrapolation

- Machine learning algorithms are *great* at **interpolation**, which is a kind-of filling-in-the-gaps between data points.

- Most algorithms are *very poor* at **extrapolation**, which is a kind-of predicting-the-future kind of problem.

Interpolation and Extrapolation

- Most algorithms are *poor* at **extrapolation**, which is a kind-of predicting-the-future kind of problem.
- “It was 38F Monday morning, and 23F Tuesday morning, so on Wednesday it will be 8F and by December it will be 270 below!”

Generalization vs Overfitting

- Two more defs:
  - **Generalization** — the performance of the model is good on unseen inputs (i.e., data which was not used during training).
  - **Overfitting** — the performance on the model becomes artificially good on the training data at the expense of generalization.

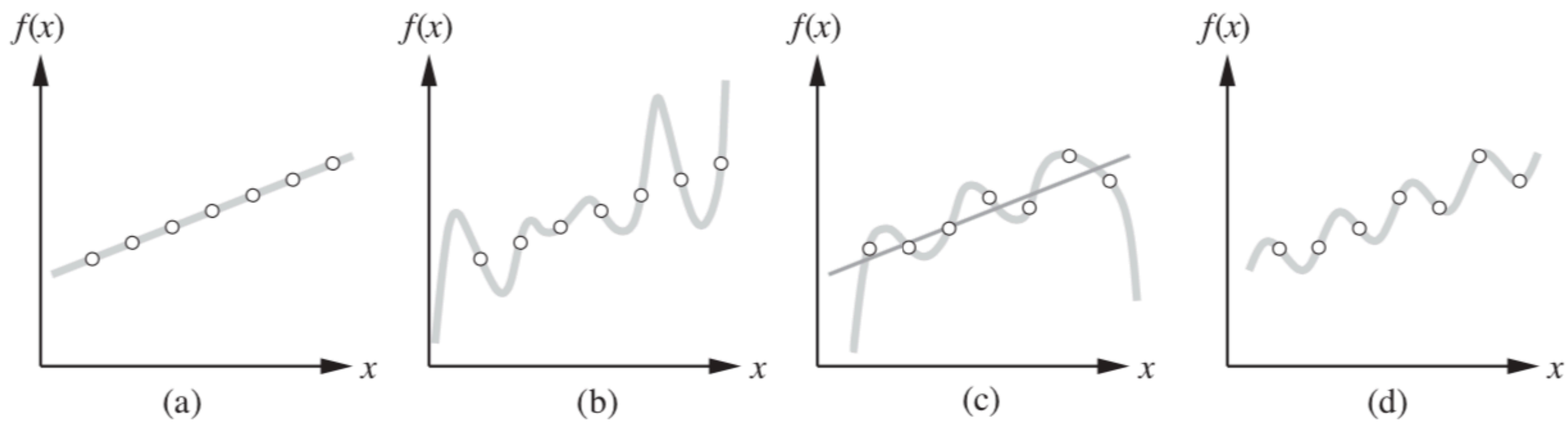


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

Causes of Overfitting

- Overfitting can happen when the hypothesis space is overly complex, especially in the presence of **noise**.

- I.e., increasing the number of parameters increases the likelihood of overfitting.

Preventing Overfitting

- Ockham's Razor — i.e., Regularization.
- ``simpler solutions are more likely to be correct than complex ones''

Model Exploration

- General idea:
  - Start with a simple model.
  - Until good enough do:
    - Test performance on training data.
    - Good? Check for overfitting.
    - Bad? Increase parameters/hypothesis space.

Detecting Overfitting

- Cross-Validation — a portion of data is 'held out' during training and used for evaluation.
- k-Fold cross-validation partitions a dataset into K equal portions, training on all but one, and evaluating on the held out portion, then repeats and average over all portions.

Training Models

- A hypothesis is (usually) determined by a set of adjustable parameters, so the challenge is to find a **good set of parameters** given some data.

Training Models

- Possible approaches:
  - Guess and check?
  - Someone gives you the parameters?
  - Guess and improve?

How do you know if the parameters are good?

Training Models: Loss

- Most machine learning models involve minimizing some form of **loss** (or maximizing some form of **score**).

Training Models: Loss

- Loss functions are usually defined per example.
- Suppose target is $y = f(\mathbf{x})$ and prediction is $\hat{y} = \hat{f}(\mathbf{x})$.
- Examples:
  - Squared Error $(y_n - \hat{y}_n)^2$
  - Absolute Error $|y_n - \hat{y}_n|$
  - Indicator $\begin{cases} 0 & \text{if } y_n = \hat{y}_n \\ 1 & \text{otherwise} \end{cases}$

Training Models: Loss

- The overall loss of a model is usually with respect to a specific data set — which is just a set of examples.
- Loss can be summarized in many ways:
  - Total $$L(\theta) = \sum_n (y_n - \hat{y_n})^2$$
  - Mean $$L(\theta) = \frac{1}{N} \sum_n (y_n - \hat{y_n})^2$$
  - Variance, Max, Min, …

Training

- A hypothesis is (usually) determined by a set of adjustable parameters, so the challenge is to find a **good set of parameters** given some data.

- Most machine learning models involve minimizing some form of **loss** (or maximizing some form of **score**).

Training Models

- Possible approaches:
  - Guess and check?
  - Someone gives you the parameters?
  - Guess and improve?

Training Models

- Possible approaches:
  - Guess and check?
  - Someone gives you the parameters?
  - **Guess and improve?**

Optimization and Gradient Descent

- By far the most common method is **gradient descent**:

$$\arg\min_{\theta} L(\theta)$$

$$\theta' = \theta + \gamma \nabla L$$

- We will discuss this in detail week when we do **logistic regression** and later when we talk about **neural networks.**

Model Exploration

- General idea:
  - Start by training a simple model.
  - Until good enough do:
    - Test performance on training data.
    - Good? Check for overfitting.
    - Bad? Increase parameters/hypothesis space and train a new model.

Note that it is possible that it may never be good enough for any number of parameters or model size, so this could go on forever!

Model Exploration

- General idea:
  - Start by training a simple model.
  - **Note that it is possible that it may never be good enough for any number of parameters or model size, so this could go on forever!**
    - Good? Check for overfitting.
    - Bad? Increase parameters/hypothesis space and train a new model.

Supervised Learning

- Supervised learning problems can be viewed as learning a functional mapping from inputs to outputs.

Supervised Learning

- Regression vs Classification
- Hypothesis Spaces and Parameters
- Interpolation vs Extrapolation
- Generalization vs Overfitting and Cross-Validation
- Ockham's Razor and Regularization
- Optimization and Gradient Descent

Next Time

- Classification with Decision Trees