Machine Learning – some basics

Bishop PRML Ch. 1

Torsten Möller

Outline

Machine Learning: What, Why, and How?

Curve Fitting: (e.g.) Regression and Model Selection

Goal Function: Decision Theory—ML, Loss Function, MAP

Probability Theory: (e.g.) Coin Tossing and Parameter Estimation

Conclusion

References

- Christopher M. Bishop: Pattern Recognition and Machine Learning; Springer, 2007
- Han, Kamber: Data Mining: Concepts and Techniques; Elsevier, 2012
- Hastie, Tibshirani, Friedman: The Elements of Statistical Learning; Springer, 2009
- James, Witten, Hastie, Tibshirani: An Introduction to Statistical Learning with Applications in R; Springer, 2015
- Kevin Murphy: Machine Learning, A Probabilistic Perspective, The MIT Press, 2012
- Tan, Steinbach, Kumar: Introduction to Data Mining; Pearson, 2005

Outline

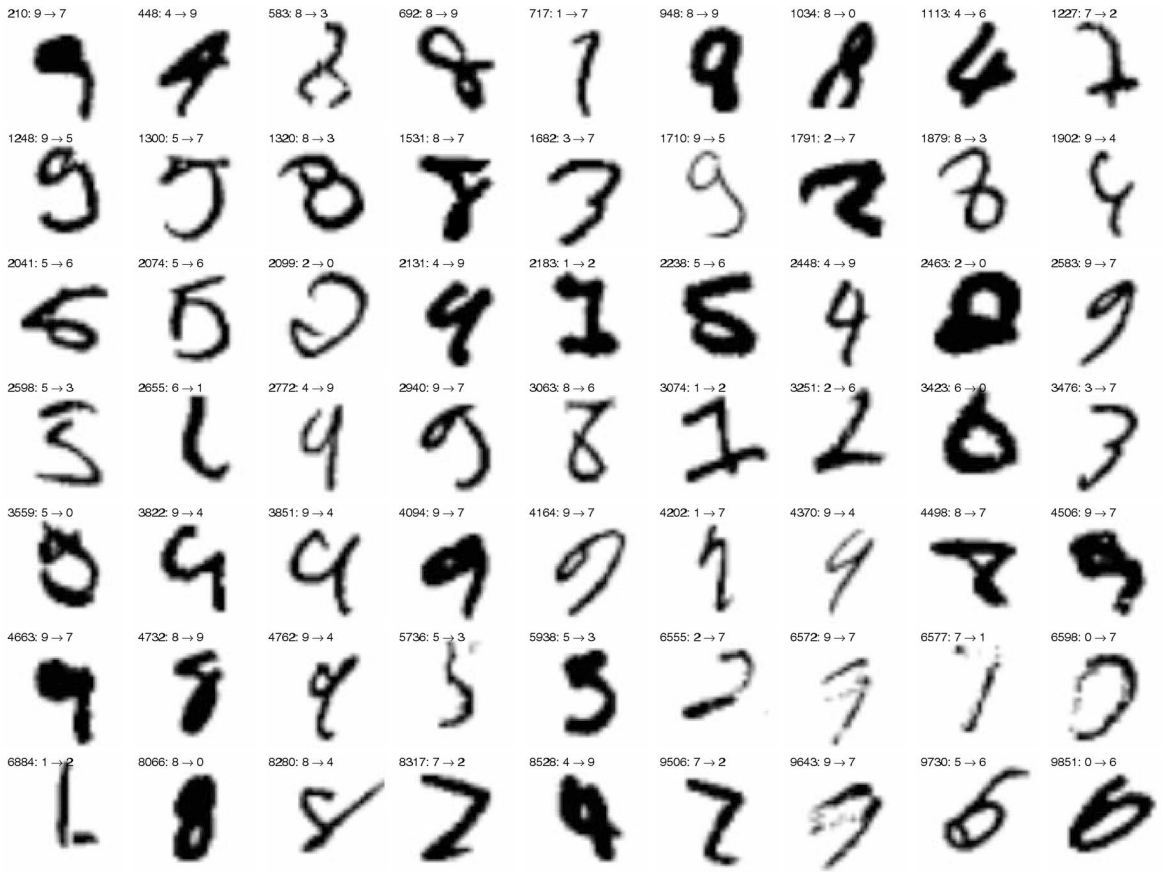
What is Machine Learning (ML)?

- Algorithms that automatically improve performance through experience
- Often this means define a model by hand, and use data to fit its parameters

Why ML?

- The real world is complex – difficult to hand-craft solutions.
- ML is the preferred framework for applications in many fields:
    - Computer Vision
    - Natural Language Processing, Speech Recognition
    - Robotics
    - …

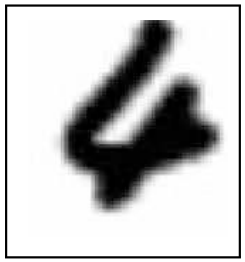
Hand-written Digit Recognition



Belongie et al. PAMI 2002

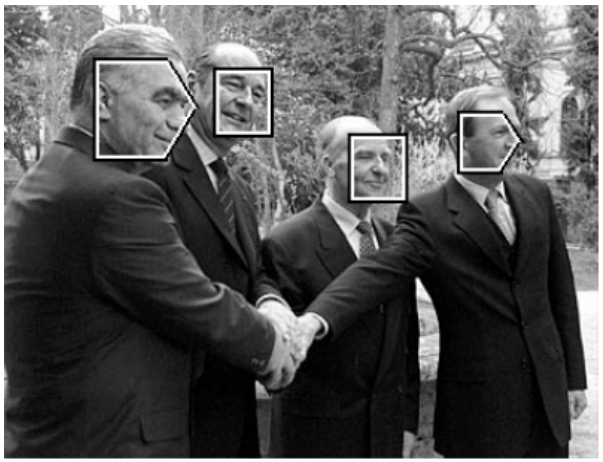
- Difficult to hand-craft rules about digits

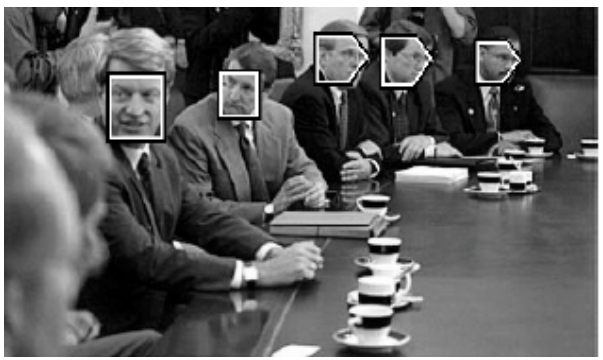
Hand-written Digit Recognition

$\boldsymbol{x}_i =$  $\boldsymbol{t}_i = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$

- Represent input image as a vector $\boldsymbol{x}_i \in \mathbb{R}^{784}$.
- Suppose we have a target vector $\boldsymbol{t}_i$
  - This is supervised learning
  - Discrete, finite label set: perhaps $\boldsymbol{t}_i \in \{0, 1\}^{10}$, a classification problem
- Given a training set $\{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$, learning problem is to construct a "good" function $\boldsymbol{y}(\boldsymbol{x})$ from these.
  - $\boldsymbol{y} : \mathbb{R}^{784} \to \mathbb{R}^{10}$

Face Detection

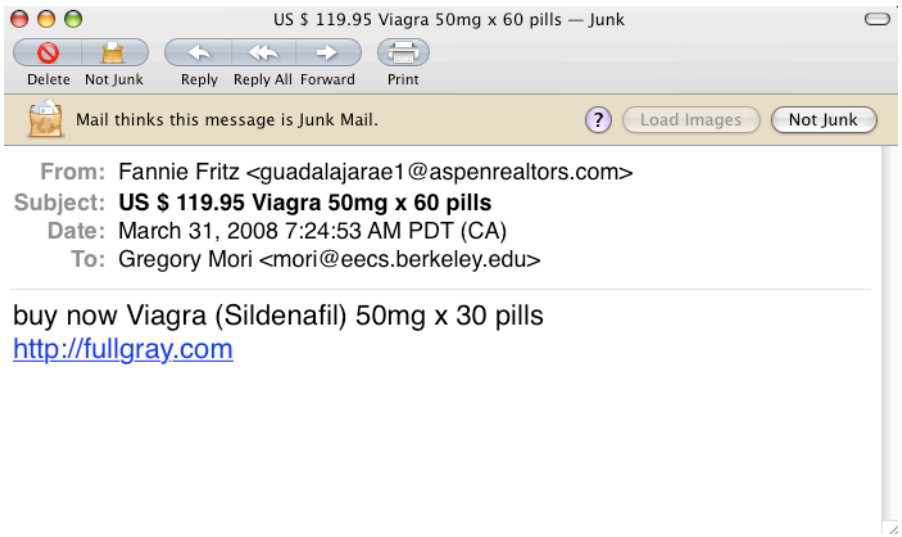


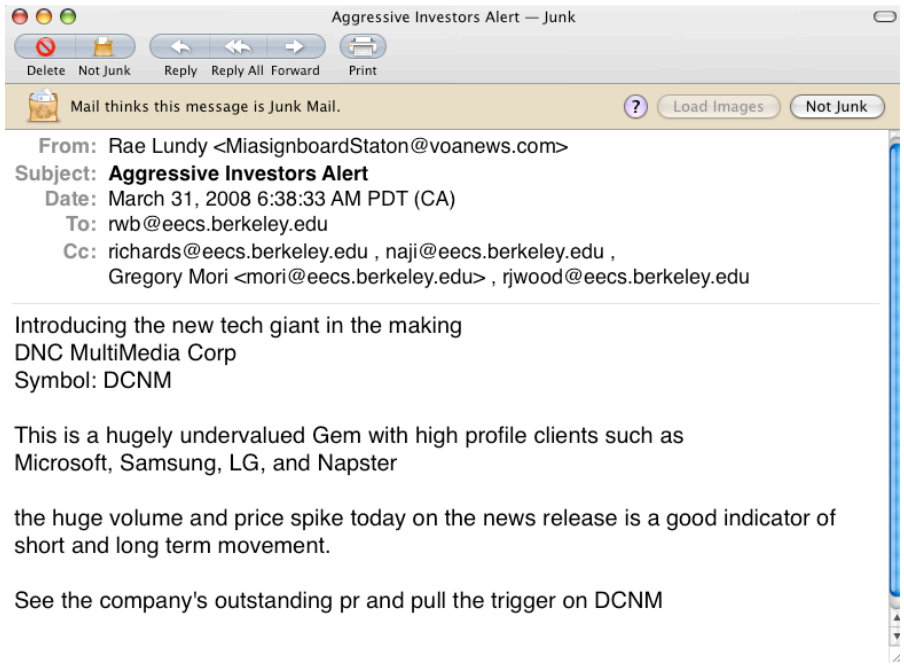


Schneiderman and Kanade, IJCV 2002

- Classification problem
- $\boldsymbol{t}_i \in \{0, 1, 2\}$, non-face, frontal face, profile face.

Spam Detection





- Classification problem
- $\boldsymbol{t}_i \in \{0, 1\}$, non-spam, spam
- $\boldsymbol{x}_i$ counts of words, e.g. `Viagra`, `stock`, `outperform`, `multi-bagger`

Caveat - Horses (source?)

- Once upon a time there were two neighboring farmers, Jed and Ned. Each owned a horse, and the horses both liked to jump the fence between the two farms. Clearly the farmers needed some means to tell whose horse was whose.
- So Jed and Ned got together and agreed on a scheme for discriminating between horses. Jed would cut a small notch in one ear of his horse. Not a big, painful notch, but just big enough to be seen. Well, wouldn't you know it, the day after Jed cut the notch in horse's ear, Ned's horse caught on the barbed wire fence and tore his ear the exact same way!
- Something else had to be devised, so Jed tied a big blue bow on the tail of his horse. But the next day, Jed's horse jumped the fence, ran into the field where Ned's horse was grazing, and chewed the bow right off the other horse's tail. Ate the whole bow!

Caveat - Horses (source?)

- Once upon a time there were two neighboring farmers, Jed and Ned. Each owned a horse, and the horses both liked to jump the fence between the two farms. Clearly the farmers needed some means to tell whose horse was whose.
- So Jed and Ned got together and agreed on a scheme for discriminating between horses. Jed would cut a small notch in one ear of his horse. Not a big, painful notch, but just big enough to be seen. Well, wouldn't you know it, the day after Jed cut the notch in horse's ear, Ned's horse caught on the barbed wire fence and tore his ear the exact same way!
- Something else had to be devised, so Jed tied a big blue bow on the tail of his horse. But the next day, Jed's horse jumped the fence, ran into the field where Ned's horse was grazing, and chewed the bow right off the other horse's tail. Ate the whole bow!

Caveat - Horses (source?)

- Once upon a time there were two neighboring farmers, Jed and Ned. Each owned a horse, and the horses both liked to jump the fence between the two farms. Clearly the farmers needed some means to tell whose horse was whose.
- So Jed and Ned got together and agreed on a scheme for discriminating between horses. Jed would cut a small notch in one ear of his horse. Not a big, painful notch, but just big enough to be seen. Well, wouldn't you know it, the day after Jed cut the notch in horse's ear, Ned's horse caught on the barbed wire fence and tore his ear the exact same way!
- Something else had to be devised, so Jed tied a big blue bow on the tail of his horse. But the next day, Jed's horse jumped the fence, ran into the field where Ned's horse was grazing, and chewed the bow right off the other horse's tail. Ate the whole bow!

Caveat - Horses (source?)

- Finally, Jed suggested, and Ned concurred, that they should pick a feature that was **less apt to change**. Height seemed like a good feature to use. But were the heights different? Well, each farmer went and measured his horse, and do you know what? The brown horse was a full inch taller than the white one!

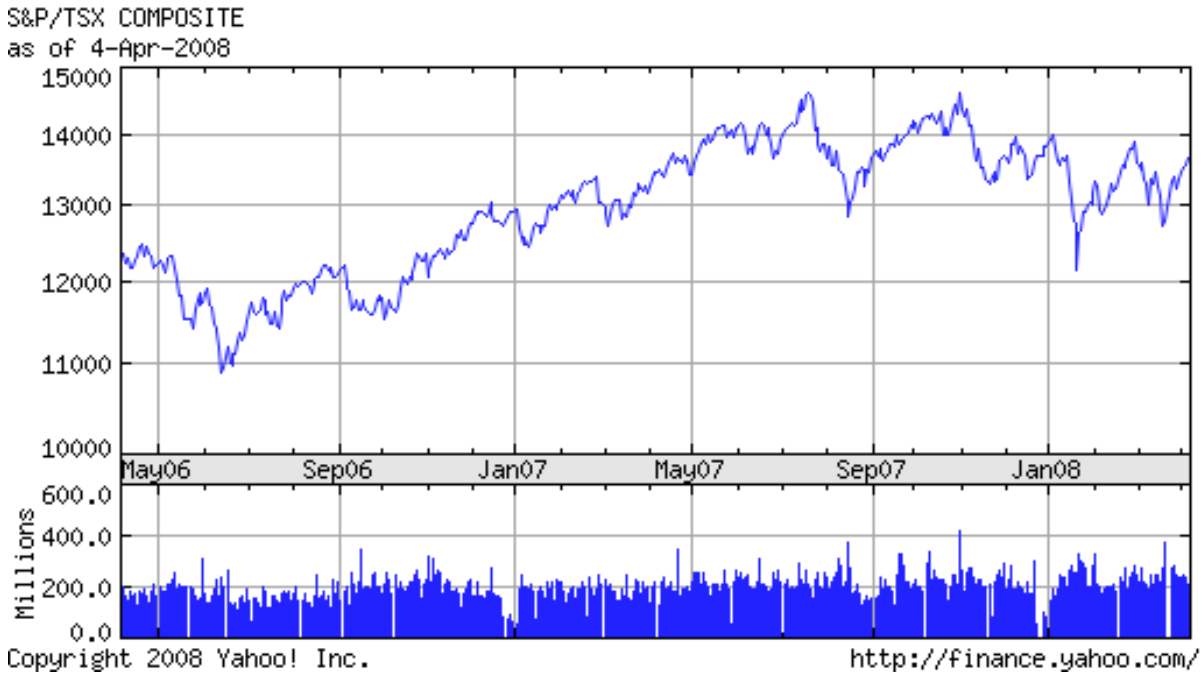
Moral of the story: ML provides theory and tools for setting parameters. Make sure you have the right model and features. Think about your "feature vector x."

Caveat - Horses (source?)

- Finally, Jed suggested, and Ned concurred, that they should pick a feature that was **less apt to change**. Height seemed like a good feature to use. But were the heights different? Well, each farmer went and measured his horse, and do you know what? The brown horse was a full inch taller than the white one!

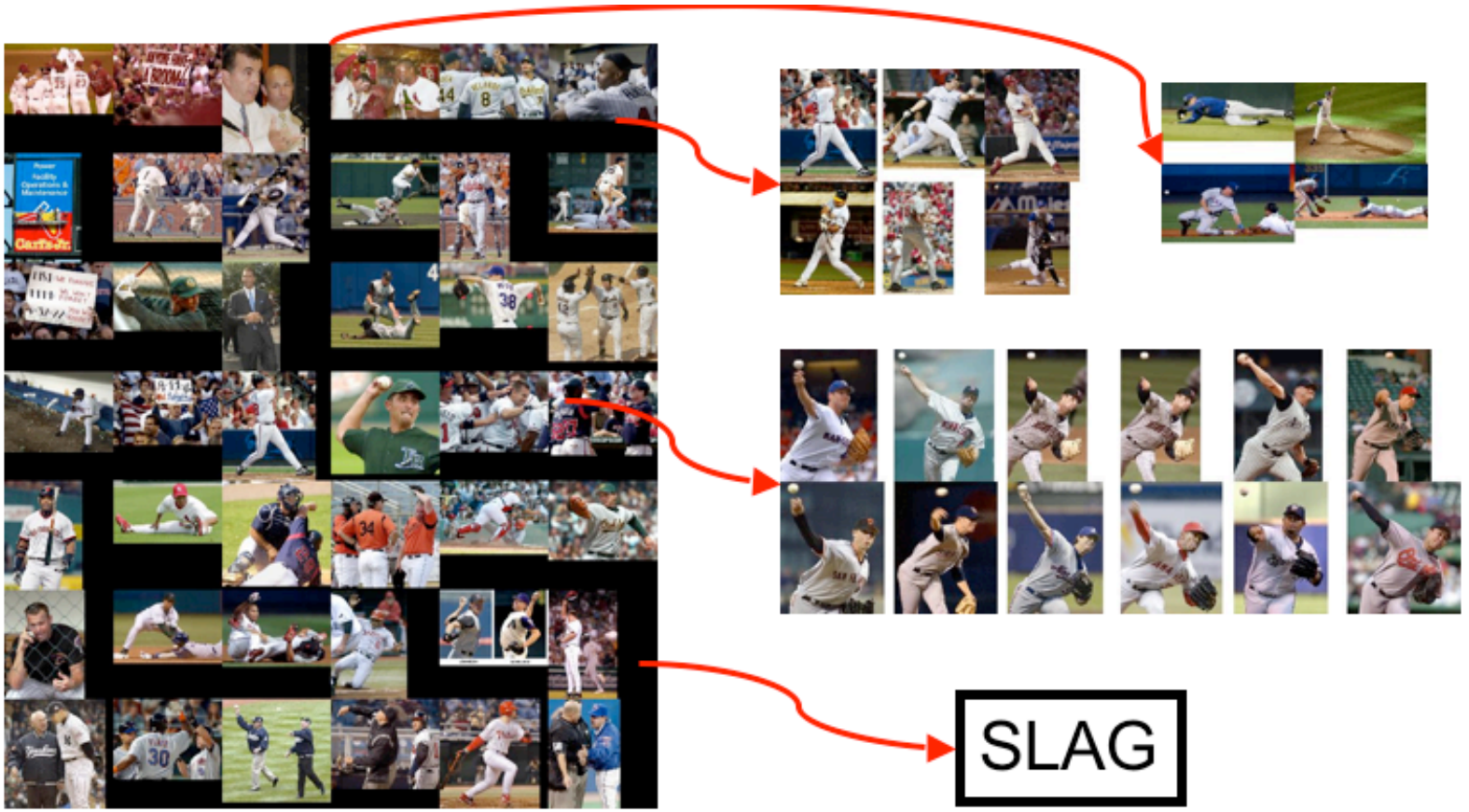
Moral of the story: ML provides theory and tools for setting parameters. Make sure you have the right model and features. Think about your "feature vector $\boldsymbol{x}$."

Stock Price Prediction



- Problems in which $\boldsymbol{t}_i$ is continuous are called regression
- E.g. $\boldsymbol{t}_i$ is stock price, $\boldsymbol{x}_i$ contains company profit, debt, cash flow, gross sales, number of spam emails sent, …

Clustering Images



Wang et al., CVPR 2006

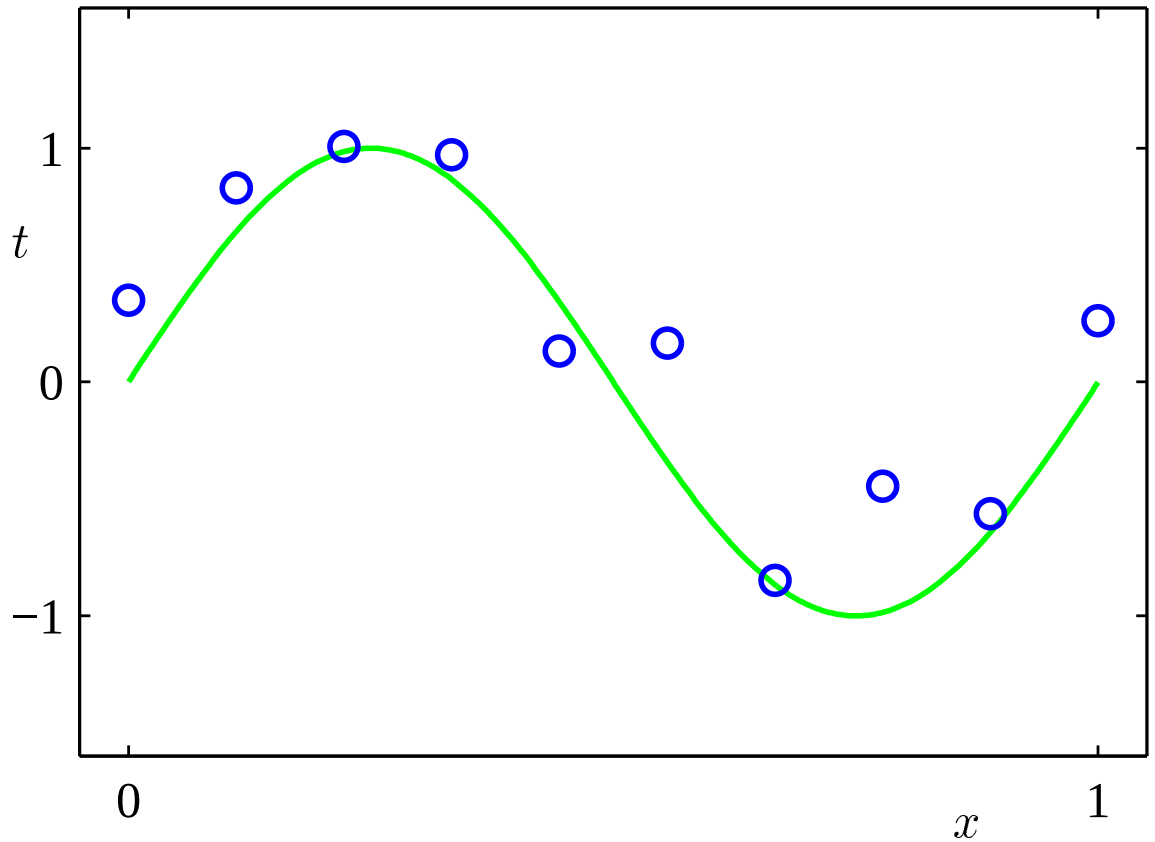
- Only $\boldsymbol{x}_i$ is defined: unsupervised learning
- E.g. $\boldsymbol{x}_i$ describes image, find groups of similar images

Types of Learning Problems

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Density estimation
  - Clustering: k-means, mixture models, hierarchical clustering
  - Hidden Markov models
- Reinforcement Learning

Outline

An Example - Polynomial Curve Fitting



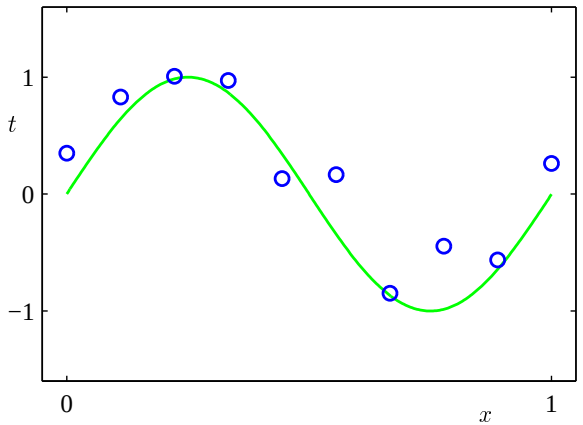
- Suppose we are given training set of N observations $(x_1, \ldots, x_N)$ and $(t_1, \ldots, t_N)$, $x_i, t_i \in \mathbb{R}$
- Regression problem, estimate $y(x)$ from these data

Polynomial Curve Fitting

- What form is $y(x)$?
  - Let's try polynomials of degree M:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

    - This is the hypothesis space.



- How do we measure success?
  - Sum of squared errors:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$

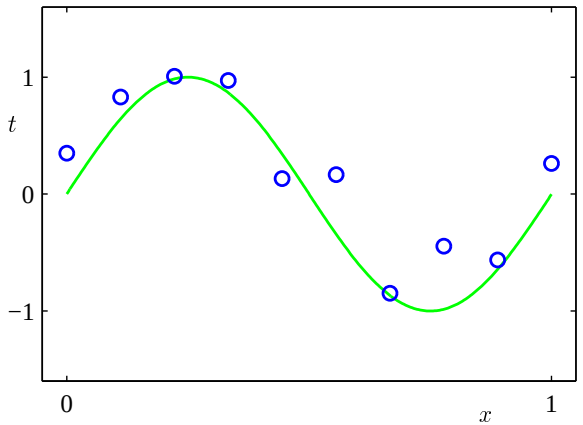
- Among functions in the class, choose that which minimizes this error

Polynomial Curve Fitting

- What form is $y(x)$?
  - Let's try polynomials of degree M:

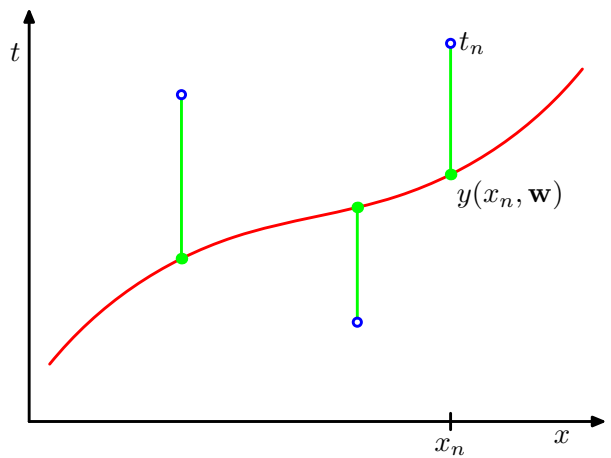
$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

    - This is the hypothesis space.



- How do we measure success?
  - Sum of squared errors:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$



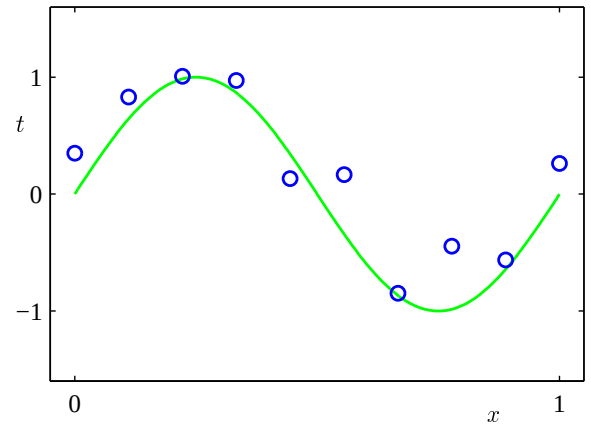
- Among functions in the class, choose that which minimizes this error

Polynomial Curve Fitting

- What form is $y(x)$?
  - Let's try polynomials of degree M:

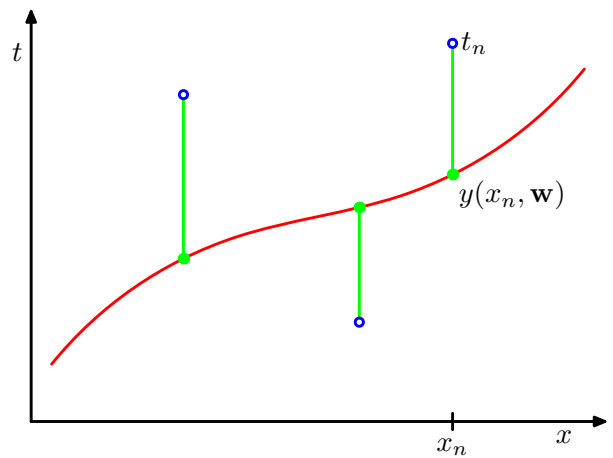
$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

    - This is the hypothesis space.



- How do we measure success?
  - Sum of squared errors:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \boldsymbol{w}) - t_n\}^2$$



- Among functions in the class, choose that which minimizes this error

Polynomial Curve Fitting

- Error function

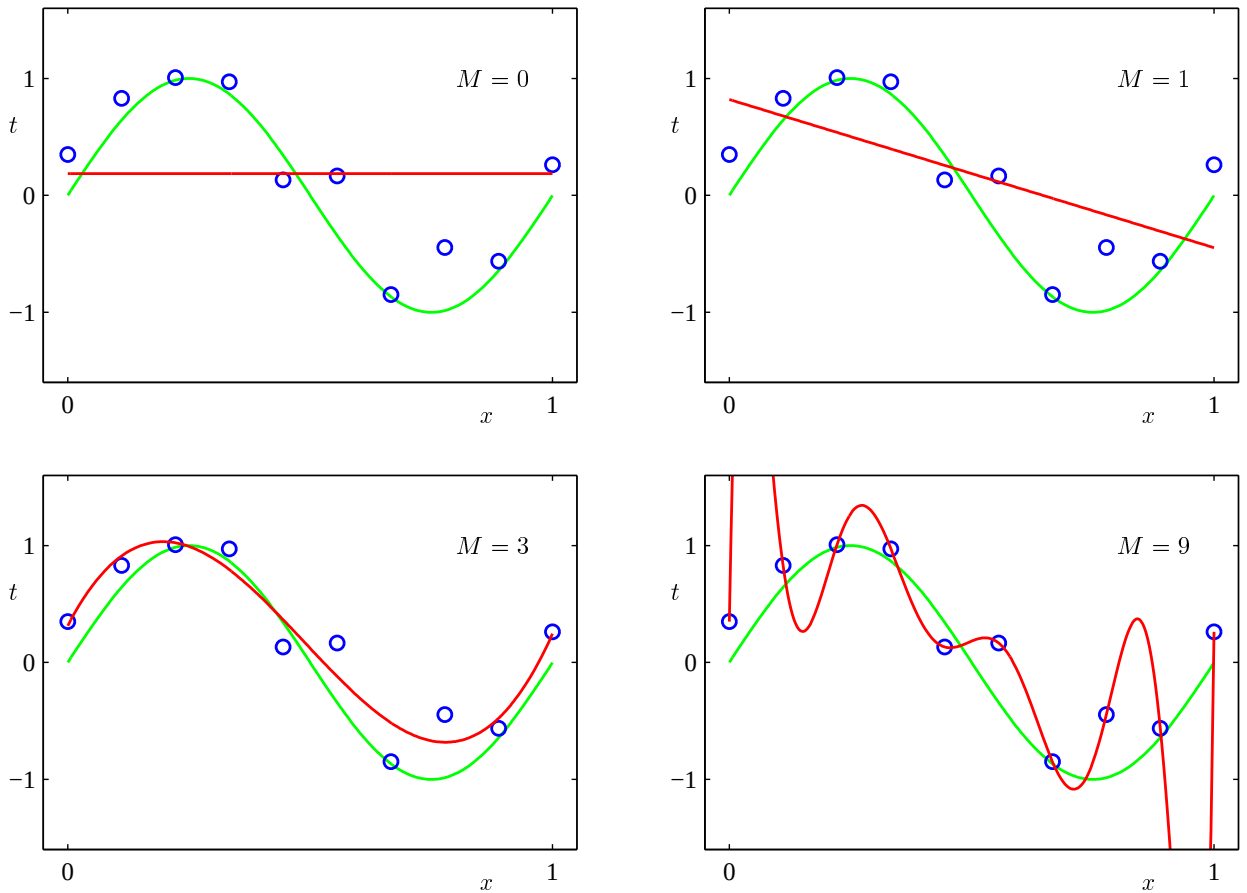
$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2$$

- Best coefficients

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$$

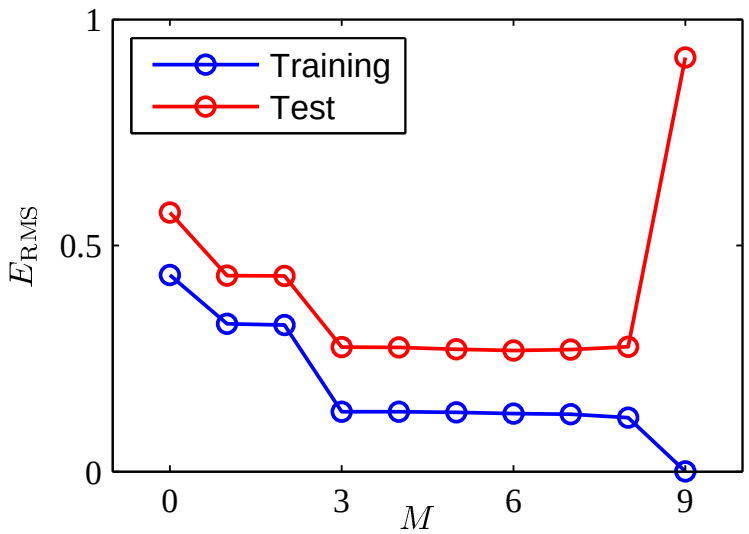
- Found using pseudo-inverse

Which Degree of Polynomial?



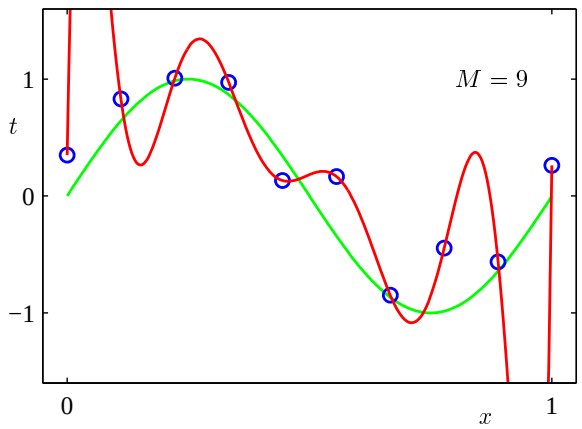
- A model selection problem
- $M = 9 \rightarrow E(\boldsymbol{w}^*) = 0$: This is over-fitting

Generalization



- Generalization is the holy grail of ML
  - Want good performance for new data
- Measure generalization using a separate set
  - Use root-mean-squared (RMS) error: $E_{RMS} = \sqrt{2E(\boldsymbol{w}^*)/N}$

Controlling Over-fitting: Regularization

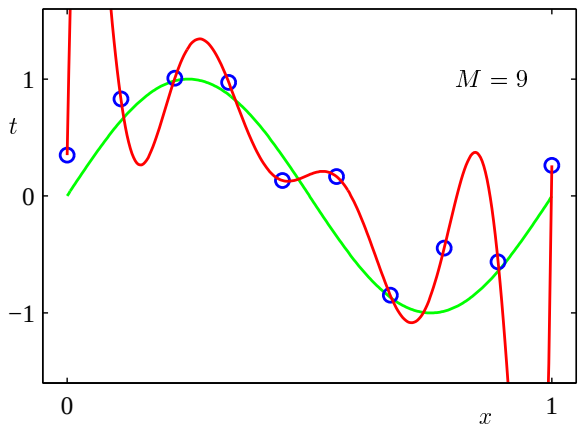


	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- As order of polynomial M increases, so do coefficient magnitudes
- Penalize large coefficients in error function:

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

Controlling Over-fitting: Regularization

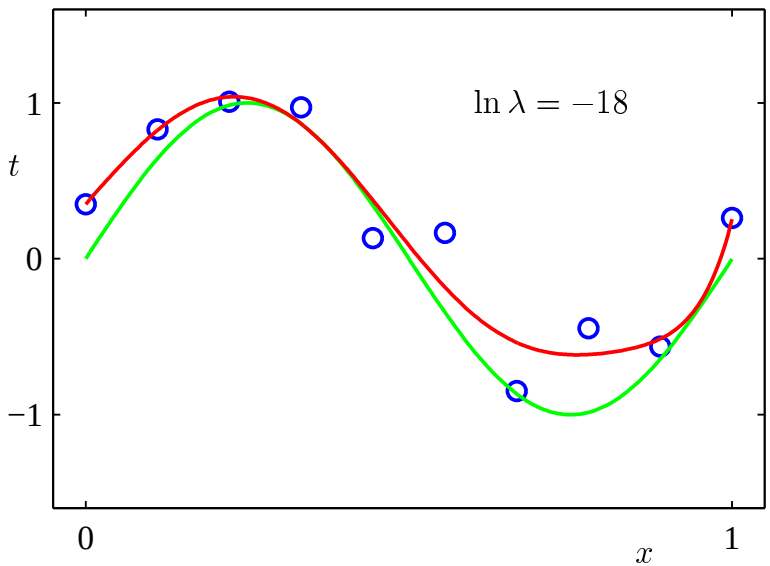


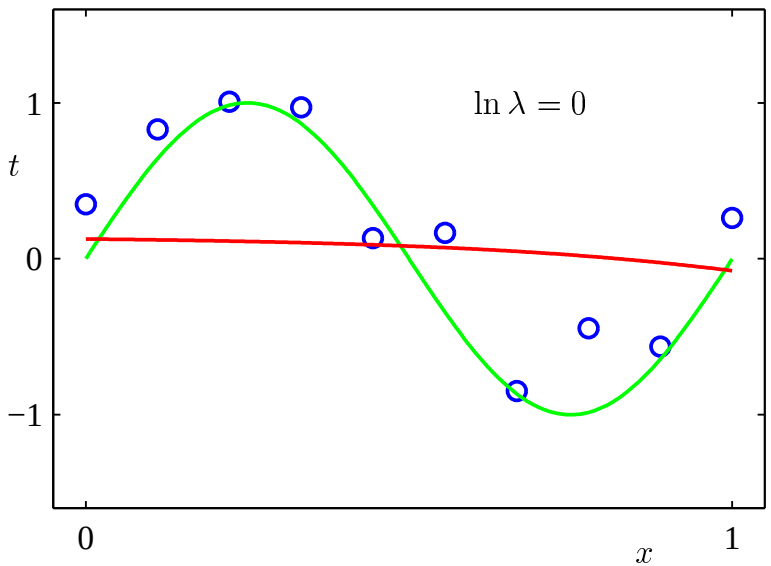
	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

- As order of polynomial M increases, so do coefficient magnitudes
- Penalize large coefficients in error function:

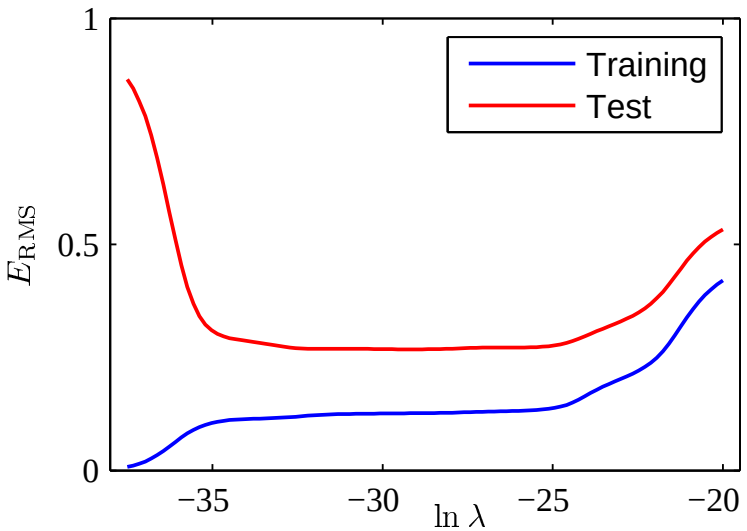
$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2$$

Controlling Over-fitting: Regularization





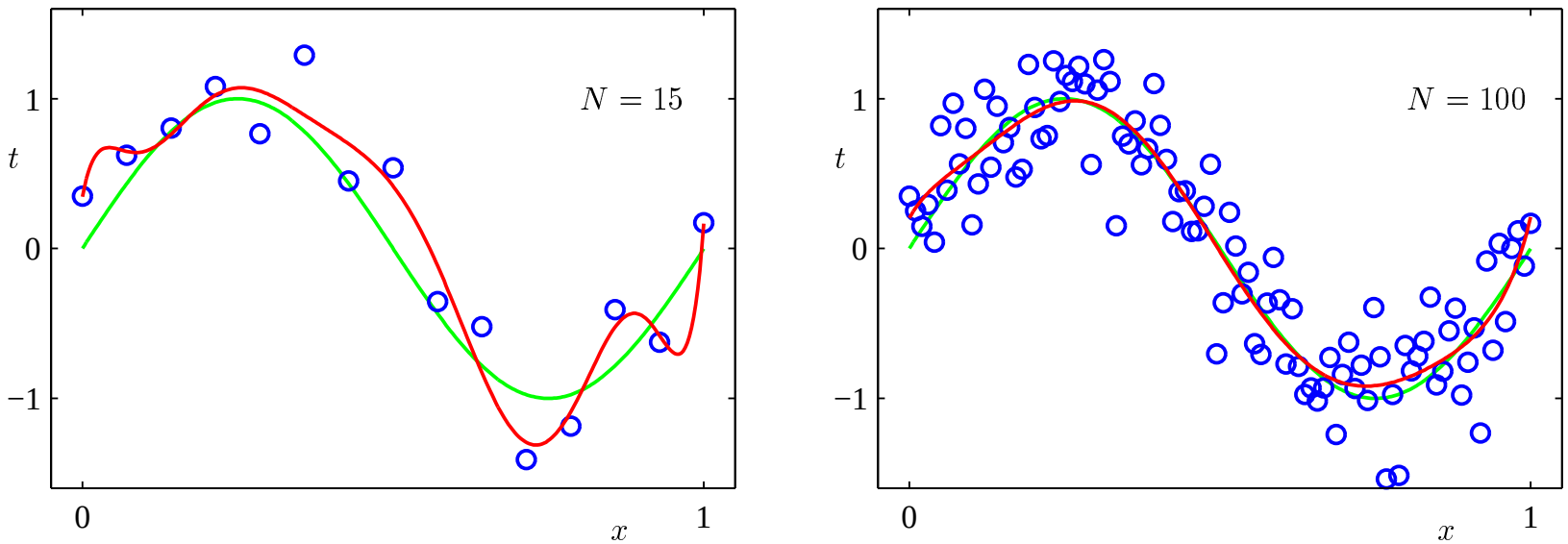
Controlling Over-fitting: Regularization



	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

- Note the E_{RMS} for the training set. Perfect match of training set with the model is a result of over-fitting
- Training and test error show similar trend

Over-fitting: Dataset size

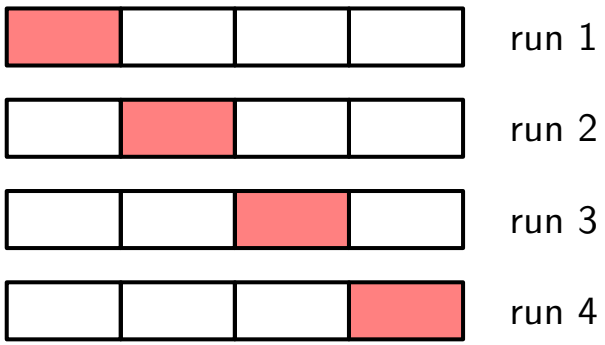


- With more data, more complex model ($M = 9$) can be fit
- Rule of thumb: 10 datapoints for each parameter

Validation Set

- Split training data into training set and validation set
- Train different models (e.g. diff. order polynomials) on training set
- Choose model (e.g. order of polynomial) with minimum error on validation set

Cross-validation



- Data are often limited
- Cross-validation creates S groups of data, use $S-1$ to train, other to validate
  - Extreme case leave-one-out cross-validation (LOO-CV): S is number of training data points
- Cross-validation is an effective method for model selection, but can be slow
  - Models with multiple complexity parameters: exponential number of runs

Summary

- Want models that generalize to new data
  - Train model on training set
  - Measure performance on held-out test set
    - Performance on test set is good estimate of performance on new data

Summary - Model Selection

- Which model to use? E.g. which degree polynomial?
  - Training set error is lower with more complex model
    - Can't just choose the model with lowest training error
  - Peeking at test error is unfair. E.g. picking polynomial with lowest test error
    - Performance on test set is no longer good estimate of performance on new data

Summary - Model Selection

- Which model to use? E.g. which degree polynomial?
  - Training set error is lower with more complex model
    - Can't just choose the model with lowest training error
  - Peeking at test error is unfair. E.g. picking polynomial with lowest test error
    - Performance on test set is no longer good estimate of performance on new data

Summary - Solutions I

- Use a validation set
  - Train models on training set. E.g. different degree polynomials
  - Measure performance on held-out validation set
  - Measure performance of that model on held-out test set
- Can use cross-validation on training set instead of a separate validation set if little data and lots of time
  - Choose model with lowest error over all cross-validation folds (e.g. polynomial degree)
  - Retrain that model using all training data (e.g. polynomial coefficients)

Summary - Solutions I

- Use a validation set
  - Train models on training set. E.g. different degree polynomials
  - Measure performance on held-out validation set
  - Measure performance of that model on held-out test set
- Can use cross-validation on training set instead of a separate validation set if little data and lots of time
  - Choose model with lowest error over all cross-validation folds (e.g. polynomial degree)
  - Retrain that model using all training data (e.g. polynomial coefficients)

Summary - Solutions II

- Use regularization
  - Train complex model (e.g high order polynomial) but penalize being “too complex” (e.g. large weight magnitudes)
  - Need to balance error vs. regularization (λ)
    - Choose λ using cross-validation
- Get more data

Summary - Solutions II

- Use regularization
  - Train complex model (e.g high order polynomial) but penalize being “too complex” (e.g. large weight magnitudes)
  - Need to balance error vs. regularization (λ)
    - Choose λ using cross-validation
- Get more data

Common Task Framework

- common to many competitions (like kaggle)
  1. A publicly available training dataset involving, for each observation, a list of (possibly many) feature measurements, and a class label for that observation.
  2. A set of enrolled competitors whose common task is to infer a class prediction rule from the training data.
  3. A scoring referee, to which competitors can submit their prediction rule. The referee runs the prediction rule against a testing dataset which is sequestered behind a Chinese wall. The referee objectively and automatically reports the score (prediction accuracy) achieved by the submitted rule

Outline

Machine Learning: What, Why, and How?

Curve Fitting: (e.g.) Regression and Model Selection

Goal Function: Decision Theory—ML, Loss Function, MAP

Probability Theory: (e.g.) Coin Tossing and Parameter Estimation

Conclusion

Goal Function – Decision Theory

For a sample $\boldsymbol{x}$, decide which class($\mathcal{C}_k$) it is from.

Ideas:

- Maximum Likelihood
- Minimum Loss/Cost (e.g. misclassification rate)
- Maximum Aposteriori (MAP)

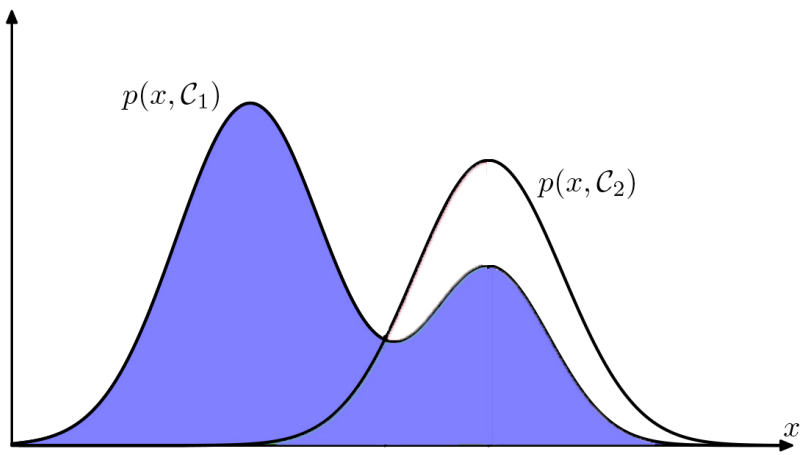
Decision: Maximum Likelihood

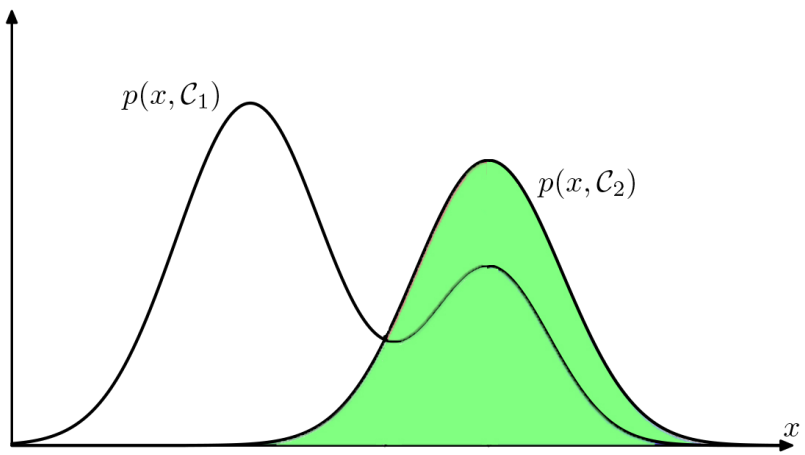
- **Inference step**: Determine statistics from training data.

$$p(\boldsymbol{x}, t) \quad \text{OR} \quad p(\boldsymbol{x}|\mathcal{C}_k)$$

- **Decision step**: Determine optimal t for test input x:

$$t = \arg\max_k \{ \underbrace{p(\boldsymbol{x}|\mathcal{C}_k)}_{Likelihood} \}$$





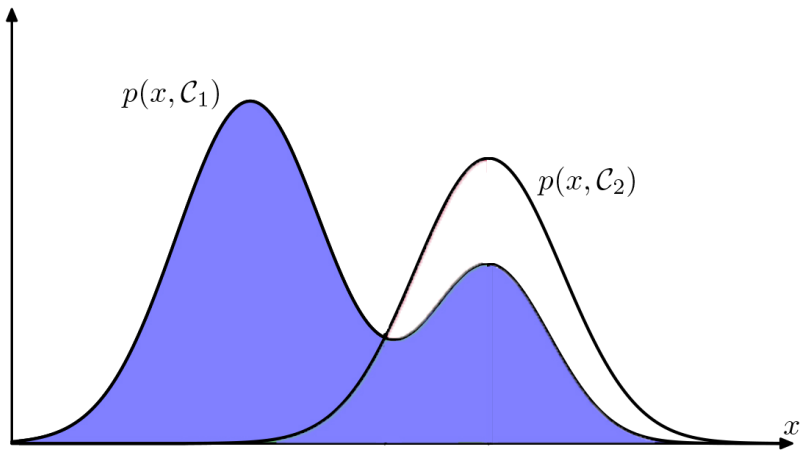
Decision: Maximum Likelihood

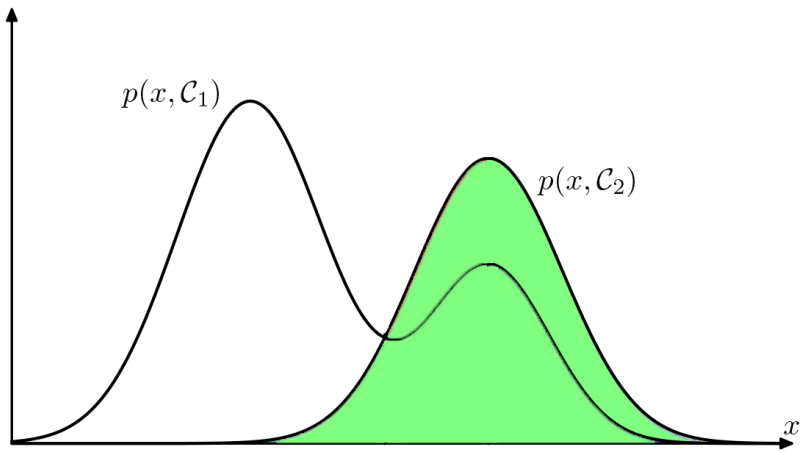
- **Inference step**: Determine statistics from training data.

$$p(\boldsymbol{x}, t) \quad \text{OR} \quad p(\boldsymbol{x}|\mathcal{C}_k)$$

- **Decision step**: Determine optimal t for test input x:

$$t = \arg\max_k \{ \quad \underbrace{p\left(\boldsymbol{x}|\mathcal{C}_k\right)}_{Likelihood} \quad \}$$





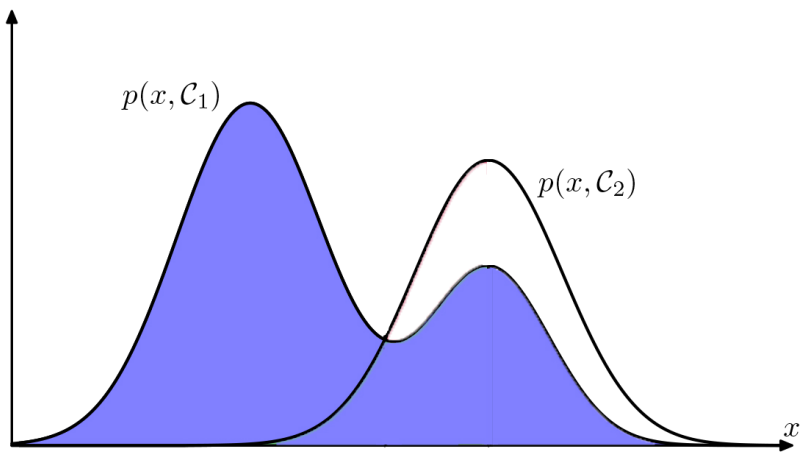
Decision: Maximum Likelihood

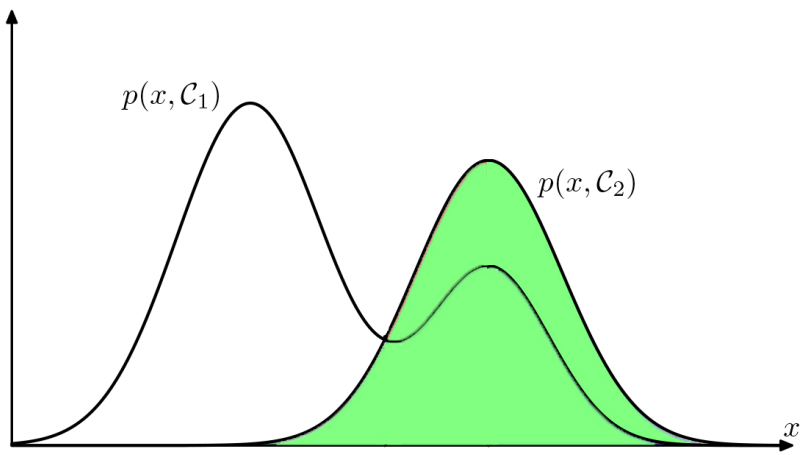
- **Inference step**: Determine statistics from training data.

$$p(\boldsymbol{x}, t) \quad \text{OR} \quad p(\boldsymbol{x}|\mathcal{C}_k)$$

- **Decision step**: Determine optimal t for test input x:

$$t = \arg\max_k \{ \quad \underbrace{p\left(\boldsymbol{x}|\mathcal{C}_k\right)}_{Likelihood} \quad \}$$

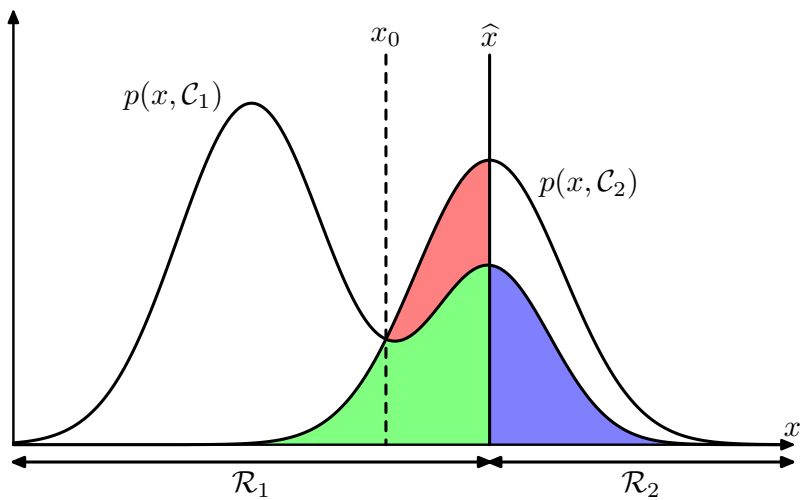




Decision: Minimum Misclassification Rate

$$\begin{aligned} p(\text{mistake}) &= p\left(\boldsymbol{x} \in \mathcal{R}_1, \mathcal{C}_2\right) + p\left(\boldsymbol{x} \in \mathcal{R}_2, \mathcal{C}_1\right) \\ &= \int_{\mathcal{R}_1} p\left(\boldsymbol{x}, \mathcal{C}_2\right) dx + \int_{\mathcal{R}_2} p\left(\boldsymbol{x}, \mathcal{C}_1\right) dx \end{aligned}$$

$$\boxed{p(\text{mistake}) = \sum_k \sum_j \int_{\mathcal{R}_j} p\left(\boldsymbol{x}, \mathcal{C}_k\right) dx}$$



$\hat{x}$: decision boundary.

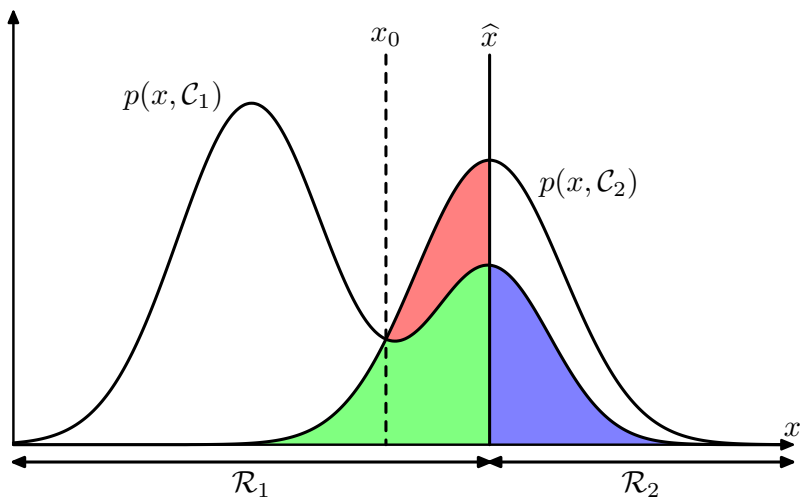
x_0: optimal decision boundary

$$x_0 : \arg\min_{\mathcal{R}_1} \{p\left(\text{mistake}\right)\}$$

Decision: Minimum Misclassification Rate

$$\begin{aligned} p(\text{mistake}) &= p\left(\boldsymbol{x} \in \mathcal{R}_1, \mathcal{C}_2\right) + p\left(\boldsymbol{x} \in \mathcal{R}_2, \mathcal{C}_1\right) \\ &= \int_{\mathcal{R}_1} p\left(\boldsymbol{x}, \mathcal{C}_2\right) dx + \int_{\mathcal{R}_2} p\left(\boldsymbol{x}, \mathcal{C}_1\right) dx \end{aligned}$$

$$\boxed{p(\text{mistake}) = \sum_k \sum_j \int_{\mathcal{R}_j} p\left(\boldsymbol{x}, \mathcal{C}_k\right) dx}$$



$\hat{x}$: decision boundary.

x_0: optimal decision boundary

$$x_0 : \arg\min_{\mathcal{R}_1} \{p\left(\text{mistake}\right)\}$$

Decision: Minimum Loss/Cost

- Misclassification rate:

$$\mathcal{R}: \underset{\{\mathcal{R}_i | i \in \{1,\cdots,K\}\}}{\arg\min} \left\{ \sum_k \sum_j \mathcal{L}\left(\mathcal{R}_j, \mathcal{C}_k\right) \right\}$$

- Weighted loss/cost function:

$$\mathcal{R}: \underset{\{\mathcal{R}_i | i \in \{1,\cdots,K\}\}}{\arg\min} \left\{ \sum_k \sum_j W_{j,k} \mathcal{L}\left(\mathcal{R}_j, \mathcal{C}_k\right) \right\}$$

Is useful when:

- The population of the classes are different
- The failure cost is non-symmetric
- ...

Decision: Maximum Aposteriori (MAP)

Bayes' Theorem: $\boxed{P\{A|B\} = \frac{P\{B|A\}P\{A\}}{P\{B\}}}$

$$\underbrace{p(\mathcal{C}_k|\boldsymbol{x})}_{Posterior} \propto \underbrace{p(\boldsymbol{x}|\mathcal{C}_k)}_{Likelihood} \underbrace{p(\mathcal{C}_k)}_{Prior}$$

- Provides an *Aposteriori Belief* for the estimation, rather than a single point estimate.
- Can utilize *Apriori Information* in the decision.

Outline

Machine Learning: What, Why, and How?

Curve Fitting: (e.g.) Regression and Model Selection

Goal Function: Decision Theory—ML, Loss Function, MAP

Probability Theory: (e.g.) Coin Tossing and Parameter Estimation

Conclusion

Coin Tossing

- Let's say you're given a coin, and you want to find out $P(heads)$, the probability that if you flip it it lands as "heads".
- Flip it a few times: $H\ H\ T$
- $P(heads) = 2/3$
- Hmm... is this rigorous? Does this make sense?

Coin Tossing

- Let's say you're given a coin, and you want to find out $P(heads)$, the probability that if you flip it it lands as "heads".
- Flip it a few times: $H\ H\ T$
- $P(heads) = 2/3$
- Hmm... is this rigorous? Does this make sense?

Coin Tossing

- Let's say you're given a coin, and you want to find out $P(heads)$, the probability that if you flip it it lands as "heads".
- Flip it a few times: $H\ H\ T$
- $P(heads) = 2/3$
- Hmm... is this rigorous? Does this make sense?

Coin Tossing - Model

- Bernoulli distribution $P(heads) = \mu$, $P(tails) = 1 - \mu$
- Assume coin flips are independent and identically distributed (i.i.d.)
  - i.e. All are separate samples from the Bernoulli distribution
- Given data $\mathcal{D} = \{x_1, \ldots, x_N\}$, heads: $x_i = 1$, tails: $x_i = 0$, the likelihood of the data is:

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

Maximum Likelihood Estimation

- Given $\mathcal{D}$ with h heads and t tails
- What should μ be?
- Maximum Likelihood Estimation (MLE): choose μ which maximizes the likelihood of the data

$$\mu_{ML} = \arg\max_{\mu} p(\mathcal{D}|\mu)$$

- Since $\ln(\cdot)$ is monotone increasing:

$$\mu_{ML} = \arg\max_{\mu} \ln p(\mathcal{D}|\mu)$$

Maximum Likelihood Estimation

- Likelihood:

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

- Log-likelihood:

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \ln \mu + (1-x_n)\ln(1-\mu)$$

- Take derivative, set to 0:

$$\frac{d}{d\mu}\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu} = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

$$\Rightarrow \mu = \frac{h}{t+h}$$

Maximum Likelihood Estimation

- Likelihood:

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

- Log-likelihood:

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \ln \mu + (1-x_n)\ln(1-\mu)$$

- Take derivative, set to 0:

$$\frac{d}{d\mu}\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu} = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

$$\Rightarrow \mu = \frac{h}{t+h}$$

Maximum Likelihood Estimation

- Likelihood:

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

- Log-likelihood:

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \ln \mu + (1-x_n)\ln(1-\mu)$$

- Take derivative, set to 0:

$$\frac{d}{d\mu}\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu} = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

$$\Rightarrow \mu = \frac{h}{t+h}$$

Maximum Likelihood Estimation

- Likelihood:

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

- Log-likelihood:

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \ln \mu + (1-x_n)\ln(1-\mu)$$

- Take derivative, set to 0:

$$\frac{d}{d\mu}\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu} = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

$$\Rightarrow \mu = \frac{h}{t+h}$$

Maximum Likelihood Estimation

- Likelihood:

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

- Log-likelihood:

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \ln \mu + (1-x_n)\ln(1-\mu)$$

- Take derivative, set to 0:

$$\frac{d}{d\mu}\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} x_n \frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu} = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

$$\Rightarrow \mu = \frac{h}{t+h}$$

Bayesian Learning

- Wait, does this make sense? What if I flip 1 time, heads? Do I believe μ=1?
- Learn μ the Bayesian way:

$$P(\mu|\mathcal{D}) = \frac{P(\mathcal{D}|\mu)P(\mu)}{P(\mathcal{D})}$$

$$\underbrace{P(\mu|\mathcal{D})}_{posterior} \propto \underbrace{P(\mathcal{D}|\mu)}_{likelihood}\underbrace{P(\mu)}_{prior}$$

- Prior encodes knowledge that most coins are 50-50
- Conjugate prior makes math simpler, easy interpretation
  - For Bernoulli, the beta distribution is its conjugate

Bayesian Learning

- Wait, does this make sense? What if I flip 1 time, heads? Do I believe μ=1?
- Learn μ the Bayesian way:

$$P(\mu|\mathcal{D}) = \frac{P(\mathcal{D}|\mu)P(\mu)}{P(\mathcal{D})}$$

$$\underbrace{P(\mu|\mathcal{D})}_{posterior} \propto \underbrace{P(\mathcal{D}|\mu)}_{likelihood}\underbrace{P(\mu)}_{prior}$$

- Prior encodes knowledge that most coins are 50-50
- Conjugate prior makes math simpler, easy interpretation
  - For Bernoulli, the beta distribution is its conjugate

Bayesian Learning

- Wait, does this make sense? What if I flip 1 time, heads? Do I believe μ=1?
- Learn μ the Bayesian way:

$$P(\mu|\mathcal{D}) = \frac{P(\mathcal{D}|\mu)P(\mu)}{P(\mathcal{D})}$$

$$\underbrace{P(\mu|\mathcal{D})}_{posterior} \propto \underbrace{P(\mathcal{D}|\mu)}_{likelihood}\underbrace{P(\mu)}_{prior}$$

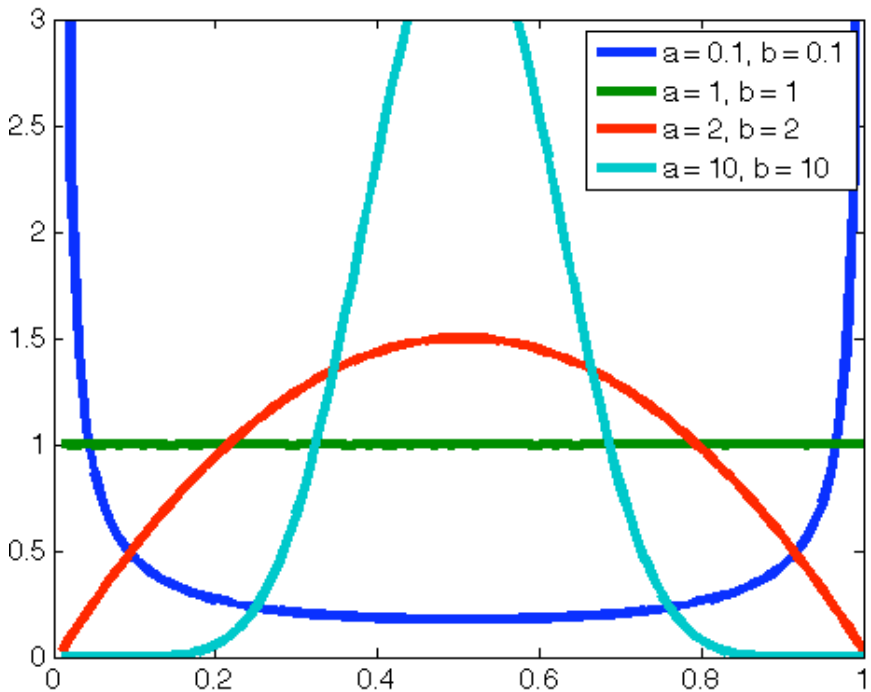
- Prior encodes knowledge that most coins are 50-50
- Conjugate prior makes math simpler, easy interpretation
  - For Bernoulli, the beta distribution is its conjugate

Beta Distribution

- We will use the Beta distribution to express our prior knowledge about coins:

$$Beta(\mu|a,b) = \underbrace{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}}_{normalization} \mu^{a-1}(1-\mu)^{b-1}$$

- Parameters a and b control the shape of this distribution



Posterior

$$
\begin{aligned}
P(\mu|\mathcal{D}) &\propto P(\mathcal{D}|\mu)P(\mu) \\
&\propto \underbrace{\prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}}_{likelihood} \underbrace{\mu^{a-1}(1-\mu)^{b-1}}_{prior} \\
&\propto \mu^{h}(1-\mu)^{t}\mu^{a-1}(1-\mu)^{b-1} \\
&\propto \mu^{h+a-1}(1-\mu)^{t+b-1}
\end{aligned}
$$

- Simple form for posterior is due to use of conjugate prior
- Parameters a and b act as extra observations
- Note that as $N = h + t \to \infty$, prior is ignored

Posterior

$$
\begin{aligned}
P(\mu|\mathcal{D}) &\propto P(\mathcal{D}|\mu)P(\mu) \\
&\propto \underbrace{\prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}}_{likelihood} \underbrace{\mu^{a-1}(1-\mu)^{b-1}}_{prior} \\
&\propto \mu^{h}(1-\mu)^{t}\mu^{a-1}(1-\mu)^{b-1} \\
&\propto \mu^{h+a-1}(1-\mu)^{t+b-1}
\end{aligned}
$$

- Simple form for posterior is due to use of conjugate prior
- Parameters a and b act as extra observations
- Note that as $N = h + t \to \infty$, prior is ignored

Posterior

$$
\begin{aligned}
P(\mu|\mathcal{D}) &\propto P(\mathcal{D}|\mu)P(\mu) \\
&\propto \underbrace{\prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}}_{likelihood} \underbrace{\mu^{a-1}(1-\mu)^{b-1}}_{prior} \\
&\propto \mu^{h}(1-\mu)^{t}\mu^{a-1}(1-\mu)^{b-1} \\
&\propto \mu^{h+a-1}(1-\mu)^{t+b-1}
\end{aligned}
$$

- Simple form for posterior is due to use of conjugate prior
- Parameters a and b act as extra observations
- Note that as $N = h + t \to \infty$, prior is ignored

Maximum A Posteriori

- Given posterior $P(\mu|\mathcal{D})$ we could compute a single value, known as the Maximum a Posteriori (MAP) estimate for μ:

$$\mu_{MAP} = \arg\max_{\mu} P(\mu|\mathcal{D})$$

  - Known as point estimation

- However, correct Bayesian thing to do is to use the full distribution over μ
  - i.e. Compute

$$\mathbb{E}_{\mu}[f] = \int p(\mu|\mathcal{D}) f(\mu) d\mu$$

  - This integral is usually hard to compute

Maximum A Posteriori

- Given posterior $P(\mu|\mathcal{D})$ we could compute a single value, known as the Maximum a Posteriori (MAP) estimate for μ:

$$\mu_{MAP} = \arg\max_{\mu} P(\mu|\mathcal{D})$$

  - Known as point estimation
- However, correct Bayesian thing to do is to use the full distribution over μ
  - i.e. Compute

$$\mathbb{E}_{\mu}[f] = \int p(\mu|\mathcal{D}) f(\mu) d\mu$$

  - This integral is usually hard to compute

Maximum A Posteriori

- Given posterior $P(\mu|\mathcal{D})$ we could compute a single value, known as the Maximum a Posteriori (MAP) estimate for μ:

$$\mu_{MAP} = \arg\max_{\mu} P(\mu|\mathcal{D})$$

  - Known as point estimation
- However, correct Bayesian thing to do is to use the full distribution over μ
  - i.e. Compute

$$\mathbb{E}_{\mu}[f] = \int p(\mu|\mathcal{D}) f(\mu) d\mu$$

  - This integral is usually hard to compute

Conclusion

- Types of learning problems
  - Supervised: regression, classification
  - Unsupervised
- Learning as optimization
  - Squared error loss function
  - Maximum likelihood (ML)
  - Maximum a posteriori (MAP)
- Want generalization, avoid over-fitting
  - Cross-validation
  - Regularization
  - Bayesian prior on model parameters
- see also Bishop: Chapters 1.1, 1.3, 1.5, 2.1