Introduction to Machine Learning

CMPT 726

Mo Chen

SFU Computing Science

Sept. 9, 2020

Bishop PRML Ch. 1

About Me

- Undergraduate from UBC
- PhD from UC Berkeley
- Postdoc at Stanford
- Assistant Professor at SFU CS since 2018
    - Multi-Agent Robotic Systems Lab (https://sfumars.com)

Outline

About This Course

Lectures: Mon 16:30-18:20, Wed 16:30-17:20, Canvas Bb Collab Ultra

- Website: https://canvas.sfu.ca/courses/56287

Piazza online discussion and Q&A :

- Links to sign-up and home page on website

Office hours:

- Tentatively Fridays 14:00-15:00, Zoom link on website
- See website for TA office hours

Teaching Assistants

- Xubo Lyu (xlv@sfu.ca)
- Roshni Shaik (rshaik@sfu.ca)
- Ria Thomas (riat@sfu.ca)

Administrivia

- We will cover techniques in the standard ML toolkit
  - maximum likelihood, regularization, neural networks, Markov random fields (MRF), graphical models, expectation-maximization (EM), mixture models, hidden Markov models (HMM), particle filters, ...
- There will be 3 assignments and 1 project
  - Assignment 0: 5%
  - Assignment 1-3: 15% each
  - Project: 50%
- No exams!

Administrivia

- Recommend doing associated readings from Bishop, *Pattern Recognition and Machine Learning* (PRML) after each lecture
  - Other reference books
    - *The Elements of Statistical Learning*, Trevor Hastie, Robert Tibshirani, and Jerome Friedman
    - *Machine Learning*, Tom Mitchell
    - *Pattern Classification (2nd ed.)*, Richard O. Duda, Peter E. Hart, and David G. Stork
    - *Information Theory, Inference, and Learning Algorithms*, David MacKay (available online)
    - *Deep Learning*, Ian Goodfellow, Yoshua Bengio and Aaron Courville (available online)
  - Online courses
    - Coursera, Udacity

Administrivia - Assignments

- Assignments will be monthly
- No late submissions other than exceptional circumstances (contact me directly)!
- Programming questions mainly use Python

Administrivia – Assignment Peer Grading

- Double blind peer grading:
  - Up to five peers will grade your assignment according to detailed solutions and rubrics
  - Assignment mark will be a combination of
    - Scores from peers (75%)
    - Quality of grading (25%)
    - Insightful comments (5% bonus)
  - Disputes to be handled directly by teaching staff
- Some benefits:
  - Frees up teaching staff's time for more help with assignments and projects
  - Ensures assignments are thoroughly reviewed
  - Ensures thorough review of assignment solutions

Administrivia – Assignments

- Assignment 0: introduce yourself to 5 peer reviewers
  - Becoming familiar with Canvas and peer review
  - Will not really be double blind this time
  - Assignment due Sept. 14 (next Monday)
  - Peer review due Sept. 18 (next Friday)
- Exact deadlines: always 23:59 on the specified date
- Assignment 1:
  - Assigned Sept. 18, due Oct. 9, reviews due Oct. 16
- Assignment 2:
  - Assigned Oct. 16, due Nov. 6, reviews due Nov. 13
- Assignment 3:
  - Assigned Nov. 13, due Dec. 4, reviews due Dec. 11

Administrivia - Project

- Project details
  - Practice doing research
  - Ideal project – take problem from your research/interests, use ML (properly)
  - We will provide a list of *pre-approved* projects proposed by TAs and SFU research labs
    - This gives some of you a chance to work with SFU research labs
  - Other projects will need to be approved: Check with a TA to make sure your project is reasonable/feasible

Administrivia - Project

- Project details
  - Work in groups (2-3 students)
  - Produce (short) research paper
  - Graded on proper research methodology, not just results
    - Choice of problem / algorithms
    - Relation to previous work
    - Comparative experiments
    - Quality of exposition
  - Students must list their contribution to the project within the group
  - Details on course webpage
    - Poster session/presentations TBA
    - Report due Dec. 16 at 23:59

Administrivia - Background

- Calculus:

$$E = mc^2 \Rightarrow \frac{\partial E}{\partial c} = 2mc$$

- Linear algebra:

$$Au_i = \lambda_i u_i;\ \frac{\partial}{\partial x}(x^\top a) = a$$

  - See PRML Appendix C

- Probability:

$$p(X) = \sum_Y p(X,Y);\ \ p(x) = \int p(x,y)dy;\ \ \mathbb{E}_x[f] = \int p(x)f(x)dx$$

  - See PRML Ch. 1.2

It will be possible to refresh, but if you've never seen these before this course will be **very** difficult.

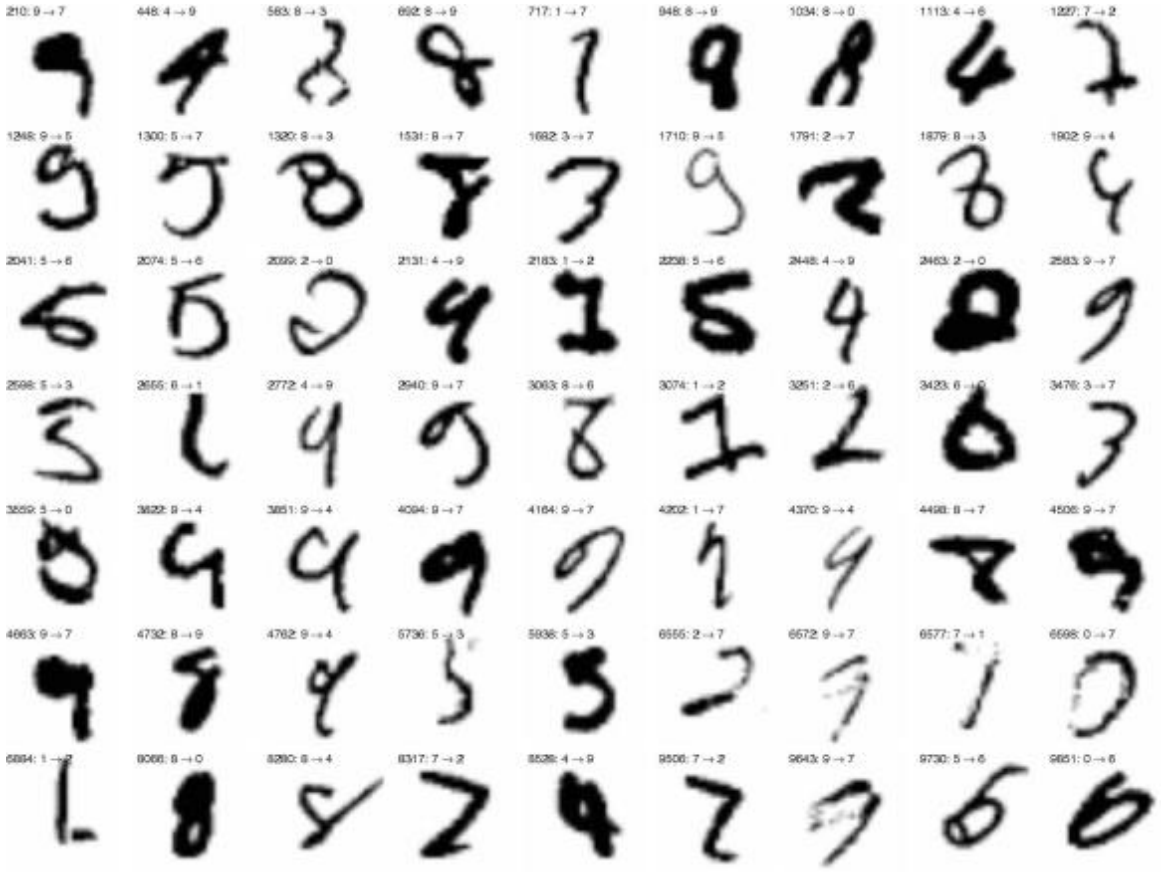
What is Machine Learning (ML)?

- 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
  - ...

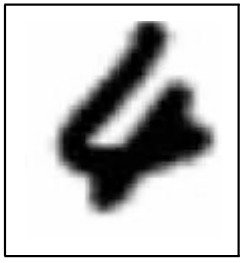
Hand-written Digit Recognition



Belongie et al. PAMI 2002

- Difficult to hand-craft rules about digits

Hand-written Digit Recognition

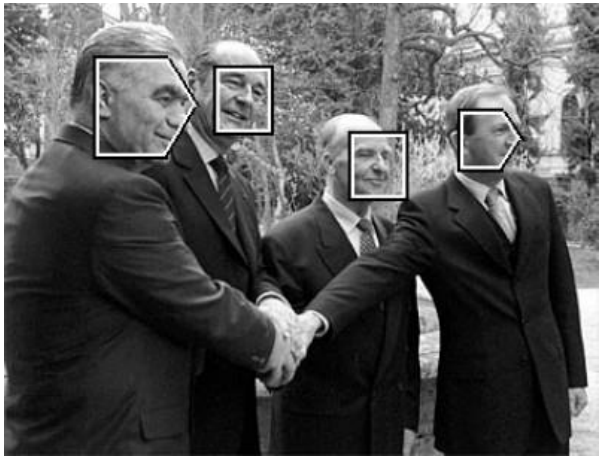
$\boldsymbol{x}_i =$ 

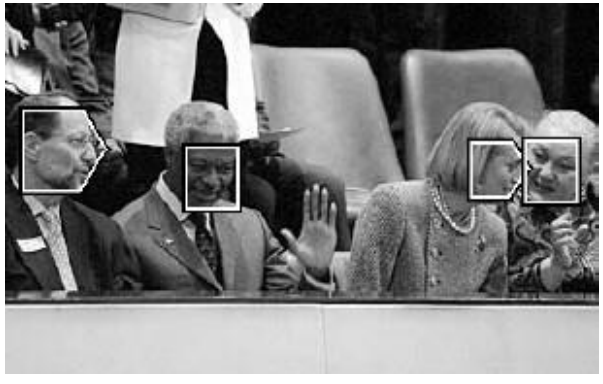
$\boldsymbol{t}_i = (0,0,0,1,0,0,0,0,0,0)$

- 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

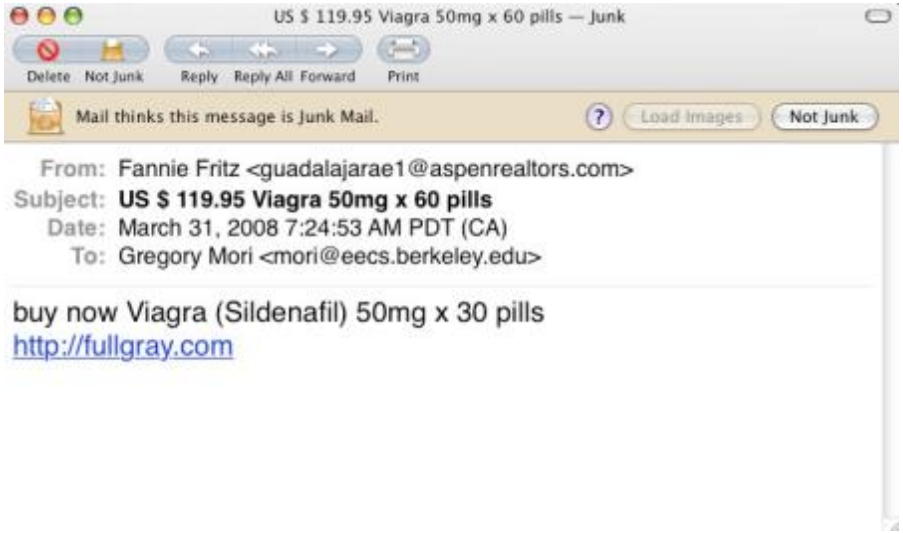


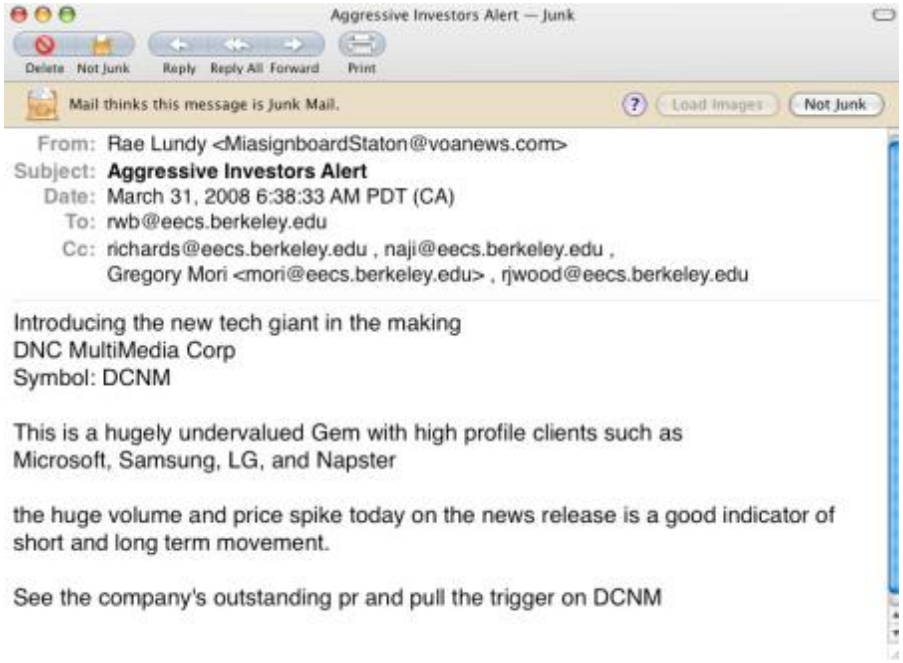


Schneiderman and Kanade, IJCV 2004

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

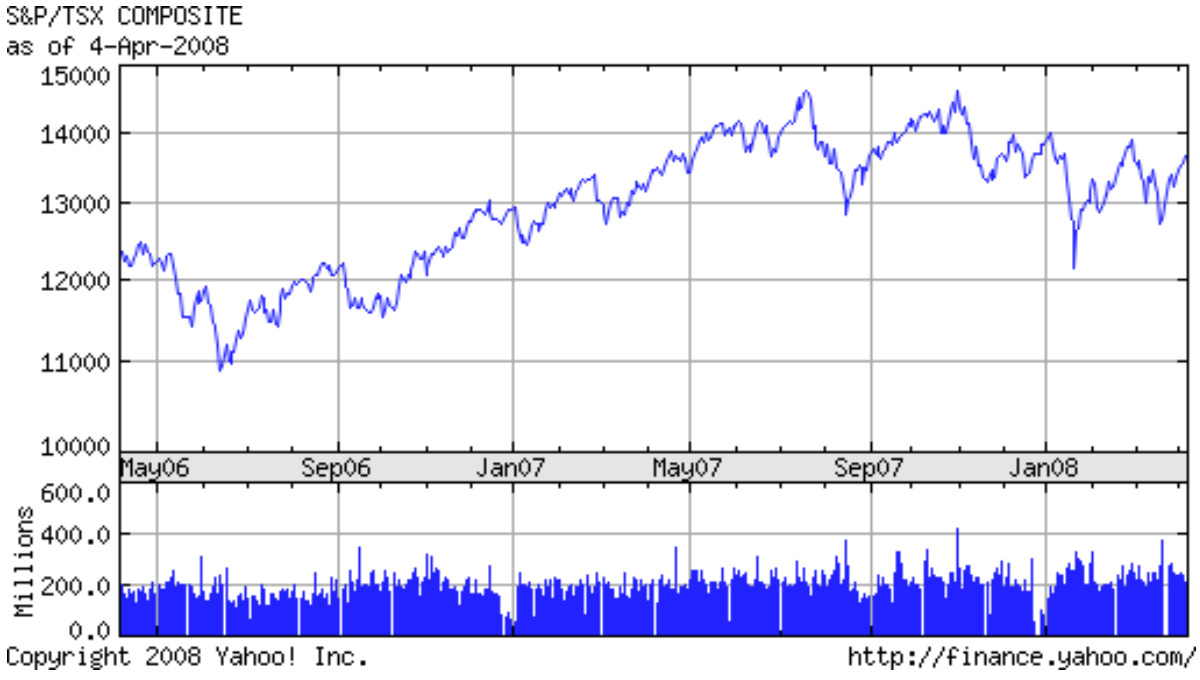
Spam Detection





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

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

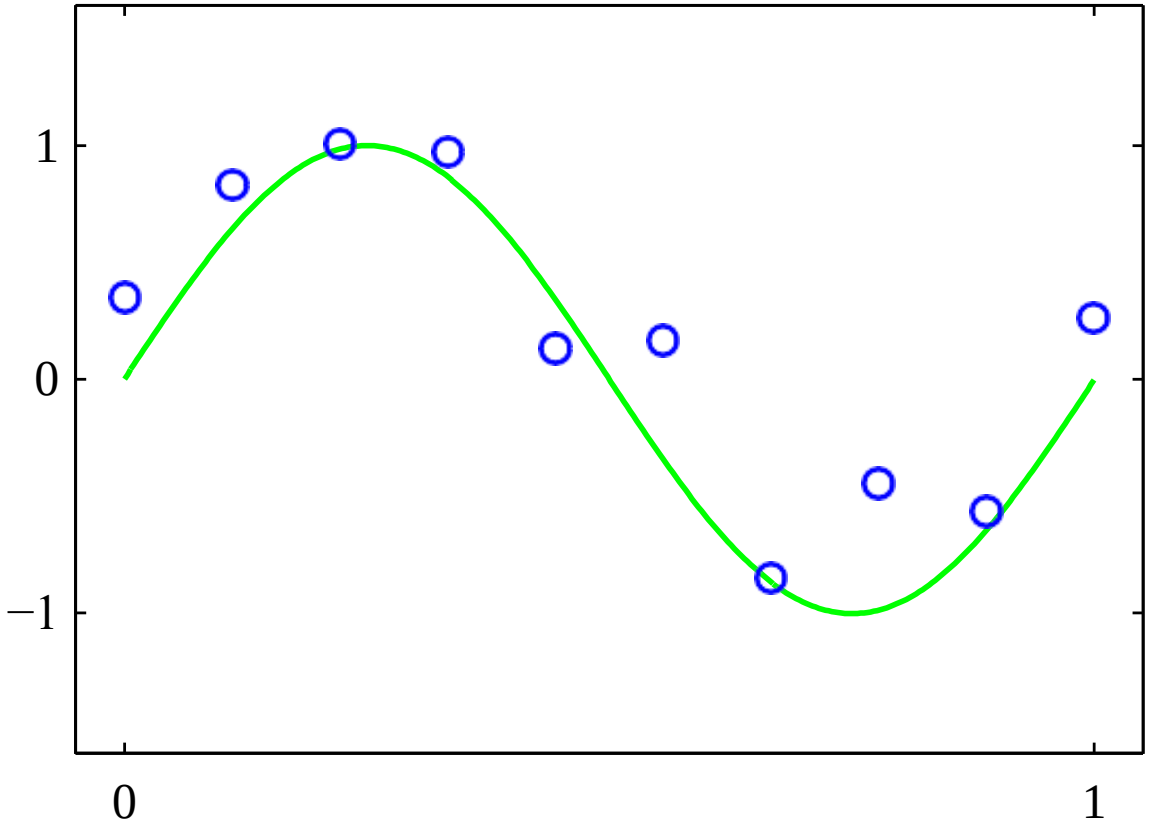
Wang et al., CVPR 2006

- 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
- Reinforcement Learning

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

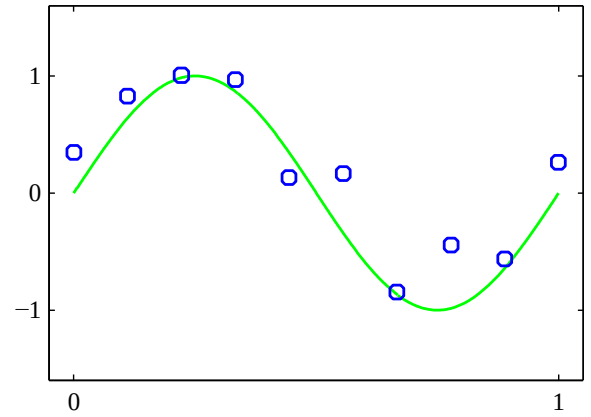
- What form is $y(x)$?
  - Let's try polynomials of degree M:

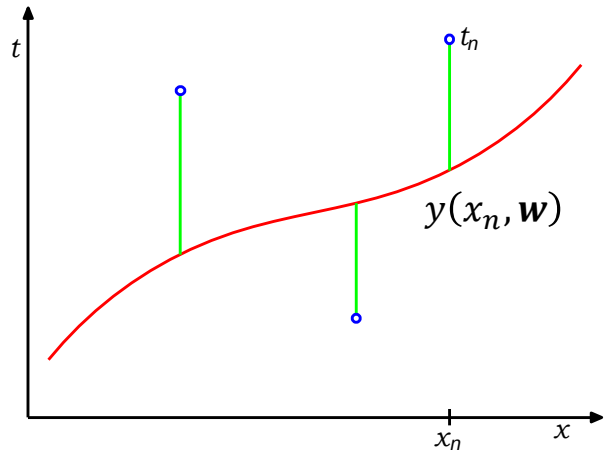
$$y(x, w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$

    - This is the hypothesis space.
- How do we measure success?
  - Sum of squared errors:

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

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





Polynomial Curve Fitting

- Error function

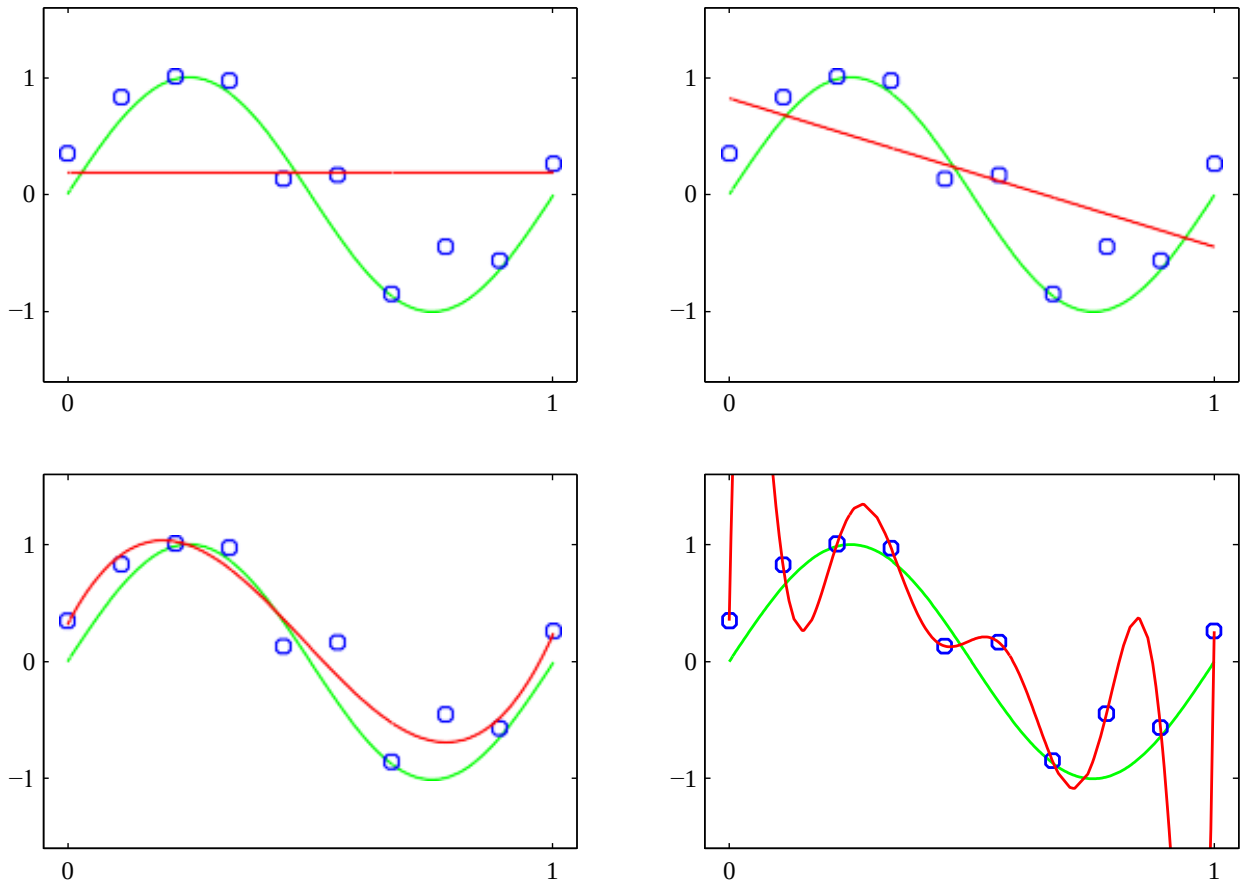
$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2$$

- Best coefficients

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

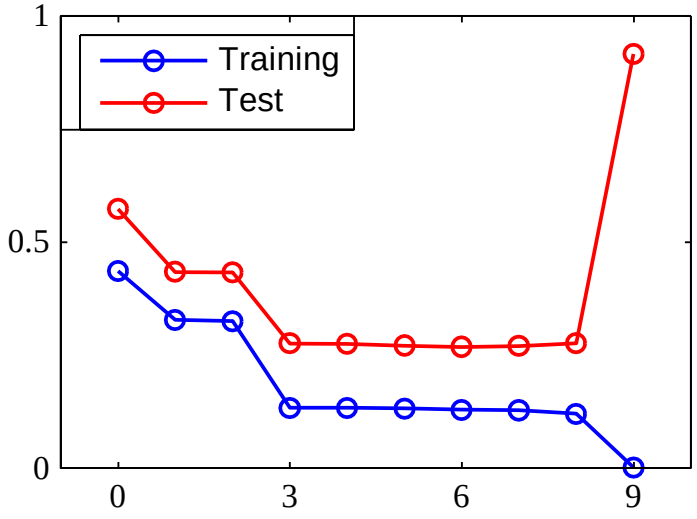
- Found using pseudo-inverse (more later)

Which Degree of Polynomial?



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

Generalization

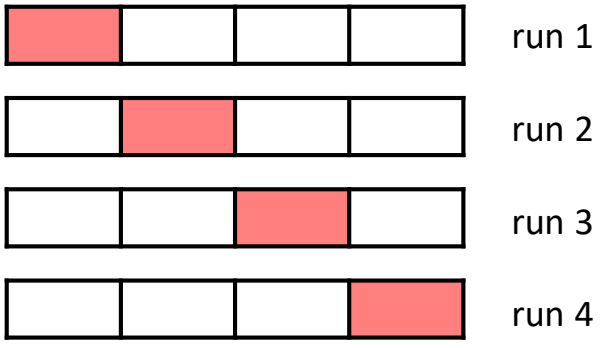


- Generalization is the holy grail of ML
  - Want good performance for new data
- Measure generalization using a separate set
  - Use root-mean-squared (RMS) error: $E_{\mathrm{RMS}} = \sqrt{2E(\boldsymbol{w}^*)/N}$

Validation Set

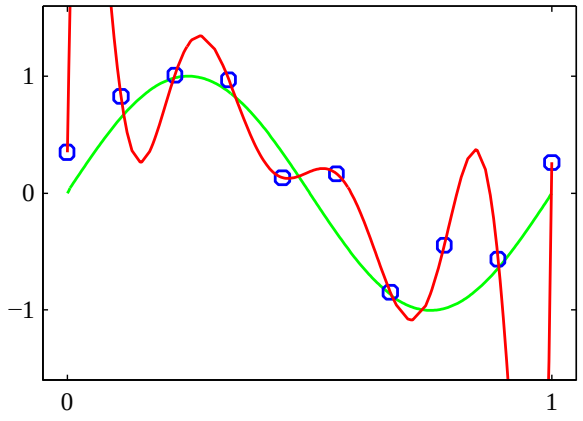
- Split training data into training set and validation set
- Train different models (e.g. diff. order polynomials) on training set
- Choose model (e.g. order of polynomial) with minimum error on validation set

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

Controlling Over-fitting: Regularization

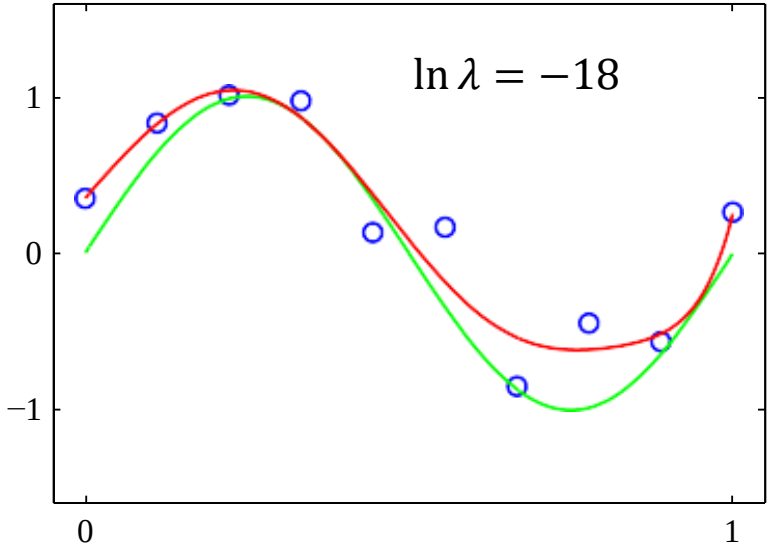


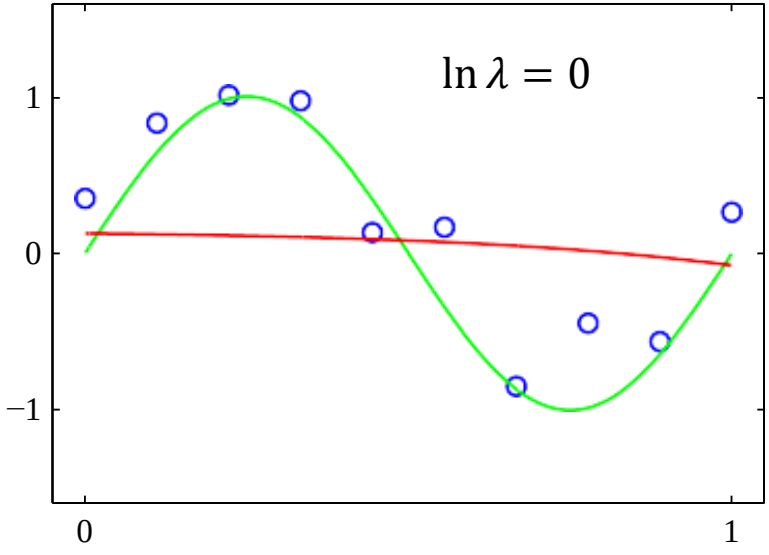
	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

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

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

Controlling Over-fitting: Regularization

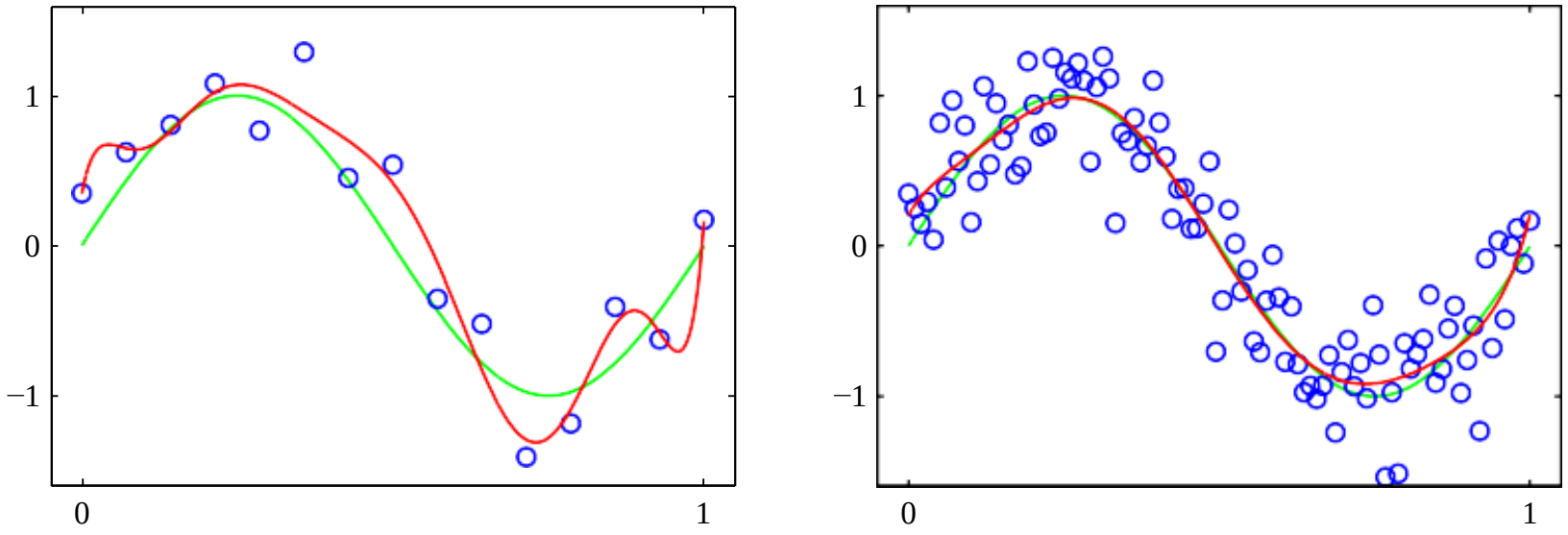




	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$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

Fits for $M = 9$

Over-fitting: Dataset size



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

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 - 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

Bayesianity

- Frequentist view – probabilities are frequencies of random, repeatable events
- Bayesian view – probability quantifies uncertain beliefs
- Important distinction for us: Bayesianity allows us to discuss probability distributions over parameters (such as $\boldsymbol{w}$)
  - Include priors (e.g. $p(\boldsymbol{w})$) over model parameters
- Later, we will see Bayesian approaches to combatting over-fitting and model selection for curve fitting
- For now, an illustrative example . . .

Coin Tossing

- Let's say you're given a coin, and you want to find out $P(H)$, the probability that if you flip it it lands as "heads".
- Flip it a few times: $H\,H\,T$
- $P(H) = 2/3$, no need for CMPT 726
- Hmm... is this rigorous? Does this make sense?

Coin Tossing - Model

- Bernoulli distribution $P(H) = \mu, P(T) = 1 - \mu$
- 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 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:

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

- Take derivative, set to 0:

$$\frac{d}{d\mu}\ln p(\mathrm{D}|\mu) = \sum_{n=1}^{N}\left[x_n\frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu}\right] = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

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

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:

$$\frac{d}{d\mu}\ln p(\mathrm{D}|\mu) = \sum_{n=1}^{N}\left[x_n\frac{1}{\mu} - (1-x_n)\frac{1}{1-\mu}\right] = \frac{1}{\mu}h - \frac{1}{1-\mu}t$$

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

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:

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

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

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:

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

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

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:

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

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

Bayesian Learning

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

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

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

- 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 $\mu = 1$?
- Learn μ the Bayesian way:

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

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

- 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 $\mu = 1$?
- Learn μ the Bayesian way:

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

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

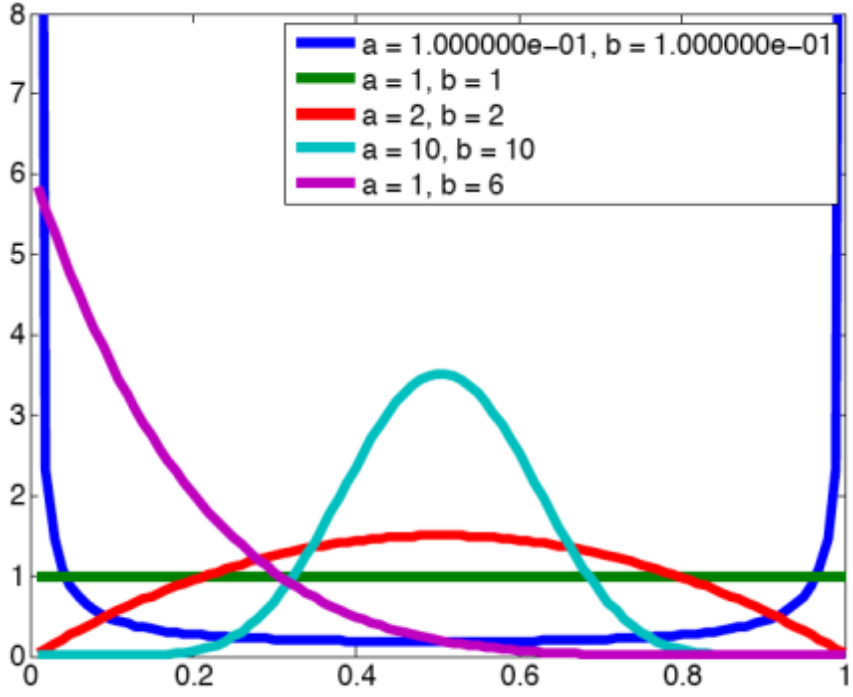
- Prior encodes knowledge that most coins are 50-50
- Conjugate prior makes math simpler, easy interpretation
  - For Bernoulli, the beta distribution is its conjugate

Beta Distribution

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

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

- Parameters a and b control the shape of this distribution



Posterior

$$P(\mu|\mathcal{D}) \propto P(\mathcal{D}|\mu)P(\mu)$$

$$\propto \underbrace{\prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}}_{\text{likelihood}} \underbrace{\mu^{(a-1)}(1-\mu)^{b-1}}_{\text{prior}}$$

$$\propto \mu^{h}(1-\mu)^{t}\mu^{(a-1)}(1-\mu)^{b-1}$$

$$\propto \mu^{h+a-1}(1-\mu)^{t+b-1}$$

- 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

$$P(\mu|\mathcal{D}) \propto P(\mathcal{D}|\mu)P(\mu)$$

$$\propto \underbrace{\prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}}_{\text{likelihood}} \underbrace{\mu^{(a-1)}(1-\mu)^{b-1}}_{\text{prior}}$$

$$\propto \mu^{h}(1-\mu)^{t}\mu^{(a-1)}(1-\mu)^{b-1}$$

$$\propto \mu^{h+a-1}(1-\mu)^{t+b-1}$$

- 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

$$P(\mu|\mathcal{D}) \propto P(\mathcal{D}|\mu)P(\mu)$$

$$\propto \underbrace{\prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}}_{\text{likelihood}} \underbrace{\mu^{(a-1)}(1-\mu)^{b-1}}_{\text{prior}}$$

$$\propto \mu^{h}(1-\mu)^{t}\mu^{(a-1)}(1-\mu)^{b-1}$$

$$\propto \mu^{h+a-1}(1-\mu)^{t+b-1}$$

- 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

Bayesian Learning

- 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

Bayesian Learning

- 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

- Readings: Ch. 1.1-1.3, 2.1
- 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