Introduction to Machine Learning  
Greg Mori - CMPT 419/726

Bishop PRML Ch. 1

Outline

Administrivia

Machine Learning

Curve Fitting

Coin Tossing

Administrivia

- We will cover techniques in the standard ML toolkit
  - maximum likelihood, regularization, support vector machines (SVM), neural networks, Fisher's linear discriminant (LDA), boosting, principal components analysis (PCA), Markov random fields (MRF), graphical models, belief propagation, expectation-maximization (EM), mixture models, mixtures of experts (MoE), hidden Markov models (HMM), particle filters, Markov Chain Monte Carlo (MCMC), Gibbs sampling, ...

- There will be 3 assignments
- Exams in class on Oct. 24 and Dec. 2

Administrivia

- Recommend doing associated readings from Bishop, *Pattern Recognition and Machine Learning* (PRML) after each lecture
- Reference books for alternate descriptions
  - *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

- Assignment late policy
  - 3 grace days, use at your discretion (not on project)
- Programming assignments use Python

Administrivia - Project

- Project details
  - Practice doing research
  - Ideal project – take problem from your research/interests, use ML (properly)
  - Other projects fine too ($1 million project: http://netflixprize.com)
  - Too late :
  - Others on http://www.kaggle.com

Administrivia - Project

- Project details
  - Work in groups (up to 5 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
  - Details on course webpage
    - Poster session Dec. 5, 10:30am-12:30pm in TASC1
    - Report due Dec. 9 at 11:59pm

Administrivia - Background

- Calculus:
  \[ E = mc^2 \Rightarrow \frac{\partial E}{\partial c} = 2mc \]
- Linear algebra:
  \[ Au_i = \lambda_i u_i; \quad \frac{\partial}{\partial x}(x^T a) = a \]
  - See PRML Appendix C
- Probability:
  \[ p(X) = \sum_Y p(X,Y); \quad p(x) = \int p(x,y)dy; \quad \mathbb{E}[f] = \int p(x)f(x)dx \]
  - 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

- Difficult to hand-craft rules about digits

\[ x_i = \begin{bmatrix} 1 \end{bmatrix}, \quad t_i = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0) \]

- Represent input image as a vector \( x_i \in \mathbb{R}^{784} \).
- Suppose we have a target vector \( t_i \)
  - This is supervised learning
  - Discrete, finite label set: perhaps \( t_i \in \{0, 1\}^{10} \), a classification problem
- Given a training set \( \{(x_1, t_1), \ldots, (x_N, t_N)\} \), learning problem is to construct a “good” function \( y(x) \) from these.
  - \( y : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10} \)
• Classification problem
• $t_i \in \{0, 1, 2\}$, non-face, frontal face, profile face.

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!

• Classification problem
• $t_i \in \{0, 1\}$, non-spam, spam
• $x_i$ counts of words, e.g. Viagra, stock, outperform, multi-bagger

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 $x$.”
Problems in which $t_i$ is continuous are called **regression**

E.g. $t_i$ is stock price, $x_i$ contains company profit, debt, cash flow, gross sales, number of spam emails sent, ...
**Polynomial Curve Fitting**

- What form is $y(x)$?
  - Let’s try polynomials of degree $M$:
    $$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$
  - This is the hypothesis space.

- How do we measure success?
  - Sum of squared errors:
    $$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2$$
  - Among functions in the class, choose that which minimizes this error

- Error function
  $$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2$$
  - Best coefficients
    $$\mathbf{w}^* = \arg \min_{\mathbf{w}} E(\mathbf{w})$$
  - Found using pseudo-inverse (more later)

**Which Degree of Polynomial?**

- A model selection problem
  - $M = 9 \rightarrow E(\mathbf{w}^*) = 0$: 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: $E_{RMS} = \sqrt{\frac{1}{2}E(\mathbf{w}^*)/N}$
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

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

As order of polynomial $M$ increases, so do coefficient magnitudes
Penalize large coefficients in error function:
\[
\tilde{E}(w) = \frac{1}{2} \sum_{n=1}^{N} [y(x_n, w) - t_n]^2 + \frac{\lambda}{2} ||w||^2
\]
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 ($\lambda$)
    - Choose $\lambda$ 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 $w$)
  - Include priors (e.g., $p(w)$) over model parameters
- Later, we will see Bayesian approaches to combating 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 $P(\text{heads})$, the probability that if you flip it it lands as "heads".
- Flip it a few times: $H H T$
- $P(\text{heads}) = 2/3$, no need for CMPT726
- Hmm... is this rigorous? Does this make sense?

Coin Tossing - Model

- Bernoulli distribution $P(\text{heads}) = \mu$, $P(\text{tails}) = 1 - \mu$
- Assume coin flips are independent and identically distributed (i.i.d.)
  - i.e. All are separate samples from the Bernoulli distribution
- Given data $D = \{x_1, \ldots, x_N\}$, heads: $x_i = 1$, tails: $x_i = 0$, the likelihood of the data is:
  $$p(D|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n} (1 - \mu)^{1-x_n}$$
Maximum Likelihood Estimation

- Given $D$ with $h$ heads and $t$ tails
- What should $\mu$ be?
- Maximum Likelihood Estimation (MLE): choose $\mu$ which maximizes the likelihood of the data
  $$\mu_{ML} = \arg \max_{\mu} p(D|\mu)$$
- Since $\ln(\cdot)$ is monotone increasing:
  $$\mu_{ML} = \arg \max_{\mu} \ln p(D|\mu)$$

Bayesian Learning

- Wait, does this make sense? What if I flip 1 time, heads?
  Do I believe $\mu=1$?
- Learn $\mu$ the Bayesian way:
  $$P(\mu|D) = \frac{P(D|\mu)P(\mu)}{P(D)}$$
  $$\propto \frac{P(D|\mu)P(\mu)}{\text{likelihood prior}}$$
- 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:
  $$\text{Beta}(\mu|a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} \mu^{a-1}(1-\mu)^{b-1}$$
- Parameters $a$ and $b$ control the shape of this distribution
### Posterior

\[ P(\mu|\mathcal{D}) \propto P(\mathcal{D}|\mu)P(\mu) \]
\[ \propto \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n} \mu^{a-1}(1-\mu)^{b-1} \]
\[ \propto \mu^{a}(1-\mu)^{b-1} \]
\[ \propto \mu^{h+a-1}(1-\mu)^{t+b-1} \]

- 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 \( \mu \):
  \[ \mu_{MAP} = \arg \max_{\mu} P(\mu|\mathcal{D}) \]
- Known as point estimation

### Bayesian Learning

- However, correct Bayesian thing to do is to use the full distribution over \( \mu \)
  - i.e. Compute
  \[ E_{\mu}[f] = \int p(\mu|\mathcal{D})f(\mu)d\mu \]
- 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