Practical Issues in Learning
-- Overfitting and Model Selection

Eric Xing

Lecture 7, October 3, 2006

Reading: Chap. 1 & 2, CB & Chap 5, 6, TM

Outline

- Overfitting
  - Instance-based learning
  - Regression

- Bias-variance decomposition

- The battle against overfitting:
  each learning algorithm has some "free knobs" that one can "tune"
  (i.e., heck) to make the algorithm generalizes better to test data.

  But is there a more principled way?
  - Cross validation
  - Regularization
  - Model selection --- Occam's razor
  - Model averaging
    - The Bayesian-frequentist debate
    - Bayesian learning (weight models by their posterior probabilities)
Recall: Vector Space Representation

- Each document is a vector, one component for each term (= word).

<table>
<thead>
<tr>
<th></th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>Word 2</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>Word 3</td>
<td>12</td>
<td>1</td>
<td>10</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>

- Normalize to unit length.
- High-dimensional vector space:
  - Terms are axes, 10,000+ dimensions, or even 100,000+
  - Docs are vectors in this space

Classes in a Vector Space

- Sports
- Science
- Arts
Test Document = ?

K-Nearest Neighbor (kNN) classifier
kNN is an instance of Instance-Based Learning

- What makes an Instance-Based Learner?
  - A distance metric
  - How many nearby neighbors to look at?
  - A weighting function (optional)
  - How to relate to the local points?

Euclidean Distance Metric

\[ D(x, x') = \sqrt{\sum_i \sigma_i^2 (x_i - x'_i)^2} \]

- Or equivalently,
  \[ D(x, x') = \sqrt{(x - x')^T \Sigma (x - x')} \]

- Other metrics:
  - L1 norm: |x-x'|
  - L∞ norm: max |x-x'| (elementwise …)
  - Mahalanobis: where \( \Sigma \) is full, and symmetric
  - Correlation
  - Angle
  - Hamming distance, Manhattan distance
  - …
1-Nearest Neighbor (kNN) classifier

2-Nearest Neighbor (kNN) classifier
3-Nearest Neighbor (kNN) classifier

5-Nearest Neighbor (kNN) classifier
Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in $D$.

- Testing instance $x$:
  - Compute similarity between $x$ and all examples in $D$.
  - Assign $x$ the category of the most similar example in $D$.

- Does not explicitly compute a generalization or category prototypes.

- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning

kNN Is Close to Optimal

- Cover and Hart 1967

- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the Bayes rate [error rate of classifier knowing model that generated data]

- In particular, asymptotic error rate is 0 if Bayes rate is 0.

- Decision boundary:
Overfitting

Another example:

- Regression
Overfitting, con'd

- The models:

- Test errors:

Bias-variance decomposition

- Now let's look more closely into two sources of errors in an functional approximator:

- In the following we show the Bias-variance decomposition using LR as an example.
Loss functions for regression

- Let $t$ be the true (target) output and $y(x)$ be our estimate. The expected squared loss is
  \[ E(L) = \int \int L(t, y(x)) p(x,t) dx dt \]
  \[ = \int \int (t - y(x))^2 p(x,t) dx dt \]

- Our goal is to choose $y(x)$ that minimize $E(L)$:
  - Calculus of variations:
    \[
    \frac{\partial E(L)}{\partial y(x)} = 2 \int (t - y(x)) p(x,t) dt = 0
    \]
    \[ y(x)p(x,t) dt = \int tp(x,t) dt \]
    \[ y^*(x) = \frac{\int p(x,t) dt}{p(x)} \] is the optimal predictor, and $y(x)$ our actual predictor, which will incur the following expected loss

Expected loss

- Let $h(x) = E[t|x]$ be the optimal predictor, and $y(x)$ our actual predictor, which will incur the following expected loss
  \[ E(y(x) - t)^2 = \int (y(x) - h(x))^2 |x| dx dt \]
  \[ = \int (y(x) - h(x))^2 + 2(y(x) - h(x))h(x) + h(x)^2 |x| dx dt \]
  \[ = \int (y(x) - h(x))^2 |x| dx + \int (h(x) - t)^2 p(x,t) dx dt \]
  There is an error on pp47

- $\int (h(x) - t)^2 p(x,t) dx dt$ is a noisy term, and we can do no better than this. Thus it is a lower bound of the expected loss.
- The other part of the error comes from $\int (y(x) - h(x))^2 p(x,t) dx$ , and let's take a close look of it.
- We will assume $y(x) = y(x|w)$ is a parametric model and the parameters $w$ are fit to a training set $D$. (thus we write $y(x;D)$)
Bias-variance decomposition

- For one data set $D$ and one test point $x$
  - since the predictor $y$ depend on the data training data $D$, write $E_D[y(x,D)]$ for the expected predictor over the ensemble of datasets, then (using the same trick) we have:
    \[ (y(x;D) - h(x))^2 = (y(x;D) - E_D[y(x;D)])^2 + h(x)^2 \]
    \[ = (y(x;D) - E_D[y(x;D)])^2 + [E_D[y(x;D)] - h(x)]^2 \]
    \[ + 2(y(x;D) - E_D[y(x;D)][E_D[y(x;D)] - h(x)] \]
  - Surely this error term depends on the training data, so we take an expectation over them:
    \[ E_D[(y(x;D) - h(x))^2] = (E_D[y(x;D)] - h(x))^2 + E_D[y(x;D) - E_D[y(x;D)]^2 \]
  - Putting things together:
    \[ \text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise} \]

Regularized Regression

\[ J(\theta, x, y) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 + \frac{1}{2} \lambda \| \theta \| \]
Bias-variance tradeoff

- $\lambda$ is a "regularization" terms in LR, the smaller the $\lambda$, is more complex the model (why?)
  - Simple (highly regularized) models have low variance but high bias.
  - Complex models have low bias but high variance.

- You are inspecting an empirical average over 100 training set.
- The actual $E_D$ can not be computed.

Bias2+variance vs regularizer

- Bias$^2$+variance predicts (shape of) test error quite well.
- However, bias and variance cannot be computed since it relies on knowing the true distribution of $x$ and $t$ (and hence $h(x) = E[t|x]$).
The battle against overfitting

Model Selection

- Suppose we are trying to select among several different models for a learning problem.
- Examples:
  1. Polynomial regression
     \[ h(x; \theta) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k) \]
     - Model selection: we wish to automatically and objectively decide if \( k \) should be, say, 0, 1, \ldots, or 10.
  2. Locally weighted regression,
     - Model selection: we want to automatically choose the bandwidth parameter \( \tau \).
  3. Mixture models and hidden Markov model,
     - Model selection: we want to decide the number of hidden states
- The Problem:
  - Given model family \( \mathcal{F} = \{M_1, M_2, \ldots, M_k\} \), find \( M_i \in \mathcal{F} \) s.t. \( M_i = \arg \max_{M \in \mathcal{F}} J(D, M) \)
Cross Validation

- We are given training data $D$ and test data $D_{\text{test}}$, and we would like to fit this data with a model $p_{i}(x; \theta)$ from the family $\mathcal{F}$ (e.g., an LR), which is indexed by $i$ and parameterized by $\theta$.

- $K$-fold cross-validation (CV)
  - Set aside $\alpha N$ samples of $D$ (where $N = |D|$). This is known as the held-out data and will be used to evaluate different values of $i$.
  - For each candidate model $i$, fit the optimal hypothesis $p_{i}(x; \theta^*)$ to the remaining $(1-\alpha)N$ samples in $D$ (i.e., hold $i$ fixed and find the best $\theta$).
  - Evaluate each model $p_{i}(x; \theta^*)$ on the held-out data using some pre-specified risk function.
  - Repeat the above $K$ times, choosing a different held-out data set each time, and the scores are averaged for each model $p_{i}(.)$ over all held-out data set. This gives an estimate of the risk curve of models over different $i$.
  - For the model with the lowest risk, say $p_{i^*}(.)$, we use all of $D$ to find the parameter values for $p_{i^*}(x; \theta^*)$.

Example:

- When $\alpha = 1/N$, the algorithm is known as Leave-One-Out-Cross-Validation (LOOCV)

\[ \text{MSE}_{\text{LOOCV}}(M_2) = 0.962 \]
\[ \text{MSE}_{\text{LOOCV}}(M_1) = 2.12 \]
Practical issues for CV

- How to decide the values for $K$ and $\alpha$
  - Commonly used $K = 10$ and $\alpha = 0.1$.
  - When data sets are small relative to the number of models that are being evaluated, we need to decrease $\alpha$ and increase $K$.
  - $K$ needs to be large for the variance to be small enough, but this makes it time-consuming.

- Bias-variance trade-off
  - Small $\alpha$ usually lead to low bias. In principle, LOOCV provides an almost unbiased estimate of the generalization ability of a classifier, especially when the number of the available training samples is severely limited; but it can also have high variance.
  - Large $\alpha$ can reduce variance, but will lead to under-use of data, and causing high-bias.

- One important point is that the test data $D_{\text{test}}$ is never used in CV, because doing so would result in overly (indeed dishonest) optimistic accuracy rates during the testing phase.
Regularization

- Maximum-likelihood estimates are not always the best (James and Stein showed a counter example in the early 60’s)
- Alternative: we "regularize" the likelihood objective (also known as penalized likelihood, shrinkage, smoothing, etc.), by adding to it a penalty term:

\[
\hat{\theta}_{\text{shrinkage}} = \arg \max_{\theta} \left[ l(\theta; D) + \lambda \| \theta \| \right]
\]

where \( \lambda > 0 \) and \( \| \theta \| \) might be the \( L_1 \) or \( L_2 \) norm.

- The choice of norm has an effect
  - using the \( L_2 \) norm pulls directly towards the origin
  - while using the \( L_1 \) norm pulls towards the coordinate axes, i.e. it tries to set some of the coordinates to 0.
  - This second approach can be useful in a feature-selection setting.

Bayesian and Frequentist

- Frequentist interpretation of probability
  - Probabilities are objective properties of the real world, and refer to limiting relative frequencies (e.g., number of times I have observed heads). Hence one cannot write \( P(\text{Katrina could have been prevented}|D) \), since the event will never repeat.
  - Parameters of models are fixed, unknown constants. Hence one cannot write \( P(\theta|D) \) since \( \theta \) does not have a probability distribution. Instead one can only write \( P(D|\theta) \).
  - One computes point estimates of parameters using various estimators, \( \hat{\theta} = f(D) \), which are designed to have various desirable qualities when averaged over future data \( D \) (assumed to be drawn from the “true” distribution).

- Bayesian interpretation of probability
  - Probability describes degrees of belief, not limiting frequencies.
  - Parameters of models are hidden variables, so one can compute \( P(\theta|D) \) or \( P(f(\theta)|D) \) for some function \( f \).
  - One estimates parameters by computing \( P(\theta|D) \) using Bayes rule:

\[
p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}
\]
Bayesian interpretation of regulation

- Regularized Linear Regression
  - Recall that using squared error as the cost function results in the LMS estimate
  - And assume iid data and Gaussian noise, LMS is equivalent to MLE of $\theta$
    \[
    l(\theta) = n \log \frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2
    \]
  - Now assume that vector $\theta$ follows a normal prior with 0-mean and a diagonal covariance matrix
    \[
    \theta - N(0, \tau^2 I)
    \]
  - What is the posterior distribution of $\theta$?
    \[
    p(\theta|D) = \frac{p(D|\theta) p(\theta)}{p(D)} = \frac{p(D|\theta)}{\int p(D|\theta) p(\theta) d\theta} = \frac{(2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2\right)}{C \exp\left(-\frac{1}{2\tau^2} \theta^T \theta\right)}
    \]

Bayesian interpretation of regulation, con’d

- The posterior distribution of $\theta$
  \[
  p(\theta|D) \propto \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2\right) \times \exp\left(-\frac{1}{2\tau^2} \theta^T \theta\right)
  \]
  - This leads to a now objective
    \[
    l_{MAP}(\theta; D) = -\frac{1}{2\sigma^2} \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 - \frac{1}{2\tau^2} \frac{1}{2} \sum_{i=1}^{k} \theta_i^2
    \] 
    \[
    = l(\theta; D) + \lambda \|	heta\| 
    \]
  - This is $L_2$ regularized LR! --- a MAP estimation of $\theta$
  - What about $L_1$ regularized LR! (homework)
  - How to choose $\lambda$.
    - cross-validation!
Feature Selection

- Imagine that you have a supervised learning problem where the number of features \( n \) is very large (perhaps \( n \gg \text{#samples} \)), but you suspect that there is only a small number of features that are "relevant" to the learning task.

- Later lecture on VC-theory will tell you that this scenario is likely to lead to high generalization error – the learned model will potentially overfit unless the training set is fairly large.

- So let's get rid of useless parameters!

How to score features

- How do you know which features can be pruned?
  - Given labeled data, we can compute some simple score \( S(i) \) that measures how informative each feature \( x_i \) is about the class labels \( y \).
  - Ranking criteria:
    - Mutual Information: score each feature by its mutual information with respect to the class labels
      \[
      MI(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}
      \]
    - Bayes error:
    - Redundancy (Markov-blank score) …
  - We need estimate the relevant \( p(i) \)'s from data, e.g., using MLE
Feature Ranking

- Bayes error of each gene
- Information gain for each gene with respect to the given partition
- KL of each removal gene w.r.t. to its MB

Feature selection schemes

- Given $n$ features, there are $2^n$ possible feature subsets (why?)
- Thus feature selection can be posed as a model selection problem over $2^n$ possible models.
- For large values of $n$, it's usually too expensive to explicitly enumerate over and compare all $2^n$ models. Some heuristic search procedure is used to find a good feature subset.
- Three general approaches:
  - Filter: i.e., direct feature ranking, but taking no consideration of the subsequent learning algorithm
    - add (from empty set) or remove (from the full set) features one by one based on $S(i)$
    - Cheap, but is subject to local optimality and may be unrobust under different classifiers
  - Wrapper: determine the (inclusion or removal of) features based on performance under the learning algorithms to be used. See next slide
  - Simultaneous learning and feature selection.
    - E.g., L1 regularized LR, Bayesian feature selection (will not cover in this class), etc.
Wrapper

- **Forward:**
  1. Initialize $F = \emptyset$
  2. Repeat
     - For $i = 1, \ldots, n$
       - If $i \notin F$, let $F = F \cup \{i\}$, and use some version of cross validation to evaluate features $F_i$ (i.e., train your learning algorithm using only the features in $F_i$, and estimate its generalization error.)
     - Set $F$ to be the best feature subset found on the last step.
  3. Select and output the best feature subset that was evaluated during the entire search procedure.

- **Backward search**
  1. Initialize $F = \text{full set}$
  2. …

Case study [Xing et al, 2001]

- **The case:**
  - 7130 genes from a microarray dataset
  - 72 samples
  - 47 type I Leukemias (called ALL)
  - 25 type II Leukemias (called AML)

- **Three classifier:**
  - kNN
  - Gaussian classifier
  - Logistic regression
Regularization vs. Feature Selection

- Explicit feature selection often outperform regularization

Model Selection via Information Criteria

- How can we compare the closeness of a learned hypothesis and the true model?
- The relative entropy (also known as the **Kullback-Leibler divergence**) is a measure of how different two probability distributions (over the same event space) are.
  - For 2 pdfs, $p(x)$ and $q(x)$, their KL-divergence is:
    \[
    D(p \parallel q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}
    \]
  - The KL divergence between $p$ and $q$ can also be seen as the average number of bits that are wasted by encoding events from a distribution $p$ with a code based on a not-quite-right distribution $q$. 
An information criterion

- Let \( f(x) \) denote the truth, the underlying distribution of the data
- Let \( g(x, \theta) \) denote the model family we are evaluating
  - \( f(x) \) does not necessarily reside in the model family
  - \( \hat{\theta}_M(y) \) denote the MLE of model parameter from data \( y \)
- Among early attempts to move beyond Fisher’s Maximum Likelihood framework, Akaike proposed the following information criterion:
  \[
  E_y [D(f \parallel g(x \mid \hat{\theta}_M(y))] 
  \]
  which is, of course, intractable (because \( f(x) \) is unknown)

AIC and TIC

- AIC (\( A \), information criterion, not Akaike information criterion)
  \[
  A = \log g(x \mid \hat{\theta}(y)) - k 
  \]
  where \( k \) is the number of parameters in the model

- TIC (Takeuchi information criterion)
  \[
  A = \log g(x \mid \hat{\theta}(y)) - \text{tr}(I(\theta_0)\Sigma) 
  \]
  where
  \[
  \theta_0 = \arg \min D(f \parallel g(\cdot \mid \theta)) \quad I(\theta_0) = -E\left[\frac{\partial^2 \log g(x \mid \theta)}{\partial \theta \partial \theta^T}\right]_{\theta = \theta_0} \quad \Sigma = E\left[\hat{\theta}(y) - \theta_0\right]\left[\hat{\theta}(y) - \theta_0\right]^T
  \]
  - We can approximate these terms in various ways (e.g., using the bootstrap)
  - \( \text{tr}(I(\theta_0)\Sigma) \approx k \)
Bayesian Model Selection

- Recall the Bayesian Theory: (e.g., for data $D$ and model $M$)

\[ P(M|D) = \frac{P(D|M)P(M)}{P(D)} \]

- the posterior equals to the likelihood times the prior, up to a constant.

- Assume that $P(M)$ is uniform and notice that $P(D)$ is constant, we have the following criteria:

\[ P(D | M) = \int_\theta P(D | \theta, M)P(\theta | M)d\theta \]

- A few steps of approximations (you will see this in advanced ML class in later semesters) give you this:

\[ P(D | M) \approx \log P(D | \hat{\theta}_{ML}) - \frac{k}{2} \log N \]

where $N$ is the number of data points in $D$. 