10701
Machine Learning
Model and feature selection
Model selection issues

• We have seen some of this before …

• Selecting features (or basis functions)
  – Linear regression
  – Logistic regression
  – SVMs

• Selecting parameter value
  – Prior strength
    • Naïve Bayes, linear and logistic regression
  – Regularization strength
    • Linear and logistic regression
  – Decision trees
    • depth, number of leaves
  – Clustering
    • Number of clusters

• More generally, these are called Model Selection Problems
Training and test set error as a function of model complexity
Simple greedy model selection algorithm

- Pick a dictionary of features
  - e.g., polynomials for linear regression
- Greedy heuristic:
  - Start from empty (or simple) set of features $F_0 = \emptyset$
  - Run learning algorithm for current set of features $F_t$
    - Obtain $h_t$
    - Select next best feature $X_i^*$
      - e.g., $X_j$ that results in lowest training error learner when learning with $F_t \cup \{X_j\}$
    - $F_{t+1} \leftarrow F_t \cup \{X_i^*\}$
  - Recurse
Greedy model selection

• Applicable in many settings:
  – Linear regression: Selecting basis functions
  – Naïve Bayes: Selecting (independent) features $P(X_i|Y)$
  – Logistic regression: Selecting features (basis functions)
  – Decision trees: Selecting leaves to expand

• Only a heuristic!
  – But, sometimes you can prove something cool about it
Simple greedy model selection algorithm

- Greedy heuristic:
  - ...  
  - Select **next best feature** $X_i^*$
    - e.g., $X_j$ that results in lowest training error learner when learning with $F_t \cup \{X_j\}$
  - $F_{t+1} \leftarrow F_t \cup \{X_i^*\}$
  - Recurse

When do you stop???

- When training error is low enough?
- When test set error is low enough?
Validation set

• Thus far: Given a dataset, \textbf{randomly} split it into two parts:
  – Training data – \{x_1,\ldots, x_{N_{\text{train}}}\}
  – Test data – \{x_1,\ldots, x_{N_{\text{test}}}\}

• But \textbf{Test data must always remain independent}!
  – Never ever ever ever learn on test data, including for model selection

• Given a dataset, \textbf{randomly} split it into three parts:
  – Training data – \{x_1,\ldots, x_{N_{\text{train}}}\}
  – Validation data – \{x_1,\ldots, x_{N_{\text{valid}}}\}
  – Test data – \{x_1,\ldots, x_{N_{\text{test}}}\}

• Use validation data for tuning learning algorithm, e.g., model selection
  – Save test data for very final evaluation
Simple greedy model selection algorithm

- Greedy heuristic:
  - ...
  - Select **next best feature** $X_i^*$
    - e.g., $X_j$ that results in lowest training error learner when learning with $F_t \cup \{X_j\}$
  - $F_{t+1} \leftarrow F_t \cup \{X_i^*\}$
  - Recurse

When do you stop???
- When training error is low enough?
- When test set error is low enough?
- When validation set error is low enough?

Sometimes, but there is an even better option …
Validating a learner, not a hypothesis (intuition only, not proof)

• With a validation set, get to estimate error of 1 hypothesis on 1 dataset
  - e.g. Should I use a polynomial of degree 3 or 4

• Need to estimate error of learner over multiple datasets to select parameters

\[ E_{\{x,y\}}[h_t] \]

Expected error over all datasets
(LOO) Leave-one-out cross validation

• Consider a validation set with 1 example:
  – \( D \) – training data
  – \( D \setminus i \) – training data with \( i \) th data point moved to validation set

• Learn classifier \( h_{D \setminus i} \) with the \( D \setminus i \) dataset

• Estimate true error as:
  – 0 if \( h_{D \setminus i} \) classifies \( i \) th data point correctly
  – 1 if \( h_{D \setminus i} \) is wrong about \( i \) th data point
  – Seems really bad estimator, but wait!

• LOO cross validation: Average over all data points \( i \):
  – For each data point you leave out, learn a new classifier \( h_{D \setminus i} \)
  – Estimate error as:

\[
\text{error}_{LOO} = \frac{1}{m} \sum_{i=1}^{m} 1 \left( h_{D \setminus i}(x^i) \neq y^i \right)
\]
LOO cross validation is (almost) unbiased estimate of true error!

- When computing **LOOCV error**, we only use \( m-1 \) data points
  - So it’s not estimate of true error of learning with \( m \) data points!
  - Usually pessimistic, though – learning with less data typically gives worse answer

- **LOO is almost unbiased!**
  - Let \( \text{error}_{true,m-1} \) be true error of learner when you only get \( m-1 \) data points
  - LOO is unbiased estimate of \( \text{error}_{true,m-1} \):
    \[
    E_D[\text{error}_{LOO}] = \text{error}_{true,m-1}
    \]

- **Great news!**
  - Use LOO error for model selection!!!
Simple greedy model selection algorithm

- Greedy heuristic:
  - ... 
  - Select **next best feature** $X_i^*$
    - e.g., $X_j$ that results in lowest training error learner when learning with $F_t \cup \{X_j\}$
  - $F_{t+1} \leftarrow F_t \cup \{X_i^*\}$
  - Recurse

When do you stop???

- When training error is low enough?
- When test set error is low enough?
- When validation set error is low enough?
- **STOP WHEN** $\text{error}_{\text{LOO}}$ **IS LOW!!!
LOO cross validation error
Computational cost of LOO

• Suppose you have 100,000 data points
• You implemented a great version of your learning algorithm
  – Learns in only 1 second
• Computing LOO will take about 1 day!!
  – If you have to do for each choice of basis functions, it will take forever!
Solution: Use $k$-fold cross validation

- Randomly divide training data into $k$ equal parts
  - $D_1, \ldots, D_k$
- For each $i$
  - Learn classifier $h_{D \setminus D_i}$ using data point not in $D_i$
  - Estimate error of $h_{D \setminus D_i}$ on validation set $D_i$:
    $$\text{error}_{D_i} = \frac{k}{m} \sum_{(x^i, y^i) \in D_i} 1 \left( h_{D \setminus D_i}(x^i) \neq y^i \right)$$
- $k$-fold cross validation error is average over data splits:
  $$\text{error}_{k-\text{fold}} = \frac{1}{k} \sum_{i=1}^{k} \text{error}_{D_i}$$
- $k$-fold cross validation properties:
  - Much faster to compute than LOO
  - More (pessimistically) biased – using much less data, only $m(k-1)/k$
Regularization

• Model selection 1 (using CV): Greedy
  – Pick subset of features that have yield low LOO error

• Model selection 2: Regularization
  – Include all possible features!
  – Penalize “complicated” hypothesis
Regularization in linear regression

- Overfitting usually leads to very large parameter choices, e.g.:
  - $-2.2 + 3.1 \, X - 0.30 \, X^2$
  - $-1.1 + 4,700,910.7 \, X - 8,585,638.4 \, X^2 + \ldots$

- Regularized least-squares (a.k.a. ridge regression):
  \[
  w^* = \arg \min_w \sum_j (w^T x_j - y_j)^2 + \lambda \sum_i w_i^2
  \]
Other regularization examples

- **Logistic regression** regularization
  - Maximize data likelihood minus **penalty for large parameters**
  
  \[ \arg \max_w \sum_j \ln P(y^j|x^j, w) - \lambda \sum_i w_i^2 \]
  
  - Biases towards small parameter values

- **Naïve Bayes** regularization
  - **Prior** over likelihood of features
  - **Biases away from zero probability** outcomes

- **Decision tree** regularization
  - Many possibilities, e.g., Chi-Square test
  - **Biases towards smaller trees**

- **Sparsity**: find good solution with few basis functions, e.g.:
  - Simple greedy model selection from earlier in the lecture
  - **L1 regularization**, e.g.:
    \[ w^* = \arg\min_w \sum_j (w^T x_j - y_j)^2 + \lambda \sum_i |w_i| \]
Regularization and Bayesian learning

\[ p(w \mid Y, X) \propto P(Y \mid X, w)p(w) \]

• For example, if we assume a zero mean, Gaussian prior for \( w \) in a logistic regression classification we would end up with an L2 regularization
  - Why?
  - Board …
  - What is \( \lambda \)?

• Similar interpretation for other learning approaches:
  – **Linear regression**: Also zero mean, Gaussian prior for \( w \)
  – **Naïve Bayes**: Directly defined as prior over parameters
How do we pick magic parameter $\lambda$?

Cross Validation!!!
Occam’s Razor

• William of Ockham (1285-1349) *Principle of Parsimony*:
  – “One should not increase, beyond what is necessary, the number of entities required to explain anything.”

• Regularization penalizes for “complex explanations”

• Alternatively (but pretty much the same), use *Minimum Description Length (MDL) Principle*:
  – minimize $\text{length}(\text{misclassifications}) + \text{length}(\text{hypothesis})$

• $\text{length}(\text{misclassifications})$ – e.g., #wrong training examples
• $\text{length}(\text{hypothesis})$ – e.g., size of decision tree
Minimum Description Length Principle

• MDL prefers small hypothesis that fit data well:

\[ h_{MDL} = \arg \min_{h} L_{C_1}(\mathcal{D} \mid h) + L_{C_2}(h) \]

  – \( L_{C_1}(\mathcal{D} \mid h) \) – description length of data under code \( C_1 \) given \( h \)
    • Only need to describe points that \( h \) doesn’t explain (classify correctly)
  – \( L_{C_2}(h) \) – description length of hypothesis \( h \)

• Decision tree example
  – \( L_{C_1}(\mathcal{D} \mid h) \) – #bits required to describe data given \( h \)
    • If all points correctly classified, \( L_{C_1}(\mathcal{D} \mid h) = 0 \)
  – \( L_{C_2}(h) \) – #bits necessary to encode tree
  – Trade off quality of classification with tree size
What you need to know about Model Selection, Regularization and Cross Validation

• Cross validation
  – (Mostly) Unbiased estimate of true error
  – LOOCV is great, but hard to compute
  – $k$-fold much more practical
  – Use for selecting parameter values!

• Model selection
  – Search for a model with low cross validation error

• Regularization
  – Penalizes for complex models
  – Select parameter with cross validation
  – Really a Bayesian approach

• Minimum description length
  – Information theoretic interpretation of regularization
Feature selection

- Choose an optimal subset from the set of all N features
  - Only use a subset of a possible words in a dictionary
  - Only use a subset of genes
- Why?
- Can we do model selection to solve this? – $2^n$ models
eg. Microarray data

1682 Genes

>8x induced  >8x repressed
Two approaches: 1. Filter

- Independent of classifier used
- Rank features using some criteria based on their relevance to the classification task
- For example, mutual information:

\[
I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p_1(x) p_2(y)} \right),
\]

- Choose a subset based on the sorted scores for the criteria used
2. Wrapper

- Classifier specific
- Greedy (large search space)

Initialize $F = \text{null set}$
  - At each step, using cross validation or an information theoretic criteria, choose a feature to add to the subset [training should be done with only features in $F + \text{new feature}$]
  - Add the chosen feature to the subset

Repeat until no improvement to CV accuracy