OK... now we’ll learn to pick those darned parameters...

- Selecting features (or basis functions)
  - Linear regression
  - Naïve Bayes
  - Logistic regression

- Selecting parameter value
  - Prior strength
    - Naïve Bayes, linear and logistic regression
  - Regularization strength
    - Naïve Bayes, linear and logistic regression
  - Decision trees
    - MaxpChance, depth, number of leaves
  - Boosting
    - Number of rounds

- More generally, these are called Model Selection Problems

Today:
- Describe basic idea
- Introduce very important concept for tuning learning approaches: Cross-Validation
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., $X_i$ 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
    - e.g., [Krause & Guestrin ’05]: Near-optimal in some settings that include Naïve Bayes

- There are many more elaborate methods out there

---

Simple greedy model selection algorithm

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

  When do you stop???
  - When training error is low enough?
Simple greedy model selection algorithm

- Greedy heuristic:
  - ... 
  - Select next best feature $X^*_i$
    - e.g., $X^*_i$ that results in lowest training error learner when learning with $F_t \cup \{X^*_i\}$
  - $F_{t+1} \leftarrow F_t \cup \{X^*_i\}$
  - 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, randomly split it into two parts:
  - Training data – $\{x_1, \ldots, x_{N_{train}}\}$
  - Test data – $\{x_{N_{train}+1}, \ldots, x_{N_{test}}\}$
- But Test data must always remain independent!
  - Never ever ever ever learn on test data, including for model selection
- Given a dataset, randomly split it into three parts:
  - Training data – $\{x_1, \ldots, x_{N_{train}}\}$
  - Validation data – $\{x_{N_{train}+1}, \ldots, x_{N_{valid}}\}$
  - Test data – $\{x_{N_{valid}+1}, \ldots, x_{N_{test}}\}$
- 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., $X_i$ that results in lowest training error learner when learning with $F_t \cup \{X_i\}$
  - $F_{t+1} \leftarrow F_t \cup \{X_i^*\}$
  - **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?

Validating a learner, not a hypothesis (disclaimer: intuition only, not proof)

- With a validation set, get to estimate error of 1 hypothesis on 1 dataset

- Need to estimate error of learner over multiple datasets to select parameter
  - Think variance
Simple greedy model selection algorithm

- Greedy heuristic:
  - Select **next best feature** $X_i^*$
    - e.g., $X_i$ that results in lowest training error learner when learning with $F_i \cup \{X_i\}$
  - $F_{t+1} \leftarrow F_t \cup \{X_i^*\}$
  - **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?
  - Man!!! OK, should I just repeat until I get tired???
    - I am tired now...
    - No, “There is a better way!”

(LOO) Leave-one-out cross validation

- Consider a **validation set with 1 example**:
  - $D$ – training data
  - $D_i$ – training data with $i$th data point moved to validation set
- Learn classifier $h_{D_i}$ with $D_i$ dataset
- Estimate true error as:
  - 0 if $h_{D_i}$ classifies $i$th data point correctly
  - 1 if $h_{D_i}$ 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 $h_{D_i}$
  - **Estimate error** as:
    $$ error_{LOO} = \frac{1}{m} \sum_{i=1}^{m} 1 \left( h_{D \setminus i}(x^i) \neq y^i \right) $$
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 an 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 \( error_{true,m-1} \) be the true error of learner when you only get \( m-1 \) data points
  - LOO is an unbiased estimate of \( error_{true,m-1} \):
    \[
    E_D[error_{LOO}] = error_{true,m-1}
    \]

- **Great news!**
  - Use LOO error for model selection!!!

---

Simple greedy model selection algorithm

- **Greedy heuristic:**
  - ...  
  - Select next best feature \( X_i^* \)
    - e.g., \( X_i \) that results in lowest training error learner when learning with \( F_t \cup \{X_i\} \)
  - \( F_{t+1} = F_t \cup \{X_i^*\} \)
  - 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** \( error_{LOO} \) **IS LOW!!!**
Using LOO error for model selection

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 fooooreeeve’!!!
- Solution 1: Preferred, but not usually possible
  - Find a cool trick to compute LOO (e.g., see homework)
Solution 2 to complexity of computing LOO:

(More typical) **Use k-fold cross validation**

- Randomly **divide training data into** $k$ *equal parts*
  - $D_1, \ldots, D_k$
- For each $i$
  - Learn classifier $h_{D_i}$ using data point not in $D_i$
  - Estimate error of $h_{D_i}$ on validation set $D_i$:
    $$error_{D_i} = \frac{k}{m} \sum_{(x^j, y^j) \in D_i} 1(h_{D_i}(x^j) \neq y^j)$$
- **k-fold cross validation error is average** over data splits:
  $$error_{k-fold} = \frac{1}{k} \sum_{i=1}^{k} error_{D_i}$$
- **k-fold cross validation properties:**
  - Much faster to compute than LOO
  - More (pessimistically) biased – using much less data, only $m(k-1)/k$
  - Usually, $k = 10$ 😊

---

**Regularization – Revisited**

- **Model selection 1**: 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), for \(\lambda \geq 0\):

  \[w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 + \lambda \sum_i w_i^2\]

Other regularization examples

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

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

- **Decision tree** regularization
  - Many possibilities, e.g., Chi-Square test and MaxPvalue parameter
  - 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.:
    \[w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 + \lambda \sum_i |w_i|\]
Geometric intuition of L2 regularization

\[ w^* = \arg\min_w \sum \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 + \lambda \sum_{i=1}^n w_i^2 \]

Geometric intuition of L1 regularization

\[ w^* = \arg\min_w \sum \left( t(x_j) - \sum_i w_i h_i(x_j) \right) + \lambda \sum_{i=1}^n |w_i| \]
How do we pick magic parameter?

Cross Validation!!!!

\( \lambda \) in Linear/Logistic Regression
(analogously for # virtual examples in Naïve Bayes, MaxPvalue in Decision Trees)

Regularization and Bayesian learning

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

- We already saw that regularization for logistic regression corresponds to MAP for zero mean, Gaussian prior for w

- Similar interpretation for other learning approaches:
  - Linear regression: Also zero mean, Gaussian prior for w
  - Naïve Bayes: Directly defined as prior over parameters
  - Decision trees: Trickier to define… but we’ll get back to this
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 \( \text{length(misclassifications)} + \text{length(hypothesis)} \)

- \( \text{length(misclassifications)} \) – e.g., #wrong training examples
- \( \text{length(hypothesis)} \) – e.g., size of decision tree

Minimum Description Length Principle

- MDL prefers small hypothesis that fit data well:
  \[
  h_{MDL} = \arg \min_h L_{C_1}(D | h) + L_{C_2}(h)
  \]
- \( L_{C_1}(D|h) \) – 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
  - \( L_{C_1}(D|h) \) – #bits required to describe data given \( h \)
    - If all points correctly classified, \( L_{C_1}(D|h) = 0 \)
  - \( L_{C_2}(h) \) – #bits necessary to encode tree
  - Trade off quality of classification with tree size
Bayesian interpretation of MDL Principle

- MAP estimate
  \[ h_{MAP} = \arg\max_h [P(D | h)p(h)] \]
  \[ = \arg\max_h [\log_2 P(D | h) + \log_2 P(h)] \]
  \[ = \arg\min_h [-\log_2 P(D | h) - \log_2 P(h)] \]

- Information theory fact:
  - Smallest code for event of probability \( p \) requires \( -\log_2 p \) bits

- MDL interpretation of MAP:
  - \( -\log_2 P(D|h) \) – length of \( D \) under hypothesis \( h \)
  - \( -\log_2 P(h) \) – length of hypothesis \( h \) (there is hidden parameter here)
  - MAP prefers simpler hypothesis:
    - minimize \( length(\text{misclassifications}) + length(hypothesis) \)

- In general, Bayesian approach usually looks for simpler hypothesis – Acts as a regularizer

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
  - Relationship to MAP