Model Selection and Feature Selection

Piyush Rai

CS5350/6350: Machine Learning

September 22, 2011
What is Model Selection

Given a set of models $\mathcal{M} = \{M_1, M_2, \ldots, M_R\}$, choose the model that is expected to do the best on the test data. $\mathcal{M}$ may consist of:

- Same learning model with different complexities or hyperparameters
  - Nonlinear Regression: Polynomials with different degrees
  - $K$-Nearest Neighbors: Different choices of $K$
  - Decision Trees: Different choices of the number of levels/leaves
  - SVM: Different choices of the misclassification penalty hyperparameter $C$
  - Regularized Models: Different choices of the regularization parameter
  - Kernel based Methods: Different choices of kernels
  - .. and almost any learning problem
- Different learning models (e.g., SVM, KNN, DT, etc.)

**Note:** Usually considered in supervised learning contexts but unsupervised learning too faces this issue (e.g., “how many clusters” when doing clustering)
Held-out Data

- Set aside a fraction (say 10%-20%) of the training data
- This part becomes our held-out data
  - Other names: validation/development data

\[\text{all the examples} \]
\[
\text{Training Set} \quad \text{Held-out/validation Set}
\]

- **Remember**: Held-out data is NOT the test data
- Train each model using the remaining training data
- Evaluate error on the held-out data
- Choose the model with the smallest held-out error

**Problems:**
- Wastes training data, so typically used when we have plenty of training data
- Held-out data may not be good if there was an unfortunate split
  - Can ameliorate unfortunate splits by repeated random subsampling
$K$-fold Cross-Validation

- Create $K$ equal sized partitions of the training data
- Each partition has $N/K$ examples
- Train using $K - 1$ partitions, validate on the remaining partition
- Repeat the same $K$ times, each with a different validation partition

Finally, choose the model with smallest \textit{average} validation error

Usually $K$ is chosen as 10
Leave-One-Out (LOO) Cross-Validation

Special case of $K$-fold CV when $K = N$ (number of training examples)
- Each partition is now an example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same $N$ times, each with a different validation example

Finally, choose the model with smallest **average** validation error
- Can be expensive for large $N$. Typically used when $N$ is small
Random Subsampling Cross-Validation

- Randomly subsample a fixed fraction $\alpha N$ ($0 < \alpha < 1$) of examples; call it the validation set.
- Train using the rest of the examples, measure error on the validation set.
- Repeat $K$ times, each with a different, randomly chosen validation set.

Finally, choose the model with smallest average validation error.

Usually $\alpha$ is chosen as 0.1, $K$ as 10.
Bootstrapping

- Given: a set of $N$ examples
- Idea: Sample $N$ elements from this set with replacement
  - An already sampled element could be picked again
- Use this new sample as the training data
- Use the set of examples not selected as the validation data
- For large $N$, training data consists of about only 63% unique examples
- Training data is inherently small $\Rightarrow$ error estimate may be pessimistic
- Use the following equation to compute the expected model error
  \[
e = 0.632 \times e_{\text{test-examples}} + 0.368 \times e_{\text{training-examples}}\]
- Note: the above estimate may still be bad if we overfit and have $e_{\text{training-examples}} = 0$. Why?
Information Criteria based methods

- Akaike Information Criteria (AIC)
  \[ AIC = 2k - 2\log(L) \]

- Bayesian Information Criteria (BIC)
  \[ BIC = k \log(N) - 2\log(L) \]

- \( k \): # of model parameters
- \( L \): maximum value of the model likelihood function
- Applicable for probabilistic models (when likelihood is defined)
- AIC/BIC penalize model complexity
  - .. as measured by the number of model parameters
  - BIC penalizes the number of parameters more than AIC
- Model with the lowest AIC/BIC will be chosen
- Can be used even for model selection in unsupervised learning
Minimum Description Length (MDL)

- MDL measures the number of bits to encode a probability distribution

\[ MDL = -\log_2 P(z) \]

- Minimum Description Length for a model \( M \)

\[ \text{Length}(M) = -\log P(Y \mid X, w, M) - \log P(w \mid M) \]

- Note: it’s just the MDL for model’s posterior distribution

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

- Complex posterior distribution \( \Rightarrow \) Complex model
- Choose the model with the lowest MDL
- Note: MDL criteria is kind of equivalent to preferring the best regularized model
Feature Selection

Selecting a useful subset from all the features

Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Removal of features can increase (relative) margin (and generalization)
- Reduces data set and resulting model size

Note: Feature Selection is different from Feature Extraction

- The latter transforms original features to get a small set of new features
- More on feature extraction when we cover Dimensionality Reduction
Feature Selection Methods

- Methods **agnostic** to the learning algorithm
  - Preprocessing based methods
    - E.g., remove a binary feature if it’s ON in very few or most examples
  - Filter Feature Selection methods
    - Use some ranking criteria to rank features
    - Select the top ranking features

- Wrapper Methods (keep the learning algorithm in the loop)
  - Requires repeated runs of the learning algorithm with different set of features
  - Can be **computationally expensive**
Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods

**Correlation Criteria:** Rank features in order of their correlation with the labels

\[
R(X_d, Y) = \frac{\text{cov}(X_d, Y)}{\sqrt{\text{var}(X_d)\text{var}(Y)}}
\]

**Mutual Information Criteria:**

\[
MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{-1,+1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}
\]

- High mutual information mean high relevance of that feature
- Note: These probabilities can be easily estimated from the data
Wrapper Methods

- Two types: Forward Search and Backward Search
  - **Forward Search**
    - Start with no features
    - Greedily include the most relevant feature
    - Stop when selected the desired number of features
  - **Backward Search**
    - Start with all the features
    - Greedily remove the least relevant feature
    - Stop when selected the desired number of features
  - Inclusion/Removal criteria uses cross-validation
Wrapper Methods

**Forward Search**

- Let $\mathcal{F} = \{\}$
- While not selected desired number of features
- For each unused feature $f$
  - Estimate model’s error on feature set $\mathcal{F} \cup f$ (using cross-validation)
  - Add $f$ with lowest error to $\mathcal{F}$

**Backward Search**

- Let $\mathcal{F} = \{\text{all features}\}$
- While not reduced to desired number of features
- For each feature $f \in \mathcal{F}$
  - Estimate model’s error on feature set $\mathcal{F} \setminus f$ (using cross-validation)
  - Remove $f$ with lowest error from $\mathcal{F}$