Lecture #8: Model Selection and Regularization

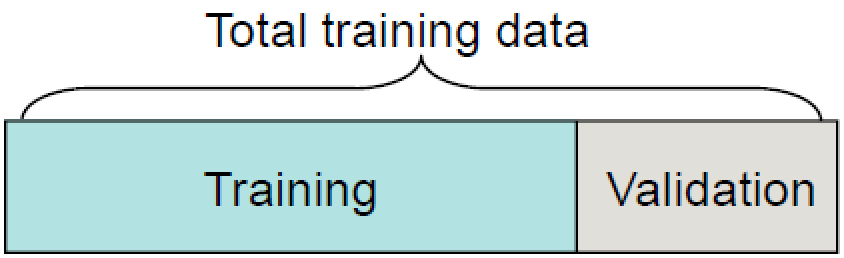
Model Selection: The Problem

- Assume that we have a set of models M=$\{M_1, M_2, \cdots, M_d\}$ that we are trying to select from. Some examples include:
- **Feature Selection:** each M_i corresponds to using a different feature subset from a large set of potential features
- **Algorithm Selection:** each M_i corresponds to an algorithm, e.g., Naïve Bayes, Logistic Regression, DT …
- **Parameter selection:** each M_i corresponds to a particular parameter choice, e.g., the choice of kernel and C for SVM

Model Selection: Approaches

- **Holdout and Cross-validation methods**
  - Experimentally determine when overfitting occurs
- **Penalty methods**
  - MAP Penalty
  - Minimum Description Length (MDL)
  - Many others
- **Ensembles**
  - Instead of choosing one, consider many possibilities and let them vote

Simple Holdout Method



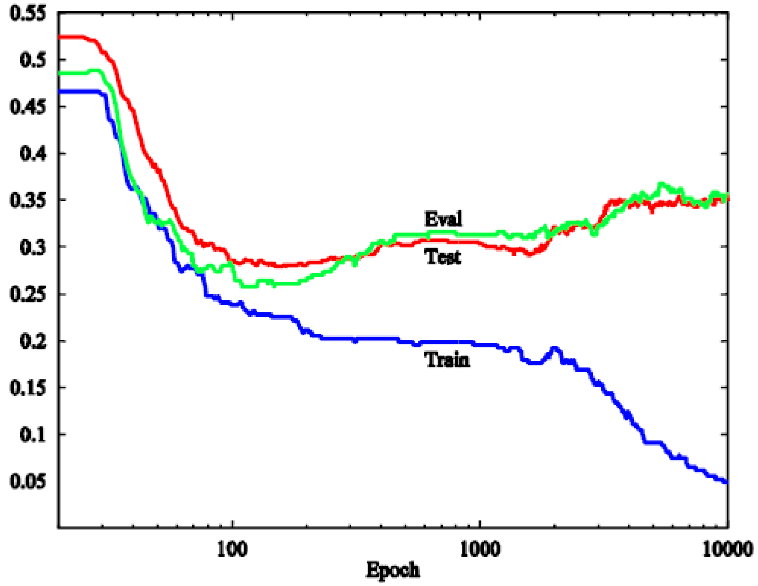
1. Divide the training set S into S_{train} and $S_{validate}$
2. Train each model M_i on S_{train} to get a hypothesis h_i
3. Choose and output h_i with the smallest error rate on $S_{validate}$

Could retrain the selected model on the whole dataset to get the final hypothesis h - this will improve the original h_i because of more training data

Notes on Holdout Method

- Hold-out method often used for choosing among nested hypotheses:
  - Deciding # of training epochs for online learner
  - Deciding when to stop growing or pruning a decision tree
  - Deciding when to stop growing an ensemble

error



Example:

Selecting # of epochs for Perceptron

Holdout Method: Issues

- It wastes part of the data
  - The model selection choice is still made using only part of the data
  - Still possible to overfit the validation data since it is a relatively small set of data

- To address these problems, we can use a method called **Cross Validation**

K-fold Cross Validation

- Partition (randomly) S into K disjoint subsets $S_1, \cdots, S_K$ (preferably in a class-balanced way)
- To evaluate model M_j:

for i=1:K

1. Train M_j on $S \backslash S_i$ (S removing S_i) $\rightarrow h_{ji}$
2. Test h_{ji} on $S_i \rightarrow \epsilon_j(i)$

End for

$$\epsilon_j = \frac{1}{K} \sum_i \epsilon_j(i)$$

- Select model that minimizes the error: $M^* = \underset{M_j}{\operatorname{argmin}}\, \epsilon_j$
- Train M^* on S and output resulting hypothesis

Comments on K-fold Cross Validation

- Computationally more expensive than simple hold-out method but better use of data
  - Every data point in the training set is used in validating the model selection choices
- If the data is really scarce, we can use the extreme choice of $k = |S|$
  - Each validation set contains only one data point
  - Often referred to as **Leave-one-out (LOO) cross-validation**

Feature Selection

- A special case of model selection problem
- **Problem:** given a supervised learning problem in which the feature dimension is very high, but only a small subset of the features is relevant
- **Goal:** identify a small subset of relevant features
- **Why?**
  - Smaller feature set size leads to less chance of overfitting
  - In some domains, user might like to know which features are important for predicting the target variable

Search Space for Feature Selection

- Given d features, there are 2^d possible subsets
- Too expensive to enumerate all possible models to evaluate and choose
- Practical solutions rely on heuristic search

Forward Search

1. Initialize $\mathcal{F} = \emptyset$

2. Repeat {
   a) For $i = 1, \ldots, d$ if $i \notin \mathcal{F}$, let $\mathcal{F}_i = \mathcal{F} \cup \{i\}$, and use cross-validation to evaluate $\mathcal{F}_i$
   b) Set $\mathcal{F}$ to be the best feature subset found in step a)

   }

3. Select the best feature subset that was evaluated during the entire search process

Backward Search

1. Initialize $\mathcal{F} = \{1, \ldots, d\}$
2. Repeat {
   a) For all $i \in \mathcal{F}$, let $\mathcal{F}_{-i} = \mathcal{F}/\{i\}$, and use cross-validation to evaluate $\mathcal{F}_{-i}$
   b) Set $\mathcal{F}$ to be the best feature subset found in step a

   }
3. Select the best feature subset that was evaluated during the entire search process

Wrapper vs. Filter Approaches

- **Both forward and backward search methods are considered wrapper approaches**
  - They wrap around a learning algorithm in order to find the subset that works the best with the given learning algorithm
- **Alternatively, filter approaches heuristically select the features without considering the learning algorithm**
  - Mutual information is one such measure frequently used by filter methods
    - Compute the mutual information between each feature and the class label, and rank them from high to low
    - Choose the top k features in the ranked order
    - How to decide k? Cross-Validation

Penalty (Regularization) Methods

- **Basic idea:** include a penalty term in the objective function to penalize complex hypothesis

- Some examples:
  - Regularized linear regression

$$J(w) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda |\mathbf{w}|^2$$

Regularization term to control model complexity

  - Regularized logistic regression

$$J(w) = L(w) - \lambda |\mathbf{w}|^2$$

Log-likelihood

- A common approach for deriving such regularization method is Maximum A Posterior (MAP) estimation

Frequentist vs. Bayesian

- When it comes to parameter estimation, there are two different statistical views
  - **Frequentist:** parameter is deterministic, it takes an unknown value
  - **Bayesian:** parameter is a random variable with a unknown distribution
    - We can express our belief about the parameter using priors
    - After observing the data, we can update our belief to obtain the posterior distribution of the parameter

Prior distribution of θ

$$p(\theta|D) = \frac{p(\theta)p(D|\theta)}{p(D)} = \frac{p(\theta)p(D|\theta)}{\int p(D|\theta)p(\theta)d\theta}$$

Posterior distribution of θ

Maximum A Posterior (MAP) as a Penalty Method

$$\hat{\theta}_{map} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|D)$$

$$= \underset{\theta}{\operatorname{argmax}}\, p(D|\theta)p(\theta)$$

$$= \underset{\theta}{\operatorname{argmax}}\, \log p(D|\theta) + \log p(\theta)$$

penalty

MAP for Logistic Regression

$$p(y=1 \mid \mathbf{x};\mathbf{w}) = p_1(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}$$

$$p(y=0 \mid \mathbf{x};\mathbf{w}) = 1 - p_1(\mathbf{x})$$

- *h* describes conditional distribution of y | x
- Parameters: **w**
- Learning goal is to find *h* (i.e. **w**) to maximize P(*h* | S):

$$\arg\max_{\mathbf{w}} P(\mathbf{w} \mid S) = \arg\max_{\mathbf{w}} P(S \mid \mathbf{w}) P(\mathbf{w})$$

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|S) = \arg\max_{\mathbf{w}} (\log P(S|\mathbf{w}) + \log P(\mathbf{w}))$$

- Our prior belief about **w**: $w_i \sim N(0, \sigma^2)$ for $i = 1, \cdots, d$
  - Large weight values correspond to more complex hypotheses, so this prior prefers simpler hypothesis ($\mu = 0$)

Logistic Regression: MAP

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|S) = \arg\max_{\mathbf{w}} (\log P(S|\mathbf{w}) + \log P(\mathbf{w}))$$

$$= \operatorname*{argmax}_{\mathbf{w}} \sum_j \log p(y^j|\mathbf{x}^j, \mathbf{w}) + \log \prod_i N(w_i; 0, \sigma^2)$$

$$= \operatorname*{argmax}_{\mathbf{w}} \sum_j \log p(y^j|\mathbf{x}^j, \mathbf{w}) + \sum_i \log\left(\frac{1}{\sqrt{2\pi\sigma}} \exp \frac{-w_i^2}{2\sigma^2}\right)$$

$$= \operatorname*{argmax}_{\mathbf{w}} \sum_j \log p(y^j|\mathbf{x}^j, \mathbf{w}) + \sum_i \frac{-w_i^2}{2\sigma^2}$$

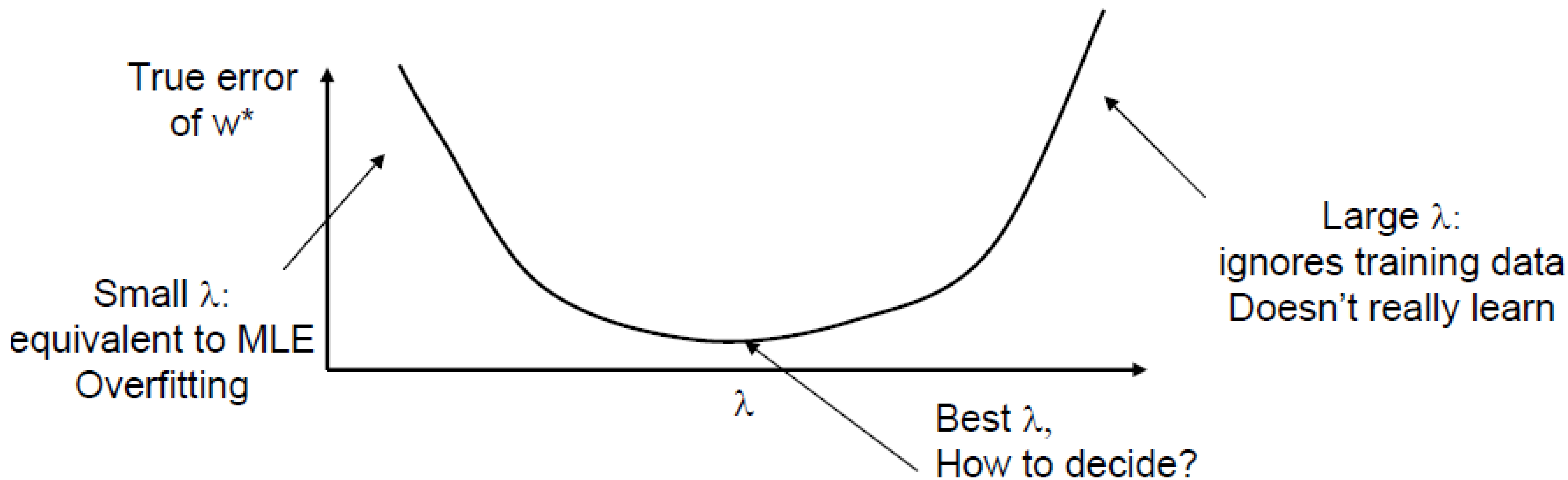
$$= \operatorname*{argmax}_{\mathbf{w}} \sum_j \log p(y^j|\mathbf{x}^j, \mathbf{w}) - \frac{\lambda}{2} \sum_i w_i^2$$

Old delta:

$$\boxed{\nabla L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \hat{y}^i)\mathbf{x}^i} \Rightarrow \boxed{\nabla L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \hat{y}^i)\mathbf{x}^i - \lambda \mathbf{w}}$$

Impact of λ

- λ is inversely proportional to the variance of our prior belief $\boxed{\lambda = \frac{1}{\sigma^2}}$



- Use cross-validation to choose

Summary

- **Minimizing training error will not necessarily minimize testing error – overfitting**
- **Hold-Out and Cross Validation**
  - Empirical methods for estimating the true error
  - Hold-out less expensive, but only uses part of the data and potentially can overfit to the validation data
  - LOO is the most accurate estimate one can get, but it is very expensive
  - K-fold cross validation is much more practical
- **Penalty method adds a penalty term to the normal objective function**
  - MAP estimation is often used to derive penalty methods
  - Often require parameter tuning – use cross validation