Practical Issues: Model/Feature Selection and Debugging Learning Algorithms

Piyush Rai

Machine Learning (CS771A)

Oct 19, 2016

Model Selection

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**. The set $\mathcal{M}$ may consist of:

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees
  - Kernel Methods: Different choices of kernels

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees
  - Kernel Methods: Different choices of kernels
  - Regularized Models: Different choices of the regularization hyperparameter

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**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees
  - Kernel Methods: Different choices of kernels
  - Regularized Models: Different choices of the regularization hyperparameter
- Different types of learning models (e.g., SVM, KNN, DT, etc.)

What is Model Selection?

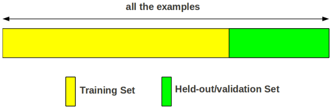
Given a set of models $\mathcal{M} = \{M_1, M_2, \ldots, M_R\}$, choose the model that is expected to do the best on the **test data**. The set $\mathcal{M}$ may consist of:

- Instances of same model with different complexities or hyperparams. E.g.,
  - K-Nearest Neighbors: Different choices of K
  - Decision Trees: Different choices of the number of levels/leaves
  - Polynomial Regression: Polynomials with different degrees
  - Kernel Methods: Different choices of kernels
  - Regularized Models: Different choices of the regularization hyperparameter
- Different types of 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)

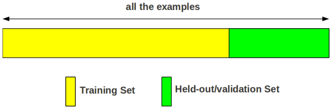
Held-out Data

- Set aside a fraction of the training data. This will be our held-out data.
    - Other names: validation/development data.



Held-out Data

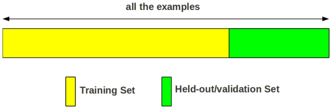
- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.



- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training

Held-out Data

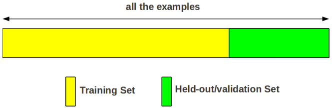
- Set aside a fraction of the training data. This will be our held-out data.
    - Other names: validation/development data.



- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training
- Train each model using the remaining training data

Held-out Data

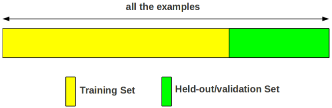
- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.



- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training
- Train each model using the remaining training data
- Evaluate error on the held-out data (cross-validation)

Held-out Data

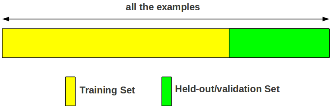
- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.



- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training
- Train each model using the remaining training data
- Evaluate error on the held-out data (cross-validation)
- Choose the model with the smallest held-out error

Held-out Data

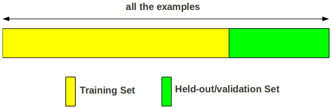
- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.



- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training
- Train each model using the remaining training data
- Evaluate error on the held-out data (cross-validation)
- Choose the model with the smallest held-out error
- Problems:
  - Wastes training data. Typically used when we have plenty of training data

Held-out Data

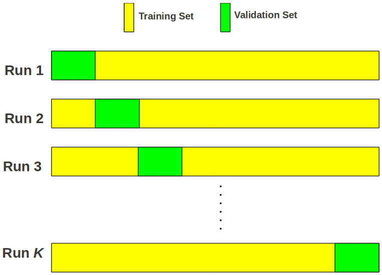
- Set aside a fraction of the training data. This will be our held-out data.
  - Other names: validation/development data.



- **Remember:** Held-out data is NOT the test data. DO NOT peek into the test data during training
- Train each model using the remaining training data
- Evaluate error on the held-out data (cross-validation)
- Choose the model with the smallest held-out error
- Problems:
  - Wastes training data. Typically used when we have plenty of training data
  - What if there was an unfortunate train/held-out split?

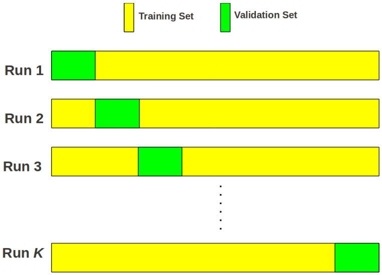
K-fold Cross-Validation

- Create K (e.g., 5 or 10) equal sized partitions of the training data
- Each partition has N/K examples
- Train using $K-1$ partitions, validate on the remaining partition
- Repeat this K times, each with a different validation partition



K-fold Cross-Validation

- Create K (e.g., 5 or 10) equal sized partitions of the training data
- Each partition has N/K examples
- Train using $K-1$ partitions, validate on the remaining partition
- Repeat this K times, each with a different validation partition



- Average the K validation errors

K-fold Cross-Validation

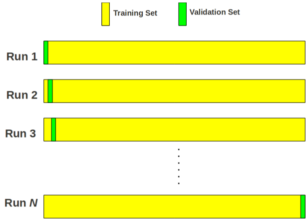
- Create K (e.g., 5 or 10) equal sized partitions of the training data
- Each partition has N/K examples
- Train using $K-1$ partitions, validate on the remaining partition
- Repeat this K times, each with a different validation partition

- Average the K validation errors
- Choose the model that gives the smallest average validation error

Leave-One-Out (LOO) Cross-Validation

Special case of K-fold CV when $K = N$

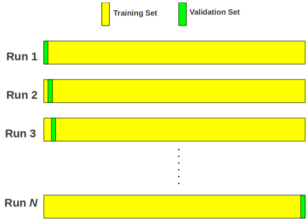
- Each partition is now a single example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same N times, each with a different validation example



Leave-One-Out (LOO) Cross-Validation

Special case of K-fold CV when $K = N$

- Each partition is now a single example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same N times, each with a different validation example

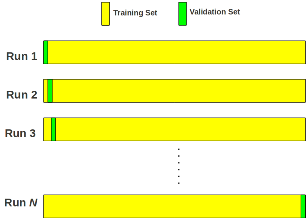


- Average the N validation errors. Choose the model with smallest error

Leave-One-Out (LOO) Cross-Validation

Special case of K-fold CV when $K = N$

- Each partition is now a single example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same N times, each with a different validation example

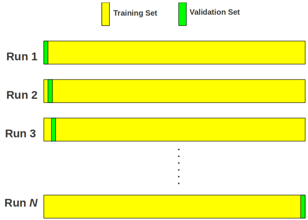


- Average the N validation errors. Choose the model with smallest error
- Can be expensive in general, especially for large N
  - But very efficient when used for selecting the number of neighbors to consider in nearest neighbor methods

Leave-One-Out (LOO) Cross-Validation

Special case of K-fold CV when $K = N$

- Each partition is now a single example
- Train using $N - 1$ examples, validate on the remaining example
- Repeat the same N times, each with a different validation example



- Average the N validation errors. Choose the model with smallest error
- Can be expensive in general, especially for large N
  - But very efficient when used for selecting the number of neighbors to consider in nearest neighbor methods (reason: NN methods require no training)

Random Subsampling based Cross-Validation

- Subsample a fixed fraction αN $(0 < \alpha < 1)$ as examples as validation set

Random Subsampling based Cross-Validation

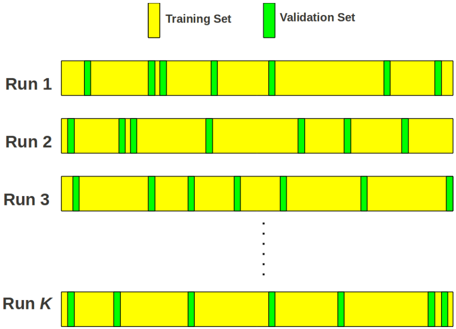
- Subsample a fixed fraction αN $(0 < \alpha < 1)$ as examples as validation set
- Train using the rest of the examples, calculate the validation error

Random Subsampling based Cross-Validation

- Subsample a fixed fraction αN $(0 < \alpha < 1)$ as examples as validation set
- Train using the rest of the examples, calculate the validation error
- Repeat K times, each with a different, randomly chosen validation set

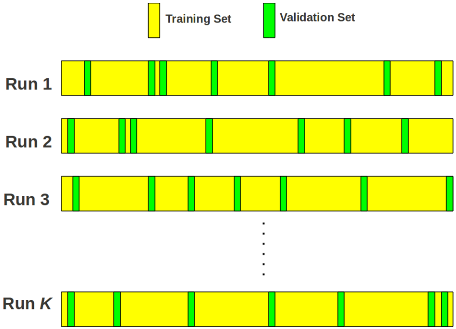
Random Subsampling based Cross-Validation

- Subsample a fixed fraction αN $(0 < \alpha < 1)$ as examples as validation set
- Train using the rest of the examples, calculate the validation error
- Repeat K times, each with a different, randomly chosen validation set



Random Subsampling based Cross-Validation

- Subsample a fixed fraction αN $(0 < \alpha < 1)$ as examples as validation set
- Train using the rest of the examples, calculate the validation error
- Repeat K times, each with a different, randomly chosen validation set



- Average the K validation errors. Choose the model with smallest error

The Bootstrap

- Idea: Given N examples, sample N elements **with replacement**
  - An already chosen example could be picked again

The Bootstrap

- Idea: Given N examples, sample N elements **with replacement**
  - An already chosen example could be picked again
- Use these N examples (with possible repeats) as the training data

The Bootstrap

- Idea: Given N examples, sample N elements **with replacement**
  - An already chosen example could be picked again
- Use these N examples (with possible repeats) as the training data
- Use the set of examples not selected as the validation data

The Bootstrap

- Idea: Given N examples, sample N elements **with replacement**
  - An already chosen example could be picked again
- Use these N examples (with possible repeats) 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

$$\text{Fraction of examples not picked:} \quad \left(1 - \frac{1}{N}\right)^N \approx e^{-1} \approx 0.368$$

- Training data is *inherently* small $\Rightarrow$ error estimate may be pessimistic

The Bootstrap

- Idea: Given N examples, sample N elements **with replacement**
  - An already chosen example could be picked again
- Use these N examples (with possible repeats) 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

$$\text{Fraction of examples not picked:} \quad \left(1 - \frac{1}{N}\right)^N \approx e^{-1} \approx 0.368$$

- Training data is *inherently* small $\Rightarrow$ error estimate may be pessimistic
- Use the following equation to compute the expected model error

$$err = 0.632 \times err_{\text{test-examples}} + 0.368 \times err_{\text{training-examples}}$$

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- $\mathcal{L}$: maximum value of the likelihood of the model

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- $\mathcal{L}$: maximum value of the likelihood of the model
- Applicable for probabilistic models (when likelihood is defined)

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- $\mathcal{L}$: maximum value of the likelihood of the model
- Applicable for probabilistic models (when likelihood is defined)
- AIC/BIC penalize model complexity
  - .. as measured by the number of model parameters

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- $\mathcal{L}$: maximum value of the likelihood of the model
- 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

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- $\mathcal{L}$: maximum value of the likelihood of the model
- 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

Information Criteria based methods

- Akaike Information Criteria (AIC)

$$AIC = 2k - 2\log(\mathcal{L})$$

- Bayesian Information Criteria (BIC)

$$BIC = k\log(N) - 2\log(\mathcal{L})$$

- k: # of model parameters
- $\mathcal{L}$: maximum value of the likelihood of the model
- 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

Feature Selection

Feature Selection

Selecting a useful subset of features from all the features

Feature Selection

Selecting a useful subset of features from all the features

Why Feature Selection?

Feature Selection

Selecting a useful subset of features from all the features

Why Feature Selection?

- Some algorithms scale poorly with increased dimension (# of features)

Feature Selection

Selecting a useful subset of features from all the features

Why Feature Selection?

- Some algorithms scale poorly with increased dimension (# of features)
- Irrelevant features can confuse some algorithms

Feature Selection

Selecting a useful subset of features from all the features

Why Feature Selection?

- Some algorithms scale poorly with increased dimension (# of features)
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization

Feature Selection

Selecting a useful subset of features from all the features

Why Feature Selection?

- Some algorithms scale poorly with increased dimension (# of features)
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Feature selection can help reduce data set size and resulting model size

Feature Selection

Selecting a useful subset of features from all the features

Why Feature Selection?

- Some algorithms scale poorly with increased dimension (# of features)
- Irrelevant features can confuse some algorithms
- Redundant features adversely affect regularization
- Feature selection can help reduce data set size 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 (e.g., PCA or other dimensionality reduction methods)

Feature Selection Methods

- Methods agnostic to the learning algorithm

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
    - In general, features that have low variance across examples can be discarded

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
    - In general, features that have low variance across examples can be discarded
  - Filter Feature Selection methods
    - Use some ranking criteria to rank features
    - Select the top ranking features

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
    - In general, features that have low variance across examples can be discarded
  - 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)

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
    - In general, features that have low variance across examples can be discarded
  - 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

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
    - In general, features that have low variance across examples can be discarded
  - 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

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
    - In general, features that have low variance across examples can be discarded
  - 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
- Learning algorithms that can identify the relevant features (e.g., sparse models with ℓ_1 regularization on the weight vector)

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
    - In general, features that have low variance across examples can be discarded
  - 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
- Learning algorithms that can identify the relevant features (e.g., sparse models with ℓ_1 regularization on the weight vector)

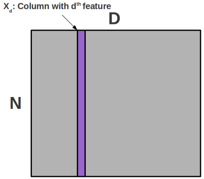
(Also see: "An Introduction to Variable and Feature Selection" by Guyon and Elisseeff)

Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually

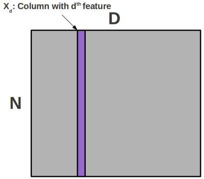
Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually



Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually

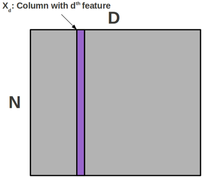


- **Correlation Criteria:** Rank features in order of their correlation with labels

$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually



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

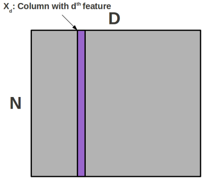
$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

- **Mutual Information Criteria:**

$$MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{0,1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}$$

Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually



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

$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

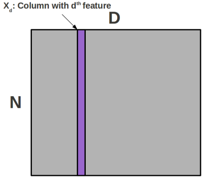
- **Mutual Information Criteria:**

$$MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{0,1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}$$

  - High mutual information mean high relevance of that feature

Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually



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

$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

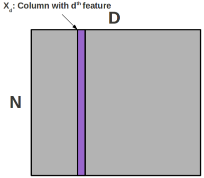
- **Mutual Information Criteria:**

$$MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{0,1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}$$

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

Filter Feature Selection

- Uses statistical tests to measure relevance of each feature individually



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

$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

- **Mutual Information Criteria:**

$$MI(X_d, Y) = \sum_{X_d \in \{0,1\}} \sum_{Y \in \{0,1\}} P(X_d, Y) \frac{\log P(X_d, Y)}{P(X_d)P(Y)}$$

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

- Various other statistical tests exist, e.g., χ^2 test

Wrapper Methods

- **Forward Search**
  - Let $\mathcal{F} = \{\}$

Wrapper Methods

- **Forward Search**
  - Let $\mathcal{F} = \{\}$
  - While not selected desired number of features
  - For each unused feature f:

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 $\mathcal{F} \bigcup f$ (using 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 $\mathcal{F} \bigcup f$ (using cross-validation)
  - Add f with lowest error to $\mathcal{F}$
- **Backward Search**
  - Let $\mathcal{F} = \{\text{all features}\}$

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 $\mathcal{F} \bigcup f$ (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 $f \in \mathcal{F}$:

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 $\mathcal{F} \bigcup f$ (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 $\mathcal{F} \backslash f$ (using 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 $\mathcal{F} \bigcup f$ (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 $\mathcal{F} \backslash f$ (using cross-validation)
  - Remove f with lowest error from $\mathcal{F}$
- In practice, these methods can be expensive. Also myopic and sub-optimal because the adding/removing of features is greedy

Debugging Learning Algorithms

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?
  - Introduce new features (can be combinations of existing features)?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?
  - Introduce new features (can be combinations of existing features)?
  - Try tuning the regularization parameter?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?
  - Introduce new features (can be combinations of existing features)?
  - Try tuning the regularization parameter?
  - Run (the iterative) optimizer longer, i.e., for more iterations?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?
  - Introduce new features (can be combinations of existing features)?
  - Try tuning the regularization parameter?
  - Run (the iterative) optimizer longer, i.e., for more iterations?
  - Change the optimization algorithm (e.g., GD to SGD or Newton..)?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?
  - Introduce new features (can be combinations of existing features)?
  - Try tuning the regularization parameter?
  - Run (the iterative) optimizer longer, i.e., for more iterations?
  - Change the optimization algorithm (e.g., GD to SGD or Newton..)?
  - Give up and switch to a different model (e.g., SVM)?

Debugging Learning Algorithms

- A notoriously hard problem in general
  - Note that code for ML algorithms is not procedural but data-driven
- What to do when our model (say logistic regression) isn't doing well (i.e., giving an acceptable level of test accuracy) but you are confident that your implementation is otherwise correct?
  - Use more training examples to train the model?
  - Use a smaller number of features?
  - Introduce new features (can be combinations of existing features)?
  - Try tuning the regularization parameter?
  - Run (the iterative) optimizer longer, i.e., for more iterations?
  - Change the optimization algorithm (e.g., GD to SGD or Newton..)?
  - Give up and switch to a different model (e.g., SVM)?
- How to know what might be going wrong and how to debug?

Bias-Variance Decomposition

- For some model $y = f(\boldsymbol{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, given its estimate $\hat{f}$ learned by a "learner" using a finite training set, the following decomposition holds

$$\mathbb{E}[(y - \hat{f}(\boldsymbol{x}))^2] = \text{Bias}[\hat{f}(\boldsymbol{x})]^2 + \text{Var}[\hat{f}(\boldsymbol{x})] + \sigma^2$$

- Note: The above expectation is over all choices of training sets

Bias-Variance Decomposition

- For some model $y = f(\boldsymbol{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, given its estimate $\hat{f}$ learned by a "learner" using a finite training set, the following decomposition holds

$$\mathbb{E}[(y - \hat{f}(\boldsymbol{x}))^2] = \text{Bias}[\hat{f}(\boldsymbol{x})]^2 + \text{Var}[\hat{f}(\boldsymbol{x})] + \sigma^2$$

- Note: The above expectation is over all choices of training sets
- $\text{Bias}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})]$:Error due to wrong (perhaps too simple) model

Bias-Variance Decomposition

- For some model $y = f(\boldsymbol{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, given its estimate $\hat{f}$ learned by a "learner" using a finite training set, the following decomposition holds

$$\mathbb{E}[(y - \hat{f}(\boldsymbol{x}))^2] = \text{Bias}[\hat{f}(\boldsymbol{x})]^2 + \text{Var}[\hat{f}(\boldsymbol{x})] + \sigma^2$$

- Note: The above expectation is over all choices of training sets
- $\text{Bias}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})]$:Error due to wrong (perhaps too simple) model
- $\text{Var}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x})^2] - \mathbb{E}[\hat{f}(\boldsymbol{x})]^2$:Learner's sensitivity to choice of training set

Bias-Variance Decomposition

- For some model $y = f(\boldsymbol{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, given its estimate $\hat{f}$ learned by a "learner" using a finite training set, the following decomposition holds

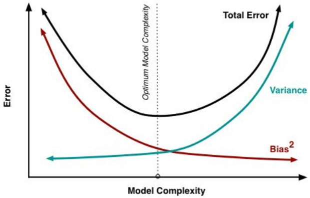
$$\mathbb{E}[(y - \hat{f}(\boldsymbol{x}))^2] = \text{Bias}[\hat{f}(\boldsymbol{x})]^2 + \text{Var}[\hat{f}(\boldsymbol{x})] + \sigma^2$$

- Note: The above expectation is over all choices of training sets
- $\text{Bias}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})]$:Error due to wrong (perhaps too simple) model
- $\text{Var}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x})^2] - \mathbb{E}[\hat{f}(\boldsymbol{x})]^2$:Learner's sensitivity to choice of training set
- The proof (note that $\mathbb{E}[y] = f(\boldsymbol{x})$):

$$\begin{aligned}
\mathrm{E}\big[(y-\hat{f})^2\big] &= \mathrm{E}[y^2 + \hat{f}^2 - 2y\hat{f}] \\
&= \mathrm{E}[y^2] + \mathrm{E}[\hat{f}^2] - \mathrm{E}[2y\hat{f}] \\
&= \mathrm{Var}[y] + \mathrm{E}[y]^2 + \mathrm{Var}[\hat{f}] + \mathrm{E}[\hat{f}]^2 - 2f\mathrm{E}[\hat{f}] \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f - \mathrm{E}[\hat{f}])^2 \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + \mathrm{E}[f - \hat{f}]^2 \\
&= \sigma^2 + \mathrm{Var}[\hat{f}] + \mathrm{Bias}[\hat{f}]^2
\end{aligned}$$

Bias-Variance Trade-off

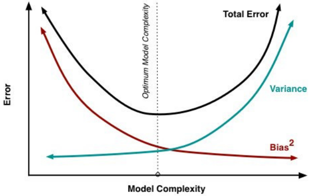
- Simple models have high bias and small variance, complex models have small bias and high variance



	Model				
	Bias	Variance	Complexity	Flexibility	Generalizability
Underfitting: you have an overly simple model	High	Low	Low	Low	High
Overfitting: your model is modelling the noise	Low	High	High	High	Low

Bias-Variance Trade-off

- Simple models have high bias and small variance, complex models have small bias and high variance



	Model				
	Bias	Variance	Complexity	Flexibility	Generalizability
Underfitting: you have an overly simple model	High	Low	Low	Low	High
Overfitting: your model is modelling the noise	Low	High	High	High	Low

- If you modified a model to reduce its bias (e.g., by increasing the model's complexity), you are likely to increase its variance, and vice-versa (if both increase then you might be doing it wrong!)

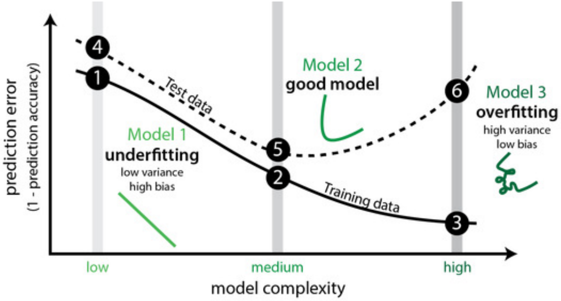
(Pic courtesy: Scott Fortmann-Roe, Latysheva and Ravarani)

High Bias or High Variance?

- The bad performance (low accuracy on test data) could be due either
  - High Bias (Underfitting)
  - High Variance (Overfitting)

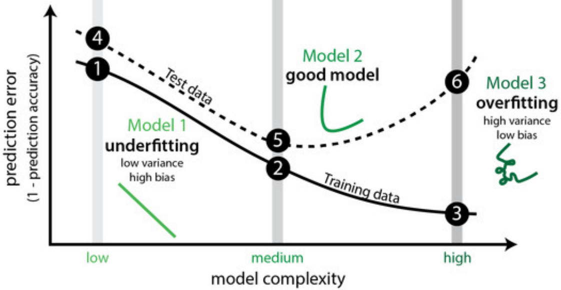
High Bias or High Variance?

- The bad performance (low accuracy on test data) could be due either
  - High Bias (Underfitting)
  - High Variance (Overfitting)
- Looking at the training and test error can tell which of the two is the case



High Bias or High Variance?

- The bad performance (low accuracy on test data) could be due either
  - High Bias (Underfitting)
  - High Variance (Overfitting)
- Looking at the training and test error can tell which of the two is the case



- High Bias: Both training and test errors are large
- High Variance: Small training error, large test error (and huge gap)

(Pic courtesy: Latysheva and Ravarani)

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)
- Suppose you have learned two models $\boldsymbol{w}_{LR}$ and $\boldsymbol{w}_{SVM}$ (LR and SVM, respectively) using the same training data, and SVM gives higher test accuracy. How do I know why LR does worse and what could I improve it?

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)
- Suppose you have learned two models $\boldsymbol{w}_{LR}$ and $\boldsymbol{w}_{SVM}$ (LR and SVM, respectively) using the same training data, and SVM gives higher test accuracy. How do I know why LR does worse and what could I improve it?
  - Is it because the optimizer for LR didn't do a good job at finding the optima?

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)
- Suppose you have learned two models $\boldsymbol{w}_{LR}$ and $\boldsymbol{w}_{SVM}$ (LR and SVM, respectively) using the same training data, and SVM gives higher test accuracy. How do I know why LR does worse and what could I improve it?
  - Is it because the optimizer for LR didn't do a good job at finding the optima?
  - Is my model choice (choosing LR over SVM) wrong for this data set?

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)
- Suppose you have learned two models $\boldsymbol{w}_{LR}$ and $\boldsymbol{w}_{SVM}$ (LR and SVM, respectively) using the same training data, and SVM gives higher test accuracy. How do I know why LR does worse and what could I improve it?
  - Is it because the optimizer for LR didn't do a good job at finding the optima?
  - Is my model choice (choosing LR over SVM) wrong for this data set?
  - Looking at the value of the LR loss function $\mathcal{L}$ can give some insights

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)
- Suppose you have learned two models $\boldsymbol{w}_{LR}$ and $\boldsymbol{w}_{SVM}$ (LR and SVM, respectively) using the same training data, and SVM gives higher test accuracy. How do I know why LR does worse and what could I improve it?
  - Is it because the optimizer for LR didn't do a good job at finding the optima?
  - Is my model choice (choosing LR over SVM) wrong for this data set?
  - Looking at the value of the LR loss function $\mathcal{L}$ can give some insights
  - If $\mathcal{L}(\boldsymbol{w}_{SVM}) < \mathcal{L}(\boldsymbol{w}_{LR})$ then improving the LR optimizer might help

Some Guidelines for Debugging Learning Algorithms

- Adding more training examples won't usually bring the bias down. If your model has a high bias, try making the model richer (e.g., adding more features or using a more sophisticated model).
- Using more training data can help bring the variance down. If your model has a high variance, try adding more training examples or make model simpler (e.g., use fewer features or regularize more)
- Suppose you have learned two models $\boldsymbol{w}_{LR}$ and $\boldsymbol{w}_{SVM}$ (LR and SVM, respectively) using the same training data, and SVM gives higher test accuracy. How do I know why LR does worse and what could I improve it?
  - Is it because the optimizer for LR didn't do a good job at finding the optima?
  - Is my model choice (choosing LR over SVM) wrong for this data set?
  - Looking at the value of the LR loss function $\mathcal{L}$ can give some insights
  - If $\mathcal{L}(\boldsymbol{w}_{SVM}) < \mathcal{L}(\boldsymbol{w}_{LR})$ then improving the LR optimizer might help
  - If $\mathcal{L}(\boldsymbol{w}_{LR}) < \mathcal{L}(\boldsymbol{w}_{SVM})$ then LR isn't a good model for this problem

Next Class: Ensemble Methods