Model Selection and Feature Selection

Piyush Rai

CS5350/6350: Machine Learning

September 22, 2011

Feature Selection

Selecting a useful subset from all the features

Feature Selection

Selecting a useful subset from all the features

Why Feature Selection?

Feature Selection

Selecting a useful subset from all the features

Why Feature Selection?

- Some algorithms scale (computationally) poorly with increased dimension

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

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

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)

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

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

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

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

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)

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

Feature Selection Methods

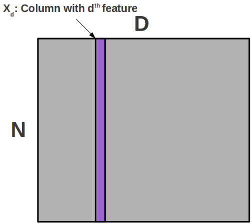
- Methods agnostic to the learning algorithm
  - Preprocessing based methods
    - E.g., remove a binary feature if it's ON in very few or most examples
  - Filter Feature Selection methods
    - Use some ranking criteria to rank features
    - Select the top ranking features
- Wrapper Methods (keep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

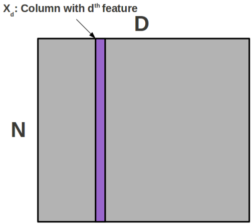
Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods

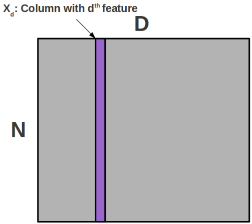


- **Correlation Critera:** Rank features in order of their correlation with the labels

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

Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



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

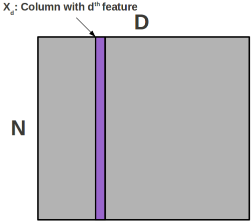
$$R(X_d, Y) = \frac{cov(X_d, Y)}{\sqrt{var(X_d)var(Y)}}$$

- **Mutual Information Criteria:**

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

Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



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

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

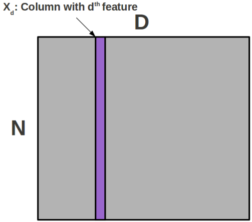
- **Mutual Information Criteria:**

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

  - High mutual information mean high relevance of that feature

Filter Feature Selection

- Uses heuristics but is much faster than wrapper methods



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

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

- **Mutual Information Criteria:**

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

  - 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

Wrapper Methods

- Two types: Forward Search and Backward Search
  - **Forward Search**

Wrapper Methods

- Two types: Forward Search and Backward Search
    - **Forward Search**
        - Start with no features

Wrapper Methods

- Two types: Forward Search and Backward Search
  - **Forward Search**
    - Start with no features
    - Greedily include the most relevant feature

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**

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

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

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

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 $\mathcal{F} = \{\}$

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 $\mathcal{F}$

Summary: feature engineering

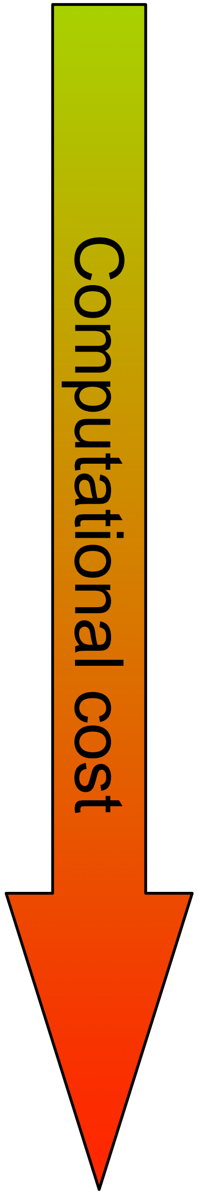
- Feature engineering is often crucial to get good results
- Strategy: overshoot and regularize
  - Come up with lots of features: better to include irrelevant features than to miss important features
  - Use regularization or feature selection to prevent overfitting
  - Evaluate your feature engineering on DEV set. Then, when the feature set is frozen, evaluate on TEST to get a final evaluation (Daniel will say more on evaluation next week)

Summary: feature selection

When should you do it?

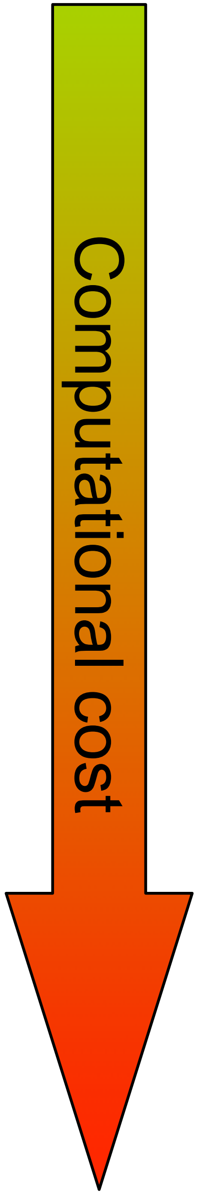
- If the only concern is accuracy, and the whole dataset can be processed, feature selection not needed (as long as there is regularization)
- If computational complexity is critical (embedded device, web-scale data, fancy learning algorithm), consider using feature selection
  - But there are alternatives: e.g. the Hash trick, a fast, non-linear dimensionality reduction technique [Weinberger et al. 2009]
- When you care about the feature themselves
  - Keep in mind the correlation/causation issues
  - See [Guyon et al., Causal feature selection, 07]

Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

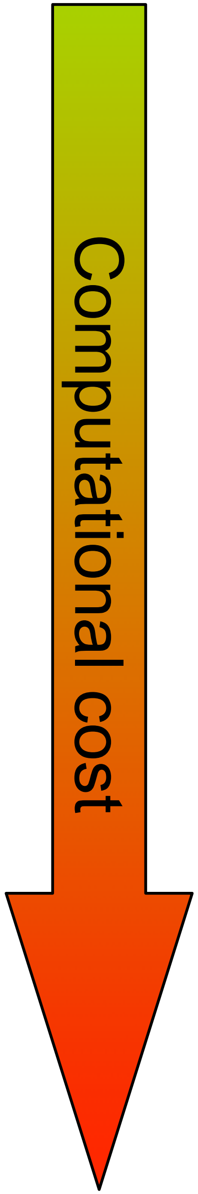
Summary: how to do feature selection



- *Filtering*
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Good preprocessing step
- Fails to capture relationship between features

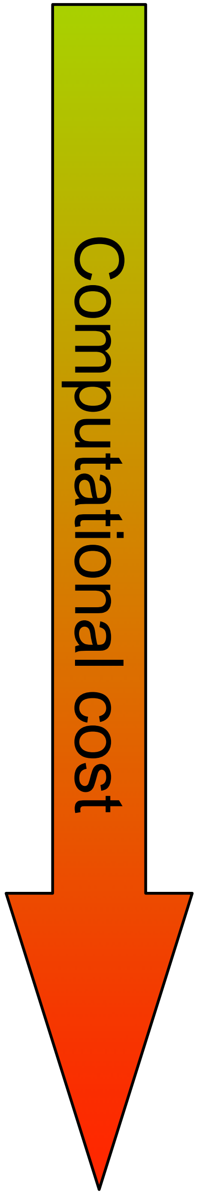
Summary: how to do feature selection



- Filtering
- *L_1 regularization (embedded methods)*
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Fairly efficient
  - LARS-type algorithms now exist for many linear models.

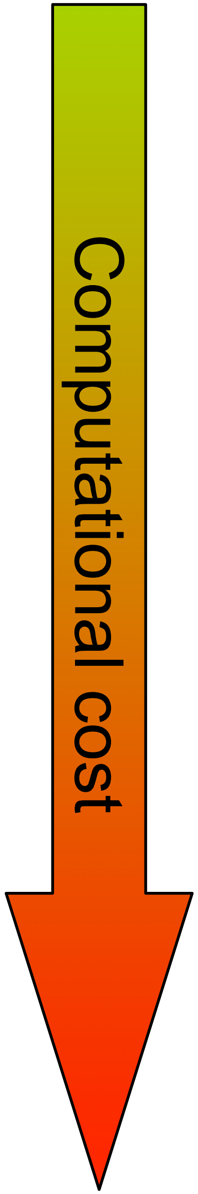
Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- *Wrappers*
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Most directly optimize prediction performance
- Can be very expensive, even with greedy search methods
- Cross-validation is a good objective function to start with

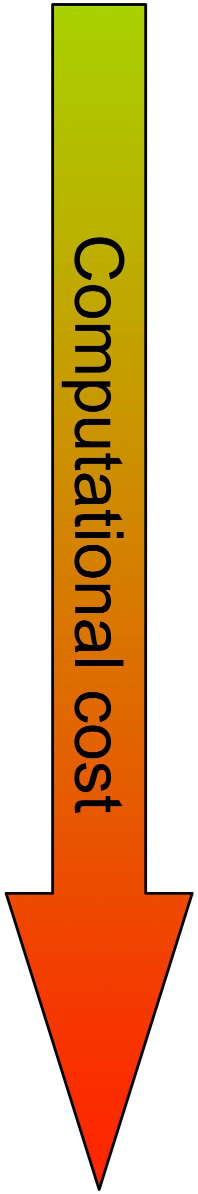
Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - *[Forward selection]*
  - *[Backward selection]*
  - Other search
  - Exhaustive

- Too greedy—ignore relationships between features
- Easy baseline
- Can be generalized in many interesting ways
  - Stagewise forward selection
  - Forward-backward search
  - Boosting

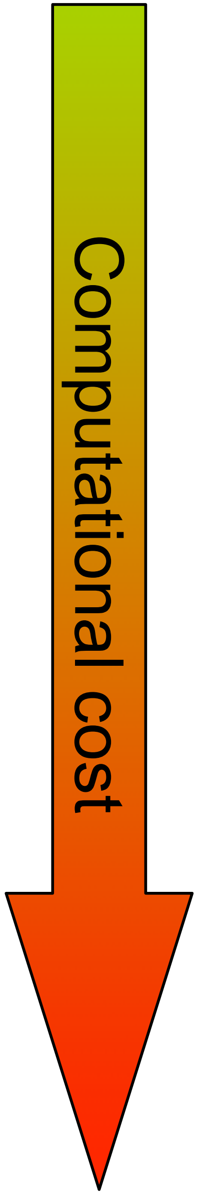
Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - *Other search*
  - Exhaustive

- Generally more effective than greedy

Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - *Exhaustive*

- The "ideal"
- Very seldom done in practice
- With cross-validation objective, there's a chance of over-fitting
  - *Some* subset might randomly perform quite well in cross-validation