Statistical Machine Learning from Data

Feature Selection

Samy Bengio

IDIAP Research Institute, Martigny, Switzerland, and
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
bengio@idiap.ch
http://www.idiap.ch/~bengio

February 7, 2006
1 Motivation

2 Filters

3 Wrappers

4 Feature Weighting
1 Motivation

2 Filters

3 Wrappers

4 Feature Weighting
Why Should We Select Features?

- Some problems are defined by 100 or even 1000 input features.
- Most Machine Learning models have to attribute parameters to handle these features (often at least linearly as much).
- Hence, capacity is determined by the number of features.
- If most features are noise, then most of the parameters will be useless → capacity is wasted.
- Worse, the algorithm might find false regularities in the input features of the training data and use the wasted capacity to represent them!
- Other problem: curse of dimensionality.
- Finally: for more interpretability and efficiency.
Classes of Feature Selection Methods

Broad classes of feature selection methods:

- **Filter Methods:**
  - Select the best features according to a reasonable criterion
  - The criterion is independent of the real problem

- **Wrapper Methods:**
  - Select the best features according to the final criterion
  - For each subset of features, try to solve the problem

- In any case, there are \( \sum_{p=1}^{n} C^n_p = \sum_{p=1}^{n} \frac{n!}{p!(1-p)!} \) combinations

- Alternative: weighting methods.
1 Motivation

2 Filters

3 Wrappers

4 Feature Weighting
Filter Methods

- **Basic idea:** select the best features according to some prior knowledge
- **Examples** of prior knowledge:
  - if we accept to transform the features...
    - features should be uncorrelated $\rightarrow$ perform a PCA and keep only the eigenvectors corresponding to $x\%$ of the variance.
    - similar ideas: linear discriminant analysis (LDA), independent component analysis (ICA)
  - features should have strong correlation with the target $\rightarrow$ select the $k$ features most linearly correlated to the target
  - features should have strong correlation with the target $\rightarrow$ select the $k$ features with highest mutual information with the target:
    \[
    I(x, y) = \sum_i \sum_j p(x = i, y = j) \log \left( \frac{p(x = i, y = j)}{p(x = i)p(y = j)} \right)
    \]
1 Motivation

2 Filters

3 Wrappers

4 Feature Weighting
Wrapper Methods

- Basic (naive) algorithm:
  1. For each subset of features, solve the problem.
  2. Select the best subset.

- Impossible because the problem is exponentially long!

- Alternatives: greedy heuristics such as forward selection or backward elimination
Forward Selection

1. let $\mathcal{P} = \emptyset$ be the current set of selected features
2. let $\mathcal{Q}$ be the full set of features
3. while size of $\mathcal{P}$ smaller than a given constant
   - for each $\nu \in \mathcal{Q}$
     1. set $\mathcal{P}' \leftarrow \{\nu\} \cup \mathcal{P}$
     2. train the model with $\mathcal{P}'$ and keep the validation performance
     2. set $\mathcal{P} \leftarrow \{\nu^*\} \cup \mathcal{P}$ where $\nu^*$ corresponds to the best validation performance obtained in step 3.1
     3. set $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{\nu^*\}$
     4. keep the validation performance obtained with current $\mathcal{P}$
4. return the best set $\mathcal{P}$
Backward Elimination

1. let $\mathcal{P}$ be the full set of features
2. while size of $\mathcal{P}$ greater than a given constant
   1. for each $v \in \mathcal{P}$
      1. set $\mathcal{P}' \leftarrow \mathcal{P} \setminus \{v\}$
      2. train the model with $\mathcal{P}'$ and keep the validation performance
   2. set $\mathcal{P} \leftarrow \mathcal{P} \setminus \{v^*\}$ where $v^*$ corresponds to the worst validation performance obtained in step 2.1
   3. keep the validation performance obtained with current $\mathcal{P}$
3. return the best set $\mathcal{P}$
Both methods are ultimately heuristics because of the combinatorial barrier.

Wrappers try to solve the real problem, hence you really optimize your criterion.

Filters solve a different problem... it might not be appropriate.

Wrappers are potentially very time consuming: you have to solve the ultimate problem numerous times.

Filters are much faster because the problem they solve is in general simpler.
1 Motivation
2 Filters
3 Wrappers
4 Feature Weighting
Instead of selecting a subset of features, which is a combinatorial problem, why not simply weight them?

Most feature weighting methods are based on the wrapper approach

**Heuristics** for feature weighting:

- **gradient descent** on the input space → train with all features, then fix the parameters and estimate the importance of each input, and loop
- **AdaBoost** when each model is trained on one feature only (→ final solution is a linear combination)