Model and Feature Selection

Dr. Víctor Uc Cetina

Facultad de Matemáticas
Universidad Autónoma de Yucatán

cetina@informatik.uni-hamburg.de
https://sites.google.com/view/victoruccetina

Content

Model Selection

- Suppose that we are trying to select among several different models for a learning problem.

Model Selection

- Suppose that we are trying to select among several different models for a learning problem.
- For instance, we might be using a polynomial regression model $h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k)$, and we wish to decide if k should be $0, 1, \ldots,$ or 10.

Model Selection

- Suppose that we are trying to select among several different models for a learning problem.
- For instance, we might be using a polynomial regression model $h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k)$, and we wish to decide if k should be $0, 1, \ldots,$ or 10.
- How can we automatically select a model that represents a good tradeoff between bias and variance?

Model Selection

- Suppose that we are trying to select among several different models for a learning problem.
- For instance, we might be using a polynomial regression model $h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k)$, and we wish to decide if k should be $0, 1, \ldots,$ or 10.
- How can we automatically select a model that represents a good tradeoff between bias and variance?
- Alternatively, suppose that we want to automatically choose the bandwidth parameter τ for locally weighted regression, or the parameter C for our SVM.

Model Selection

- Suppose that we are trying to select among several different models for a learning problem.
- For instance, we might be using a polynomial regression model $h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_1 x + \theta_2 x^2 + \ldots + \theta_k x^k)$, and we wish to decide if k should be $0, 1, \ldots,$ or 10.
- How can we automatically select a model that represents a good tradeoff between bias and variance?
- Alternatively, suppose that we want to automatically choose the bandwidth parameter τ for locally weighted regression, or the parameter C for our SVM.
- How can we do that?

Model Selection

- We will assume that we have some finite set of models $\mathcal{M} = \{M_1, M_2, \ldots, M_d\}$ that we are trying to select among.

Model Selection

- We will assume that we have some finite set of models $\mathcal{M} = \{M_1, M_2, \dots, M_d\}$ that we are trying to select among.
- For instance, in our first example above, the model M_i, would be an i-th order polynomial regression model.

Model Selection

- We will assume that we have some finite set of models $\mathcal{M} = \{M_1, M_2, \ldots, M_d\}$ that we are trying to select among.
- For instance, in our first example above, the model M_i, would be an i-th order polynomial regression model.
- Alternatively, if we are trying to decide between using an SVM, a neural network or logistic regression, then $\mathcal{M}$ may contain these models.

Cross Validation

- Given a training set S a possible empirical risk minimization algorithm might look like the following one:

Cross Validation

- Given a training set S a possible empirical risk minimization algorithm might look like the following one:
  1. Train each model M_i on S, to get some hypothesis h_i.

Cross Validation

- Given a training set S a possible empirical risk minimization algorithm might look like the following one:
  1. Train each model M_i on S, to get some hypothesis h_i.
  2. Pick the hypothesis with the smallest training error.

Cross Validation

- Given a training set S a possible empirical risk minimization algorithm might look like the following one:
  1. Train each model M_i on S, to get some hypothesis h_i.
  2. Pick the hypothesis with the smallest training error.
- Unfortunately, this algorithm does not work. Why?

Cross Validation

- Given a training set S a possible empirical risk minimization algorithm might look like the following one:
  1. Train each model M_i on S, to get some hypothesis h_i.
  2. Pick the hypothesis with the smallest training error.
- Unfortunately, this algorithm does not work. Why?
- Consider choosing the order of the polynomial. The higher the order of the polynomial, the better it will fit the training set S, and thus the lower the training error.

Cross Validation

- Given a training set S a possible empirical risk minimization algorithm might look like the following one:
  1. Train each model M_i on S, to get some hypothesis h_i.
  2. Pick the hypothesis with the smallest training error.
- Unfortunately, this algorithm does not work. Why?
- Consider choosing the order of the polynomial. The higher the order of the polynomial, the better it will fit the training set S, and thus the lower the training error.
- Hence, this method will always select a high-variance, high-degree polynomial model, which we saw previously is often a poor choice.

Cross Validation

- An algorithm that works better is the **hold-out cross validation**:

Cross Validation

- An algorithm that works better is the **hold-out cross validation**:
    1. Randomly split S into S_{train} (say 70% of the data) and S_{cv} (the remaining 30%). Here, S_{cv} is called the hold-out cross validation set.

Cross Validation

- An algorithm that works better is the **hold-out cross validation**:
  1. Randomly split S into S_{train} (say 70% of the data) and S_{cv} (the remaining 30%). Here, S_{cv} is called the hold-out cross validation set.
  2. Train each model M_i on S_{train} only, to get some hypothesis h_i.

Cross Validation

- An algorithm that works better is the **hold-out cross validation**:
  1. Randomly split S into S_{train} (say 70% of the data) and S_{cv} (the remaining 30%). Here, S_{cv} is called the hold-out cross validation set.
  2. Train each model M_i on S_{train} only, to get some hypothesis h_i.
  3. Select and output the hypothesis h_i that had the smallest error $\hat{\varepsilon}_{\text{cv}}(h)$ on the hold out cross validation set. Recall that $\hat{\varepsilon}_{\text{cv}}(h)$ denotes the empirical error of h on the set of examples in S_{cv}.

Cross Validation

- An algorithm that works better is the **hold-out cross validation**:
  1. Randomly split S into S_{train} (say 70% of the data) and S_{cv} (the remaining 30%). Here, S_{cv} is called the hold-out cross validation set.
  2. Train each model M_i on S_{train} only, to get some hypothesis h_i.
  3. Select and output the hypothesis h_i that had the smallest error $\hat{\varepsilon}_{\text{cv}}(h)$ on the hold out cross validation set. Recall that $\hat{\varepsilon}_{\text{cv}}(h)$ denotes the empirical error of h on the set of examples in S_{cv}.
- By testing on a set of examples S_{cv} that the models were not trained on, we obtain a better estimate of each hypothesis h_i's true generalization error, and can then pick the one with the smallest estimated generalization error.

Cross Validation

- Usually, somewhere between 1/4-1/3 of the data is used in the hold out cross validation set, and 30% is a typical choice.

Cross Validation

- Usually, somewhere between 1/4-1/3 of the data is used in the hold out cross validation set, and 30% is a typical choice.
- Optionally, step 3 in the algorithm may also be replaced with selecting the model M_i according to $\arg\min_i \hat{\varepsilon}_{s_{\mathrm{cv}}}(h_i)$, and then retraining M_i on the entire training set S.

Cross Validation

- Usually, somewhere between 1/4-1/3 of the data is used in the hold out cross validation set, and 30% is a typical choice.
- Optionally, step 3 in the algorithm may also be replaced with selecting the model M_i according to $\arg\min_i \hat{\varepsilon}_{s_{\mathrm{cv}}}(h_i)$, and then retraining M_i on the entire training set S.
- The disadvantage of using hold out cross validation is that it wastes about 30% of the data. It is as if we were training a model using only $0.7m$ training examples.

Cross Validation

- Usually, somewhere between 1/4-1/3 of the data is used in the hold out cross validation set, and 30% is a typical choice.
- Optionally, step 3 in the algorithm may also be replaced with selecting the model M_i according to $\arg\min_i \hat{\varepsilon}_{S_{\mathrm{cv}}}(h_i)$, and then retraining M_i on the entire training set S.
- The disadvantage of using hold out cross validation is that it wastes about 30% of the data. It is as if we were training a model using only $0.7m$ training examples.
- While this is fine if data is abundant and/or cheap, in learning problems in which data is scarce, we would like to do something better.

Cross Validation

- There is another method called k-**fold cross validation** that holds out less data each time:

Cross Validation

- There is another method called k-**fold cross validation** that holds out less data each time:
  1. Randomly split S into k disjoint subsets of m/k training examples each. Lets call this subsets $S_1, \ldots, S_k$.

Cross Validation

- There is another method called k-**fold cross validation** that holds out less data each time:
  1. Randomly split S into k disjoint subsets of m/k training examples each. Lets call this subsets $S_1, \ldots, S_k$.
  2. For each model M_i, we evaluate it as follows:
     For $j = 1, \ldots, k$

Cross Validation

- There is another method called k-**fold cross validation** that holds out less data each time:
  1. Randomly split S into k disjoint subsets of m/k training examples each. Lets call this subsets $S_1, \ldots, S_k$.
  2. For each model M_i, we evaluate it as follows:
     For $j = 1, \ldots, k$
     - Train the model M_i on $S_1 \cup \ldots \cup S_{j-1} \cup S_{j+1} \cup \ldots S_k$ (train on all the data except S_j) to get some hypothesis h_{ij}.

Cross Validation

- There is another method called k-**fold cross validation** that holds out less data each time:
  1. Randomly split S into k disjoint subsets of m/k training examples each. Lets call this subsets $S_1, \ldots, S_k$.
  2. For each model M_i, we evaluate it as follows:
     For $j = 1, \ldots, k$
     - Train the model M_i on $S_1 \cup \ldots \cup S_{j-1} \cup S_{j+1} \cup \ldots S_k$ (train on all the data except S_j) to get some hypothesis h_{ij}.
     - Test the hypothesis h_{ij} on S_j, to get $\hat{\varepsilon}_{s_j}(h_{ij})$. The estimated generalization error of model M_i is then calculated as the average of the $\hat{\varepsilon}_{s_j}(h_{ij})$'s (average over j).

Cross Validation

- There is another method called k-**fold cross validation** that holds out less data each time:
    1. Randomly split S into k disjoint subsets of m/k training examples each. Lets call this subsets $S_1, \ldots, S_k$.
    2. For each model M_i, we evaluate it as follows:
       For $j = 1, \ldots, k$
        - Train the model M_i on $S_1 \cup \ldots \cup S_{j-1} \cup S_{j+1} \cup \ldots S_k$ (train on all the data except S_j) to get some hypothesis h_{ij}.
        - Test the hypothesis h_{ij} on S_j, to get $\hat{\varepsilon}_{s_j}(h_{ij})$. The estimated generalization error of model M_i is then calculated as the average of the $\hat{\varepsilon}_{s_j}(h_{ij})$'s (average over j).
    3. Pick the model M_i with the lowest estimated generalization error, and retrain that model on the entire training set S. The resulting hypothesis is then output as our final answer.

Cross Validation

- A typical choice for the number of folds to use here would be $k = 10$.

Cross Validation

- A typical choice for the number of folds to use here would be $k = 10$.
- This procedure may also be more computationally expensive than hold-out cross validation, since now we need to train each model k times.

Cross Validation

- A typical choice for the number of folds to use here would be $k = 10$.
- This procedure may also be more computationally expensive than hold-out cross validation, since now we need to train each model k times.
- While $k = 10$ is a commonly used choice, in problems in which data is really scarce, sometimes we will use the extreme choice of $k = m$ in order to leave out as little data as possible each time.

Cross Validation

- A typical choice for the number of folds to use here would be $k = 10$.
- This procedure may also be more computationally expensive than hold-out cross validation, since now we need to train each model k times.
- While $k = 10$ is a commonly used choice, in problems in which data is really scarce, sometimes we will use the extreme choice of $k = m$ in order to leave out as little data as possible each time.
- This method is called **leave-one-out cross validation**.

Cross Validation

- Even though we have described the different versions of cross validation as methods for selecting a model, they can also be used more simply to evaluate a single model or algorithm.

Cross Validation

- Even though we have described the different versions of cross validation as methods for selecting a model, they can also be used more simply to evaluate a single model or algorithm.
- For example if you have implemented some learning algorithm and want to estimate how well it performs for your application, cross validation would give a reasonable way of doing so.

Feature Selection

- One special and important case of model selection is called feature selection.

Feature Selection

- One special and important case of model selection is called feature selection.
- Machine learning algorithms are known to degrade in performance (prediction accuracy) when faced with many inputs (aka features) that are not necessary for predicting the desired output.

Feature Selection

- One special and important case of model selection is called feature selection.
- Machine learning algorithms are known to degrade in performance (prediction accuracy) when faced with many inputs (aka features) that are not necessary for predicting the desired output.
- In the feature selection problem, a learning algorithm is faced with the problem of selecting some subset of features upon which to focus its attention, while ignoring the rest.

Feature Selection

- One special and important case of model selection is called feature selection.
- Machine learning algorithms are known to degrade in performance (prediction accuracy) when faced with many inputs (aka features) that are not necessary for predicting the desired output.
- In the feature selection problem, a learning algorithm is faced with the problem of selecting some subset of features upon which to focus its attention, while ignoring the rest.
- In many bioinformatics problems the number of features amounts to several thousands: for example in cancer classification tasks the number of variables (i.e. the number of genes for which expression is measured) may range from 6000 to 60000.

Feature Selection

There are many potential benefits of feature selection:

- Facilitating data visualization and data understanding.

Feature Selection

There are many potential benefits of feature selection:

- Facilitating data visualization and data understanding.
- Reducing the measurement and storage requirements

Feature Selection

There are many potential benefits of feature selection:

- Facilitating data visualization and data understanding.
- Reducing the measurement and storage requirements
- Reducing training and utilization times of the final model.

Feature Selection

There are many potential benefits of feature selection:

- Facilitating data visualization and data understanding.
- Reducing the measurement and storage requirements
- Reducing training and utilization times of the final model.
- Defying the curse of dimensionality to improve prediction performance.

Feature Selection

There are also some drawbacks:

- The search for a subset of relevant features introduces an additional layer of complexity in the modelling task. The search in the model hypothesis space is augmented by another dimension: the one of finding the optimal subset of relevant features.

Feature Selection

There are also some drawbacks:

- The search for a subset of relevant features introduces an additional layer of complexity in the modelling task. The search in the model hypothesis space is augmented by another dimension: the one of finding the optimal subset of relevant features.
- You will need additional time for learning.

Two main approaches to feature selection

- Wrapper methods: these methods assess subsets of variables according to their usefulness to a given predictor. The method conducts a search for a good subset using the learning algorithm itself as part of the evaluation function. The problem boils down to a problem of stochastic state space search.

Two main approaches to feature selection

- Wrapper methods: these methods assess subsets of variables according to their usefulness to a given predictor. The method conducts a search for a good subset using the learning algorithm itself as part of the evaluation function. The problem boils down to a problem of stochastic state space search.
- Filter methods: they are preprocessing methods. They attempt to assess the merits of features from the data, ignoring the effects of the selected feature subset on the performance of the learning algorithm. Examples are methods that select variables by ranking them through compression techniques, like PCA or clustering (SOM's and K-means), or by computing correlation with the output.

Wrapper methods

- Pros: interaction between feature subset search and model selection, and the ability to take into account feature dependencies.

Wrapper methods

- Pros: interaction between feature subset search and model selection, and the ability to take into account feature dependencies.
- Cons: higher risk of overfitting than filter techniques and are very computationally intensive, especially if building the classifier has a high computational cost.

Wrapping search

- The wrapper search can be seen as a search in a space $W = \{0, 1\}^n$ where a generic vector $w \in W$ is such that $w[j] = 1$ if the input j does not belong to the feature set, and $w[j] = 0$ otherwise.

Wrapping search

- The wrapper search can be seen as a search in a space $W = \{0,1\}^n$ where a generic vector $w \in W$ is such that $w[j] = 1$ if the input j does not belong to the feature set, and $w[j] = 0$ otherwise.
- We look for the optimal vector $w^* \in \{0,1\}^n$ such that $w^* = \arg\min_{w \in W} MSE_w$ where MSE_w is the generalization error of the model based on the set of variables described by w.

Wrapping search

- The wrapper search can be seen as a search in a space $W = \{0,1\}^n$ where a generic vector $w \in W$ is such that $w[j] = 1$ if the input j does not belong to the feature set, and $w[j] = 0$ otherwise.
- We look for the optimal vector $w^* \in \{0,1\}^n$ such that $w^* = \arg\min_{w \in W} MSE_w$ where MSE_w is the generalization error of the model based on the set of variables described by w.
- The number of vectors in W is equal to 2^n.

Wrapping search

- The wrapper search can be seen as a search in a space $W = \{0, 1\}^n$ where a generic vector $w \in W$ is such that $w[j] = 1$ if the input j does not belong to the feature set, and $w[j] = 0$ otherwise.
- We look for the optimal vector $w^* \in \{0, 1\}^n$ such that $w^* = \arg\min_{w \in W} MSE_w$ where MSE_w is the generalization error of the model based on the set of variables described by w.
- The number of vectors in W is equal to 2^n.
- For moderately large n, the exhaustive search is no more possible.

Wrapping search strategies

Greedy methods have been developed for evaluating only a $O(n^2)$ number of variables by either adding or deleting one variable at a time. Three are the main categories:

- Forward search: the procedure starts with no variables and progressively incorporate features. The first input selected is the one which allows the lowest generalization error. The second input selected is the one that, together with the first, has the lowest error, and so on, till when no improvement is made.

Wrapping search strategies

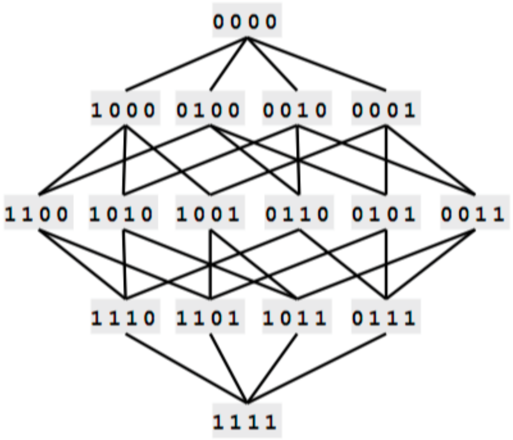
Greedy methods have been developed for evaluating only a $O(n^2)$ number of variables by either adding or deleting one variable at a time. Three are the main categories:

- Forward search: the procedure starts with no variables and progressively incorporate features. The first input selected is the one which allows the lowest generalization error. The second input selected is the one that, together with the first, has the lowest error, and so on, till when no improvement is made.
- Backward search: it works in the opposite direction of the forward approach by progressively removing feature from the full set. We begin with a model that contains all the n variables. The first input to be removed is the one that allows the lowest generalization error.

Forward Search

```
1. Initialize $\mathcal{F} = \emptyset$.
2. Repeat {
     • For $i = 1, \ldots, n$ if $i \notin \mathcal{F}$, let $\mathcal{F}_i = \mathcal{F} \cup \{i\}$, and use some
       version of cross validation to evaluate features $\mathcal{F}_i$.
     • Set $\mathcal{F}$ to be the best feature subset found on the previous step.
   }
3. Select and output the best feature subset that was evaluated
   during the entire search procedure.
```

Forward Search



Backward Search

- Backward search starts off with $\mathcal{F} = \{1, \dots, n\}$ as the set of all features, and repeatedly deletes features one at a time until $\mathcal{F} = \emptyset$

Backward Search

- Backward search starts off with $\mathcal{F} = \{1, \ldots, n\}$ as the set of all features, and repeatedly deletes features one at a time until $\mathcal{F} = \emptyset$
- Forward search and backward search are called wrapper feature selection algorithms. Both algorithms work quite well, however, they can be computationally expensive, given that they need to make many calls to the learning algorithm.

Filter methods

- Pros: easily scale to very high-dimensional datasets, computationally simple and fast, and independent of the classification algorithm. Feature selection needs to be performed only once, and then different classifiers can be evaluated.

Filter methods

- Pros: easily scale to very high-dimensional datasets, computationally simple and fast, and independent of the classification algorithm. Feature selection needs to be performed only once, and then different classifiers can be evaluated.
- Cons: they ignore the interaction with the classifier. They are often univariate or low-variate. This means that each feature is considered separately, thereby ignoring feature dependencies, which may lead to worse classification performance when compared to other types of feature selection techniques.

Filter Feature Strategy

- Filter feature selection algorithms give heuristic, but computationally less expensive ways of choosing a feature subset.

Filter Feature Strategy

- Filter feature selection algorithms give heuristic, but computationally less expensive ways of choosing a feature subset.
- The idea is to compute some simple score $S(i)$ that measures how informative each feature x_i is about the class labels y. Then we simply pick the k features with the largest scores $S(i)$.

Filter Feature Strategy

- Filter feature selection algorithms give heuristic, but computationally less expensive ways of choosing a feature subset.
- The idea is to compute some simple score $S(i)$ that measures how informative each feature x_i is about the class labels y. Then we simply pick the k features with the largest scores $S(i)$.
- One possible choice of the score would be define $S(i)$ to be the correlation between x_i and y, as measured on the training data. So, we will choose the features that are the most strongly correlated with the class labels.

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.
- Measures of relevance which are commonly used are:

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.
- Measures of relevance which are commonly used are:
  - Pearson correlation (the greater the more relevant) which assumes linear dependency;

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.
- Measures of relevance which are commonly used are:
  - Pearson correlation (the greater the more relevant) which assumes linear dependency;
  - Significance p-value of an hypothesis test (the lower the more relevant) which aims at detect the genes that split well the dataset. Parametric (t-test) and nonparametric (Wilcoxon) tests have been proposed in literature;

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.
- Measures of relevance which are commonly used are:
    - Pearson correlation (the greater the more relevant) which assumes linear dependency;
    - Significance p-value of an hypothesis test (the lower the more relevant) which aims at detect the genes that split well the dataset. Parametric (t-test) and nonparametric (Wilcoxon) tests have been proposed in literature;
    - Mutual information (the greater the more relevant).

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.
- Measures of relevance which are commonly used are:
  - Pearson correlation (the greater the more relevant) which assumes linear dependency;
  - Significance p-value of an hypothesis test (the lower the more relevant) which aims at detect the genes that split well the dataset. Parametric (t-test) and nonparametric (Wilcoxon) tests have been proposed in literature;
  - Mutual information (the greater the more relevant).
- After the univariate assessment the method ranks the variable in a decreasing order of relevance.

Ranking methods

- They assess the importance (or relevance) of each variable with respect to the output by using a univariate measure.
- Measures of relevance which are commonly used are:
  - Pearson correlation (the greater the more relevant) which assumes linear dependency;
  - Significance p-value of an hypothesis test (the lower the more relevant) which aims at detect the genes that split well the dataset. Parametric (t-test) and nonparametric (Wilcoxon) tests have been proposed in literature;
  - Mutual information (the greater the more relevant).
- After the univariate assessment the method ranks the variable in a decreasing order of relevance.
- These methods are fast (complexity $O(n)$) and their output is intuitive and easy to understand. At the same time they disregard redundancies and higher order interactions between variables.

Mutual Information

- In practice it is more common to choose $S(i)$ to be the **mutual information** $\mathrm{MI}(x_i, y)$, between x_i and y:

$$\mathrm{MI}(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}.$$

Mutual Information

- In practice it is more common to choose $S(i)$ to be the **mutual information** $\mathrm{MI}(x_i, y)$, between x_i and y:

$$\mathrm{MI}(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}.$$

- This equation assumes that x_i and y are binary-valued; more generally the summations will be over the domains of the variables.

Mutual Information

- In practice it is more common to choose $S(i)$ to be the **mutual information** $\mathrm{MI}(x_i, y)$, between x_i and y:

$$\mathrm{MI}(x_i, y) = \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}.$$

- This equation assumes that x_i and y are binary-valued; more generally the summations will be over the domains of the variables.
- Moreover, the probabilities $p(x_i, y)$, $p(x_i)$ and $p(y)$ can all be estimated according to their empirical distributions on the training set.

PCA

A few words on principal component analysis (PCA):

- PCA is one of the most popular methods for linear dimensionality reduction. It can project the data from the original space into a a lower dimensional space in an unsupervised manner.

PCA

A few words on principal component analysis (PCA):

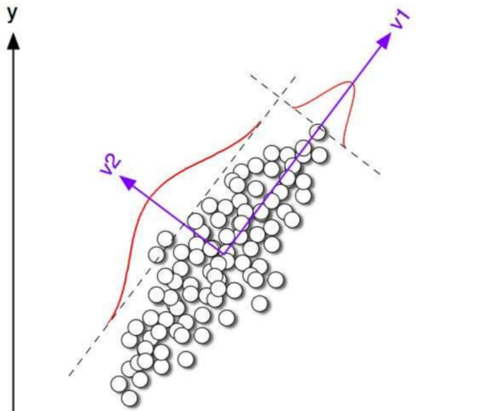
- PCA is one of the most popular methods for linear dimensionality reduction. It can project the data from the original space into a a lower dimensional space in an unsupervised manner.
- Each of the original dimensions is an axis. However, other axes can be created as linear combinations of the original ones.

PCA

A few words on principal component analysis (PCA):

- PCA is one of the most popular methods for linear dimensionality reduction. It can project the data from the original space into a a lower dimensional space in an unsupervised manner.
- Each of the original dimensions is an axis. However, other axes can be created as linear combinations of the original ones.
- PCA creates a completely new set of axes (principal components) that like the original ones are orthogonal to each other.

PCA



Evolutionary Algorithms

Simple Genetic Algorithm:

- Generate initial population (of subsets of features)
- Repeat for n generations:
  1. Selection
  2. Crossover
  3. Mutation

This kind of heuristic algorithm could be used for example when we have thousands of features, as it is common in bioinformatics, so complexities $\mathcal{O}(2^n)$ or $\mathcal{O}(n^2)$ are not viable.

Simple Genetic Algorithm

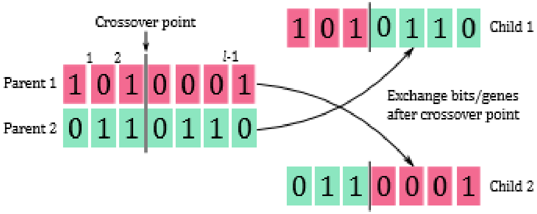


Image taken from

https://www.dataminingapps.com/2017/03/solving-the-knapsack-problem-with-a-simple-genetic-algorithm/

Assessing the significance of findings

- Most of the previous techniques aim to return insight about the data by performing a large number of comparisons and selections. Despite the use of validation procedures, it is highly possible that high correlations or low prediction errors are found only due to chance.

Assessing the significance of findings

- Most of the previous techniques aim to return insight about the data by performing a large number of comparisons and selections. Despite the use of validation procedures, it is highly possible that high correlations or low prediction errors are found only due to chance.
- A bad practice consists in using the same set of observations to select the feature set and to assess the generalization accuracy of the classifier. The use of external validation sets is recommended.

Assessing the significance of findings

- If no additional data are available, an alternative consists in estimating how often random data would generate correlation or classification accuracy of the magnitude obtained in the original analysis. Typically this is done through use of permutation testing.

Assessing the significance of findings

- If no additional data are available, an alternative consists in estimating how often random data would generate correlation or classification accuracy of the magnitude obtained in the original analysis. Typically this is done through use of permutation testing.
- This consists in repeating the same procedure several times with data where the dependency between the input and the output is artificially removed (for example by permuting the inputs ordering). This would provide us with a distribution of the accuracy in case of random data where no information is present in the data.

Combining instead of selecting

- Instead of choosing one particular FS method, and accepting its outcome as the final subset, different FS methods can be combined using ensemble FS approaches.

Combining instead of selecting

- Instead of choosing one particular FS method, and accepting its outcome as the final subset, different FS methods can be combined using ensemble FS approaches.
- Since there is not an optimal feature selection technique and due to the possible existence of more than one subset of features that discriminates the data equally well, model combination approaches have been adapted to improve the robustness and stability of final, discriminative methods.

Reference

- Gianluca Bontempi. **Machine learning methods for bioinformatics**.
- Andrew Ng. **Machine Learning Course Notes**. 2003.

Thank you!

Dr. Victor Uc Cetina
cetina@informatik.uni-hamburg.de