Machine Learning, Module 7

Mattias Ohlsson

December 5, 2013
Outline

**Monday**
- Artificial neural networks
  - MLP
- Model selection and generalization performance
  - K-fold cross validation
  - Bootstrap
- Self-Organizing Feature Map (SOFM)

**Wednesday**
- Ensemble methods
  - Random forest classification
  - Support vector machines
A few slides from last time

Artificial neural networks
   MLP

Model selection and generalization performance
   K-fold cross validation
   Bootstrap

Self-Organizing Feature Map (SOFM)
The overall function of the machine learning tool

Machine learning tool

"Grey" box

Healthy

0
No

0.12

1.2
45.4
-34.2
1.2
7.7
-30.7
21.4
The Multi Layer Perceptron

\[ y_i(n) = \varphi_o \left( \sum_i \omega_{ij} \varphi_h \left( \sum_k \tilde{\omega}_{jk} x_k(n) \right) \right) = \varphi_o \left( \tilde{\omega}_i^T h(n) \right) \]
Classification problems (two classes)

\[ E(\omega) = - \sum_{n=1}^{N} \left( d(n) \log y(n) + (1 - d(n)) \log(1 - y(n)) \right) \]

\[ d = 1 \text{ for class for the first class and } d = 0 \text{ for the second class.} \]
If minimization works, complex enough network and large dataset!

\[ y_i(x, \omega^*) = \langle d_i | x \rangle \]

For classification problems, one gets

\[ y_i(x, \omega^*) = P(C_i | x) \]

Outputs are probabilities!
How to detect overfitting?

Data

- Training data
- Validation data

Error

- Validation error
- Training error

Number of iterations
Use data better

\[ \hat{P} \text{ is the estimate of the true generalization performance} \]

\[ \hat{P} = \frac{1}{K} \sum_{i=1}^{K} P_i \]

\[ P_i \text{ is the performance on validation data set } i \]

\[ \left( \hat{P} = \frac{1}{NK} \sum_{n=1}^{N} \sum_{i=1}^{K} P_{in} \right) \]

**K-fold crossvalidation**
Estimate generalization performance with a model that needs model selection.
SOFM – some features

- Artificial neural network trained using unsupervised learning (Only inputs, no targets)
- Input space is represented using a discretized low-dimensional (often 2-dim) space.
- A neighborhood function is used to preserve the topological properties of the input space (see also multidimensional scaling (MDS)).
A number of simple perceptrons

\[ y_i = e^{-||x-\omega_i||^2} \]

Define the **winning** node, as the one with the largest output, (i.e smallest distance to the input data point).
Update the weights

$$\omega_i \rightarrow \omega_i + \eta \Lambda_{ii^*} (x - \omega_i)$$

where $\Lambda_{ij}$ is the neighborhood function. $i^*$ is the winning node!
Neighborhood functions

\[
\Lambda_{ij} = \begin{cases} 
\Lambda_o & \text{if } |i - j| \leq \Delta \\
0 & \text{if } |i - j| > \Delta 
\end{cases}
\]

\[
\Lambda_{ij} = \Lambda_o e^{-|i-j|^2/2\sigma^2}
\]

Note!

|i − j| is distance in the discrete space!
Some examples (2-D to discrete 2-D)
SOFM 10x15 nodes, 500 datapoints. Result after 20 epochs
Ensemble of models

Basic idea

We combine many networks to form an *ensemble of networks* (or sometimes called committee machines)

**Static structures:**
- Ensemble averaging
- Boosting

**Dynamic structures:**
- Mixture of exports
- Hierarchical mixture of experts
Theoretical arguments

$E_D [\cdot]$ is the average of many data sets $D$

$E_{DI} [\cdot]$ is the average of many data sets $D$ and many initial conditions $I$.

$\bar{y}_D(x)$ is the average output of many models trained with different initial conditions $I$

(e.g. $= \frac{1}{L} \sum_I^L y_{DI}(x)$)

One can show

$$E_D \left[ (\bar{y}_D(x) - < d|x >)^2 \right] \leq E_{DI} \left[ (y_{DI}(x) - < d|x >)^2 \right]$$
Two questions:

1. How to average the different models?

\[ y_{\text{ens}}(\mathbf{x}) = \sum_{i=1}^{L} \alpha_i y_i(\mathbf{x}) \]

often

\[ \alpha_i = 1/L \]
2. How construct the different models? We can gain some insight into this by the following “decomposition”.

\[ y_{\text{ens}}(\mathbf{x}) = \sum_{i}^{L} \alpha_{i} y_{i}(\mathbf{x}) \text{ and } \sum_{i}^{L} \alpha_{i} = 1 \]

Then one can show

\[
\mathbb{E} \left[ (y_{\text{ens}}(\mathbf{x}) - h(\mathbf{x}))^2 \right] = \\
\sum_{i} \alpha_{i} \mathbb{E} \left[ (y_{i}(\mathbf{x}) - h(\mathbf{x}))^2 \right] - \sum_{i} \alpha_{i} \mathbb{E} \left[ (y_{i}(\mathbf{x}) - y_{\text{ens}}(\mathbf{x}))^2 \right]
\]

NOTE!
In summary we want good models that are different from each other!

**Bagging:** Bagging creates diverse models by training on `slightly` different training data sets. Each model in the bagging ensemble is trained on a bootstrap sample of the original training data set.

Bagging is a very common ensemble creation technique!
Cross-validation ensemble

Original training data

Split into K parts

Training set for member 1

Training set for member 2

Training set for member 3

...  

Training set for member K
Random Forest Classifiers

Some features

- Ensemble classifier using many decision tree models
- Works for both regression and classification
- Generalization performance estimates and variable importance measurements are provided with the results
- Resistant to overfitting – generalizes well to new data
- Only a few parameters in the model
What about decision tree models

- A model that uses a set of binary rules to calculate a target value
- Both for classification and regression
- The “best” split at a node can be determined using different strategies

Example

```
        A
       / \  
      /   \  
     /     \  
    /       \  
   /         \  
  /           \  
 /             \  
/               \  
/                 \  
B                   B
 /\                 /\  
/  \               /  \  
B < 4.5 B ≥ 4.5 B ≥ 8.1 B < 8.1
 /\             /\  
/  \           /  \  
K=y K=x K=x K=y
```

CBBP, Theoretical Physics
More on trees

- Training data is used to build the tree (of course!).
- The tree builder have to determine what variable(s) to use at a given node and what value to split on.
- Decision to stop (create a terminal node or a new decision node).
- A tree can be fully developed, that is all training data can be correctly classified (overtrained).
A small classification problem

Pima Indians

- Classification (having diabetes or not)
- 384 training data and 384 test data
- 8 features
Test performance

72.6 % correct

A single decision tree (build using matlab)
Test performance

75.8 % correct

A single decision tree (pruned)
Test performance

70.6 % correct

A single decision tree (full tree)
Random forest

- Train a ensemble of decision trees. All trees are fully developed i.e. training data is correctly classified for all trees in the ensemble. About ~100 – 1000 trees are used in the ensemble.

- A bootstrap sample of the original training data set is used when training individual trees. About 2/3 of the will be used for training each time.

- Remaining training data (OOB) are used to estimate error and variable importance.

- Majority vote is used when predicting new data.

- A randomly selected subset of variables is used to split each node. Typically sqrt(NVar) is used!
Test performance
77.1 % correct

OOB performance
75.3 % correct

Number of variables for each node split: 3
Variable importance can be estimated on the OOB data.
Support Vector Machine Classifiers

What are SVM's?

- Learning algorithms for classification of objects into two classes (works also for regression). Multiple classes can be handled.

Some features

- Good performance in real-world applications
- Computational efficiency
- Works well in high dimension (Bioinformatics)
- Sound theoretical foundations
- Few free parameters to tune
Linear separable data

How to find the best separating line?
Maximize the margin!

This can be formulated as a classical *quadratic programming problem.*

(Can be solved using a Lagrangian multiplier method)

*A few support vectors are obtained!*

CBBP, Theoretical Physics
For non-linear separable problems, introduce a *soft* margin!

A trade-off between:

- Minimizing training error
- Maximizing the margin
If we want a non-linear classifier?

The kernel trick!

\[ \mathbf{x} \cdot \mathbf{x}' \rightarrow \mathcal{K}(\mathbf{x}, \mathbf{x}') \]
Common kernels for SVM

Polynomial: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^d$

Gaussian: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2}$

Hyperbolic tangent: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \tanh(\kappa \mathbf{x} \cdot \mathbf{x}' + c)$
Test performance
75.0 % correct

Test performance for an ensemble of ANN's
77.0 % correct