

Machine learning exam 2017

Henrik Lund Mortensen

January 2, 2018

1 Linear Models

2 Learning Theory

- Supervised learning
- In-sample error vs out-of-sample error

Learning feasibility

In general, we have an unknown function $f(x)$ and only a finite set of samples $\mathcal{D} = \{x_i, f(x_i)\}$. Our job is to use \mathcal{D} to estimate f outside of \mathcal{D} . Consider a set of hypotheses, \mathcal{H} . The goal is to choose the hypothesis, $g \in \mathcal{H}$ that approximates $f(x)$ best. There are two things we must consider:

1. Can we find a g such that the training samples are well explained, i.e. low $E_{\text{in}}(g)$?
2. Can we make sure that the $E_{\text{out}}(g)$ is close to $E_{\text{in}}(g)$?

The answer to the first question depends on the complexity of f and the \mathcal{H} . It is clear that we can always find a set \mathcal{H} which contains a hypothesis that matches the training data perfectly, i.e. $E_{\text{in}} = 0$. However, we get in trouble in the part 2 above. We cannot guarantee that $E_{\text{out}}(g)$ is close to $E_{\text{in}}(g)$, but we can estimate the probability that is it. This is given by the Hoeffding bound

$$\mathcal{P}[|E_{\text{out}}(g) - E_{\text{in}}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}, \quad (1)$$

where N is the number of samples in \mathcal{D} and M is the size of \mathcal{H} (number of hypotheses). There is a trade-off between choosing \mathcal{H} complex enough to describe f well, while low enough to have a reasonable bound in Eq. (1). The Hoeffding bound only applies to finite \mathcal{H} , as it becomes meaningless when $M \rightarrow \infty$.

When the hypothesis space becomes infinite we must replace M by something else. We call the replacement the *growth function*. We restrict ourselves to

binary target functions, such that the target function (and our hypotheses, $h \in \mathcal{H}$) map from \mathcal{X} to $\{+1, -1\}$. Further, we define a dichotomy as an N -tuple of $+1$'s and -1 's such that the dichotomies generated by \mathcal{H} on some set $x_1, x_2, \dots, x_N \in \mathcal{X}$ is given by

$$\{h(x_1), h(x_2), \dots, h(x_N) | \forall h \in \mathcal{H}\}. \quad (2)$$

It is clear that there can be a maximum of 2^N different dichotomies for N points. However, it is not true that any \mathcal{H} can generate all dichotomies. We say that if \mathcal{H} generates all possible dichotomies for some set, it *shatters* this set.

3 Support Vector Machines

In a binary classifier labels inputs either $+1$ or -1 (for example). However, points close to the decision boundary are more uncertain than those far from it.

4 Neural Nets

5 Decision Trees and Ensemble Methods

6 Hidden Markov Models - Decoding

7 Hidden Markov Models - Training

8 Unsupervised Learning - Clustering

Unsupervised learning is used when data is available, but comes without labels, i.e. there are no training examples. An example of unsupervised learning is clustering. Here data are labeled into groups (or clusters).

K-means

The K-means problem is a clustering problem, where the goal is to find k means, μ_i , such that the Euclidean distance to each cluster member is minimized,

$$SSE(\mathcal{C}) = \sum_{i=1}^k \sum_{\mathbf{x}_j \in \mathcal{C}_i} \|\mathbf{x}_j - \mu_i\|^2. \quad (3)$$

In general, it is hard to find the optimal μ_i . However, an iterative method known as Lloyd's algorithm is often used as an approximation. Here, we start by choosing k random μ_i and determine for all data points which μ_i is closest. The new μ_i are now chosen as the mean of all $\mathbf{x}_j \in \mathcal{C}_i$. This is repeated until no data points change cluster.

Lloyd's algorithm is fast, but can only give convex clusters (draw non-convex example). Also, sometimes a mean is not well-defined.

***K*-medoids**

Similar to *K*-means, but here a representative datapoint is chosen as cluster center. Also, the distance is the L_1 -norm (Manhattan distance) instead of the Euclidian distance in *K*-means (why?).

Expectation Maximization Clustering

Unlike *K*-means and *K*-medoids, EM clustering is a *soft clustering* method. A probability for belonging to a certain cluster is assigned to each data point, instead of a hard label.

In EM clustering we find n clusters represented by multivariate normal distributions.

$$f_i(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^d |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp \left[-\frac{(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)}{2} \right]. \quad (4)$$

The assumption is the \mathbf{x} is generated by a Gaussian mixture model probability distribution defined by

$$f(\mathbf{x}) = \sum_i f_i(\mathbf{x}) P(\mathcal{C}_i), \quad (5)$$

where the mixture parameters satisfies $\sum_i P(\mathcal{C}_i) = 1$. The model parameters thus include $\theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, P(\mathcal{C}_i)\}$. This goal is to find the set of parameters, such that the probability of observing the dataset, \mathbf{D} , is maximized,

$$\theta^* = \arg \max_{\theta} P(\mathbf{D}|\theta) = \arg \max_{\theta} \ln P(\mathbf{D}|\theta). \quad (6)$$

Finding θ^* is generally hard. Again, we can use an iterative algorithm to approximate θ^* in a two-step process. The first step (*estimation step*) is to calculate $w_{ij} = P(\mathcal{C}_i|\mathbf{x}_j)$, which is ...

The second step (*maximization step*) is to reestimate the model parameters using the weights, w_{ij} . The means are estimated as

$$\boldsymbol{\mu}_i = \frac{\sum_j w_{ij} \mathbf{x}_j}{\sum_j w_{ij}}. \quad (7)$$

The cluster variance matrix is assumed diagonal for simplicity (otherwise the number of parameters is huge). The covariance matrix is esimated as the weighted covariance for each cluster,

$$\boldsymbol{\Sigma}_i = \frac{\sum_j w_{ij} \mathbf{z}_{ji} \cdot \mathbf{z}_{ji}^T}{\mathbf{w}_i^T \mathbf{1}}, \quad (8)$$

where $\mathbf{z}_{ji} = \mathbf{x}_j - \boldsymbol{\mu}_i$. Finally, the mixture parameters are estimated as

$$P(\mathcal{C}_i) = \frac{\mathbf{w}_i^T \mathbf{1}}{n}. \quad (9)$$

These update rules follows from differentiation of the log likelihood function, $\ln P(\mathbf{D}|\theta)$, with respect to the parameters, and setting it to zero.

Performance measures

9 Unsupervised Learning - Outlier Detection and Dimensionality Reduction