Machine learning exam 2017

Henrik Lund Mortensen

January 4, 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 finte set of samples $\mathcal{D} = \{x_i, f(x_i)\}$. Our job is to use $\mathcal{D}$ to estimate f outisde 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 thing 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

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

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 finte $\mathcal{H}$, as it becomes meaningless when $M \to \infty$.

When the hypothesis space becomes infinite we must replace M by something else. We call the replacement the *growth function*. We restrict outselves 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 $x_1, x_2, \ldots, x_N \in \mathcal{X}$ is given by

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

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 that 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 s clustering problem, where the goal is to find k means, μ_i, such that the Euclidian distance to each cluster member is minimized,

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

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

$$\mathrm{TD} = \sum_i \sum_{p \in \mathcal{C}_i} \mathrm{dist}(p, x_i). \tag{4}$$

Then, for each cluster center, x_i, and data point that is not a cluster center, q, we swap such that q becomes a cluster center and calculates TD again. The swap with the lowest cost is now chosen. This repeats untill there is no change in TD.

k-medoids is sometimes applied instead of k-means, e.g. if a mean is ill-defined in the specific problem. It is also not as sensitive to outliers. The implementation cost is $\mathcal{O}(N^2k^2)$ (I think), which makes it much more expensive than k-means.

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.

$$f_i(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^d |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_i)}{2}\right]. \tag{5}$$

The assumption is the x is generated by a Gaussian mixture model probability distribution defined by

$$f(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x}) P(\mathcal{C}_i), \tag{6}$$

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, $\boldsymbol{D}$, is maximized,

$$\theta^* = \arg\max_\theta \ P(\boldsymbol{D}|\theta) = \arg\max_\theta \ \ln P(\boldsymbol{D}|\theta). \tag{7}$$

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 $w_{ij} = P(\mathcal{C}_i|\boldsymbol{x}_j)$. This is done using Bayes rule

$$P(\mathcal{C}_i|x_j) = \frac{P(x_j)P(x_j|\mathcal{C}_i)}{P(\mathcal{C}_i)}. \tag{8}$$

Here, $P(\mathcal{C}_i)$ is initialized as $\frac{1}{n}$. $P(x_j|\mathcal{C}_i)$ is easily calculated by Eq. (5) and $P(x_j) = \sum_i P(x_j|\mathcal{C}_i)P(\mathcal{C}_i)$.

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

$$\boldsymbol{\mu}_i = \frac{\sum_j w_{ij} \boldsymbol{x}_j}{\sum_j w_{ij}}. \tag{9}$$

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,

$$\boldsymbol{\Sigma}_i = \frac{\sum_j w_{ij} \boldsymbol{z}_{ji} \cdot \boldsymbol{z}_{ji}^T}{\boldsymbol{w}_i^T \mathbf{1}}, \tag{10}$$

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

$$P(\mathcal{C}_i) = \frac{\boldsymbol{w}_i^T \mathbf{1}}{n}. \tag{11}$$

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

DBSCAN

DBSCAN is a density based clustering method. Instead of considering distances between all data points, only the neighbourhoods of the data points are considered. The idea is to cluster data points based on the local density.

Consider a data point q in the dataset $\mathcal{D}$. Any point, p, which is closer to q than some threshold, ϵ, is said to be in the neighbourhood of q,

$$\mathcal{N}_\epsilon(q) = \{\, p \;\mid \operatorname{dist}(p, q) \leq \epsilon \,\}. \tag{12}$$

Some definitions;

- q is a *core point* of the cluster if $|\mathcal{N}_\epsilon(q)| \geq \mathit{MinPts}$.
- p is *directly density reachable* from q if $p \in \mathcal{N}_\epsilon(q)$.
- p is *density reachable* from s if there exists a chain of core points such that the chain elements are directly reachable from its chain neighbour.
- Two points, p and s, are *density connected* if they are both density connected to some core point, q.

The algorithm first computes $N_\epsilon(p)$ for all points $p \in \mathcal{D}$. Based on this, all core points are identified. Next, the density connected clusters are located by going through $N_\epsilon(q)$ for all core points.

The complexity depends on how efficiently the neighbourhood can be determined. Worst case is $\mathcal{O}(N^2)$, but if the dimension is not to high it can be done in $\mathcal{O}(N \log N)$.

An advantage is that no number of clusters is to be specified before hand. Also, non-convex cluster shapes can be found. A disadvantage is that ϵ is har to choose, especially if some clusters are on different scales.

DENCLUE

The DENCLUE algorithm is also a density based clustering algorithm.

Performance measures

. . .

9 Unsupervised Learning - Outlier Detection and Dimensionality Reduction

PCA

In Principal Component Analisys (PCA), we seek to represent data in a high dimentional space in a low dimensional containing the most important features. Data may be correlated in a way that makes the high dimensional representation essentially redundant. Consider images of faces represented by 100 pixels $\times$ 100 pixels = 10000 dimensional space. With PCA, this can be reduced to $\sim$ 20 dimensions representing the most essential face features.

The first step in PCA is to normalize the data, such that it has zero mean and unit variance. This is done by

$$x^{(i)} \to x^{(i)} - \mu \,, \quad \mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \tag{13}$$

and then

$$x_j^{(i)} \to \frac{x_j^{(i)}}{\sigma_j} \,, \quad \sigma_j = \frac{1}{m}\sum_{i=1}^{m} \left(x_j^{(i)}\right)^2 . \tag{14}$$

The goal is now to find a vector $\boldsymbol{u}$ such that when the data is projected onto $\boldsymbol{u}$ it has the highest variance. That is, we seek to maximize

$$\frac{1}{m}\sum_{i=1}^{m} \left(\boldsymbol{x}^{(i)^T}\boldsymbol{u}\right)^2 = \boldsymbol{u}^T \left(\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)^T}\right)\boldsymbol{u}. \tag{15}$$

Maximizing this quantity with the constraint that $\boldsymbol{u}$ is a unit vector reveals that $\boldsymbol{u}$ must be an eigenvector of the estimated covariance matrix

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}^{(i)}\boldsymbol{x}^{(i)^T}. \tag{16}$$

To reduce the high dimentional space to a k-dimensional space we simply project onto the first k eigenvectors of Σ,

$$y^{(i)} = \begin{bmatrix} \boldsymbol{u}_1^T \boldsymbol{x}^{(i)} \\ \boldsymbol{u}_2^T \boldsymbol{x}^{(i)} \\ \vdots \\ \boldsymbol{u}_k^T \boldsymbol{x}^{(i)} \end{bmatrix} . \tag{17}$$