Machine Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Semester 2016, Homework 2 (90 points + 15 bonus)

Prof. Dr. J. Peters, F. Veiga, S. Parisi

Due date: Wednesday, 08 June 2016 (before the lecture)

Problem 2.1 Bayesian Decision Theory [20 Points]

In this exercise, we consider data produced by a mixture of two Gaussian distributions with parameters $\{\mu_1, \sigma_1\}$ and $\{\mu_2, \sigma_2\}$. Each Gaussian represents a class labeled C_1 and C_2, respectively.

a) **Optimal Boundary [4 Points]**
Explain in one short sentence what Bayesian Decision Theory is. What is its goal? What condition does hold at the optimal decision boundary? When do we decide for class C_1 over C_2?

Bayesian Decision Theory models needs an explicit distribution of features. It's goal is to minimize the error probability. If x is a feature vector, than the corresponding sample is classifies as member of C_1 if $p(C_1|x) > p(C_2|x)$ and vice versa. Therefore, the optimal decision boundary $x_{optimal}$ fulfills the equation $p(C_1|x_{optimal}) = p(C_2|c_{optimal})$

b) **Decision Boundaries [8 Points]**
If both classes have equal prior probabilities $p(C_1) = p(C_2)$ and the same variance $\sigma_1 = \sigma_2$, derive the decision boundary x^* analytically as a function of the two means μ_1 and μ_2.

Gaussian distribution: $\frac{1}{\sqrt{2\pi\sigma^2 2}} exp\left(-\frac{x-\mu}{2\sigma^2}\right)$

Since prior probabilities are equal, the optimal decision boundary is at the point where the distributions are equal.

$$\begin{aligned}
\frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right) &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu_2)^2}{2\sigma^2}\right) \\
\frac{(x-\mu_1)^2}{2\sigma^2} &= \frac{(x-\mu_2)^2}{2\sigma^2} \\
-2x\mu_1 + \mu_1^2 &= -2x\mu_2 + \mu_2^2 \\
x &= \frac{\mu_2^2 - \mu_1^2}{2(\mu_2 - \mu_1)} \\
x &= \frac{\mu_2 + \mu_1}{2}
\end{aligned}$$

The symmetry of the Gaussian distribution could be used as well to derive this value.

c) **Different Miss-Classification Costs [8 Points]**
Assume $\mu_1 > 0$, $\mu_1 = 2\mu_2$, and $\sigma_1 = \sigma_2$. If misclassifying sample $x \in C_2$ as class C_1 is four times more expensive than the opposite, how does the decision boundary change? Derive the boundary analytically. (There is no cost for correctly classifying samples.)

TODO

Machine Learning - Homework 2

Name, Vorname: _____ Matrikelnummer: ⌊_⌊_⌊_⌊_⌊_⌊_⌊_⌊_

Problem 2.2 Density Estimation [30 Points + 15 Bonus]

In this exercise, you will use the datasets `densEstClass1.txt` and `densEstClass2.txt`. The datasets contain 2D data belonging to two classes, C_1 and C_2.

a) **Gaussian Maximization Likelihood Estimate [10 Points]**
Derive the ML estimate for the mean and covariance of the **multivariate** Gaussian distribution. Start your derivations with the function you optimize. Assume that you can collect i.i.d data. (Hint: you can find many matrix identities on the Matrix Cookbook and at http://en.wikipedia.org/wiki/Matrix_calculus.)

$$
\begin{aligned}
L(X|\theta) &= max \log p(X|\theta) \\
&= \log \prod_{i=1}^{N} p(x_i|\theta) \\
&= \sum_{i=1}^{N} \log p(x_i|\theta) \\
&= \sum_{i=1}^{N} \log \left((2\pi)^{-\frac{K}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu) \right) \right) \\
&= \sum_{i=1}^{N} \left(\log\left((2\pi)^{-\frac{K}{2}} \Sigma^{-\frac{1}{2}} \right) - \frac{1}{2} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu) \right) \\
&= -\frac{NK}{2} \log(2\pi) - \frac{N}{2} \log|\Sigma| - \frac{1}{2} \sum_{i=1}^{N} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu)
\end{aligned}
$$

1. In order to compute the ML estimate for the mean we take the derivative for μ:

$$
\begin{aligned}
\frac{\delta L}{\delta \mu} &= \frac{\delta \left(-\frac{1}{2} \sum_{i=1}^{N} (x_i-\mu)^T \Sigma^{-1} (x_i-\mu) \right)}{\delta \mu} \\
&= -\frac{1}{2} \sum_{i=1}^{N} \left(-(x_i-\mu)^T \Sigma^{-1} - (x_i-\mu)^T \Sigma^{-1} \right) \\
&= \sum_{i=1}^{N} (x_i-\mu)^T \Sigma^{-1}
\end{aligned}
$$

Setting it to zero gives us the following formular for the mean:

$$
\begin{aligned}
& 0 = \sum_{i=1}^{N} (x_i-\mu)^T \Sigma^{-1} \\
\Leftrightarrow \quad & 0 = \sum_{i=1}^{N} (x_i-\mu) \\
\Leftrightarrow \quad & 0 = \sum_{i=1}^{N} x_i^T - N\mu^T \\
\Leftrightarrow \quad & \mu = \frac{1}{N} \sum_{i=1}^{N} x_i
\end{aligned}
$$

2. In order to compute the ML estimate for the covariance we take the derivative for Σ:

$$\begin{aligned}
\frac{\delta L}{\delta \Sigma} &= \frac{\delta\left(-\frac{N}{2}\log|\Sigma|\right)}{\delta\Sigma} - \frac{1}{2}\frac{\delta\left(\sum_{i=1}^{N}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\right)}{\delta\Sigma} \\
&= -\frac{N|\Sigma|\Sigma^{-1}}{2|\Sigma|} - \frac{1}{2}\frac{\sum_{i=1}^{N}\delta\left(Tr\,(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\right)}{\delta\Sigma} \\
&= -\frac{N}{2}\Sigma^{-1} - \frac{1}{2}\frac{\sum_{i=1}^{N}\delta\left(Tr\,\Sigma^{-1}(x_i-\mu)(x_i-\mu)^T\right)}{\delta\Sigma} \\
&= -\frac{N}{2}\Sigma^{-1} + \frac{1}{2}\sum_{i=1}^{N}\Sigma^{-1}(x_i-\mu)(x_i-\mu)^T\Sigma^{-1} \\
&= -\frac{N}{2} + \frac{1}{2}\sum_{i=1}^{N}\Sigma^{-1}(x_i-\mu)(x_i-\mu)^T
\end{aligned}$$

Then we set the derivative to zero:

$$\begin{aligned}
& 0 = \frac{\delta L}{\delta\Sigma} \\
\Leftrightarrow \quad & 0 = -\frac{N}{2} + \frac{1}{2}\sum_{i=1}^{N}\Sigma^{-1}(x_i-\mu)(x_i-\mu)^T \\
\Leftrightarrow \quad & \frac{N}{2} = \frac{1}{2}\Sigma^{-1}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T \\
\Leftrightarrow \quad & N = \Sigma^{-1}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T \\
\Leftrightarrow \quad & \Sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T
\end{aligned}$$

b) **Prior Probabilities [2 Points]**
Compute the prior probability of each class from the dataset.

Dataset 1 has 239 elements, dataset 2 has 761 elements, resulting in 1000 elements total. Therefore:
$p(C_1) = \frac{239}{1000} = 0.239$, $p(C_2) = \frac{761}{1000} = 0.761$

c) **Biased ML Estimate [5 Points]**
Define the bias of an estimator and write how we can compute it. Then calculate the biased and unbiased estimates of the conditional distribution $p(x|C_i)$, assuming that each class can be modeled with a Gaussian distribution. Which parameters have to be calculated? Show the final result and attach a snippet of your code. Do not use existing functions, but rather implement the computations by yourself!

d) **Class Density [5 Points]**
Using the unbiased estimates from the previous question, fit a Gaussian distribution to the data of each class. Generate a single plot showing the data points and the probability densities of each class. (Hint: use the contour function for plotting the Gaussians.)

e) **Posterior [8 Points]**
I n a single graph, plot the posterior distribution of each class $p(C_i|x)$ and show the decision boundary. For convenience, show the probabilities in the log domain.

f) **Bayesian Estimation [15 Bonus Points]**
State the generic case of a Bayesian treatment of the parameters θ used to model the data, .i.e., $p(\mathbf{X}|\theta)$. Which are the advantages of being Bayesian?
Derive the estimate of the mean, assuming that the model is a Gaussian distribution with a fixed variance. (Hint: work approximately, complete the square.)

Machine Learning - Homework 2

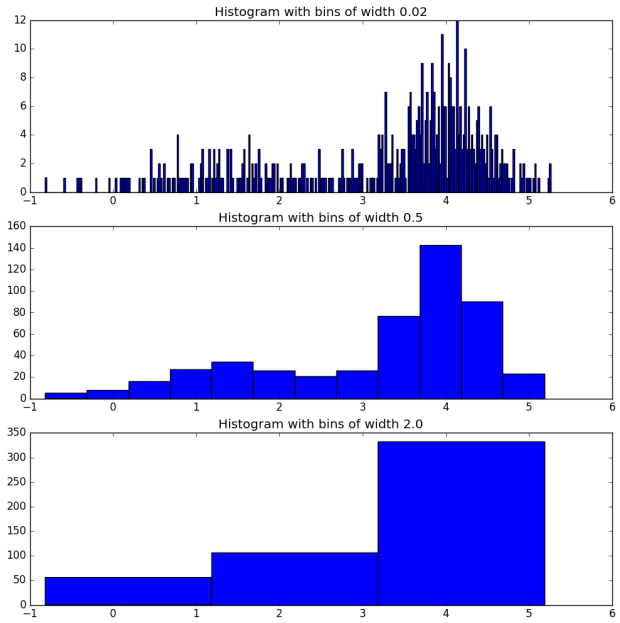
Name, Vorname: _____ Matrikelnummer: ⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋

Problem 2.3 Non-parametric Density Estimation [20 Points]

In this exercise, you will use the datasets `nonParamTrain.txt` for training and `nonParamTest.txt` for evaluating the performance of your model.

a) **Histogram [4 Points]**
Compute and plot the histograms using 0.02, 0.5, 2.0 size bins. Intuitively, indicate which bin size performs the best in and explain why. Knowing only these three trials, would you be sure that the one you picked is truly the best one? Attach the plot of the histograms and a snippet of your code.

Intuitively, the bins of size 0.5 perform the best. They show sufficient details of the distribution without having gaps. However, it is theoretically possible that the underlying distribution looks like the one that results from the histogram of bin size 0.02.

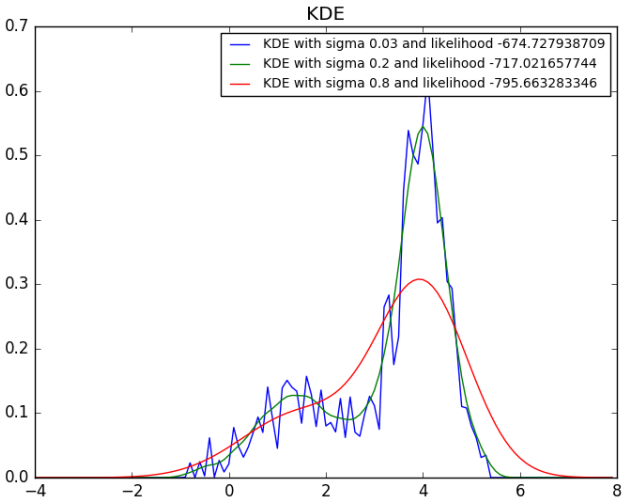



```
# Load the dataset
data = np.loadtxt(fname='../nonParamTrain.txt')

# Create figure with 3 subplots
f, axarr = plt.subplots(3)

# Iterate through all bin-widths
n = 0
for width in [0.02, 0.5, 2.0]:
    # Calculate boundaries of bins
    boundaries = np.arange(data.min(), data.max(), width)

    # Plot histogram
    axarr[n].hist(data, bins=boundaries)
    axarr[n].set_title("Histogram with bins of width {}".format(width))
    n += 1
```

```
# Show plot
plt.show()
```

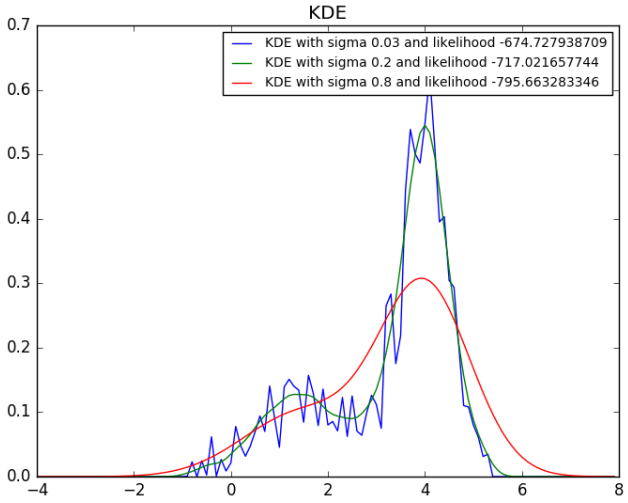
b) **Kernel Density Estimate [6 Points]**
Compute the probability density estimate using a Gaussian kernel with $\sigma = 0.03$, $\sigma = 0.2$ and $\sigma = 0.8$. Compute the log-likelihood of the data for each case, compare them and show which parameter performs the best. Generate a single plot with the different density estimates and attach it to your solutions. Plot the densities in the interval $x \in [-4, 8]$, attach a snippet of your code and comment the results.

The plot shows that the distribution resulting from the lowest sigma 0.02 has the highest log-likelihood. The function first creates two large arrays that are subtracted from each other to calculate all values of the sum simultaneously and then performs further steps to calculate the probability values, according to the gaussian kernel. In order to calculate the log-likelihood, the distribution can be evaluated at the samples by setting evalat==sample. The logarithm of every element of the resulting vector has to be taken and then summed up.




```
def calculateDensity(evalat, sample, sigma):
    nelements = sample.size
    nsteps = evalat.size
    #repeat vectors to facilitate calculation
    evalat = evalat.repeat(nelements, 0)
    sample = sample.repeat(nsteps, 1)

    # calculate exponent
    p = np.negative(np.power(np.abs(np.subtract(evalat, sample)), 2))
    p = np.divide(p, 2 * sigma * sigma)
    # exponantiate and sum up
    p = np.exp(p)
    p = np.sum(p, 0)
    # divide
    p = np.divide(p, nelements * np.sqrt(2 * np.pi) * sigma)

    return p
```

c) **K-Nearest Neighbors [6 Points]**
E stimate the probability density with the K-nearest neighbors method with $K = 2, K = 8, K = 35$. Generate a single plot with the different density estimates and attach it to your solutions. Plot the densities in the interval $x \in [-4, 8]$, attach a snippet of your code and comment the results. (Note: you need to normalize your solution according to K as the volume is not constant.)




```
def calculateknn(evalat, samples, K):
    nelements = samples.size
    nsteps = evalat.size
    # repeat vectors to facilitate calculation
    evalat = evalat.repeat(nelements, 0)
    samples = samples.repeat(nsteps, 1)

    # Calculate distance from evaluation points to all samples
    distances = np.sort(np.abs(np.subtract(samples, evalat)), 0)

    # Get distances to K-nearest neighbour
    distances = distances[K-1, ...]

    # Calculate probability according to formula
    p = np.divide(K, np.multiply(distances, nelements*2) )

    return p
```

d) **Comparison of the Non-Parametric Methods [4 Points]**
Estimate the log-likelihood of the testing data using the KDE estimators and the K-NN estimators. Why do we need to test them on a different data set? Compare the log-likelihoods of the estimators w.r.t. both the training and testing sets in a table. Which estimator would you choose?

Name, Vorname: _____ Matrikelnummer: ⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋

Problem 2.4 Expectation Maximization [20 Points]

In this exercise, you will use the datasets `gmm.txt`. It contains data from a Gaussian Mixture Model with four 2-dimensional Gaussian distributions.

a) **Gaussian Mixture Update Rules [2 Points]**
Define the model parameters and the update rules for your model. Specify the E- and M-steps of the algorithm.

b) **EM [18 Points]**
Implement the Expectation Maximization algorithm for Gaussian Mixture Models. Initialize your model uniformly. Generate plots at different iterations $t_i \in [1, 3, 5, 10, 30]$, showing the data and the mixture components, and plot the log-likelihood for every iteration $t_i = 1 : 30$.