Statistical Machine Learning 2018
Assignment 4
Deadline: 11th of January 2019

Christoph Schmidl
s4226887
c.schmidl@student.ru.nl

Mark Beijer
s4354834
mbeijer@science.ru.nl

December 7, 2018

Exercise 1 - Logistic regression (weight 5)

Part 1 - The IRLS algorithm

Many machine learning problems require minimizing / maximizing some function $f(x)$. For this, an alternative to the familiar gradient descent technique, is the so called Newton-Raphson iterative method:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \mathbf{H}^{-1}\nabla f(\mathbf{x}^{(n)}) \tag{1}$$

where $\mathbf{H}$ represents the Hessian matrix of second derivatives of $f(\mathbf{x})$, see Bishop, §4.3.3.

1.1.1

Derive an expression for the minimization / maximization of the function $f(x) = \sin(x)$, using the Newton-Raphson iterative optimization scheme (1), and verify (using Matlab, just up to, e.g., five iterations) how quickly it converges when starting from $x^{(0)} = 1$. What happens when you start from $x^{(0)} = -1$?
Hint: The Hessian of a 1-dimensional function $f(x)$ is just the second derivative f''. So, the Newton-Raphson iterative method reduces in 1-d to

$$x^{(n+1)} = x^{(n)} - \frac{f'(x^{(n)})}{f''(x^{(n)})} \tag{2}$$

Answer:

Expression for the minimization / maximization of the function $f(x) = \sin(x)$:

$$\begin{aligned} x^{(x+1)} &= x^{(n)} - \frac{\sin'(x^{(n)})}{\sin''(x^{(n)})} \\ &= x^{(n)} - \frac{\cos\left(x^{(n)}\right)}{-\sin\left(x^{(n)}\right)} \\ &= x^{(n)} + \frac{\cos\left(x^{(n)}\right)}{\sin\left(x^{(n)}\right)} \end{aligned}$$

```python
import numpy as np
import sympy as sp

# Exercise 1.1.1

## See also: https://docs.sympy.org/latest/tutorial/calculus.html

```

```python
def f(x):
    return np.sin(x)

def iterative_optimization(x):
    """ Using Newton-Raphson iterative optimization scheme """
    return x + (np.cos(x)/np.sin(x))

x_t = 1
n_iterations = 5

print("Starting the optimization process with x_0: {}".format(x_t))

for i in range(n_iterations):
    x_t = iterative_optimization(x_t)
    print("Step {}: {}".format(i, x_t))

# Starting the optimization process with x_0: 1
# Step 0: 1.6420926159343308
# Step 1: 1.5706752771612507
# Step 2: 1.5707963267954879
# Step 3: 1.5707963267948966
# Step 4: 1.5707963267948966

# Starting the optimization process with x_0: -1
# Step 0: -1.6420926159343308
# Step 1: -1.5706752771612507
# Step 2: -1.5707963267954879
# Step 3: -1.5707963267948966
# Step 4: -1.5707963267948966
```

If we take $x_0 = 1$, then the algorithm converges after 4 steps towards 1.5707963267948966 which is an approximation of $\frac{\pi}{2}$.
If we take $x_0 = -1$, then the algorithm converges after 4 steps towards -1.5707963267948966 which is an approximation of $-\frac{\pi}{2}$.

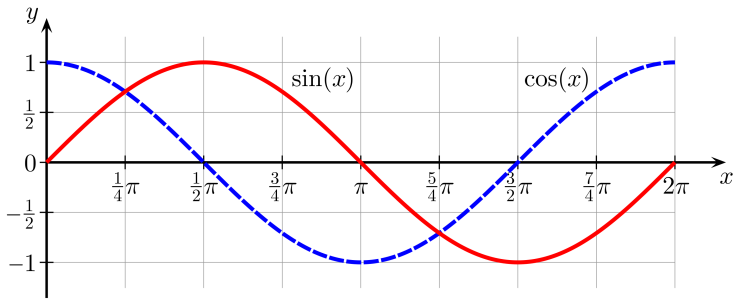

Figure 1: Sine and Cosine

When we take a look at figure 1 then we can see that the maximum and minimum of the sine function is indeed at $(x = \frac{\pi}{2}, f(x) = 1)$ and $(x = -\frac{\pi}{2}, f(x) = -1)$ with possible shiftings to the right and left based on the periodic nature of the sine function.

Note: Also see `https://en.wikipedia.org/wiki/Newton%27s_method#Applications`

1.1.2

We want to apply this method to the logistic regression model for classification (see Bishop, §4.3.2):

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(w^T\phi) \tag{3}$$

For a data set $\{\phi_n, t_n\}_{n=1}^N$, with $t_n \in \{0, 1\}$, using $y_n = p(\mathcal{C}_1|\phi_n)$ the corresponding cross entropy error function to minimize is

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \tag{4}$$

With one basis function ϕ and the dummy basis function 1, the feature vector in (3) becomes $\phi = [1, \phi]^T$. The weight vector including the bias term is then also two dimensional, $\mathbf{w} = [w_0, w_1]^T$. Expressions for

the gradient $\nabla E(\mathbf{w})$ and Hessian $\mathbf{H}$ in terms of the data set are given in Bishop, eq.4.96-98. As both are implicitly dependent on the weights $\mathbf{w}$, they have to be recalculated after each step: hence this is known as the 'Iterative Reweighted Least Squares' algorithm.

Consider the following data set: $\{\phi_1, t_1\} = \{0.3, 1\}, \{\phi_2, t_2\} = \{0.44, 0\}, \{\phi_3, t_3\} = \{0.46, 1\}$ and $\{\phi_4, t_4\} = \{0.6, 0\}$, and initial weight vector $\mathbf{w}^{(0)} = [1.0, 1.0]^T$.

Show using e.g. a Matlab implementation that for this situation the IRLS algorithm converges in a few iterations to the optimal solution $\hat{\mathbf{w}}^T \approx [9.8, -21.7]$, and show that this solution corresponding to a decision boundary $\phi = 0.45$ in the logistic regression model. (The IRLS algorithm should take about five lines of Matlab code inside a loop + initialization).

Answer:

TODO:

- Read Bishop, §4.3.2
- Read Bishop, eq.4.96 - 98
- The IRLS algorithm

Useful internet resources which helped solving this exercise:

- `https://thelaziestprogrammer.com/sharrington/math-of-machine-learning/solving-logreg-newtons-method`
- `http://cs229.stanford.edu/notes/cs229-notes1.pdf`
- `https://www.stat.cmu.edu/~cshalizi/350/lectures/26/lecture-26.pdf`
- `https://www.stat.cmu.edu/~cshalizi/402/lectures/14-logistic-regression/lecture-14.pdf`
- `https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/quadratic-approximations/a/the-hessian`
- `https://web.stanford.edu/group/sisl/k12/optimization/#!index.md`
- `http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex4/ex4.html`

Part 2 - Two-class classification using logistic regression

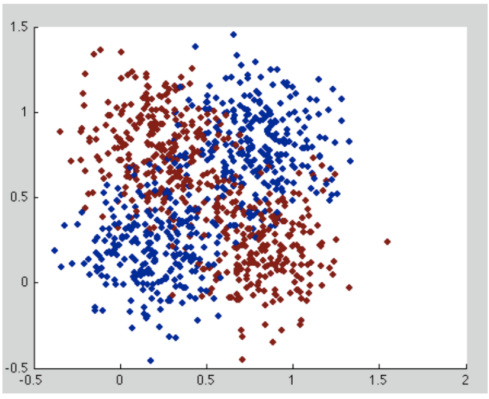

Figure 2: Two class data for logistic regression

Two-class classification using logistic regression in the IRLS algorith,. Data consists of 1000 pairs $\{x_1, x_2\}$ with corresponding class labels $\mathcal{C}_1 = 0$ or $\mathcal{C}_2 = 1$. Load it into Matlab using

```
data = load('a010_irlsdata.txt', '-ASCII');
X = data(:,1:2); Y = data(:,3);
```

1.2.1

Make a scatter plot of the data, similar to Figure 2. (Have a look at Matlab file `a010plotideas.m` in Brightspace for some ideas to make such a scatter plot and the plots later on.) Do you think logistic regression can be a good approach to classification for this type of data? Explain why.

Answer:

1.2.2

Modify the Iterative Reweighted Least Squares algorithm from part 1 to calculate the optimal weights for this data. Use again a dummy basis function. Initialize with the weight vector $\mathbf{w}^T = [0, 0, 0]$. With these initial weights, what are the class probabilities according to the logistic regression model (i.e., before optimization)?

Answer:

1.2.3

Run the algorithm. Make a scatter plot of the data, similar to figure 2, but now with color that represent the data point probabilities $P(C = 1, X_n)$ according to the model after optimization. Compare the cross entropy error with the initial value. Did it improve? Much? Explain your findings.

Answer:

1.2.4

Introduce two Gaussian basis functions as features ϕ_1, ϕ_2, similar to Bishop, fig.4.12. Use identical isotropic covariance matrices $\Sigma = \sigma^2 I$ with $\sigma^2 = 0.2$, and center the basis functions around $\mu_1 = (0, 0)$ and $\mu_2 = (1, 1)$. Make a scatter plot of the data in the feature domain. Do you think logistic regression can be a good approach to classification with these features? Explain why.

Answer:

1.2.5

Modify the IRLS algorithm to use the features $\{\phi_1, \phi_2\}$ and the dummy basis function. Initialize with the weight vector $\mathbf{w}^T = [0, 0, 0]$.
Run the algorithm. Make a scatter plot of the data, similar to Figure 2, but now with colors that represent the data point probabilities $P(C = 1|X_n)$ according to this second model (after optimization). Compare the cross entropy error with the initial value. Did it improve? Much? Explain your findings.

Answer:

Exercise 2 - Neural network regression (weight 5)

We train a neural network using backpropagation, to learn to mimic a 2D multimodal probability density. First, we implement the network and test its regression capabilities on a standard Gaussian; then we train it on the real data set. Visualization of the network output plays an important role in monitoring the progress.

2.1

Create a plot of an isotropic 2D Gaussian $y = 3 \cdot \mathcal{N}(\mathbf{0}|\frac{2}{5}\mathbf{I}_2)$ centered at the origin using the `meshgrid()`, `mvnpdf()` and `surf()` functions. Sample the density at 0.1 intervals over the range $[-2, 2] \times [-2, 2]$ and store the data in colum vector variables $\mathbf{X}$ (2D) and $\mathbf{Y}$ (1D).

Answer:

2.2

Implement a 2-layer neural network with $D = 2$ input nodes, $K = 1$ output nodes and M hidden nodes in the intermediate layer that can be trained using a sequential error backpropagation procedure, as described in Bishop §5.3. Use $tanh(\cdot)$ activation functions for the hidden nodes and a linear activation function (regression) for the output node. Introduce appropriate weights and biases, and set the learning rate parameter $\eta = 0.1$. Initialize the weights to random values in the interval $[-0.5, 0.5]$. Plot a 2D graph of the initial output of the network over the same $[-2, 2] \times [-2, 2]$ grid as the Gaussian (again using `surf()`).

Answer:

2.3

Train the network for $M = 8$ hidden nodes on the previously stored $\mathbf{X}$ and $\mathbf{Y}$ values (the $\{x_1, x_2\}$ input coordinates and corresponding output probability density y), by repeatedly looping over all datapoints and updating the weights in the network after each point. Repeat for at least 500 complete training cycles and monitor the progress of the training by plotting the output of the network over the $\mathbf{X}$ grid after each full cycle. Verify the output starts to resemble the Gaussian density after some 200 cycles (all be it with lots of 'wobbles').

Answer:

2.4

Permute the $\mathbf{X}$ and $\mathbf{Y}$ arrays to a random order using the `randperm()` function, keeping corresponding x and y together. Repeat the network training session using this randomized data set. Verify that convergence is now much quicker. Can you understand why? Try out the effect of different numbers of hidden nodes, different initial weights and different learning rates on speed and quality of the network training. Explain your results.

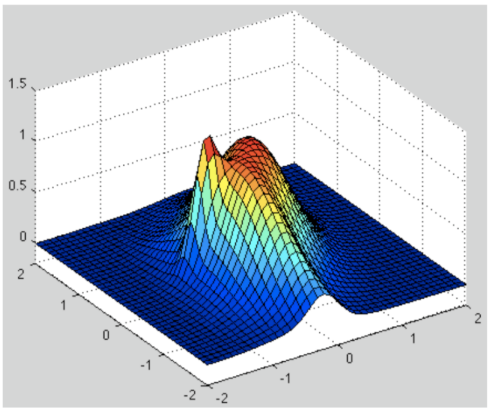

Figure 3: Multi-modal probability density

After these preliminaries we are now going to train the network on the real data set.
Load the data using

```
data = load('a017_NNpdfGaussMix.txt', '-ASCII');
X = data(:,1:2); Y = data(:,3);
```

Answer:

2.5

Create a $2D$-plot of the target probability density function. Notice that the data is in the correct sequence to use in `surf()`.

Answer:

2.6

Train the network on this data set. Use at least 40 hidden nodes and a learning rate parameter no higher than $\eta = 0.01$. Make sure the input data is properly randomized. Run the training phase for at least 2000 complete cycles and follow the progress by plotting the updated network output after every 20 full cycles. How does the final output of the network compare to the target distribution in the data? Explain. How could you improve the neural network in terms of speed of convergence and/or quality of the approximation?

Exercise 3 - Gaussian processes (weight 5)

Part 1 - Sampling from Gaussian stochastic processes

One widely used kernel function for Gaussian process regression is given by the exponential of quadratic form, with the addition of constant and linear terms (eq. 6.63 Bishop):

$$k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left(-\frac{\theta_1}{2}||\mathbf{x} - \mathbf{x}'||^2\right) + \theta_2 + \theta_3 \mathbf{x}^T \mathbf{x}' \tag{5}$$

We denote by $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3)$ the hyperparameter vector governing the kernel function k.

3.1.1

Implement the kernel given by Equation (5) in Matlab as a function of $\mathbf{x}, \mathbf{x}'$ and $\boldsymbol{\theta}$. Note that $\mathbf{x}$ can have any dimension.

Answer:

3.1.2

We first consider the univariate case. For the parameter values $\boldsymbol{\theta} = (1, 1, 1, 1)$ and $N = 101$ equally spaced points $\mathbf{X}$ in the interval $[-1, 1]$, compute the Gram matrix $\mathbf{K(X,X)}$ (eq. 6.54 Bishop). What is the dimension of $\mathbf{K}$? How can we show that $\mathbf{K}$ is positive semidefinite?

Note: Even when $\mathbf{K}$ is positive definitive, some of its eigenvalues may be too small to accurately compute (same for the determinant). This may pose a problem when generating a multivariate Gaussian distribution using $\mathbf{K}$ as its covariance matrix. You can alleviate this issue by adding a small diagonal term to $\mathbf{K}$.

Answer:

3.1.3

We will now use the previously computed matrix $\mathbf{K(X,X)}$ to produce samples from the Gaussian process prior $\mathbf{y(X)}\ \mathcal{N}(\mathbf{0,\ K(X,X)})$, with $\mathbf{X}$ being the previously determined N equally spaced points. Generate five functions $\mathbf{y(X)}$ with Matlab and plot them against the N input values $\mathbf{X}$. Repeat the process (remember to compute a new $\mathbf{K}$ each time) for the hyperparameter configurations from Bishop, Figure 6.5:

$$\boldsymbol{\theta} \in \{(1, 4, 0, 0), (9, 4, 0, 0), (1, 64, 0, 0), (1, 0.25, 0, 0), (1, 4, 10, 0), (1, 4, 0, 5)\}.$$

Describe the differences between the plots. Explain in which way each of the kernel parameters affects the generated samples.

Answer:

3.1.4

We now move to the bivariate case. Instead of an interval, we now consider a 2-D grid of equally spaced points of size $N = 21 \times 21$ in $[-1, 1] \times [-1, 1]$. We collect all these grid points in a data matrix $\mathbf{X}$, where each one of the 441 observations has two dimensions. What is the dimension of $\mathbf{K}$ now? What does this tell you about the scalability of sampling multivariate functions from Gaussian processes in higher dimensions?

Answer:

3.1.5

Using the same kernel from (5), compute the Gram matrix $\mathbf{K(X,X)}$ on the grid for each hyperparameter configuration $\boldsymbol{\theta} \in \{(1, 1, 1, 1), (1, 10, 1, 1), (1, 1, 1, 10)\}$. For each $\mathbf{K}$, generate and plot four random surfaces from the Gaussian process prior $\mathbf{yX()}\ \mathcal{N}\mathbf{(0,\ K(X,X))}$. Compare the observed differences to the univariate case.

Answer:

Part 2 - Gaussian processes for regression

We would like to apply Gaussian process models to the problem of regression (Bishop 6.4.2). We consider a noisy model of the form

$$t_n = y_n + \epsilon_n,$$

where $y_n = y(x_n)$ and ϵ_n are i.i.d. samples from a random noise variable on the observed target values. Furthermore, we assume that the noise process has a Gaussian distribution given by:

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{6}$$

Going back to a one-dimensional input space, we consider the following training data consisting of four data points

$$\mathcal{D}\{(x_1 = -0.5, t_1 = 0.5), (x_2 = 0.2, t_2 = -1), (x_3 = 0.3, t_3 = 3), (x_4 = -0.1, t_4 = -2.5)\}$$

3.2.1

Just as before, compute the Gram matrix of the training data for $\boldsymbol{\theta} = (1, 1, 1, 1)$. Then, taking $\beta = 1$ in Equation 6, compute the covariance matrix $\mathbf{C}$ corresponding to the marginal distribution of the training target values: $p(\mathbf{t}) = \mathcal{N}(\mathbf{t}|\mathbf{0},\ \mathbf{C})$.

Answer:

3.2.2

Using the previous results, compute the mean and the covariance of the conditional distribution $p(t|\mathbf{t})$ of a new target value t corresponding to the input $x = 0$. Which equations from Bishop do you need?

Answer:

3.2.3

Does the mean of the conditional distribution $p(t|\mathbf{t})$ go to zero in the limit $x \to \pm\infty$? If so, explain why this happens. If not, how would you set the parameters $\boldsymbol{\theta}$ of the kernel function to make it happen?

Answer:

Exercise 4 - EM and doping (weight 5)

In a certain hypothetical sport, banned substance '**X**' has become popular as a performance enhancing drug, as its presence is hard to establish in blood samples directly. Recently, it has been discovered that users of the drug tend to show a strong positive correlation between concentrations of two other quantities, x_1 and x_2, present in the blood. In contrast, 'clean' athletes tend to fall in one of two or three groups, that either show no or a negative correlation between x_1 and x_2. Unfortunately, as each sample contains only a single, instantaneous, measurement for each variable, it is not possible to establish this correlation from the sample. However, in many cases it is possible to distinguish to which *class* a certain sample belongs by also looking at the values of two other measured variables, x_3 and x_4: certain combinations of measured values are often typical for one class but highly unusual for others.
After a high profile event, a large scale test has resulted in 2000 samples. Rumours suggest the number of positives could be as high as 20%. However, the exact relationship between different classes and typical x values is still not clear. This is where the EM-algorithm comes in...

The blood sample measurements are modelled as a mixture of K Gaussians, one for each class

$$p(x|\mu, \Sigma, \pi) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \sigma_k) \tag{7}$$

where $\mathbf{x} = [x_1, x_2, x_3, x_4]$ represents the values for the measured quantities in the blood sample, $\mu = \{\mu_1, ..., \mu_K\}$ and $\Sigma = \{\Sigma_1, ..., \Sigma_K\}$ are the means and covariance matrices of the Gaussians for each class, and $\pi = \{\pi_1, ..., \pi_K\}$ are the mixing coefficients in the overall data set.

Load the data using

```
data = load('a011_mixdata.txt', '-ASCII');
```

and set N to the number of datapoints and D to the number of variables in the dataset X.

4.1

Try to give an estimate of the number, size and shape of the classes in the data, by plotting the distribution of the variables, e.g. using `hist()`, `scatter()` or `scatter3()`.

Answer:

4.2

Implement an EM-algorithm using the description and formulas given in Bishop, §9.2.2. Use variable K for the number of classes and choose a priori equal mixing coefficients π_k. Initialize the means $\boldsymbol{mu}_k$, to random values around the sample mean of each variable, e.g. set $\mu_{k,1}$ to $\bar{x}_1 + [-1 \leq \epsilon \leq +1]$. Initialize the Σ_k to diagonal matrices with reasonably high variances, e.g. $4 * \texttt{rand()} + 2$, to avoid very small responsibilities in the first step. Make sure the EM-loop runs over at least 100 iterations. Display relevant quantities, at least the log likelihood (9.28), after each step so you can monitor progress and convergence. Write a plot routine that plots the x_1, x_2 coordinates of the data points, and color each data point according to the most probable component according to the mixture model.

Answer:

4.3

Set $K = 2$, initialize your random generator and run the EM-algorithm on the data. Describe what happens. Try different random initializations and compare results.
(Should converge within 50 steps to two clusters, accounting for $\pm 1/3$ *resp.* $2/3$ *of the data).* Plot the x_1, x_2 coordinates colored according to the most probable component. Compute the correlation coefficients

$$p_{12} = \frac{cov[x_1, x_2]}{\sqrt{var[x_1] var[x_2]}} \tag{8}$$

of each of the components (i.e., use their covariance matrices to compute variances and covariances in (8), see also (Bishop, eq. (2.93))). Does either class show the characteristic strong[1] positive correlation for $\{x_1, x_2\}$?

Answer:

4.4

Increase the number of classes to $K = 3$ and rerun your algorithm on the data, again trying different random initializations. Plot the x_1, x_2 coordinates colored according to the most probable component and compute the correlation coefficients of each of the components. Check both your plot and your coefficients if one of the clusters now displays the strong positive $\{x_1, x_2\}$ correlation we are looking for.
Increase to $K = 4$, do the same, and see if this improves your result (in terms of detection of the doping-clusters). Based on your findings, is the rumoured 1-in-5 estimate for users of X credible?

Answer:

4.5

Having found the offending cluster in the data using the EM-algorithm, we are now presented with four samples $\{A, B, C, D\}$, with values for $[x_1, x_2, x_3, x_4]$ given as

$$A = [11.85, 2.2, 0.5, 4.0]$$
$$B = [11.95, 3.1, 0.0, 1.0]$$
$$C = [12.00, 2.5, 0.0, 2.0]$$
$$D = [12.00, 3.0, 1.0, 6.3]$$

One of these is from a subject who took drug X, and one is from a subject who tried to tamper with the test by artificially altering one or more of the x_i levels in his/her blood sample.

Identify which sample belongs to the suspected user and which one belongs to the 'fraud'.

Answer:

[1] According to Wikipedia, the correlation is none if $|p| < 0.1$, small if $0.1 < |p| < 0.3$, medium if $0.3 < |p| < 0.5$ and strong if $|p| > 0.5$