# Statistical Machine Learning: Exercise 3

**Linear Regression, Linear Classification and Principal Component Analysis**

**Prof. Stefan Roth, Dr. Simone Schaub-Meyer**

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2021

**Due date: Jul 6th, 23:59 PM**

## Task 1: Linear Regression (44 Points)

In this exercise, you will implement various kinds of linear regressors using the data `lin_reg_train.txt` and `lin_reg_test.txt`. The files contain noisy observations from an unknown function $f : \mathbb{R} \mapsto \mathbb{R}$. In both files, the first column represents the input and the second column represents the output. You can load the data using `numpy.loadtxt` and built-in functions for computing the mean.

For all subtasks, assume that the data is identically and independently distributed according to

$$y_i = \mathbf{\Phi}(\mathbf{x_i})^\top \mathbf{w} + \epsilon_i,$$

where

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2),$$

and $\Phi : \mathbb{R} \to \mathbb{R}^n$ is a feature transformation such that

$$\mathbf{y} \sim \mathcal{N}(\mathbf{\Phi}(\mathbf{X})^\top \mathbf{w}, \sigma^2 \mathbf{I}).$$

Additionally, make sure that your implementations support multivariate inputs. The feature transformations are given in each task; if no basis function is stated explicitly, use the data as is, i.e. $\mathbf{\Phi}(x) = x$.

### 1a) Linear Features (12 Points)

Implement linear ridge regression using linear features, i.e. the data itself. Include an additional input dimension to represent a bias term and use the ridge coefficient $\lambda = 0.01$.

1. Explain: What is the ridge coefficient and why do we use it? (1)
2. Derive the optimal model parameters by minimizing the squared error loss function. (3)
3. Attach snippets of your code showing how you fit the model. (4)
4. Report the root mean squared error of the train and test data under your linear model with linear features. (2)
5. Include a single plot that shows the training data as black dots and the predicted function as a blue line. (2)

## 1b) Polynomial Features (8 Points)

Implement linear ridge regression using a polynomial feature projection. Include an additional input dimension to represent a bias term and use the ridge coefficient $\lambda = 0.01$.

For polynomials of degrees 2, 3 and 4:

1. Attach snippets of your code showing how you generate polynomial features and how you fit the model. (3)
2. Report the root mean squared error of the training data and of the testing data under your model with polynomial features. (2)
3. Include a single plot that shows the training data as black dots and the predicted function as a blue line. (2)
4. Why do we call this method *linear* regression despite using polynomials? (1)

## 1c) Bayesian Linear Regression (11 Points)

Implement Bayesian linear ridge regression, assuming that $\mathbf{w}$ follows a multivariate Gaussian distribution, such that

$$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0^{-1}),$$

where ridge regression dictates $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{\Lambda}_0 = \lambda\mathbf{I}$.

Here, $\boldsymbol{\mu}_0$ is the prior weight mean and $\boldsymbol{\Lambda}_0$ is the prior weight precision matrix, i.e. the inverse of the covariance matrix. The corresponding posterior parameters can be denoted as $\boldsymbol{\mu}_n$ and $\boldsymbol{\Lambda}_n$.

Assume $\sigma = 0.1$, use $\lambda = 0.01$, and include an additional input dimension to represent a bias term. Use all of the provided training data for a single Bayesian update.

1. State the posterior distribution of the model parameters $p(\mathbf{w} \mid \mathbf{X}, \mathbf{y})$ (no derivation required). (1)
2. State the predictive distribution $p(\mathbf{y}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y})$ (no derivation required). (1) lecture_08_Regression page50
3. Attach snippets of your code showing how you compute the parameters of the posterior and predictive distribution. (4)
4. Report the RMSE of the train and test data under your Bayesian model (use the predictive mean). (1)
5. Report the average log-likelihood of the train and test data under your Bayesian model. (1)
6. Include a single plot that shows the training data as black dots, the mean of the predictive distribution as blue line and 1, 2 and 3 standard deviations of the predictive distribution in shades of blue (you can use matplotlib's `fill_between` function for that). (2)
7. Explain the differences between linear regression and Bayesian linear regression. (1)

## 1d) Squared Exponential Features (9 Points)

Implement Bayesian linear ridge regression using squared exponential (SE) features. In other words, replace your observed data matrix $\mathbf{X} \in \mathbb{R}^{n\times 1}$ by a feature matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n\times k}$, where

$$\boldsymbol{\Phi}_{ij} = \exp\left(-\frac{1}{2}\beta(\mathbf{X}_i - \alpha_j)^2\right).$$

Set $k = 20$, $\alpha_j = j * 0.1 - 1$ and $\beta = 10$. Use the ridge coefficient $\lambda = 0.01$ and assume known Gaussian noise with $\sigma = 0.1$. Include an additional input dimension to represent a bias term.

1. Attach snippets of your code showing how you generate SE features and apply your Bayesian model. (3)
2. Report the RMSE of the train and test data under your Bayesian model with SE features. (1)

3. Report the average log-likelihood of the train and test data under your Bayesian model with SE features. (1)
4. Include a single plot that shows the training data as black dots, the mean of the predictive distribution as blue line and 1, 2 and 3 standard deviations of the predictive distribution in shades of blue (you can use matplotlib's `fill_between` function for that). (2)
5. How can SE features be interpreted from a statisticians point of view? What are $\alpha$ and $\beta$ in that context? (2)

### 1e) Cross validation (4 Points)

So far, we have split our dataset in two sets: training data and testing data. Cross-validation is a more sophisticated approach for model selection. Discuss it and its variants, pointing out their pro and cons.

## Task 2: Linear Classification (16 Points)

In this exercise, you will use the dataset `ldaData.txt`, containing 137 feature points $\vec{x}$. The first 50 points belong to class $C_1$, the second 43 to class $C_2$, the last 44 to class $C_3$.

### 2a) Discriminative and Generative Models (4 Points)

Explain the difference between discriminative and generative models and give an example for each case. Which model category is generally easier to learn and why?

### 2b) Linear Discriminant Analysis (12 Points)

Use Linear Discriminant Analysis to classify the points in the dataset. Attach two plots with the data points using a different color for each class: one plot with the original dataset, one with the samples classified according to your LDA classifier. Attach a snippet of your code and discuss the results. How many samples are misclassified? (You are allowed to use built-in functions for computing the mean, the covariance, eigenvalues and eigenvectors.)

## Task 3: Principal Component Analysis (28 Points)

In this exercise, you will use the dataset `iris.txt`. It contains data from three kind of Iris flowers ('Setosa', 'Versicolour' and 'Virginica') with 4 attributes: sepal length, sepal width, petal length, and petal width. Each row contains a sample while the last attribute is the label ($0$ means that the sample comes from a 'Setosa' plant, $1$ from a 'Versicolour' and $2$ from 'Virginica'). (You are allowed to use built-in functions for computing the mean, the covariance, eigenvalues and eigenvectors.)

### 3a) Data Normalization (3 Points)

Normalizing the data is a common practice in machine learning. Normalize the provided dataset such that it has zero mean and unit variance per dimension. Why is normalizing important? Attach a snippet of your code.

### 3b) Principal Component Analysis (8 Points)

Apply PCA on your normalized dataset and generate a plot showing the proportion (percentage) of the cumulative variance explained (you can use the function `numpy.cumsum`). How many components do you need in order to explain at least $95\%$ of the dataset variance? Attach a snippet of your code.

### 3c) Low Dimensional Space (6 Points)

Using as many components as needed to explain $95\%$ of the dataset variance, generate a scatter plot of the lower-dimensional projection of the data. Use different colors or symbols for data points from different classes. What do you observe? Attach a snippet of your code.

### 3d) Projection to the Original Space (6 Points)

Reconstruct the original dataset by using different number of principal components. Using the normalized root mean square error (NRMSE) as a metric, fill the table below (error per input versus the amount of principal components used).

| N. of components | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

Attach a snippet of your code. (Remember that in the first step you normalized the data.)

### 3e) Kernel PCA (5 Points)

Throughout this class we have seen that PCA is an easy and efficient way to reduce the dimensionality of some data. However, it is able to detect only linear dependences among data points. A more sophisticated extension to PCA, *Kernel PCA*, is able to overcome this limitation. This question asks you to deepen this topic by conducting some research by yourself: explain what Kernel PCA is, how it works and what are its main limitations. Be as concise (but clear) as possible.