Statistical Machine Learning: Exercise 3

Linear Regression, Linear Classification and Principal Component Analysis

Prof. Dr. Kristian Kersting, Karl Stelzner, Claas Voelcker

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2020

Due date: Jul 12th, 23:59 PM

Task 1: Linear Regression (8 + 5 + 5 + 7 + 6 [+ 10] = 31 [+ 10])

In this exercise, you will implement various kinds of linear regression 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 inputs and the second column represents the outputs. You can load the data using `numpy.loadtxt`.

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 $\Phi : \mathbb{R}^1 \to \mathbb{R}^n$ 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 transformation are given in each task, if no basis functions are stated explicitely use the data as is $\mathbf{\Phi}(x) = x$.

1a)

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. Report the root mean squared error of the train and test data under your linear model with linear features. (2)
4. Include a single plot that shows the training data as black dots and the predicted function as a blue line. (2)

1b)

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. Report the root mean squared error of the training data and of the testing data under your model with polynomial features. (2)

2. Include a single plot that shows the training data as black dots and the predicted function as a blue line. (2)
3. Why do we call this method *linear* regression despite using polynomials? (1)

1c)

Implement 5-fold cross-validation to select the optimal degree for your polynomial regression.

- Start by splitting the provided data into 5 distinct subsets with each subset consisting of 20% of the original data.
- Use subsets 1 - 4 to train your model with polynomial features of degrees 2, 3 and 4.
- Compute the train RMSEs using your trained models and subsets 1 - 4.
- Compute the validation RMSEs using your trained models and subset 5.
- Compute the test RMSEs using your trained models and the test data.
- Repeat the previous steps, cycling through the subsets until every subset has been used for validation.

Provide the following results and answers:

1. For each polynomial degree, report the average train, validation and test RMSEs among all folds. (2)
2. Explain: Do the resulting numbers meet your expectations? Why (not)? (2)
3. Which polynomial degree should be chosen for the given data? Why? (1)

1d)

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. inverse of 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)
3. Report the RMSE of the train and test data under your Bayesian model (use the predictive mean). (1)
4. Report the average log-likelihood of the train and test data under your Bayesian model. (1)
5. 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)
6. Explain the differences between linear regression and Bayesian linear regression. (1)

1e)

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 $\mathbf{\Phi} \in \mathbb{R}^{n \times k}$, where

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

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. Report the RMSE of the train and test data under your Bayesian model with SE features. (1)
2. Report the average log-likelihood of the train and test data under your Bayesian model with SE features. (1)
3. 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)
4. How can SE features be interpreted from a statisticians point of view? What are α and β in that context? (2)

1f)

In this bonus assignment, you will perform a grid search using $\beta \in \{1, 10, 100\}$ to select a 'better' β for your squared exponential features from the previous subtask. Keep using the same settings as in the previous subtask, except β.

Grid search is a simple method to select hyperparameters, such as β. First, a list of possible values, or a grid, in case of multiple hyperparameters, is created to define a discrete search space. For every value in this search space, a model is trained and evaluated using a score or loss function. Finally, the hyperparameters that yield the highest score or smallest loss are selected. Here, the log-marginal likelihood will be used as a score function.

The log-marginal likelihood of our Bayesian linear model can be expressed as (see Bishop Ch. 3.5)

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) &= \log \int p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) p(\mathbf{w}) \, \mathrm{d}\mathbf{w}, \\ &= \frac{k+1}{2}\log\lambda - \frac{n}{2}\log\sigma^2 - \frac{1}{2}\frac{\|\mathbf{y} - \mathbf{\Phi}\boldsymbol{\mu}\|_2^2}{\sigma^2} + \frac{\lambda}{2}\boldsymbol{\mu}^\top\boldsymbol{\mu} - \frac{1}{2}\log|\mathbf{\Lambda}| - \frac{n}{2}\log 2\pi,\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{\mu} &= \sigma^{-2}\mathbf{\Lambda}^{-1}\mathbf{\Phi}^\top\mathbf{y}, \\ \mathbf{\Lambda} &= \sigma^{-2}\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\mathbf{I}.\end{aligned}$$

Here, k is the dimensionality of the feature space, the $+1$ comes from the extra bias term.

Using the marginal likelihood to select hyperparameters is typically referred to as empirical Bayes, type-II maximum likelihood or evidence approximation. Applying the logarithm (analytically) ensures numerical stability.

1. What is the difference between the marginal likelihood $p(\mathbf{y} \mid \mathbf{X})$ and the likelihood $p(\mathbf{y} \mid \mathbf{X}, \mathbf{w})$? (1)
2. For each β, report RMSE and average log-likelihood of the train and test data and the log-marginal likelihood. (3)
3. For each beta, 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)
4. According to the grid search, which value for β is the best? Why? (2)
5. Compare the log-marginal likelihood values to the average train and test log-likelihood values. What do you observe? Is the log-marginal likelihood a 'good' score function compared to the train log-likelihood? (2)

Task 2: Linear Classification (4 + 12 = 16)

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)

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)

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 and the covariance.)

Task 3: Principal Component Analysis (3 + 8 + 6 + 6 + 5 [+ 5] = 28 [+ 5])

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, eigenvectors and singular value decomposition.)

3a)

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)

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

3c)

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)

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)

In machine learning, it is often desirable to 'whiten' the data before applying a model or an algorithm. In this context, 'whitening' the data refers to a transformation that warps the data into a spherical shape, such that the data dimensions become uncorrelated and the individual means and variances are 0 and 1, respectively. In particular, PCA can be used to compute such a transformation.

Recommended reading: ufldl.stanford.edu/tutorial/unsupervised/PCAWhitening

1. Explain the difference between PCA and ZCA whitening. (1)
2. State the equation(s) to compute the ZCA whitening parameters, given the data. (1)
3. State the equation(s) to whiten a (new) data example $\mathbf{x}$, given the ZCA parameters. (1)

4. Compute and report the ZCA whitening parameters for the unnormalized IRIS data (including numerical values!). For numerical stability, use $\epsilon = 1e - 5$. (2)

3f)

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.