Deep Vision
Summer Semester 20
Prof. Björn Ommer
ommer@uni-heidelberg.de

Exercise sheet 2: Classification

Due on *14.05.2020*, 11am.

Patrick Esser (patrick.esser@iwr.uni-heidelberg.de).

You can find starter code for this exercise in `ex02.py`.

Task 1: k-Nearest Neighbors Classification (4P)

In this exercise, we will implement a k-Nearest Neighbor (KNN) classifier with an interface similar to that provided by `sklearn.neighbors.KNeighborsClassifier`. The function `task1` contains code to create a synthetic, 2D dimensional dataset consisting of 2D points `x` and binary labels `y`.

- Visualize the dataset using a scatter plot, with data points colored according to their label. You can use `matplotlib` or any other plotting library of your choice.
- Implement the `kneighbors` method of the `KNN` class. For given query points, it should return the indices of the k-nearest neighbors in the training set, where k is given by `KNN.n_neighbors`, together with the distances. Use the euclidean metric for distances. You should only use `numpy` in your implementation and for full points your implementation should be vectorized.
- Implement the `predict` method of the `KNN` class. It should return the predicted label for each query point.
- For `k=5`, fit both your kNN implementation and that from `sklearn` to the training data and check that their predictions on the test set are the same.
- Evaluate and plot the accuracy of the kNN classifier on the test set against different values of $k = 2^i, i = 0, \dots, 9$.
- For the same values of k, plot the decision boundary of the kNN classifier with the data points on top. Plot the decision boundary within the box $[-1.5, 2.5] \times [-1.0, 1.5]$. *Hint:* Evaluate the kNN classifier on a grid within this box. Use roughly 100 points in each direction for this grid, which can be achieved with `numpy.meshgrid`. Use the results to draw a filled contour plot (`contourf` in `matplotlib`).
- What is the effect of k on the decision boundary? What would happen in the case where k equals the number of training examples?

Task 2: kNN Classification of Images **(2P)**

In this exercise, we will apply kNN classification to image data. `task2` contains code to load images of digits together with labels between 0 and 9. You can either use your own KNN implementation from the previous task or that of `sklearn`.

- Evaluate and plot the accuracy of the kNN classifier on the test set against different values of $k = 2^i, i = 0, \ldots, 3$.
- For $k = 8$, produce a plot that shows 10 test images together with their k-nearest neighbors. Include examples for both successful and unsuccesful predictions.

Task 3: Linear Least Squares **(4P)**

In this exercise, we will implement a Linear Least Squares (LLS) classifier. Let $\bar{x}_i$ denote the i-th training example, and $y_i \in \{-1, 1\}$ its corresponding label. The linear score function f with parameters $\bar{w}$ and b is given by

$$f(\bar{x}, \bar{w}, b) = \langle \bar{x}, \bar{w} \rangle + b \tag{1}$$

Using the bias trick, we write x for $\bar{x}$ with a one prepended, and w for $\bar{w}$ with b prepended. Then we can write the above as $f(x, w) = \langle x, w \rangle$. To fit the discriminant function to the training data, we want to minimize the squared difference between the value of f and the labels over the parameter w:

$$L(w) = \frac{1}{2} \sum_i (f(x_i, w) - y_i)^2 \tag{2}$$

Stacking all training points into the rows of a matrix X and labels into a vector y, this is equivalent to

$$L(w) = \frac{1}{2} \|Xw - y\|^2 \tag{3}$$

- Derive the following formula for the minimizer w^* of L:

$$w^* = (X^t X)^{-1} X^t y \tag{4}$$

Hint: These are the first-order-optimality conditions for L, stating that its gradient must be zero at the minimizer. To compute the gradient, expand the squared norm into an inner product, $\|h\|^2 = \langle h, h \rangle$, and use the fact that the gradient of $h \mapsto \langle Ah, h \rangle$ is $2Ah$ and that of $h \mapsto \langle v, h \rangle$ is v.

- Implement the computation of w^* in the `fit` method of `LinearLeastSquares`. *Hint:* Make sure to prepend ones to the input to use the bias trick and use `numpy.linalg.inv` for matrix inversion.
- Implement the `predict` method of the class `LinearLeastSquares` returning class predictions according to the sign of $f(\cdot, w^*)$.

The function `task3` contains code to generate synthetic, 2D data. The training data includes an outlier, and the parameter `outlier` controls the magnitude of this outlier. For outlier $= 2^i, i = 0, \ldots, 4$,

- Visualize the training dataset (including the outlier in `xtrain`).
- Fit both your implementation of `LinearLeastSquares` and `LinearSVC` from `sklearn.svm` to the training data, evaluate its accuracy on the test data and plot its decision boundary within the box $[-1.5, 2.5] \times [-1.0, 1.5]$.
- How are the two methods affected by the outlier? Give a short explanation.

Note: Submit exactly one ZIP file and one PDF file via Moodle before the deadline. The ZIP file should contain your executable code. Make sure that it runs on different operating systems and use relative paths. Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. The PDF file should contain your written code, all figures, explanations and answers to questions. Make sure that plots have informative axis labels, legends and captions.