183.605
Machine Learning for Visual Computing
Assignment 1

Michael Reiter

16. Oktober 2018

Assignment via TUWEL. Please be aware of the deadlines in TUWEL.

- Upload a zip-file with the required programs. You can choose the programming language.
- Add a PDF document with answers to all of the questions of the assignment (particularly all required plots) and description and discussion of results.

1 Assignment 1

The aim of this first assignment is to gather experience with linear models. We will apply a linear model to synthetic examples of binary classification (part 1) and polynomial regression (part 2). In both cases we will employ basis functions to allow for modeling non-linear functions of the original data.

You have free choice of the programming language (we recommend Matlab, R or Python). You are asked to implement the required functions by yourself, without using pre-packaged programs providing these functions. The required algorithms are introduced in the lecture (you are referred to recommended literature and lecture slides).

1.1 Part 1: Binary classification and the perceptron

1.1.1 The MNIST data set

MNIST is a database of handwritten digits available on `http://yann.lecun.com/exdb/mnist/`. There are packages for reading the data in all recommended programming languages (e.g. mnistHelper for Matlab or mnist for Python).

Tasks:

- Read the data using available packages for your programming language resp. simulation software.
- Choose two classes (e.g., all images of digits '0' and '1') for the following two-class classification task. Select a small subset $\mathcal{T}$(with less than 1000 images in total) of corresponding images from the MNIST training set
- Extract two suitable image features from the subset $\mathcal{T}$. For example Matlab's *regionprop*-function allows to calculate image features such as *FilledArea* and *Solidity* from binary images.
- Plot the input vectors in $\mathbb{R}^2$ and visualize corresponding target values (e.g. by using color).

1.1.2 Perceptron training algorithm

The function

```
y = perc(w,X).
```

simulates a perceptron. The first argument is the weight vector $\mathbf{w}$ and the second argument is a matrix with input vectors in its columns $\mathbf{X}$. The output $\mathbf{y}$ is a binary vector with class labels 1 or -1.

The function

```
w = percTrain(X,t,maxIts,online).
```

returns a weight vector $\mathbf{w}$ corresponding to the decision boundary separating the input vectors in $\mathbf{X}$ according to their target values $\mathbf{t}$.

The argument `maxIts` determines an upper limit for iterations of the gradient based optimization procedure. If this upper limit is reached before a solution vector is found, the function returns the current $\mathbf{w}$, otherwise it returns the solution weight vector. `online` is *true* if the *online*-version of the optimization procedure is to be used or *false* for the *batch*-version.

Tasks:

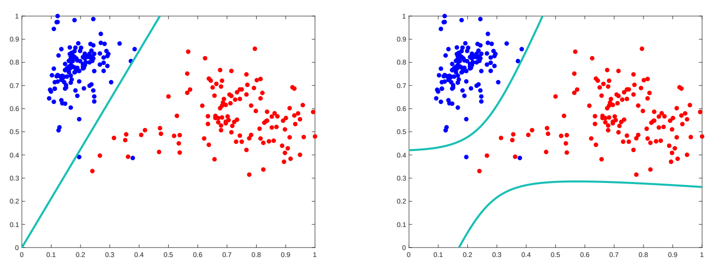
- Implement both functions. Use homogeneous coordinates and the corresponding augmented weight vector $\mathbf{w} \in \mathbb{R}^m$, where m is the dimensionality of the augmented input space.

- Train the perceptron using $\mathcal{T}$ and plot the data together with the resulting decision boundary in $\mathbb{R}^2$. Is the data set $\mathcal{T}$ linearly separable? (Depending on your selected subset and images features *yes* or *no* is possible.) Is the perceptron training algorithm capable of detecting linearly non-separable data?

- Use the feature transformation $\Phi(\mathbf{x}) : (x_1, x_2) \rightarrow (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$ and plot the data and decision boundary in the original data space $\mathbb{R}^2$ (see e.g. Figure 1) after training. *Hint:* Sample the relevant region of the input space using a *meshgrid*. Compute $y = \mathbf{w}^T \Phi(\mathbf{x})$ for all grid points and use a *contour*-function or a surface-plot to visualize the approximation of the curve $y = 0$.

- Train the perceptron using all $28 \times 28 = 784$ pixels of MNIST images of $\mathcal{T}$ as input, resulting in augmented input vectors with dimensionality of $m = 785$ and visualize $w_1, \ldots, w_m$ as a 28×28 gray-scale image (see Figure 2).

- Compare the error rate (percentage of falsely classified input vectors) of all three experiments (2 features, 5 features, whole images) on the independent MNIST test set.



(a) (b)

Figure 1: Plot of the decision boundary (curve) in the original feature space (*FilledArea* vs. *Solidity*) found by the perceptron (a) in the original 2D-space of $\mathbf{x}$ and (b) in the transformed feature space of $\Phi(\mathbf{x})$ together with labelled training vectors.

1.2 Part 2: Linear basis function models for regression

Aim of this exercise to deepen understanding of parameter optimization of an error function in the context of a regression problem.

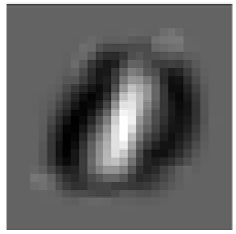


Figure 2: Visualization of weights $w_1, \ldots, w_m$ after training directly on input images of digits '0' and '1'.

1.2.1 Experimental setup

A row vector of scalar inputs $x \in [0, 5]$ obtained by sampling the interval in steps of 0.1 (resulting in 51 values) and a corresponding output vector $\mathbf{y}$ with values $y = f(x) = 2x^2 - Gx + 1$ is the basis of this experiment. The coefficient G is your group number. These 51 points are to be used for the visualization of the target function and the predictions of the fitted model. A training set is generated by subsampling the 51 values as follows: Every eighth value ($x_0 = 0$, $x_1 = 0.8$, $x_2 = 0.16$, ...) is assigned to the training set and the target values t_i are obtained by adding to the corresponding y_i a random value from the normal distribution $\mathcal{N}(\mu = 0, \sigma = 4)$[1]. Thus, the training set contains $N = 7$ pairs of observations x_i, t_i.

We will employ a linear basis function model of the form $f_{\mathbf{w}}(x) = \mathbf{w}^T \Phi(x)$, where

$$\mathbf{\Phi}(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{pmatrix}, \tag{1}$$

and $\mathbf{w} \in \mathbb{R}^{d+1}$. The model will be fitted to the training set by minimization of the training error

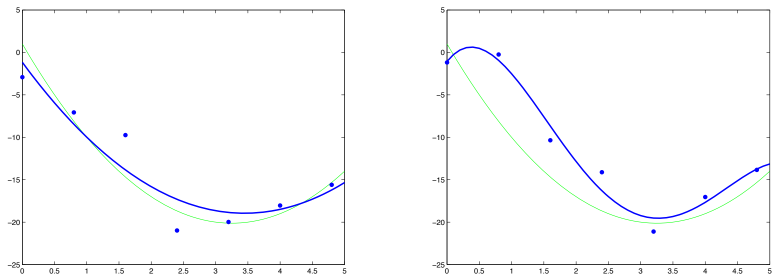
$$E(\mathbf{w}) = \sum_{i=1}^{N} (t_i - \mathbf{w}^T \mathbf{\Phi}(x_i))^2 \tag{2}$$

also known as the *residual sum of squares* (RSS). The optimal weight vector is given by $\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w})$.

1.2.2 Optimization: LMS-learning rule vs. closed form

Use a linear unit (*online* LMS-learning rule) for regression on transformed input data. In a first step use a linear basis function model with $d = 2$ (in Matlab you

[1] i.e., variance $\sigma^2 = 16$



(a) (b)

Figure 3: An example of a true target function (thin green curve) from which the training data was generated, training set (without feature transformation) with $N = 7$ (blue dots) and prediction of the fitted model $\mathbf{w}^T \mathbf{\Phi}(x)$ (blue curve). The basis functions are $\mathbf{\Phi}(x) \to (1, x, x^2, x^3, ..., x^d)^T$. (a) $d = 2$ (b) $d = 4$.

can calculate the power elementwise: e.g. `[x x x].∧ [0 1 2]`). Hint: Visualize y and its prediction during the training or observe the change of the weight vector to determine useful values for γ.

Tasks:

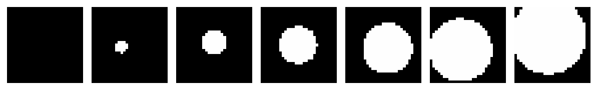
- What is the resulting weight vector when using the LMS-rule?
- How can you determine the optimal $\mathbf{w}^*$ in closed form? Compare $\mathbf{w}^*$ with the outcome of the LMS-rule training.
- What is the influence of γ? Which value for γ represents a good tradeoff between number of iterations and convergence?

1.2.3 Image data

As a last experiment use the same data set as in Section 1.2.1 but instead of features $\Phi(x)$ represent the scalar input variable x by a 29×29 grey-scale image augmented with $x_0 = 1$. x is represented by a circular region where every pixel has a value 1 if $(i - m_1)^2 + (j - m_2)^2 - (3 * x)^2 > 0$ where (i, j) is the pixel's position. Outside the circular region, where $(i - m_1)^2 + (j - m_2)^2 - (3 * x)^2 \leq 0$ the pixel value is 0. Instead of noisy target values, this time $t_i = y_i = f(x_i) = 2x_i^2 - Gx_i + 1$ (no noise added) and instead the center of the circles are distorted by noise, i.e., m_1 and m_2 is random with a normal distribution around the center of the image by $\mathcal{N}(\mu = 15, \sigma = 2)$. Figure 4 shows an example of training images corresponding to values $0, 0.8, 1.6, \ldots, 4.8$.
Tasks:

- Calculate $\mathbf{w}^*$ in closed form.

- Plot the predicted $\hat{y}_i$ vs. the true y_i for the 7 training images. Compute the training error.
- Plot the predicted $\hat{y}_i$ vs. the true y_i for all 51 images. What happens if we increase the noise variance for the centers?



(a) (b) (c) (d) (e) (f) (g)

Figure 4: Representation of values $0, 0.8, 1.6, \ldots, 4.8$ by binary images with circular regions with corresponding radius and random center.