582744 Advanced Course in Machine Learning

Exercise 2 **Due Mar 28, 23:55 AM**

Rules:

1. Return your solutions in Moodle by the deadline.
2. The submission consists of two parts: (a) A single PDF file containing your answers to all questions. (b) A single file containing your code (either a single plain text source code or a compressed file). Do not include datasets, plots etc in this file, only the code.
3. If you feel comfortable, add an estimate of how many hours you worked on the problems in the beginning of your report.
4. Please typeset your work using appropriate software such as LaTeX. However, there is no need to typeset the pen and paper answers – you can also include a scanned hand-written version.
5. Pay attention to how you present the results. Be concise.
6. Spotted a mistake? Something is unclear? Ask for clarifications in Moodle.

This set of exercises is due on Tuesday Mar 28, before 23:55 AM.

1 Training and generalization error (programming, 3 pts)

Read in the training data matrix $\mathbf{X}$ given in Moodle. Each row is a data point of $D = 10$ dimensions and there are in total $N = 500$ data points. The last element on each row is a scalar output value.

In this exercise we will study a simple linear regression model. The model is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^D$ and the loss function is given by $L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}_n - y_n)^2$. We will also consider regularizers of the form $g(\boldsymbol{\theta}) = \boldsymbol{\theta}^T\boldsymbol{\theta}$. The gradients of these two are given by

$$\nabla L = \frac{2}{N}\sum_n(\boldsymbol{\theta}^T\mathbf{x}_n - y_n)\mathbf{x}_n,$$
$$\nabla g = 2\boldsymbol{\theta}.$$

Start by implementing the standard gradient descent method (Section 8.3.2) for this problem. For this you need to write a function that computes the loss for a given set of data points and the parameters of the model, a function that evaluates the gradient for given parameters, and the actual gradient descent loop. For this exercise, simply use the fixed step-size 0.5 and start with the initial solution of $\boldsymbol{\theta} = 0$.

We will not study overfitting. You will produce three alternative estimates for the parameters and compare the resulting errors on a separate test set not available during training.

(a) *Empirical risk minimization (ERM)*: Minimize the empirical risk for the first 50 samples of the data set. Run the algorithm until convergence, recognized as the squared norm (sum of the squared elements) of the gradient dropping below 10^{-6}.

(b) *Early-stopping*: Minimize the empirical risk for the same subset of 50 samples, but after each iteration compute also the validation loss on the remaining $N - 50 = 450$ samples. Stop the iteration when the validation loss no longer decreases.

(c) *Regularization*: Minimize the regularized risk for the same set of samples, now including also the regularization term $\lambda g(\boldsymbol{\theta})$ as part of the cost function. Use the same criterion for detecting convergence as for ERM. Try out different values for the regularization constant $\lambda \in \{0.001, 0.003, 0.01, 0.03, 0.1\}$ and plot the validation loss (the loss itself, without the regularization term) as a function of λ. Choose λ that gives the best validation loss.

Now you have three alternative solutions for the model. Evaluate their accuracy on the test data provided as a separate file in Moodle. The data is in the same format as the training data, but has 2500 samples. Which of the three solutions is the best? Are the other solutions good as well? Report also the number of iterations taken until convergence.

Above we used 10% of the available data for training and 90% for validation. Repeat the same process but exchanging the training and validation data. How do the results change? Report the final test accuracy and number of iterations for all three methods.

What to report:

(a) A plot showing the training and validation errors as a function of the iteration for ERM, indicating somehow the point where early-stopping terminated the iteration.

(b) A plot of the final validation error as a function of the regularization parameter.

(c) The final test errors and numbers of iterations for all variants.

2 Comparison of optimization methods (3 pts)

We use the same model as in Problem 2, but now use it to model a simple two-dimensional data set provided in Moodle. Again the last element on each row is the output value. No regularization is used in this case.

We will now compare three optimization algorithms for minimizing the empirical risk. For each method start from the parameter values $\boldsymbol{\theta} = [-0.3, 1.5]$ and compare the following algorithms:

1. *Gradient descent with fixed step size*: Use the algorithm you implemented in Problem 2, trying out alternative values for the step-size. Which value worked the best?

2. *Newton's method (Section 8.3.3)*: The Hessian matrix for this loss is constant $H = \frac{2}{N} X^T X$ (a 2×2 matrix). Instead of the regular gradient updates, use the update scheme $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - H^{-1} \nabla L(\boldsymbol{\theta})$. How many iterations do we need for convergence? Why?

3. *Stochastic gradient descent (SGD; Section 8.5.2)*: Implement SGD that supports computing the gradient only over a random subset of the data samples, taking the subset size as a parameter. Implement also support for adaptive step-size control with the *adaGrad* algorithm that works as follows:

   - For each iteration t, store the gradient $g_t = \nabla L$
   - For each dimension d of the parameter vector, compute $s_t(d) = \sum_{i=1}^{t} g_i(d)^2$
   - The step-size at iteration t for each dimension is given by $\frac{\eta}{\tau + \sqrt{s_t(d)}}$, where you can use $\eta = 1$ and $\tau = 1$ (or some other values if you wish to play around).

   Note that here the step-size is not a scalar but a vector, controlling the steps separately for each dimension of the parameter vector. Try the algorithm so that you always pick just one sample for estimating the gradient. Try it out also with *mini-batch* of 50 randomly chosen samples. How do the results differ?

To understand what the algorithms do you can plot the cost function as a function of the iteration, but since the problem is two-dimensional we can also track how the actual parameter values change. The following python code produces a contour plot of the loss surface (assuming you have implemented the cost function). You can overlay the progress of the algorithm on top of this plot.

```
import numpy as np
import matplotlib.pyplot as plt
I = np.arange(-2.0, 2, 0.05)
J = np.arange(-2.0, 2, 0.05)

Z = np.zeros((len(I),len(J)))

for i in range(len(I)):
    for j in range(len(J)):
        theta = [I[i], J[j]]
        Z[i,j] = cost_function(theta, X, Y)

levels=np.arange(0.1,1.2,0.3)
I1, J1 = np.meshgrid(I, J)
plt.contour(I1, J1, Z.T, levels)

#
# Plot the progress of the algorithm here
#

plt.show()
```

3 Plate diagrams and independence (2 points)

Draw the plate diagrams corresponding to the following two factorizations of a joint distribution:

(a)

$$p(\{x_t\}, \{z_t\}, \boldsymbol{\phi}, \boldsymbol{\pi}) = p(\boldsymbol{\phi}|\phi_0)p(\boldsymbol{\pi}|\pi_0)p(z_0)\prod_{t=1}^{3}\Big[p(x_t|z_t, \boldsymbol{\phi})p(z_t|z_{t-1}, \boldsymbol{\pi})\Big],$$

where x_t are observed.

(b)

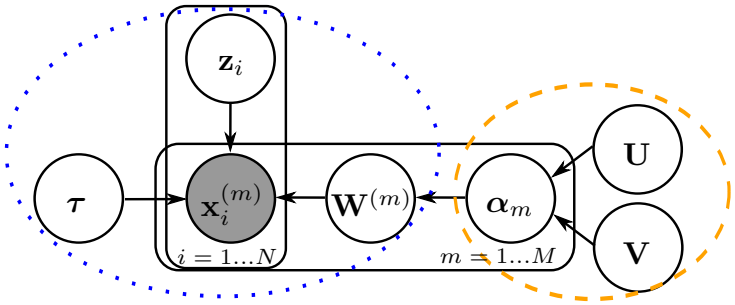
$$p(\{y_n\}, \{\mathbf{x}_n\}, \{z_n\}, \boldsymbol{\pi}, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}, \{\mathbf{w}_k\}) = p(\boldsymbol{\pi})\prod_{k=1}^{K}\left[p(\boldsymbol{\mu}_k)p(\boldsymbol{\Sigma_k})p(\mathbf{w}_k)\prod_{n=1}^{N}\Big(p(z_n|\boldsymbol{\pi})p(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)p(y_n|\mathbf{x}_n, \mathbf{w}_k)\Big)^{\mathbb{I}(z_n=k)}\right],$$

where both $\mathbf{x}_n$ and y_n are observed.

The curly brackets $\{y_n\}$ denote a collection of N random variables y_n, and $\mathbb{I}(\cdot)$ is the indicator function, obtaining value 1 if the argument is true and 0 otherwise.

Now do the opposite: Write down the joint probability density for the variables depicted by the following plate diagram. You can ignore the blue and orange circles; the figure is taken from a publication that used those to help understanding some details of the model.

(c)



Can you name these models? Feel free to guess – the right answers are not required for maximum grade.