SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 5: February 4–February 8, 2019

Exercises consist of both pen&paper and computer assignments. Pen&paper questions are solved at home before exercises, while computer assignments are solved during exercise hours. The computer assignments are marked by `python` and Pen&paper questions by `pen&paper`

1. `pen&paper` *Compute the gradient of the log-loss.*

   In the lectures we defined the *logistic loss function*:

   $$\ell(\mathbf{w}) = \sum_{n=0}^{N-1} \ln(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)), \tag{1}$$

   and computed its gradient $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}}$. Here, $\mathbf{x}_n \in \mathbf{R}^P$ and $y_n \in \{-1, 1\}$ are the inputs and labels for the samples $n = 0, 1, \ldots, N-1$, and $\mathbf{w} \in \mathbf{R}^P$ are the model parameters to be learnt.

   The *L_2-regularized logistic loss* is defined by:

   $$\ell(\mathbf{w}) = \sum_{n=0}^{N-1} \ln(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)) + C \cdot \mathbf{w}^T \mathbf{w}, \tag{2}$$

   with $C \geq 0$ the regularization strength parameter. Compute the gradient of the regularized loss.

   Hint: For finding the gradients of vector functions, check the document at `http://www.kamperh.com/notes/kamper_matrixcalculus13.pdf`

2. `pen&paper` *Manually compute the update step.*

   There are two alternative strategies for using the gradient.

   - **Batch gradient:** Compute the gradient for many input samples and then apply the gradient descent rule once: $\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}}$.
   - **Stochastic gradient:** Compute the gradient for one sample and apply the gradient descent rule after every input: In other words, pretend $N = 1$ in formula (2).

   In the latter case, compute the result after updating $\mathbf{w}$ when its current value is $\mathbf{w} = [2, 1]^T$, and we feed input $\mathbf{x}_n = [-1, 1]^T$ with label $y[n] = -1$. Assume $C = 1$.

3. `python` *Implement gradient descent for log-loss.*

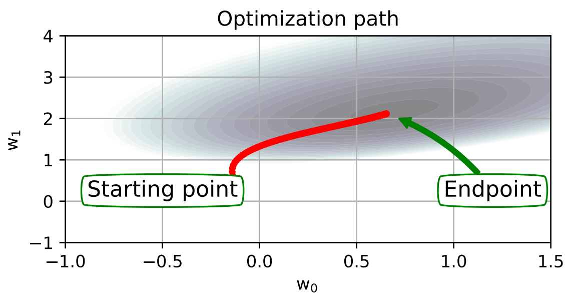
   a) Implement a log-loss minimization algorithm. You may use the template provided by the teaching assistant.

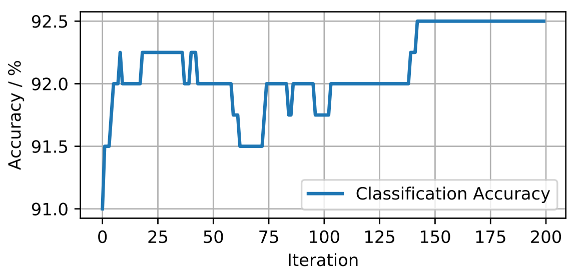
   b) Apply the code for the data downloaded from

      `https://github.com/mahehu/SGN-41007/tree/master/exercises/Ex5/log_loss_data.zip`

      The data is in CSV format. Load `X` and `y` using `numpy.loadtxt`.

c) Plot the path of w over 100 iterations and check the accuracy (see plots below).





4. **python** *Select appropriate hyperparameters for the GTSRB data.*

Last week we trained classifiers for the German Traffic Sign Recognition Benchmark (GTSRB) dataset. It turned out that the SVM was really poor with default arguments, but changing the kernel pushed it to the top. In this exercise, we use brute force to find good hyperparameters for the classifiers (kernel, C, number of trees, etc.).

Consider the following two classifiers

```
clf_list = [LogisticRegression(), SVC()]
clf_name = ['LR', 'SVC']
```

Most important hyperparameters are the *regularization strength* `C` and the penalty type parameter `penalty`, which can have values "`l1`" and "`l2`".

In order to use the same range for the two methods, you need to scale the data to zero mean and unit variance using `sklearn.preprocessing.Normalizer`.

Implement a grid search over these two parameters along the following lines:

```
for clf,name in zip(clf_list, clf_name):
    for C in C_range:
    for penalty in ["l1", "l2"]:
            clf.C = C
            clf.penalty = penalty
            clf.fit(X_train, y_train)
            y_pred = clf.predict(X_test)
            score = accuracy_score(y_test, y_pred)
```

A reasonable range for `C` is $C \in \{10^{-5}, ..., 10^{0}\}$.

5. **`python`** *Select hyperparameters using sklearn.*

   Instead of writing your own for-loop, scikit-learn provides a convenient interface for testing different hyperparameter combinations. Study the documentation of the following two approaches:

   - `model_selection.GridSearchCV`: same as task 4, but with many data splits (we did just one).
   - `model_selection.RandomizedSearchCV`: same as `GridSearchCV`, but chooses parameter combinations at random.

   Implement a search using both two methods for the parameter ranges of question 4.