assignment5_w12_Zimmermann_4165446_Haege_4179267

May 22, 2022

Assignment 5

Due May 23 at 12:00

Please note:

- Read the instructions in the exercise PDF and in this notebook carefully.
- Add your solutions *only* at `YOUR CODE HERE`/`YOUR ANSWER HERE` and remove the corresponding `raise NotImplementedError()`.
- Do not chance the provided code and text, if not stated.
- Do not *add* or *delete* cells.
- Do not `import` additional functionality.
- Before submitting: Please make sure, that your notebook can be executed from top to bottom `Menu -> Kernel -> Restart & Run all`.

1 Exercise 1 (Lagrange multipliers, 2 points)

$$
\begin{aligned}
&\max x + y \\
&\text{s.t. } x^2 + 2y^2 \leq 5 \\
L(x, y, \alpha) &= (x + y) - \alpha(x^2 + 2y^2 - 5) \\
\frac{\partial L}{\partial x} &= 1 - 2\alpha x \stackrel{!}{=} 0 \\
\alpha &= \frac{1}{2x} \\
\frac{\partial L}{\partial y} &= 1 - 4\alpha y \stackrel{!}{=} 0 \\
\alpha &= \frac{1}{4y} \\
\frac{1}{2x} &= \frac{1}{4y} \\
x &= 2y \\
L(y, \alpha) &= (2y + y) - \alpha((2y)^2 + 2y^2 - 5) \\
\frac{\partial L}{\partial \alpha} &= (2y)^2 + 2y^2 - 5 \stackrel{!}{=} 0 \\
4y^2 + 2y^2 &= 5 \\
y^2 &= \frac{5}{6} \\
y &= \sqrt{\frac{5}{6}} \\
x &= 2\sqrt{\frac{5}{6}} = \sqrt{\frac{10}{3}} \\
\alpha &= \frac{1}{2x} \geq 0
\end{aligned}
$$

The constraint is active. Geometrically this can be explained, by thinking about the uncontrained solution of $\max x + y$ and the ellipsoid, that $x^2 + 2y^2 = 5$ describes. We can then easily see, that the solution of the constrained problem lies at a point on said ellipsoid, not inside it.

2 Exercise 2 (Linear and quadratic programs, 3+1+1=5 points)

2.1 a)

2.2 b)

$$
\begin{aligned}
E &= \frac{1}{2} x^T Q x + c^T x \\
\frac{\partial E}{\partial x} &= x^T Q + c \\
\frac{\partial^2 E}{\partial x^2} &= Q
\end{aligned}
$$

When we derive the problem twice, we can see the Q is the Hessian matrix. Therefore we require Q to be positive semi-definite for the problem to be convex.

2.3 c)

This means that, as the solutions for the dual and the primal are equal, it does not matter wether we solve the primal or the dual problem.

3 Exercise 3 (Primal hard margin SVM problem, 3 points)

Any solution of this problem is subject to:

$$Y_i(w^T X_i + b) \geq 1 \; \forall i = 1, \ldots, n$$

If the hyperplane is not in canonical representation, it holds:

$$\min_i |w^T X_i + b| \neq 1$$

And thus, because $Y_i \in -1, 1$:

$$Y_i(w^T X_i + b) > 1 \; \forall i = 1, \ldots, n$$

Then

$$\exists \hat{w} \text{ with } \frac{1}{2}\|\hat{w}\|^2 < \frac{1}{2}\|w\|^2$$

in other words, w does not minimize the objective, and is not a solution.
While

$$Y_i(\hat{w}^T X_i + b) \geq 1 \; \forall i = 1, \ldots, n$$

and

$$\exists i \; Y_i(\hat{w}^T X_i + b) = 1$$

and therefore

$$\min_i |\hat{w}^T X_i + b| = 1$$

in other words, $\hat{w}$ is a valid solution and a canonical Hyperplane.

[1]:
```
import time

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from numpy.testing import assert_equal, assert_almost_equal

from sklearn import preprocessing
from sklearn.datasets import load_breast_cancer
```

```
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression

#  Hide warnings of LinearSVC, LogisticRegression

import warnings
from sklearn.exceptions import ConvergenceWarning
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=ConvergenceWarning)

np.random.seed(42)
```

3.1 a) Know the Dataset

- **Features:** 30 different tissue measurements of the samples (like concavitivity, radius, …)
- **Labels:** malignant/benign {1,0} tells if the sample contains harmful tissue

[2]:
```
dataset = load_breast_cancer()

xs = dataset.data
ys = dataset.target
```

Split `xs, ys` to the training set `xs_train, ys_train` (size 70%) and the test set `xs_test, ys_test` (size 30%)

3.2 b)/c) Optimize the Hyperparameters: Linear SVM vs Logistic Regression

[3]:
```
# center and normalize

scaler = preprocessing.StandardScaler().fit(xs)
xs = scaler.transform(xs)

# split into training and test test
n_train = int(len(xs) * .7)

xs_train, xs_test = xs[:n_train], xs[n_train:]
ys_train, ys_test = ys[:n_train], ys[n_train:]

# evaluate standard SVM
svm_estimator = LinearSVC().fit(xs_train,ys_train)
print('We can do better than ', np.mean(svm_estimator.predict(xs_test) !=␣
 ↪ys_test))
```

```
We can do better than  0.03508771929824561
```

Find good hyperparameters *without* the test set.

[4]:
```
# DO NOT USE xs_test, ys_test here!
# Please make your optimization reproducable (e.g. set random_state, seed, …)
```

```
# LinearSVC and LogisticRegression

#best_params_svm, best_params_lr = {}, {}
folds = 5
Cs=np.logspace(-3,3,100)


def get_best_params(xs_train, ys_train, Cs, folds, rdm_state = 42):
    svc_error = 10
    logreg_error = 10
    svc_C = 0
    logreg_C = 0


    for c in Cs:
        svc_err = []
        logreg_err = []
        for i in list(range(0,folds - 1)):
            # Split the folds in train and validation set
            x_test = xs_train[int(i*(len(xs_train)/folds)):␣
 ↪int((i+1)*(len(xs_train)/folds))]
            x_train = np.append(xs_train[:int(i*len(xs_train)/
 ↪folds)],xs_train[int((i+1)*len(xs_train)/folds):], axis=0)
            y_test = ys_train[int(i * len(ys_train)/folds):␣
 ↪int((i+1)*len(ys_train)/folds)]
            y_train = np.append(ys_train[:int(i*len(ys_train)/folds):
 ↪],ys_train[int((i+1)*len(ys_train)/folds):])

            svc = LinearSVC(random_state=rdm_state, C=c).fit(x_train, y_train)
            logreg=LogisticRegression(random_state=rdm_state,C=c).
 ↪fit(x_train,y_train)

            svc_err.append(1-svc.score(x_test,y_test))
            logreg_err.append(1-logreg.score(x_test,y_test))
        svc_err = np.mean(svc_err)
        logreg_err = np.mean(logreg_err)
        if logreg_err < logreg_error:
            logreg_error = logreg_err
            logreg_C = c
        if svc_err < svc_error:
            svc_error = svc_err
            svc_C = c
    return {'C': svc_C, 'Error': svc_error}, {'C': logreg_C, 'Error':␣
 ↪logreg_error}

best_params_svm, best_params_lr = get_best_params(xs_train, ys_train, Cs, folds)
```

```
print("best params Logistic Regression: ",best_params_lr)
print("best params Linear SVC: ",best_params_svm)
```

```
best params Logistic Regression:  {'C': 0.40370172585965536, 'Error':
0.03136867088607595}
best params Linear SVC:  {'C': 0.003511191734215131, 'Error':
0.028243670886075933}
```

[5]:
```
# if you did not solve the cross validation, use this:
# svm_estimator = LinearSVC(C=0.0035, random_state=42).fit(xs_train, ys_train)
# lr_estimator = LogisticRegression(C=0.4037,random_state=42).fit(xs_train,␣
 ↪ys_train)
# else, use this:
svm_estimator = LinearSVC(random_state=42, C=best_params_svm['C']).
 ↪fit(xs_train, ys_train)
lr_estimator = LogisticRegression(random_state=42,C=best_params_lr['C']).
 ↪fit(xs_train,ys_train)

svc_error = (1-svm_estimator.score(xs_test,ys_test))
logreg_error = (1-lr_estimator.score(xs_test,ys_test))

print('Linear SCV Error: ', svc_error)
print('Logistic Regressen Error: ' ,logreg_error)
```

```
Linear SCV Error:  0.011695906432748537
Logistic Regressen Error:  0.0292397660818714
```

What do you observe and why?

Answer: SVC performs way worse on test set than on validation sets. For all other folds the validation sets are trained as well. Therefore the validation set is not independent of the training set

3.3 d) State concerns

1. **Ethical:** It is an ethical issue to only look at the classification error because it implicitly assumes that false positives and false negatives are equally bad, when in fact it is way worse to not detect if a patient has cancer
2. **Technical/Statistical:** dataset does not have an representative proportion of malignant/benigne samples. In real life the proportion of actually cancerous tissue is way less than benigne tissue.
3. **Any:** The classification does not contain a measure of certainty. Samples close to the decision boundary are treated the same as clear cases