STA 208: Assignment #3

Due on Monday, April 25, 2016

Prof. James Sharpnack MW 12:00 - 2:00 P.M.

Wenhao Wu

Contents

1.

This question is about your final project.

(a) List the members of your group.

Answer: Xingtai Li, Guicheng Wu, Wenhao Wu.

(b) What is the title of your project?

Answer: Where should a taxi driver in New York City pick up passenger?

(c) Give a brief description of what you intend to do.

Answer:

- Objective: Given the current location and time of a taxi driver in NYC, we want to come up with some locations where the driver has better chance of picking up a passenger and will gain more profits.
- Data set: The dataset we have considered is from *data.gov*, including Green Taxi Trip Data from 2013 to 2015. Each taxi in dataset has twenty attributes such as pick up or drop off time and location. There are about seventy thousands driving records for each year.
- Methodology: We want to implement a Bayesian space temporal analysis to model the pattern of pick-up and drop-off location based on our dataset. A regression model is also included to predict the distribution of the driver's benefit. As a result, the driver is recommended to a nearby location where there is a better chance of picking up a passenger and gain more profits.

2.

Consider the following linear classification loss function:

$$l(\boldsymbol{\beta}; y_i, \mathbf{x}_i) = (1 - y_i \mathbf{x}_i^T \boldsymbol{\beta})_+^2$$

where $z_+ = \max\{0, z\}$. Consider the modified support vector machine program,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} l(\boldsymbol{\beta}; y_i, \mathbf{x}_i) + \lambda \|\boldsymbol{\beta}\|_2^2. \tag{1}$$

(a) Compute the gradient of the objective in (1) and verify that it is continuous.

Answer: Note that $l(\boldsymbol{\beta}; y_i, \mathbf{x}_i) = f(y_i \mathbf{x}_i^T \boldsymbol{\beta})$, where $f(x) = (1-x)_+^2$. Since

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \begin{cases} -2(1-x) & x < 1 \\ 0 & x \geq 1 \end{cases} \tag{2}$$

therefore $f(x)$ is smooth. Consequently, we have

$$\begin{aligned} \frac{\mathrm{d}l}{\mathrm{d}\boldsymbol{\beta}} &= f'(y_i \mathbf{x}_i^T \boldsymbol{\beta}) y_i \mathbf{x}_i \\ &= \begin{cases} -2(1 - y_i \mathbf{x}_i^T \boldsymbol{\beta}) y_i \mathbf{x}_i & y_i \mathbf{x}_i^T \boldsymbol{\beta} < 1 \\ 0 & y_i \mathbf{x}_i^T \boldsymbol{\beta} \geq 1 \end{cases} \end{aligned} \tag{3}$$

which is also continuous.

continued on next page...

Algorithm 1 Sequential Optimization and Grouping of MBMS based on Weighted Sum Efficiency

1: Initialize $\boldsymbol{\beta} = \boldsymbol{\beta}_0$. Define the objective function $l(\boldsymbol{\beta}) = (1/n)\sum_{i=1}^{n} l(\boldsymbol{\beta}; y_i, \mathbf{x}_i) + \lambda\|\boldsymbol{\beta}\|_2^2$.
2: **repeat**
3: Evaluate the gradient descent direction $\Delta\boldsymbol{\beta} = -\sum_{i=1}^{n}(1/n)dl(\boldsymbol{\beta}; y_i, \mathbf{x}_i)/d\boldsymbol{\beta} - 2\lambda\boldsymbol{\beta}$
4: Choose a step with exact line search, i.e. $\hat{t} = \arg\min_{t\geq 0} l(\boldsymbol{\beta} + t\Delta\boldsymbol{\beta})$
5: Update $\boldsymbol{\beta} := \boldsymbol{\beta} + \hat{t}\Delta\boldsymbol{\beta}$
6: **until** Stopping criterion is satisfied.

(b) Write the pseudocode for gradient descent with exact line search for this gradient (you do not need to write out the bisection algorithm for exact line search, only state what exact line search is.).

Answer:

(c) Suppose that we make a transformation $\mathbf{z}_i = \Phi(\mathbf{x}_i)$ and we use this as the design matrix. Rewrite the minimization for (1) with the generalized kernel trick (Hint: make the substitution $\boldsymbol{\beta} = \mathbf{Z}^T\boldsymbol{\gamma}$ and write it as a function of $\boldsymbol{\gamma}$ and define $\mathbf{K} = \mathbf{Z}\mathbf{Z}^T$.).

Answer: Substitute $\mathbf{x}_i$ with $\mathbf{z}_i$ and $\boldsymbol{\beta} = \mathbf{Z}^T\boldsymbol{\gamma}$, we have

$$l(\boldsymbol{\gamma}; y_i, \mathbf{z}_i) = (1 - y_i\mathbf{z}_i^T\boldsymbol{\beta})_+^2 = (1 - y_i\mathbf{z}_i^T\mathbf{Z}^T\boldsymbol{\gamma})_+^2 = (1 - y_i\mathbf{k}_i^T\boldsymbol{\gamma})_+^2 \tag{4}$$

where $\mathbf{k}_i$ is the i-th column of $\mathbf{K}$. On the other hand,

$$\|\boldsymbol{\gamma}\|_2^2 = \boldsymbol{\gamma}^T\mathbf{Z}\mathbf{Z}^T\boldsymbol{\gamma} = \boldsymbol{\gamma}^T\mathbf{K}\boldsymbol{\gamma} \tag{5}$$

Consequently, (1) can be rewritten as

$$\min_{\boldsymbol{\gamma}\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n}(1 - y_i\mathbf{k}_i^T\boldsymbol{\gamma})_+^2 + \lambda\boldsymbol{\gamma}^T\mathbf{K}\boldsymbol{\gamma}. \tag{6}$$

3.

Consider the following way in which we encode logical statements. For $\mathbf{x} \in \{0,1\}^p$, we will encode the "and" and "or" statements: ($x_i > 0$ and $x_j > 0$) is the same as ($x_i x_j > 0$); ($x_i > 0$ or $x_j > 0$) is the same as ($x_i + x_j > 0$).

(a) Suppose that $y_i = 1$ if and only if ($x_{i,1} = 1$ and $x_{i,2} = 1$) or $x_{i,3} = 1$ and $y_i = 0$ otherwise. Write a polynomial function $f(\mathbf{x}_i)$ such that $f(\mathbf{x}_i) > 0$ if and only if $y_i = 1$.

Answer:

$$f(\mathbf{x}_i) = x_{i,1}x_{i,2} + x_{i,3} \tag{7}$$

(b) Define an embedding of $\mathbf{x}$ such that the dataset has a separating hyperplane, and write an equation for the separating hyperplane as a function of $\mathbf{z}_i = \Psi(\mathbf{x}_i)$. Specifically, consider any dataset that is consistent with this rule ($y_i = 1$ iff this rule holds) then your separating hyperplane should separate the positive from negative examples.

Answer: The embedding is defined as

$$\mathbf{z}_i = \Phi(\mathbf{x}_i) = \begin{bmatrix} x_{i,1}x_{i,2} \\ x_{i,3} \end{bmatrix} \tag{8}$$

continued on next page...

and a separating hyperplane for $\mathbf{z}_i$ is simply $z_{i,1} + z_{i,2} = 0.5$

(c) From this embedding construct the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$.

Answer: The kernel is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = x_{i,1}x_{i,2}x_{j,1}x_{j,2} + x_{i,3}x_{j,3} \tag{9}$$

(d) Suppose that $y_i = 1$ if and only if ($x_{i,1} = 1$ or $x_{i,2} = 1$) and ($x_{i,3} = 1$ and $x_{i,4} = 1$) and $y_i = 0$ otherwise. Write a polynomial function $f(\mathbf{x}_i)$ such that $f(\mathbf{x}_i) > 0$ if and only if $y_i = 1$.

Answer:

$$f(\mathbf{x}_i) = (x_{i,1} + x_{i,2})x_{i,3}x_{i,4} = x_{i,1}x_{i,3}x_{i,4} + x_{i,2}x_{i,3}x_{i,4} \tag{10}$$

(e) Define an embedding of $\mathbf{x}$ such that this new dataset has a separating hyperplane, and write an equation for the separating hyperplane as a function of $\mathbf{z}_i = \Phi(\mathbf{x}_i)$, as you did in (b).

Answer: The embedding is defined as

$$\mathbf{z}_i = \Phi(\mathbf{x}_i) = \begin{bmatrix} x_{i,1}x_{i,3}x_{i,4} \\ x_{i,2}x_{i,3}x_{i,4} \end{bmatrix} \tag{11}$$

and a separating hyperplane for $\mathbf{z}_i$ is simply $z_{i,1} + z_{i,2} = 0.5$

(f) From this embedding construct the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$.

Answer: The kernel is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) = x_{i,1}x_{i,3}x_{i,4}x_{j,1}x_{j,3}x_{j,4} + x_{i,2}x_{i,3}x_{i,4}x_{j,2}x_{j,3}x_{j,4} \tag{12}$$

(g) For simplicity let $p = 3$ (if you did it more generally then great!). Define the degree of a logical statement as the maximum number of variables involved in any monomial after we apply the substitutions defined above. Give an embedding and kernel that separates a dataset where $y_i = 1$ if and only if some logical statements of degree 3 is true. For example, a possible statement of degree 3 is $x_1 = 1$ and $x_2 = 1$ and $x_3 = 0$, which would be encoded as the polynomial $x_1x_2(1 - x_3) = x_1x_2 - x_1x_2x_3$, so $y_i = 1\{x_1x_2x_1x_2x_3 > 0\}$ (i.e. the binary response variable is 1 iff the statement is true). You must provide an embedding that separates the positive and negative cases for any such logical statement (of degree 3).

Answer: Suppose that $y_i = 1$ if and only if $x_{i,1} = 0$ and ($x_{i,2} = 1$ or $x_{i,3} = 0$) and $x_{i,4} = 1$ and $y_i = 0$ otherwise. Such a logical statement can be encoded as plynomial

$$\begin{aligned} f(\mathbf{x}_i) &= (1 - x_{i,1})[x_{i,2} + (1 - x_{i,3})]x_{i,4} \\ &= (1 - x_{i,1})[x_{i,2}x_{i,4} + x_{i,4} - x_{i,3}x_{i,4}] \\ &= x_{i,2}x_{i,4} + x_{i,4} - x_{i,3}x_{i,4} - x_{i,1}x_{i,2}x_{i,4} - x_{i,1}x_{i,4} + x_{i,1}x_{i,3}x_{i,4} \end{aligned} \tag{13}$$

Corespondingly, the embedding is defined as

$$\mathbf{z}_i = \Phi(\mathbf{x}_i) = \begin{bmatrix} x_{i,2}x_{i,4} \\ x_{i,4} \\ x_{i,3}x_{i,4} \\ x_{i,1}x_{i,2}x_{i,4} \\ x_{i,1}x_{i,4} \\ x_{i,1}x_{i,3}x_{i,4} \end{bmatrix} \tag{14}$$

continued on next page...

Table 1: Comparison of different classifier-design matrix.

Classifier-Design Matrix	$\hat{C}$	Training score	Test score
SVC-Raw	1	0.9838	0.9882
SVC-tf-idf	10	0.9872	0.9904
LogReg-Raw	10^3	0.9862	0.9865
LogReg-tf-idf	10^6	0.9874	0.9910

and a separating hyperplane for $\mathbf{z}_i$ is simply $z_{i,1} + z_{i,2} - z_{i,3} - z_{i,4} - z_{i,5} + z_{i,6} = 0.5$. The kernel is defined as

$$\begin{aligned} k(\mathbf{x}_i, \mathbf{x}_j) &= \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \\ &= x_{i,2}x_{i,4}x_{j,2}x_{j,4} + x_{i,4}x_{j,4} + x_{i,3}x_{i,4}x_{j,3}x_{j,4} + x_{i,1}x_{i,2}x_{i,4}x_{j,1}x_{j,2}x_{j,4} \\ &+ x_{i,1}x_{i,4}x_{j,1}x_{j,4} + x_{i,1}x_{i,3}x_{i,4}x_{j,1}x_{j,3}x_{j,4}. \end{aligned} \tag{15}$$

4.

Download the training and test datasets from the website for the Reuters data. The first column is the response variable and the rest are the counts of each term in the dictionary for each document.

(a) Tune a linear SVM with the raw counts as the design matrix and tune with 5-fold cross validation

(b) Apply the tf-idf transformation and then tune a linear SVM with this as the design matrix, again with 5-fold cross validation.

(c) Tune logistic regression to these and tune it with 5-fold cross validation.

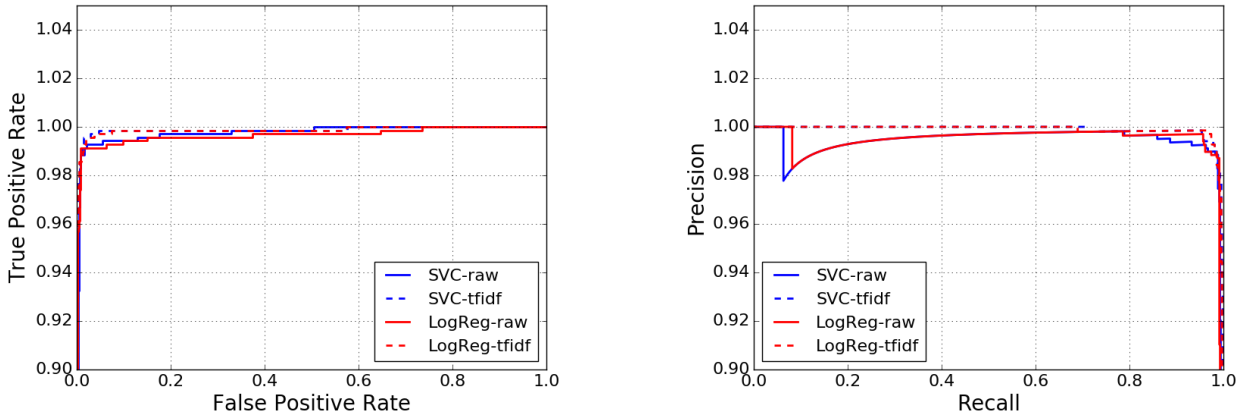
(d) Construct scores for each of these methods for the test set (this should be $\mathbf{x}_i^T \boldsymbol{\beta}$), and construct the ROC and PR curves for them.

Answer: We select the range of tuning parameters to be $\log C = \text{range}(-1, 7)$ (where larger C means less regularization) for both SVC and logistic regression and select the best one with a 5-fold cross validation. The training and testing scores of the 4 classifiers are shown in Table. 1. The ROC and PR curves are shown in Figure 1. It appears that all the classifiers perform pretty well and training with design matrix after tf-idf transformation is advantageous.
The source code for this simulation is as follows:

```
import pandas as pd
import numpy as np
import timeit
import sys

from sklearn import linear_model
from sklearn.svm import SVC
from sklearn.cross_validation import KFold
from sklearn.grid_search import GridSearchCV
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import precision_recall_curve, roc_curve
```

continued on next page...



(a) ROC (b) PR

Figure 1: Model performances.

```
from scipy.sparse import csr_matrix

# Import the training file
df_train = pd.read_csv('spmats/train_Xsp.csv', header=None)
df_test = pd.read_csv('spmats/test_Xsp.csv', header=None)
d = max(df_train.values[:, 1].max(), df_test.values[:, 1].max()) + 1
n_train, n_test = df_train.values[:, 0].max() + 1, df_test.values[:, 0].
    ↪ max() + 1

X_train_raw = csr_matrix((df_train.values[:, 2], (df_train.values[:, 0],
    ↪ df_train.values[:, 1])), shape=(n_train, d), dtype="float64")
y_train = pd.read_csv('spmats/train_y.csv', header=None).values[:, 0]
n = X_train_raw.shape[0]

# Evaluate the tf-idf transformation on the counts
transformer = TfidfTransformer()
X_train_tfidf = transformer.fit_transform(X_train_raw)

# Import the testing file
X_test_raw = csr_matrix((df_test.values[:, 2], (df_test.values[:, 0],
    ↪ df_test.values[:, 1])), shape=(n_test, d), dtype="float64")
y_test = pd.read_csv('spmats/test_y.csv', header=None).values[:, 0]

X_test_tfidf = transformer.fit_transform(X_test_raw)

# Construct classifier objects for cross validation
cv = KFold(n, n_folds=5)
C_range = np.logspace(-1, 7, 9) # Range of Parameter C, larger C means
    ↪ less regularization
```

continued on next page...

```
classifier_svc = SVC(kernel='linear') # The linear svm classifier on the
    ↪ raw counts
classifier_logreg = linear_model.LogisticRegression() # The logistic
    ↪ classifier

# Tune the classifiers and evaluate their performances
case = 0

precision = dict()
recall = dict()
fpr = dict()
tpr = dict()

for classifier in [classifier_svc, classifier_logreg]:
    for X_train, X_test in [(X_train_raw, X_test_raw), (X_train_tfidf,
        ↪ X_test_tfidf)]:
        start_time = timeit.default_timer()

        print("Tuning␣case␣{0}...".format(case))

        grid = GridSearchCV(classifier, param_grid=dict(C=C_range), cv=cv,
            ↪  verbose=1, n_jobs=4)
        grid.fit(X_train, y_train)

        print("␣-␣The␣best␣parameters␣are␣{0}␣with␣a␣score␣of␣{1}".format(
            ↪ grid.best_params_, grid.best_score_))
        print("␣-␣The␣score␣on␣the␣test␣set␣is␣{0}".format(grid.score(
            ↪ X_test, y_test)))

        y_score = grid.best_estimator_.decision_function(X_test)
        precision[case], recall[case], _ = precision_recall_curve(y_test,
            ↪ y_score)
        fpr[case], tpr[case], _ = roc_curve(y_test, y_score)

        print("␣-␣Elapsed␣time␣is␣{0}".format(timeit.default_timer() -
            ↪ start_time))
        sys.stdout.flush()

        case = case + 1

# Visualization
import matplotlib.pyplot as plt
import matplotlib as mpl
\%matplotlib qt

axis_font = {'size':'20'}
mpl.rcParams['xtick.labelsize'] = 16
mpl.rcParams['ytick.labelsize'] = 16
```

continued on next page...

```
labels = ['SVC-raw', 'SVC-tfidf', 'LogReg-raw', 'LogReg-tfidf']
line_specs = ['b-', 'b--', 'r-', 'r--']

# The PR curve
plt.figure()
for case in range(4):
    plt.plot(recall[case], precision[case], line_specs[case], label=labels
        ↪ [case], linewidth=2)

plt.xlim([0.0, 1.0])
plt.ylim([0.9, 1.05])
plt.grid()
plt.xlabel('Recall',  **axis_font)
plt.ylabel('Precision',  **axis_font)
plt.legend(loc="lower␣left", prop={'size':16})

# The ROC curve
plt.figure()
for case in range(4):
    plt.plot(fpr[case], tpr[case], line_specs[case], label=labels[case],
        ↪ linewidth=2)

#plt.plot([0, 1], [0, 1], 'k--', linewidth=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.9, 1.05])
plt.grid()
plt.xlabel('False␣Positive␣Rate',  **axis_font)
plt.ylabel('True␣Positive␣Rate',  **axis_font)
plt.legend(loc="lower␣right", prop={'size':16})
```