CMPSC 448: Machine Learning and AI

Homework 3 (Friday April 19, 11:59PM)

Instruction

This HW includes both theory and implementation problems. Please note,

- Your code must work with Python 3.5+ (you may install the Anaconda distribution of Python)
- You need to submit a report in PDF including all written deliverable and plots via Gradescope, and all implementation codes via Canvas so one could regenerate your results.
- For non-linear classifier problem, you need to submit a Jupyter notebook for each algorithm and all complementary python codes so one could **reproduce** your results. Submit your code in `Problem5*.py` and replace $*$ witn the name of classifier.

The submission for this homework should be a single PDF file with **solutions of theory problems, figures, and any text explaining your results of programming problems** via Gradescope and a single 'zip' file containing all of the relevant code via Canvas.

SVM & Kernel Methods

Problem 1 (10 points) In this problem we would like to compare the solutions of hard and soft SVMs on a linearly seperable dataset.

Let $n > 1$ be a fixed number. Is it true that there exists $C > 0$ such that for every sample $\mathcal{S}$ of n training examples with binary label, which are linearly separable, the hard-SVM and the soft-SVM (with particular choice of parameter C) solutions will return exactly the same weight vector $\boldsymbol{w}_*$. Justify your answer. (Hint: consider $n = 2, d = 1$ and $S = \{(x_1, y_1), (x_2, y_2)\}$. Let $a > 0$ and consider $x_1 = a, y_1 = 1, x_2 = -a, y_2 = -1$. Derive the optimal solution for hard and soft SVM and compare the results.)

Problem 2 (10 points) In this problem you are asked to find the explicit mapping function $\Phi(\cdot)$ for the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{z}\|^2 / 2\gamma^2\right)$ by showing that it can be expressed as the inner product of an infinite-dimensional feature space by a mapping Φ.

a) Assume $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^1$ (only one feature per training example). Use the Taylor expansion of the kernel function κ (to be precise, the $\exp(\cdot)$ function) and derive the explicit mapping Φ the kernel correspond to.

b) Answer the above question in general case where $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^d$. Assume $\|\boldsymbol{x}\|_2 = \|\boldsymbol{z}\|_2 = 1$.

Problem 3 (20 points) In this problem we aim at utilizing the kerenl trick in Ridge regression and propose its kernalized version. Recall the Ridge regression training objective function:

$$f(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

for $\lambda > 0$.
a) Show that for $\boldsymbol{w}$ to be a minimizer of $f(\boldsymbol{w})$, we must have $\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w} + \lambda\boldsymbol{I}\boldsymbol{w} = \boldsymbol{X}^\top\boldsymbol{y}$, where $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ is the data matrix with n samples each with d features, and $\boldsymbol{I}$ is identity matrix (please check lectures for more details). Show that the minimizer of $f(\boldsymbol{w})$ is $\boldsymbol{w} = \left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$. Justify that the matrix $\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}$ is invertible, for $\lambda > 0$. (Hint: use SVD decomposition of data matrix $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and show all the eigenvalues of $\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}$ are larger than zero).

b) Rewrite $\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w} + \lambda\boldsymbol{I}\boldsymbol{w} = \boldsymbol{X}^\top\boldsymbol{y}$ as $\boldsymbol{w} = \frac{1}{\lambda}\left(\boldsymbol{X}^\top\boldsymbol{y} - \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w}\right)$. Based on this, show that we can write $\boldsymbol{w} = \boldsymbol{X}^\top\boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^n$, and give an expression for $\boldsymbol{\alpha}$.

c) Based on the fact that $\boldsymbol{w} = \boldsymbol{X}^\top\boldsymbol{\alpha}$, explain why we say $\boldsymbol{w}$ is "in the span of the data."

d) Show that $\boldsymbol{\alpha} = \left(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}\boldsymbol{y}$. Note that $\boldsymbol{X}\boldsymbol{X}^\top$ is the $n \times n$ Gram (kernel) matrix for the standard vector dot product. (Hint: Replace $\boldsymbol{w}$ by $\boldsymbol{X}^\top\boldsymbol{\alpha}$ in the expression for $\boldsymbol{\alpha}$, and then solve for $\boldsymbol{\alpha}$.)

e) Give a kernelized expression for the $\boldsymbol{X}\boldsymbol{w}$, the predicted values on the training points. (Hint: Replace $\boldsymbol{w}$ by $\boldsymbol{X}^\top\boldsymbol{\alpha}$ and α by its expression in terms of the kernel matrix $\boldsymbol{X}\boldsymbol{X}^\top$.)

f) Give an expression for the prediction $\boldsymbol{w}_*^\top\boldsymbol{x}$ for a test sample $\boldsymbol{x}$, not in the training set, where $\boldsymbol{w}_*$ is the optimal solution. The expression should only involve $\boldsymbol{x}$ via inner products training data samples $\boldsymbol{x}_i, i = 1, \ldots, n$.

g) Based on (f), propose a kernalized version of the Ridge regression.

Boosting

Problem 4 (15 points) Consider the AdaBoost algorithm we discussed in the class [1]. AdaBoost is an example of ensemble classifiers where the weights in next round are decided based on the training error of the weak classifier learned on the current weighted training set. We wish to run the AdaBoost on the dataset provided in Table 1.

Instance	**Color**	**Size**	**Shape**	**Edible?**
D1	Yellow	Small	Round	Yes
D2	Yellow	Small	Round	No
D3	Green	Small	Irregular	Yes
D4	Green	Large	Irregular	No
D5	Yellow	Large	Round	Yes
D6	Yellow	Small	Round	Yes
D7	Yellow	Small	Round	Yes
D8	Yellow	Small	Round	Yes
D9	Green	Small	Round	No
D10	Yellow	Large	Round	No
D11	Yellow	Large	Round	Yes
D12	Yellow	Large	Round	No
D13	Yellow	Large	Round	No
D14	Yellow	Large	Round	No
D15	Yellow	Small	Irregular	Yes
D16	Yellow	Large	Irregular	Yes

Table 1: Mushroom data with 16 instances, three categorical features, and binary labels.

a) Assume we choose the following decision stump f_1 (a shallow tree with a single decision node), as the first predictor (i.e., when training instances are weighted uniformly):

```
if(Color is Yellow):
      predict  Eadible = Yes
else:
      predict Eadible = No
```

What would be the weight of f_1 in final ensemble classifier (i.e., α_1 in $f(\boldsymbol{x}) = \sum_{i=1}^{K} \alpha_i f_i(\boldsymbol{x})$)?

b) After computing f_1, we proceed to next round of AdaBoost. We begin by recomputing data weights depending on the error of f_1 and whether a point was (mis)classified by f_1. What is the weight of each instance in second boosting iteration, i.e., after the points have been re-weighted? Please note that the weights across the training set are to be uniformly initialized.

c) In AdaBoost, would you stop the iteration if the error rate of the current weak classifier on the weighted training data is 0?

[1] Please read reading assignement from textbook and (optional) the Introduction chapter from Boosting book for detailed explanation https://bit.ly/2HvKObl

Experiment with non-linear classifiers

Problem 5 (40 points) For this problem, you will need to learn to use software libraries for at least two of the following non-linear classifier types:

- Boosted Decision Trees (i.e., boosting with decision trees as weak learner)
- Random Forests
- Support Vector Machines with Gaussian Kernel

All of these are available in `scikit-learn`, although you may also use other external libraries (e.g., XGBoost [2] for boosted decision trees and LibSVM for SVMs). You are welcome to implement learning algorithms for these classifiers yourself, but this is neither required nor recommended.

Pick two different types of non-linear classifiers from above for classification of Adult dataset. You can the download the data from a9a in libSVM data repository. The a9a data set comes with two files: the training data file `a9a` with 32,561 samples each with 123 features, and `a9a.t` with 16,281 test samples. Note that a9a data is in LibSVM format. In this format, each line takes the form `<label> <feature-id>:<feature-value> <feature-id>:<feature-value> ....`. This format is especially suitable for sparse datasets. Note that `scikit-learn` includes utility functions (e.g., `load_svmlight_file` in example code below) for loading datasets in the LibSVM format.

For each of learning algorithms, you will need to set various hyperparameters (e.g., the type of kernel and regularization parameter for SVM, tree method, max depth, number of weak classifiers, etc for XGBoost, number of estimators and min impurity decrease for Random Forests). Often there are defaults that make a good starting point, but you may need to adjust at least some of them to get good performance. Use hold-out validation or K-fold cross-validation to do this (`scikit-learn` has nice features to accomplish this, e.g., you may use `train_test_split` to split data into train and test data and `sklearn.model_selection` for K-fold cross validation). Do **not** make any hyperparameter choices (or any other similar choices) based on the test set! You should only compute the test error rates after you have settled on hyperparameter settings and trained your two final classifiers.

What to submit (in PDF file and Jupyter/python codes):

1. Names of the two classifiers you opt to learn and a brief description of each algorithm and how it works.
2. Description of your training methodology, with enough details so that another machine learning enthusiast can reproduce the your results. You need to submit all the codes (python and Jupyter notebooks) to reproduce your code. Please use prefix `Problem5*.py` where you need to replace '*' with the name of non-linear classifier for your coding files.
3. The list of hyperparameters and breif description of each hyperparameter you tuned in training, their default values, and the final hyperparameter settings you use to get the best result.

[2] A simple blog post on how to use XGBoost please check this.

4. Training error rates, hold-out or cross-validation error rates, and test error rates for your two final classifiers. You are also encouraged to report other settings you tried with the accuracy it achieved (please make a table with a column with each hyperparamter and accuracy of configuration of parameters.

5. Please do your best to obtain the best achievable accuracy for each classifier on given dataset.

   Note: The amount of effort you put on tuning the parameters will be determined based on the discrepancy between the accuracy you get and the best achievable accuracy on a9a data for each algorithm. You may wanna try a simple algorithm first such as nearest neighbour to get a sense of base accuracy.

Parameters to be tuned for XGBoost:

1. `n_estimators`
2. `max_depth`
3. `lambda`
4. `learning_rate`
5. `missing`
6. `objective`

Parameters to be tuned for SVM:

1. `kernel_type`
2. `gamma`
3. `C`

Parameters to be tuned for Random Forests:

1. `n_estimators`
2. `bootstrap`
3. `max_depth`
4. `min_impurity_decrease`
5. `min_samples_leaf`

Example code to use XGBoost

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_svmlight_file

from xgboost import XGBClassifier


# load data in LibSVM sparse data format

X, y = load_svmlight_file("a9a")
```

```
# split data into train and test sets
seed = 6
test_size = 0.4
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=seed)

# fit model on training data
# for simplicity we fit based on default values of hyperparameters

model = XGBClassifier()
model.fit(X_train, y_train)

# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```