Homework 6, Due 11:59 p.m., June 2

MPCS 53111 Machine Learning
University of Chicago

For all questions in which you obtain an empirical result, such as a plot, briefly discuss the result: Is the result as expected? Does the result reveal some interesting aspect of the data or the algorithm.

Practice problems, do not submit

P-1 Read §9.5 in ISLR for a discussion on the relationship between Logistic Regression and SVMs and summarize it.

Graded problems, submit

1. When the data for a SVM is not linearly separable, or we are willing to tolerate a few points inside the margin, a *soft margin* classifier is used. In this the optimization function has a *regularization term* as shown below:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \max\left\{0, 1 - y^{(i)}(\mathbf{w}\cdot\mathbf{x} + b)\right\}$$

   (a) Describe in a few words how this change makes sense. (Hint: See §12.3.3, "Mining of Massive Datasets.")

   (b) `sklearn` uses an apparently different formulation. Explain why the two formulations are equivalent.

   (c) The loss corresponding to SVMs, shown above, is called the *hinge loss*. Specfically, given example $(\mathbf{x}, y)$, in which $y \in \{-1, 1\}$, let $f(\mathbf{x})$ denote the "score" of the model. Then the hinge loss for the example is $\max\{0, 1 - yf(\mathbf{x})\}$. In SVMs, $f(\mathbf{x})$ is given by $\mathbf{w}\cdot\mathbf{x} + b$. Suppose, however, that you were allowed to choose any $f(\mathbf{x})$ to minimize the total expected hinge loss. In particular, assume that for every $\mathbf{x}$ in the universe of examples, $P(Y = 1|\mathbf{x})$ and $P(Y = -1|\mathbf{x})$ are known. Explain how would you choose $f(\mathbf{x})$ to minimize the expected hinge loss over all examples in the universe? (The answer is given in Table 12.1, page 427, in ESL. Please give the derivation. Hint: Use calculus to minimize the expected loss.

Since the loss function is not differentiable w.r.t. $f(\mathbf{x})$ at all points, divide the possible values of $f(\mathbf{x})$ into three cases[1].)

2. I have a classification problem for SVMs with inputs x_1, x_2. I believe the data is linearly separable in the space of the following features: $x_i^3, x_i^2 x_j, x_i^2, x_i x_j, x_i$, in which i and j are distinct indices from $\{1, 2\}$. What feature vector should I use, if I want to use the kernel trick? What is the corresponding kernel function?

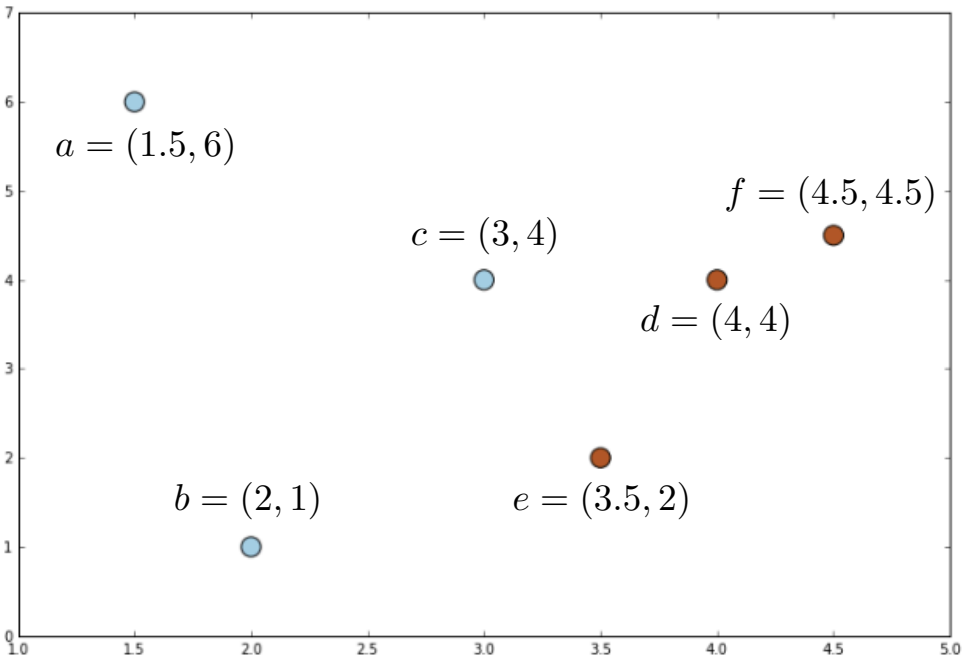

Figure 1: SVM classification problem.

3. In this problem you will get a feel for maximum margin separators. Refer to the dataset shown in Figure 1

   (a) Make your best guess on the optimal separator and determine the distance of the closest example to it. (There is no easy way to

[1] As given in Table 12.1 in ESL, the binary cross entropy loss for the same situation is $\log[1+\exp(-yf(\mathbf{x}))]$, and to minimize the expected binary cross entropy loss, $f(\mathbf{x})$ should be the log of the odds ratio or $\log[P(Y=1|\mathbf{x})/P(Y=-1|\mathbf{x})]$. For both kinds of loss, the final answers are intuitive.

manually determine the maximum margin separator, so correctness is not required.)

(b) Use `sklearn.svm.SVC` to find the maximum margin separator, and plot the corresponding separator and the points. (Use a linear kernel, and $C \geq 100$ to prevent a soft margin.) What is the distance of the closest example to the separator? (Compute this distance for this and part 3c, but not for parts 3d and 3e. See below.)

(c) Repeat 3a and 3b but with point e at $(5, 2)$.

(d) Repeat 3b but switch the class of point f. Plot the soft margin separators (see 1) for $C \in \{0.1, 1, 10\}$.

(e) Repeat 3d but with the kernels *polynomial* and *rbf*. Set C to a value at least 100 again. These kernels use special parameters; you may have to change them to correctly classify all points. (For an intuitive explanation on why the rbf kernel works, see pg 16, Andrew Ng's notes on SVMs.)

4. The accompanying starter code implements an `sklearn`-style regressor class `KNNRegressor` for k-nearest-neighbor linear regression. It also generates data using the following underlying model—

   - $x \sim \text{Uniform}(0, 100)$.
   - $y(x) = 3 + 4x - 0.05x^2 + \epsilon$, in which $\epsilon \sim \text{Uniform}(-25, 25)$.

   Finally it plots the data points, the underlying function without the noise ($y(x) - \epsilon$), and the hypothesis function for k-nearest-neighbor linear regression.

   (a) Similarly, write the class `LWRegressor` to implement locally weighted regression in which the kernel is the rbf_kernel that takes a user specified γ parameter.

   (b) Create a combined plot that includes (i) data points, (ii) the underlying function without the noise, (iii) the hypothesis function for k-nearest-neighbor linear regression with $k = 5$, and (iv) the hypothesis function for locally weighted regression with $\gamma = 1/40$.

(c) Using crossvalidation on the above dataset, determine the best value of (i) k for k-nearest-neighbor linear regression, and (ii) γ for locally weighted regression, and clearly state those values. Which model do you recommend using for this artificial dataset?

5. Consider an ensemble learning classification algorithm that uses simple majority voting among K learned hypotheses. Suppose each hypothesis has error ϵ, but the errors made by each hypothesis are not necessarily independent of the others. Prove or disprove that the error of the ensemble is never worse than ϵ.

6. Recreate Figure 8.8, page 318, in ISLR. For this download the Heart dataset available at the ISLR website. It contains data from 303 patients, including 13 predictors such as age, sex, presence of chest pain, etc., and a binary target indicating presence of heart disease.

   For this please use the `BaggingClassifier` and `RandomForestClassifier` from `sklearn`.

7. Ex 10.4, page 385, from ESL. You should implement AdaBoost in Python on your own as given in Algorithm 10.1, page 339, in ESL. You may, however, use `DecisionTreeClassifier` and other helper functions from `sklearn`.

Submit, on Gradescope, your code, plots, discussions of your results, and solutions to the theory problems as `hw6.py` and `hw6.pdf`.