Introduction to Machine Learning, Spring 2016

Problem Set 2: Support vector machines

Due: Monday, February 15, 2016 at 10pm (uploaded to NYU Classes.)

Your submission should include a PDF file called "solutions.pdf" with your answers to the below questions (including plots), and all of the code that you write.

Important: *See problem set policy on the course web site.*

---

1. Consider a (hard margin) support vector machine and the following training data from two classes:

$$\begin{array}{lllll} +1: & (2,2) & (4,4) & (4,0) \\ -1: & (0,0) & (2,0) & (0,2) \end{array}$$

   (a) Plot these six training points, and construct by inspection the weight vector for the optimal hyperplane. In your solution, specify the hyperplane in terms of $\vec{w}$ and b such that $w_1 x_1 + w_2 x_2 + b = 0$. Calculate what the margin is (i.e., 2γ, where γ is the distance from the hyperplane to its closest data point), showing all of your work.

   (b) What are the support vectors? Explain why.

2. Show that, irrespective of the dimensionality of the data space, a data set consisting of just two data points (call them $\vec{x}_1$ and $\vec{x}_2$), one from each class, is sufficient to determine the maximum-margin hyperplane. Fully explain your answer, including *giving an explicit formula* for the solution to the hard margin SVM (i.e., $\vec{w}$) as a function of $\vec{x}_1$ and $\vec{x}_2$.

3. **Instructions.** You may use the programming language of your choice (we recommend Python, and using `matplotlib` for plotting). However, you are **not** permitted to use or reference any machine learning code or packages not written by yourself.

   In this question, you will implement the Pegasos algorithm [1] to optimize the SVM objective using stochastic sub-gradient descent and revisit the spam data. To be consistent with the paper, this question uses the following form of the SVM objective: $\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\max\{0, 1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)\}$. This can be seen to be equivalent to the optimization problem given in class and lab, $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\max\{0, 1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)\}$, when $C = 1/m\lambda$ (after which they only differ by a multiplicative constant).

Initialize: Choose $\boldsymbol{w}_1 = 0, t = 0$.
1. For iter $= 1, 2, \cdots, 20$
2. For $j = 1, 2, \cdots, |\text{data}|$
3. $t = t + 1; \quad \eta_t = \frac{1}{t\lambda}$
5. If $y_j(\boldsymbol{w}_t \cdot \boldsymbol{x}_j) < 1$
6. $\boldsymbol{w}_{t+1} = (1 - \eta_t\lambda)\boldsymbol{w}_t + \eta_t y_j \boldsymbol{x}_j$
7. Else
8. $\boldsymbol{w}_{t+1} = (1 - \eta_t\lambda)\boldsymbol{w}_t$
8. Output: $\boldsymbol{w}_{t+1}$

We use the identical setting as in the problem set 1. Split the data in `spam_train.txt` into a training and validate set, putting the last 1000 emails into the validation set. Transform all of the data into feature vectors. Build a vocabulary list using only the 4000 e-mail training set by finding all words that occur across the training set. Ignore all words that appear in fewer than $X = 30$ e-mails of the 4000 e-mail training set. For each email, transform it into a feature vector $\vec{x}$ where the ith entry, x_i, is 1 if the ith word in the vocabulary occurs in the email, and 0 otherwise.

Note: To keep your algorithm simple, we will not use an offset term b when optimizing the SVM primal objective using Pegasos.

(a) Implement the function `pegasos_svm_train(data, lambda)`. The function should return $\boldsymbol{w}$, the final classification vector. For simplicity, the stopping criterion is set so that the total number of passes over the training data is 20. After each pass through the data, evaluate the SVM objective $f(\boldsymbol{w}_t) = \frac{\lambda}{2}\|\boldsymbol{w}_t\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i(\boldsymbol{w}_t \cdot \boldsymbol{x}_i)\}$ and store its value ($m = |\texttt{data}|$). Plot $f(\boldsymbol{w}_t)$ as a function of iteration (i.e. for $t = |\texttt{data}|, \ldots, 20|\texttt{data}|$), and submit the plot for $\lambda = 2^{-5}$.

(b) Implement the function `pegasos_svm_test(data, w)`.

(c) Run your learning algorithm for various values of the regularization constant, $\lambda = 2^{-9}, 2^{-8}, \cdots, 2^{1}$. Plot the average training error, average hinge loss of the training samples (i.e. $\frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i)\}$), and average validation error as a function of $\log_2 \lambda$. You should expect that the hinge loss upper-bounds the training error. What is the minimum of your validation error? For the classifier that has the smallest validate error: What is the test error? How many training samples are support vectors? How did you find them? Compare your test error with your result from problem set 1.

Note: Using a smaller value of X such as 0 (i.e., not filtering the vocabulary) would give even better results (the SVM's regularization prevents overfitting). If you try this, we recommend you use sparse matrices (e.g., with Python's `scipy.sparse`) as the feature vectors will be large but sparse and this will improve efficiency.

4. The multi-class SVM generalizes the binary SVM to multi-class classification. This involves introducing a weight vector $\vec{w}^{(k)}$ and $b^{(k)}$ for each class $k = 1, \ldots, K$ (where K is the number of classes). Learning solves the following optimization problem, where there is still only one slack variable ξ_j for each data point, but now there are $K - 1$ constraints per data point:

$$\min_{\{\vec{w}^{(k)}, b^{(k)}\}} \sum_{k=1}^{K} ||\vec{w}^{(k)}||_2^2 + C \sum_j \xi_j$$

subject to

$$\begin{aligned} \vec{w}^{(y_j)} \cdot \vec{x}_j + b^{(y_j)} &\geq \vec{w}^{(k)} \cdot \vec{x}_j + b^{(k)} + 1 - \xi_j \qquad \forall j \text{ and } k \neq y_j \\ \xi_j &\geq 0 \qquad \forall j. \end{aligned}$$

Prediction for a new data point $\vec{x}$ is performed using the rule

$$\hat{y} \leftarrow \arg\max_k \ \vec{w}^{(k)} \cdot \vec{x} + b^{(k)}.$$

This problem compares the binary prediction rule $\text{sign}(\vec{w} \cdot \vec{x} + b)$ to the multi-class prediction rule in the case that $K = 2$, and shows how to reduce between the two of them.

(a) Demonstrate $\vec{w}$ and b as a function of $\vec{w}^{(1)}$, $b^{(1)}$, $\vec{w}^{(2)}$ and $b^{(2)}$ such that the predictions made for all data points $\vec{x}$ using the new binary prediction rule are the same as what would have been made using the multi-class prediction rule with $\vec{w}^{(1)}$, $b^{(1)}$, $\vec{w}^{(2)}$.

(b) Next you should show the converse. Given $\vec{w}$ and b, demonstrate $\vec{w}^{(1)}$, $b^{(1)}$, $\vec{w}^{(2)}$ and $b^{(2)}$ (as a function of $\vec{w}$ and b) such that the predictions made for all data points $\vec{x}$ using the multi-class prediction rule are the same as what would have been made using the binary prediction rule with $\vec{w}$ and b.

As always, you must show all of your work to obtain full credit.

References

[1] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.