

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.867 MACHINE LEARNING, FALL 2009

Problem Set 1 Solutions

Problem 1 Perceptron

(a) We will derive the two parts of the convergence proof with the modified update and show that we arrive to the same result.

Part 1:

$$(\underline{\theta}^*)^T \underline{\theta}^{(k)} = (\underline{\theta}^*)^T \underline{\theta}^{(k-1)} + y_t (\underline{\theta}^*)^T \frac{\underline{x}_t}{\|\underline{x}_t\|} \tag{1}$$

$$\geq (\underline{\theta}^*)^T \underline{\theta}^{(k-1)} + \frac{\gamma_g \|\underline{\theta}^*\|}{R} \tag{2}$$

$$(\underline{\theta}^*)^T \underline{\theta}^{(k)} \geq k \frac{\gamma_g \|\underline{\theta}^*\|}{R} \tag{3}$$

Part 2:

$$\|\underline{\theta}^{(k)}\|^2 = \|\underline{\theta}^{(k-1)} + y_t \frac{\underline{x}_t}{\|\underline{x}_t\|}\|^2 \tag{4}$$

$$= \|\underline{\theta}^{(k-1)}\|^2 + 2y_t \underline{\theta}^{(k-1)} \frac{\underline{x}_t}{\|\underline{x}_t\|} + \frac{\|\underline{x}_t\|^2}{\|\underline{x}_t\|^2} \tag{5}$$

$$\leq \|\underline{\theta}^{(k-1)}\|^2 + \frac{\|\underline{x}_t\|^2}{\|\underline{x}_t\|^2} \tag{6}$$

$$= \|\underline{\theta}^{(k-1)}\|^2 + 1 \tag{7}$$

$$\|\underline{\theta}^{(k)}\|^2 \leq k \tag{8}$$

$$\cos(\underline{\theta}^*, \underline{\theta}^{(k)}) = \frac{(\underline{\theta}^*)^T \underline{\theta}^{(k)}}{\|\underline{\theta}^*\| \|\underline{\theta}^{(k)}\|} \stackrel{(1)}{\geq} \frac{k \frac{\gamma_g \|\underline{\theta}^*\|}{R}}{\|\underline{\theta}^*\| \|\underline{\theta}^{(k)}\|} \stackrel{(2)}{\geq} \frac{k \frac{\gamma_g \|\underline{\theta}^*\|}{R}}{\|\underline{\theta}^*\| \sqrt{k}} = \frac{k \frac{\gamma_g}{R}}{\sqrt{k}} \tag{9}$$

Since cosine is bounded by one, we get

$$1 \geq \frac{k \frac{\gamma_g}{R}}{\sqrt{k}} \text{ or } k \leq \frac{R^2}{\gamma_g^2} \tag{10}$$

(b) After the first and only update, we will get $\underline{\theta} = y_i \underline{x}$. The decision boundary line is perpendicular to \underline{x} . The farthest point between \underline{x} and a line passing through the origin is the line perpendicular to \underline{x} . Hence, we found our max margin classifier.

(c) Let $\gamma_g \rightarrow R$, without loss of generality, the possible sequences in this case are $\forall t : \underline{x}_t \rightarrow [x1, x2]$ for positive labels or $\underline{x}_t \rightarrow [-x1, -x2]$ for the negative labels where $\|\underline{x}_t\| = \sqrt{(x1^2 + x2^2)} = R$

Using part b), we can see that we only need one update to classify all the points correctly. After the first and only update $\underline{\theta}$ is equal to either $[x1, x2]$ or $[-x1, -x2]$ the maximum margin classifier.

(d) The perceptron will make a mistake on each \underline{x}_t . We can see that all \underline{x}_t are orthogonal. If you go step by step over the perceptron algorithm, first $\underline{\theta} = 0$. We go through the examples one by one to check $y_t(\underline{\theta} \cdot \underline{x}_t) \leq 0$ and update $\underline{\theta}$ if this test holds. Let's assume that after the i-1th iteration $\underline{\theta} = \sum_{j=1}^{i-1} y_j \underline{x}_j$ then we test for the ith example if $y_i \underline{\theta} \cdot \underline{x}_i \leq 0$ to update $\underline{\theta}$ or not. Since $\underline{\theta}$ is a linear combination of the previous \underline{x}_t and all the \underline{x}_t are orthogonal, $y_i \underline{\theta} \cdot \underline{x}_i = y_i \sum_{j=1}^{i-1} y_j \underline{x}_j \cdot \underline{x}_i = y_i \sum_{j=1}^{i-1} y_j 0 = 0$ which is a mistake. Clearly after the nth example, $\underline{\theta} = \sum_{j=1}^n y_j \underline{x}_j = [y_1, \dots, y_n]^T$.

The geometric margin in this case is $\frac{\min(\underline{\theta} \cdot \underline{x}_i)}{\|\underline{\theta}\|} = \frac{1}{\|\underline{\theta}\|} = \frac{1}{\sqrt{n}}$ since $\|\underline{\theta}\| = \sqrt{\sum_{j=1}^n y_j^2} = \sqrt{\sum_{j=1}^n 1} = \sqrt{n}$.

(e) Perceptron code:

```
function perc = perceptron_build(data)

% Initialize weight vector to all zeros
% NOTE: final w vector should be in perc.w
perc.w = zeros(size(data.X,2),1);

% Record number of mistakes we make on each data point
perc.mistakes = zeros(size(data.X,1),1);

% Fill in algorithm here:

mistakes = 0;
for i=1:size(data.X,1),
    if (data.y(i)*(data.X(i,:)*perc.w)<=0), %mistake
        perc.w = perc.w + data.y(i)*data.X(i,:);
        perc.mistakes(i) = perc.mistakes(i)+1;
    end;
end;

% show the solution

%perceptron_plot(data,perc);
```

The number of mistakes depends on the order from the sequential nature of the perceptron algorithm. In some cases like d) the order of the sequence didn't matter since all examples were orthogonal. We update $\underline{\theta}$ based on the sequence order we receive, and various orderings produce different separators since there a linear combination of different $y_i \underline{x}_i$. In the general case, if we receive first the \underline{x}_i 's with the smallest norms value then correcting $\underline{\theta}$ will converge faster since we need to traverse a small distance to correct $\underline{\theta}$. Similarly, if we receive \underline{x}_i 's with the largest norms the correction of $\underline{\theta}$ need to go through a much greater distance to classify the points correctly.

We can use few runs of the perceptron to estimate the value of the geometric margin γ_g by recording the highest number of mistakes from those runs. Since $k \leq \frac{R^2}{\gamma_g^2}$ then $\gamma_g \leq \frac{R}{\sqrt{k}}$.

Problem 2 Support vector machines

In many cases the data points to be classified are not linearly separable. Even if they are, however, the maximum margin separator may be unduly influenced by a single or only a few data points. It would be great if we could just “give up” on a few points and construct the classifier on the basis of the remaining points. This would be a hard problem to solve but we can do something close to it by introducing slack variables for the margin constraints in support vector machines. Here we will try to understand the effect of the slack variables on the SVM solution and how the penalty constant C should be chosen. The SVM optimization problem is given by

$$\text{minimize } \frac{1}{2} \|\underline{\theta}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{subject to} \quad (11)$$

$$y_i(\underline{\theta} \cdot \underline{x}_i + \theta_0) \geq 1 - \xi_i, \quad i = 1, \dots, n \quad (12)$$

$$\xi_i \geq 0, \quad i = 1, \dots, n \quad (13)$$

$$(14)$$

We have provided you with MATLAB code to experiment with different slack penalties. You can get the SVM solution by running `svm = svm_build(data,@K_linear,C)` where C is the slack penalty and `@K_linear` passes a function reference to a “linear kernel” (see file “k_linear.m”). The data is the same as before. You can plot the points and the SVM solution by running `svm_plot(data,svm)` which will highlight the support vectors as well as the margin around the linear separator.

After solving the SVM, the variable “`svm.NS`” contains the number of support vectors in the resulting solution. “`svm.w0`” contains the offset (bias) of the hyperplane, “`svm.XS`” contains the support vectors (1 per row), and “`svm.beta`” for row i gives the dual parameters for the quadratic programming problem (equal to $\alpha_i y_i$). See “`svm_discrim_func.m`” for an example of how they are used.

(a) Try $C = 100$ and $C = 1000$. Why doesn’t the solution change? Explain this in terms of the optimization problem.

This dataset is a modification of the standard ‘Iris’ dataset (<http://archive.ics.uci.edu/ml/datasets/Iris>). ‘Iris’ is a three class classification problem to identify the iris plant type (Iris Setosa, Iris Versicolour, or Iris Virginica). It has the following 4 features:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

In our dataset, we let +1 represent Iris Setosa, and -1 represent the other two types.

By inspection, we see that our data set is linearly separable with offset. Therefore, the hard-margin SVM is feasible; denote the optimal setting for $\underline{\theta}$ as $\hat{\underline{\theta}}$. In terms of the dual problem, this means there exists an optimal setting for the dual variables $\underline{\alpha} = \hat{\underline{\alpha}}$ in the nonnegative orthant ($\underline{\alpha} \succeq \underline{0}$) and on the hyperplane $\underline{y}^T \underline{\alpha} = 0$. Note that $\hat{\underline{\alpha}}$ is not necessarily unique, but that $\hat{\underline{\theta}}$ must be.

Now let $C_0 = \max_i \hat{\alpha}_i$. Consider the soft-margin SVM for any $C \geq C_0$. As $\underline{0} \preceq \hat{\underline{\alpha}} \preceq \underline{C}$, this dual objective is also maximized at the same setting $\underline{\alpha} = \hat{\underline{\alpha}}$. It follows that the corresponding $\hat{\underline{\theta}}$ is also optimal for this soft-margin SVM.

Therefore, for C sufficiently large, the hard-margin SVM optimal solution, $\hat{\theta}$, will also optimize the soft-margin SVM. We can check that $C = 100$ does indeed yield the hard-margin SVM optimal solution, as the slacks ξ_i are all 0. Then, any $C > 100$ will also yield the same solution.

(b) Recall that, for the hard-margin SVM, the geometric margin is defined as the minimum distance from any point to the separating hyperplane. This distance can be shown to be $\frac{1}{\|\hat{\theta}\|}$, where the 1 in the numerator comes from the classification constraints $y_t(\hat{\theta} \cdot x_t + \theta_0) \geq 1$ in the optimization problem.

When we use slack variables (regardless of whether the data is separable), some of the points may be misclassified, so we can no longer measure the margin directly in terms of the minimum distance to the separating hyperplane. However, we can still use the expression $\frac{1}{\|\hat{\theta}\|}$ as the definition of the margin, where $\hat{\theta}$ comes from solving the SVM optimization problem with slack variables.

Try $C = 0.1, 1, 10, 100$. What happens to the margin $1/\|\hat{\theta}\|$ as C increases? Will this always happen as we increase C ? What happens to the number of support vectors as C increases?

The margin $\frac{1}{\|\hat{\theta}\|}$ must decrease as C increases. The most intuitive way to see this is to divide the objective by C , which does not affect the solution. Then note that, as we increase the slack penalty C , $\|\hat{\theta}\|^2$ can become larger without costing as much.

$$\frac{1}{2C} \|\theta\|^2 + \sum_{i=1}^n \xi_i$$

To show this rigorously, suppose we have two values of the slack penalty C_1, C_2 with corresponding optimal settings $\hat{\theta}^1$ and $\hat{\theta}^2$. It suffices to show that $(C_1 - C_2)(\|\hat{\theta}^1\|^2 - \|\hat{\theta}^2\|^2) \geq 0$. We outline one possible proof. First, verify the following lemma for any functions f, g with the same domain: if $a = \operatorname{argmin}_x g(x)$ and $b = \operatorname{argmin}_x g(x) + f(x)$, then $f(b) \leq f(a)$. Then simply take $g(\theta) = \frac{1}{2C_1} \|\theta\|^2 + \min_{\theta_0} \sum_{i=1}^n \xi_i(\theta, \theta_0)$ (using the notation of 2(e)) and $f(\theta) = (\frac{1}{2C_2} - \frac{1}{2C_1}) \|\theta\|^2$.

For smaller C , as the margin is larger, the number of support vectors $\#SV$ should intuitively also be larger. (By the way, we are not yet aware of a rigorous justification for this, so let us know if you find one.) To be clear, here we say that x_i is a *support vector* if and only if its margin constraint is tight (either with slack or without). By complementary slackness, if the dual variable $\hat{\alpha}_i > 0$, then x_i is a *support vector*. Thus, a bound on the the number of support vectors is $\#SV \geq \#\{i : \hat{\alpha}_i > 0\}$. Remarkably, this bound is invariant to the choice of C (Can you see why?). Then, how is it possible that $\#SV$ increases as C decreases? It is because, even if the dual variable $\hat{\alpha}_i = 0$, it is still possible that x_i is a *support vector*. And this is where the action really happens: as C decreases, x_i for which $\hat{\alpha}_i = 0$ may become support vectors, even while their dual variable remains fixed at $\hat{\alpha}_i = 0$.

(c) The value of C will typically change the resulting classifier and therefore also affects the accuracy on test examples. The best choice of C would be to optimize the test performance. With access only to the few training examples, we cannot hope to do that. However, we can simulate test performance via leave-one-out cross-validation. In other words, we can hold out each training example-label pair in turn, training the classifier on the remaining examples, and testing the classifier on the held out “test” example. Implement the cross-validation procedure. Do choices $C = 0.1, 1, 10, 100$ have any significant effect on the cross-validation performance? Could we instead use a much simpler procedure by training the classifier only once for each value of C and relying on the bound on the cross-validation performance given by the number of support vectors? Explain why or why not.

As leave-one-out cross-validation (LOOCV) is computationally expensive, one might consider simply training a classifier on the full dataset only once (rather than n times) for each value of C . In this case, how might we pick the ‘best’ C ? Here, we have suggested using the support vector bound, which holds for any value of C : $\epsilon_{LOOCV}(C) \leq \frac{\#SV(C)}{n}$. How helpful is this bound to determining a ‘best’ C ? Note that this bound is monotonically decreasing with C (as we empirically observed in 2(b)), so it will prefer a large C (namely C sufficiently large for which $\#SV$ becomes constant).

In general, this bound is less than desirable, because it favors a very tight fit (possibly an overfit) to our training data, one with few support vectors.

(d) If we fix C and obtain $\hat{\underline{\theta}}$, $\hat{\theta}_0$, and $\hat{\xi}_i$ as the solution to the quadratic programming problem. The slack is relevant (non-zero) only for support vectors. How does the value of slack $\hat{\xi}_i$ relate to the distance of a support vector \underline{x}_i from the decision boundary?

Recall that the signed distance from a point \underline{b} to the hyperplane $\underline{a}^T \underline{x} + a_0 = 0$ is given by $\frac{\underline{a}^T \underline{b} + a_0}{\|\underline{a}\|}$. For a support vector \underline{x}_i , the slack constraint is tight: $y_i(\hat{\underline{\theta}}^T \underline{x}_i + \hat{\theta}_0) = 1 - \hat{\xi}_i$. Thus, the signed distance from \underline{x}_i to the decision boundary $y_i(\hat{\underline{\theta}}^T \underline{x} + \hat{\theta}_0) = 0$ is $\frac{y_i(\hat{\underline{\theta}}^T \underline{x}_i + \hat{\theta}_0)}{\|\hat{\underline{\theta}}\|} = \frac{1 - \hat{\xi}_i}{\|\hat{\underline{\theta}}\|}$.

(e) The quadratic programming problem involves both $\underline{\theta}, \theta_0$ and the slack variables ξ_i . We can rewrite the optimization problem in terms of $\underline{\theta}, \theta_0$ alone. By doing so we explicate what the loss-function is on the training points corresponding to the relaxed max-margin formulation. You can do this as follows. First, fix $\underline{\theta}, \theta_0$ and find the optimal values of the slack variables $\xi_i = \xi_i(\underline{\theta}, \theta_0)$ as functions of $\underline{\theta}, \theta_0$. What are these values? Are all the margin constraints satisfied with these values for the slack variables?

The resulting minimizing problem over $\underline{\theta}, \theta_0$ can be formally written as

$$\frac{1}{2} \|\underline{\theta}\|^2 + C \sum_{i=1}^n \xi_i(\underline{\theta}, \theta_0) \tag{15}$$

where the first (regularization) term biases our solution towards zero in the absence of any data and the remaining terms give rise to the loss functions. What are the loss functions? Do we need any additional constraints? Many learning problems can be understood and compared in the above form (regularization + loss).

We wish to express the optimal slack values ξ_i as a function of $\underline{\theta}, \theta_0$. To do this, for a fixed setting of $\underline{\theta}, \theta_0$, we shall find the setting for the slack variables ξ_i that solves the following optimization:

$$\begin{aligned} \text{minimize}_{\xi_i} \quad & \frac{1}{2} \|\underline{\theta}\|^2 + C \sum_{i=1}^n \xi_i \quad \text{subject to} \\ & y_i(\underline{\theta} \cdot \underline{x}_i + \theta_0) \geq 1 - \xi_i, \quad i = 1, \dots, n \\ & \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned}$$

As $\underline{\theta}, \theta_0$ are fixed and $C > 0$, this problem is equivalent to:

$$\begin{aligned} \text{minimize}_{\xi_i} \quad & \sum_{i=1}^n \xi_i \quad \text{subject to} \\ & \xi_i \geq 1 - y_i(\underline{\theta} \cdot \underline{x}_i + \theta_0), \quad i = 1, \dots, n \\ & \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned}$$

This, in turn, can be decomposed over the variables ξ_i individually:

$$\begin{aligned} \text{minimize}_{\xi_i} \quad & \xi_i \quad \text{subject to} \\ & \xi_i \geq 1 - y_i(\underline{\theta} \cdot \underline{x}_i + \theta_0), \\ & \xi_i \geq 0 \end{aligned}$$

And the solution to these individual optimization problems is given by $\xi_i = \max(0, 1 - y_i(\underline{\theta} \cdot \underline{x}_i + \theta_0))$. The function $f(z) = \max(0, 1 - z)$ is known as the *hinge loss* function, which is occasionally written as $f(z) = (1 - z)_+$.

Thus, $\xi_i(\underline{\theta}, \theta_0) = (1 - y_i(\underline{\theta} \cdot \underline{x}_i + \theta_0))_+$. Writing ξ_i in this way, the optimization problem becomes unconstrained in $\underline{\theta}, \theta_0$.