CS 4780 Assignment 3

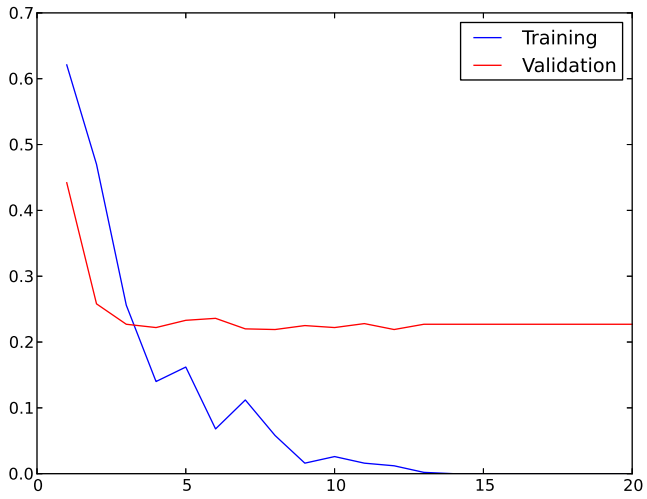
Justin Cheng and Sunling Selena Yang

October 2, 2011

---

1 Perceptron and SVM

a.



The training error decreases dramatically after the first few iterations, although it is not strictly decreasing. However, after the 15th iteration the training error is 0, meaning that a separating hyperplane was learnt.

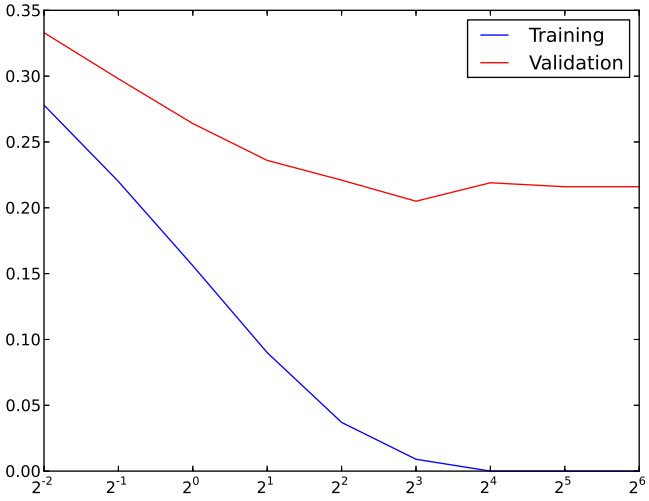
The validation error fluctuates wildly for the first few iterations, but then stabilizes at about 0.217.

This does appear to confirm the fact that the Perceptron algorithm converges after a finite number of iterations, and slightly overfits when iterated too many times.

Prefer a lower learning rate for improved accuracy, though over a longer number of iterations, since increasing the learning rate speeds up convergence around the true separating hyperplane.. If the learning rate is too high, over-correction is likely to occur and you might end up oscillating around the true separating hyperplane.

Prefer to use the primal algorithm, since computation in each iteration requires using the whole matrix of training examples and storing the alpha values for each example, rather than a single $1 \times n$ dimensional matrix if n is the number of unique features. Either algorithm should be equally correct. However, the dual algorithm does provide us with additional expressive power since we can see how important certain examples are by examining their corresponding α values.

b.

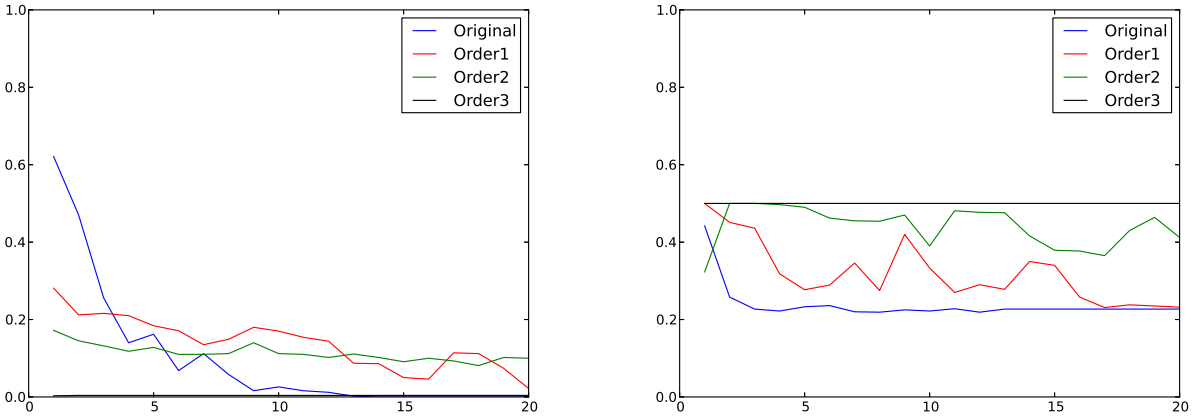


The smallest validation error rate is achieved at $C = 8$. This error rate, 0.205, is superior to that of the Perceptron algorithm, which obtains 0.227 after 20 iterations.

While training error goes to 0 as C increases, validation error decreases until $C = 8$ and then increases slightly afterward.

The training error rate goes to 0 as C increases, but the validation error steadily decreases to a minimum at $C = 8$, and increases after that. This is because C controls for how much we penalize incorrectly classified points. If C is too low, it means that we allow too many points to be incorrectly classified, and when it is too high, we over-fit and overly penalize incorrectly classified points.

c.



The graph on the left shows the training error, and the graph on the right shows the validation error. Perceptron works badly on reorder-3 because in that reordering, all the positive examples come before the negative examples. Thus, the algorithm makes very few updates in each iteration (at most 4), since after the first mistake on the first positive example, all examples are classified correctly until the first negative example, and then after

a couple of corrections, all subsequent examples (all negative) are all classified correctly. Thus, the hypothesis learned is not indicative of a true weighing of all the features.

Using SVM_{light} with $C = 8$, the training error in all cases is 0.009, and the validation error in all cases is 0.205. We observe that SVM is not order-dependent.

d.

Let H_1 be the hypothesis produced by the perceptron algorithm. Let H_2 be the hypothesis produced by SVM. Let d_i be the # of examples H_i makes an error on, but not $H_{i'}$, $i' \neq i$.

$d_1 = 61$

$d_2 = 39$

$k = d_1 + d_2 = 100$

If $Err_p(h_1) = Err_p(h_2)$, then d_1, d_2 are binomially distributed with $p = \frac{1}{2}$.

Null hypothesis: D_1 is binomial with $p = \frac{1}{2}$ and $k = 100$.

$P(D_1 \geq d_1 \mid p = 0.5, k) = 0.0104 < 0.025$

Reject null hypothesis. There is evidence to suggest that the number of errors produced by both hypothesis are significantly different.

2 Linear Separability

a.

So for $\tilde{S}$ to be linearly separable, it has to satisfy the constraint for $\delta > 0$, $\forall 1 \leq i \leq n, y_i \left(w^T x_i\right) + y_i w_i jk \geq \delta$, where $w_i j$ = the weight coefficients for the corresponding $t_i j$.

If $t_i j = 0 \forall$ i,j, data is separable as given by the problem. For $t_i j \neq 0, let w_i j = \frac{My_i}{k}, and since \mid x_i \mid \leq M$, the extra terms are always > 0 and guaranteed to be larger than terms generated by any missing attributes so if δ is the lower bound for datasets with no missing features,

$y_i < \tilde{x}^T, \tilde{x}_i > \geq \delta \forall x_i, y_i$.

b.

An upper bound on the radius of $\tilde{S}$ is $\sqrt{R^2 + k^2}$ because since there is now at most one additional dimension with length k (since the other additional dimensions all have length 0), the ball now has a new radius $\sqrt{R^2 + k^2 - \sum x_i^2}$, where x_i are the missing features that would otherwise have nonzero values. Since we don't know what the values for the missing features, $\sqrt{R^2 + k^2}$ is an upper bound on the new radius.

The upper bound on the number of mistakes made is then $\frac{(R^2+k^2)}{\delta^2}$.

3 Linear SVM

a.

We can use the constraint equation to find the minimum ξ_i and see if it falls near the boundary after adding the slack variable.

$$y_i \left(w^T x + b\right) \geq 1 - \xi_i$$
$$2\alpha_i R^2 + \xi_i \geq 1$$

i	y_i	$w^T x_i + b$	α_i	ξ_i	$2\alpha_i R^2 + \xi_i$	≥ 1?
1	1	1	0	0	0	No
2	-1	-0.4	0.1	0.6	0.8	No
3	1	0.3	0.2	0.7	1.1	Yes

So from this the upper bound is 1/3

One can also count the number of support vectors $\alpha_i > 0$ that lie on the boundary, but that yields a bound of 2/3, which is not as tight as above.

b.

For the hard margin unbiased SVM, the dual QP is

$$\begin{aligned} \max \quad & D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \end{aligned}$$

Since $x_i^T x_j = 0$ for $i \neq j$, the objective function simplifies to $\max \sum_i \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \alpha_i^2 l_i$, since $y_i y_i = 1 \ \forall i$.

Further simplifying, we get $\max \sum_i \left(\alpha_i - \frac{1}{2}\alpha_i^2 l_i\right)$, and since we know there is exactly one maximum for this function, we take the derivative of the function with respect to α_i, set that to zero, and solve for x_i:

$$\begin{aligned} \frac{d}{d\alpha_i} \sum_{i=1}^{n} \left(\alpha_i - \frac{1}{2}\alpha_i^2 l_i\right) &= 0 \\ 1 - \alpha_i l_i &= 0 \\ 1 &= \alpha_i l_i \\ \alpha_i &= \frac{1}{l_i} \end{aligned}$$

, where $l_i = x_i^T x_i$ for each training example (x_i, y_i). Again, we know that this is a maximum point since the second derivative is $-l_i < 0$.

c.

First, note that in the perceptron primal algorithm, for each update, you add $y_i x_i$ to w_k. If all x_i are scaled up, such that $x_i^\star = kx_i$, in each update, you will instead add $ky_i x_i$ to w_k. Thus, after normalizing, w will still be the same as before, since we simply multiplied w (unnormalized) by k.

Now, the error ξ_i will be multiplied by k for each i. To see why this is the case, notice that this value depends on the distance to the hyperplane, $w^T x_i$. In the new case, this value is now $w^T x_i^\star = kw^T x_i$, so that this distance is multiplied by k.

Thus, in our objective, we want to set C such that our new objective $\min \frac{1}{2}||w^{\star 2}|| + C^\star \sum_{i=1}^n \xi_i^\star$ is a constant factor of our original objective function. But we found out the new values of w and xi, so we know our new objective is equivalently $\min \frac{1}{2}||w^2|| + C^\star \sum_{i=1}^n k\xi_i$. Comparing terms, $kC^\star = C \Rightarrow C^\star = \frac{C}{k}$.

Thus, C should be divided by k so that the new solution defines the same linear classifier.

The resulting $(w^\star, \xi^\star)$ is equivalently $(w, k\xi)$.