16-831 RoboStats Homework 2

2. Perceptron (40 Points)

2.1.1 (5 points)

We measured the performance of Perceptron using the zero-one loss. However, this loss is usually hard to optimize, as it is non-convex and non-continuous. In class we learned that a common trick (i.e., convexification trick) to get around this is to upper bound the zero-one loss by some surrogate loss. An example of this for the zero-one loss is the hinge loss:

Answer:

We need to show that the following inequality holds (equality 2 in the writeup):

$$\ell\big(w, (x_t, y_t)\big) \geq l[\hat{y}_t \neq y_t] - \mathrm{w}^{\mathrm{T}} u_t \qquad (1)$$

Now let us consider equation 1:

$$\ell\big(w, (x_t, y_t)\big) = \max\{0, 1 - y_t x_t^T w\}$$

Plug this into equation 1 to get:

$$\max\{0, 1 - y_t x_t^T w\} \;\geq\; l[\hat{y}_t \neq y_t] - \mathrm{w}^{\mathrm{T}} u_t \qquad (2)$$

Now since $u_t = y_t \, x_t^T \, l[\hat{y}_t \neq y_t]$ we can express equation (2) as:

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; l[\hat{y}_t \neq y_t] - \mathrm{w}^{\mathrm{T}} \, y_t \, x_t^T \, l[\hat{y}_t \neq y_t] \qquad (3)$$

What a ghastly thing.

Anyways, let us now consider two cases, one where the output of the perceptron $\hat{y}_t$ is equal to the ground truth y_t and the other where it isn't. So the two cases that we should look into are: $\hat{y}_t \neq y_t$ and $\hat{y}_t = y_t$.

Let's plug these into equation (3), if the inequality holds during these 2 test cases, we have proven it.

Starting with $\hat{y}_t \neq y_t$, if this is the case:

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; l[\hat{y}_t \neq y_t] - \mathrm{w}^{\mathrm{T}} \, y_t \, x_t^T \, l[\hat{y}_t \neq y_t]$$

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; 1 - \mathrm{w}^{\mathrm{T}} \, y_t \, x_t^T \, 1$$

In the case, that the prediction is inaccurate, $y_t x_t^T w$ is negative, which means that the left hand side is $\max\{\, 0, Val > +1 \,\}$, and so will be a value greater than 1. Therefore the left hand sign and right hand sign are equal to each other. And no matter, this is the case no matter the magnitude of $y_t x_t^T w$.

Now if $\hat{y}_t = y_t$, then:

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; l[\hat{y}_t = y_t] - \mathrm{w}^{\mathrm{T}} \, y_t \, x_t^T \, l[\hat{y}_t = y_t]$$

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; 0 - \mathrm{w}^{\mathrm{T}}\, y_t\, x_t^T\, 0$$

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; 0 - \;0$$

$$\max\{\, 0, 1 - y_t x_t^T w \,\} \;\geq\; 0$$

If the prediction is accurate, $y_t x_t^T w$ is positive, therefore we get the following set of inequalities depending on the magnitude of $y_t x_t^T w$:

Left Hand Side	**Magnitude of** $y_t x_t^T w$
$\max\{\, 0, 1 > Val > 0\}$,	$\lvert y_t x_t^T w \rvert > 1$
$\max\{\, 0, Val < 0 \,\}$	$1 > \lvert y_t x_t^T w \rvert > 0$

Therefore if $\lvert y_t x_t^T w \rvert > 1$,

$$\max\{\, 0, 1 > Val > 0\} \;\geq\; 0$$

This proves the $\geq$ operator.

If $1 > \lvert y_t x_t^T w \rvert > 0$,

$$\max\{\, 0, Val < 0 \,\} \geq\; 0$$

This proves the $=$ operator, which itself is part of the $\geq$ operator.

2.1.2 (5 points)

We begin with induction for the equation:

$$u_t^T w^* = \phi(W_{t+1}) - \phi(W_t)$$

Let's isolate the $\phi(W_{t+1})$ so that we can prove $\phi(W_{t+1}) \geq M - L$

$$\phi(W_{t+1}) = u_t^T w^* + \phi(W_t) \qquad (5)$$

We now plug in the following relationship $\phi(W_t) = w_t^T w^*$ into equation (5) to obtain:

$$\phi(W_{t+1}) = u_t^T w^* + w_t^T w^* \qquad (6)$$

So from this position, this is where variable *loss* is included in this equation. Without changing the logic of expression (6) we both add *loss* and subtract it from this equation to get:

$$\phi(W_{t+1}) = u_t^T w^* + w_t^T w^* + loss - loss$$

Rearranging the terms in this equation as part of the master plan to prove the inequality, we get:

$$\phi(W_{t+1}) = u_t^T w^* - loss + w_t^T w^* + loss$$

Using the same strategy, express $u_t^T w^*$ as – $(-u_t^T w^*)$ to get:

$$\phi(W_{t+1}) = - (-u_t^T w^*) - loss + w_t^T w^* + loss$$

We can now expressing this as:

$$\phi(W_{t+1}) = - (-u_t^T w^* + loss) + (w_t^T w^*) + (loss)$$

If we now proceed with this equation as it holds for all possible times (t) we get:

$$\phi(W_{t+1}) = -\sum_{t=1}^{T} (-u_t^T w^* + loss) + \sum_{t=1}^{T} (w_t^T w^*) + \sum_{t=1}^{T} (loss)$$

As mentioned before, the *loss* can be expressed as $l[\hat{y}_t \neq y_t]$:

$$\phi(W_{t+1}) = -\sum_{t=1}^{T} (-u_t^T w^* + l[\hat{y}_t \neq y_t]) + \sum_{t=1}^{T} (w_t^T w^*) + \sum_{t=1}^{T} (l[\hat{y}_t \neq y_t])$$

According to the documentation, in the equation $\phi(W_{t+1}) \geq M - L$:

$$M = \sum_{t=1}^{T} (l[\hat{y}_t \neq y_t])$$

And

$$L = \sum_{t=1}^{T} (-u_t^T w^* + l[\hat{y}_t \neq y_t])$$

Therefore we have:

$$\phi(W_{t+1}) = -L + M + \sum_{t=1}^{T} (w_t^T w^*)$$

Now the term below is positive or at least greater than 0:

$$\sum_{t=1}^{T} (w_t^T w^*)$$

Therefore

$$\phi(W_{t+1}) = M - L + Positive\ Value$$

So thus

$$\phi(W_{t+1}) \geq M - L$$

Q2.1.3

Again if use induction, we start off with:

$$W_{t+1} = W_t + U_t \quad (7)$$

We now start off by finding the squared L2 norm on both sides of this equation:

$$|| W_{t+1} ||_2^2 = || W_t + U_t ||_2^2$$

Expanding we get:

$$|| W_{t+1} ||_2^2 = || W_t ||^2_2 + || U_t ||^2_2 + \mathbf{2} < W_t, U_t >$$

If we make a mistake at time t, $\mathbf{2} < W_t, U_t >$ must be negative. Therefore:

$$|| W_{t+1} ||_2^2 \leq || W_t ||^2_2 + || U_t ||^2_2$$

We also know that $|| W_t ||^2_2$ is positive or at least larger than 0. We can thus logically make the following inequality:

$$|| W_{t+1} ||_2^2 \leq || W_t ||^2_2 + || U_t ||^2_2 \leq || U_t ||^2_2 \quad (8)$$

Now referring back to the pseudo code for the perceptron algorithm, if we look at the weight update line:

$$W_t = W_{t-1} + y_t \cdot \boldsymbol{x}^t \cdot \boldsymbol{l}[y^t \neq \hat{y}^t]$$

We can see that if we refer back to equation (7), that:

$$U_t = y_t \cdot \boldsymbol{x}^t \cdot \boldsymbol{l}[y^t \neq \hat{y}^t]$$

Now if we plug U_t into (8) we get:

$$\begin{aligned} || W_{t+1} ||_2^2 &\leq || W_t ||^2_2 + || y_t \cdot \boldsymbol{x}^t \cdot \boldsymbol{l}[y^t \neq \hat{y}^t] ||^2_2 \\ &\leq || y_t \cdot \boldsymbol{x}^t \cdot \boldsymbol{l}[y^t \neq \hat{y}^t] ||^2_2 \end{aligned}$$

This equation still holds, if we remove W_t:

$$|| W_{t+1} ||_2^2 \leq || y_t \cdot \boldsymbol{x}^t \cdot \boldsymbol{l}[y^t \neq \hat{y}^t] ||^2_2$$

We can breakdown the right hand side as:

$$|| W_{t+1} ||_2^2 \leq || \boldsymbol{x}^t ||^2_2 \cdot || y_t \cdot \boldsymbol{l}[y^t \neq \hat{y}^t] ||^2_2$$

The upper bound deals with the situation where every ground truth $y_t = 1$

$$\therefore || W_{t+1} ||_2^2 \leq || \boldsymbol{x}^t ||^2_2 \cdot || \boldsymbol{l}[y^t \neq \hat{y}^t] ||^2_2$$

Now summing the cumulative losses, we get:

$$|| \ W_{t+1} \ ||_2^{\mathbf{2}} \leq \ || \ \boldsymbol{x}^t \ ||^2_{\mathbf{2}} \cdot || \ \sum_{\boldsymbol{t=1}}^{\boldsymbol{T}} \boldsymbol{l}[y^t \neq \hat{y}^t] \ ||^2_{\mathbf{2}}$$

Therefore, since $|| \ \sum_{t=1}^{T} l[y^t \neq \hat{y}^t] \ ||^2_2$ is $\boldsymbol{M}$:

$$|| \ W_{t+1} \ ||_2^{\mathbf{2}} \leq \ || \ \boldsymbol{x}^t \ ||^2_{\mathbf{2}} \cdot \boldsymbol{M}$$

Also R bounds x^t therefore:

$$|| \ \boldsymbol{x}^t \ ||^2_{\mathbf{2}} = \boldsymbol{R^2}$$

$$\therefore || \ W_{t+1} \ ||_2^{\mathbf{2}} \leq \boldsymbol{MR^2}$$

Therefore demonstrating $|| \ W_{t+1} \ ||_2^2$ is linearly proportional to the number of mistakes M.

Q2.1.4

In the previous question, we were given a series of inequalities:

$$\phi_{(W_{T+1})} = W_{T+1}^T W^* \leq \left||W_{T+1}|\right|_2 \left||W^*|\right|_2 \leq D\left||W_{T+1}|\right|_2$$

However from the proof at the end of the previous question: $||\ W_{t+1}\ ||_2^{\ 2} \leq \boldsymbol{MR^2}$
And so we know:

$$\phi_{(W_{T+1})} \leq D\left||W_{T+1}|\right|_2$$

$$\therefore \phi_{(W_{T+1})} \leq D\sqrt{\boldsymbol{MR^2}}$$

$$\therefore \phi_{(W_{T+1})} \leq D\sqrt{\boldsymbol{M}}\boldsymbol{R}$$

Referring back to the proof at the end of the second ($\phi_{(W_{T+1})} \geq M - L$) question we can say:

$$M - L \leq D\sqrt{\boldsymbol{M}}\boldsymbol{R}$$

Bringing everything to one side we get:

$$M - L - D\sqrt{\boldsymbol{M}}\boldsymbol{R} \leq 0$$

If we solve this quadratically, where $M = x^2$ we can begin be representing this equation as:

$$x^2 - L - D\boldsymbol{x}\boldsymbol{R} \leq 0$$

Solving this we get:

$$x \leq \frac{DR + \sqrt{(DR)^2 + 4L}}{2(1)}$$

$$\therefore M \leq \frac{DR + \sqrt{(DR)^2 + 4L}}{2(1)} * \frac{DR + \sqrt{(DR)^2 + 4L}}{2(1)}$$

$$\therefore M \leq \frac{(DR)^2 + 2DR\sqrt{(DR)^2 + 4L} + (DR)^2 + 4L}{4}$$

$$\therefore M \leq \frac{2(DR)^2 + 2DR\sqrt{(DR)^2 + 4L} + 4L}{4}$$

Now if we look at the equation that our derivation should end up looking like:

$$M \leq L + O\left(\sqrt{L}\right) + O(1)$$

We see that we have a match as we have an L, a constant $2(DR)^2$ and a $O\left(\sqrt{L}\right)$ as shown by $2DR\sqrt{(DR)^2 + 4L}$.

Q2.2 Multi-class perceptron

Q2.2.1 Understanding Multi-class Hinge Loss

Answer:

If the algorithm does not make a mistake that is the case where $\hat{y} = y$. In such a situation, the right-handed term encapsulated within the hinge loss maximum expression 1 - $(W^{y_T}x_T - \max_{\underline{j\epsilon\{1,2,..,k\}}} W^j x_T)$ needs to smaller than 0, as the loss is 0 for cases where the algorithm does not make a mistake.

In order for this to be the case, $W^{y_T}x_T - \max_{\underline{j\epsilon\{1,2,..,k\}}} W^j x_T$ has to be larger than 1 so that the right hand side is negative. This would work in this situation and the given definition for multi-class hinge loss would work.

Now if the agent were not perfectly classifying but was only a tiny bit off, yes the loss would no longer be 0, but would be a positive value as shown by the right hand term 1 - $(W^{y_T}x_T - \max_{\underline{j\epsilon\{1,2,..,k\}}} W^j x_T)$.
However $W^{y_T}x_T - \max_{\underline{j\epsilon\{1,2,..,k\}}} W^j x_T$ would be smaller than 1 in this situation, which fits within bounds, as your computed loss is still less than 1, and in fact if the difference between $W^{y_T}x_T$ and the best performance over all other classes was small (showing that you are assymptoting towards good performance), you would be subtracting a large value smaller than 1 from 1, meaning that your loss would be a small value, closer to 0 than 1. This is perfectly logical when you think about it.

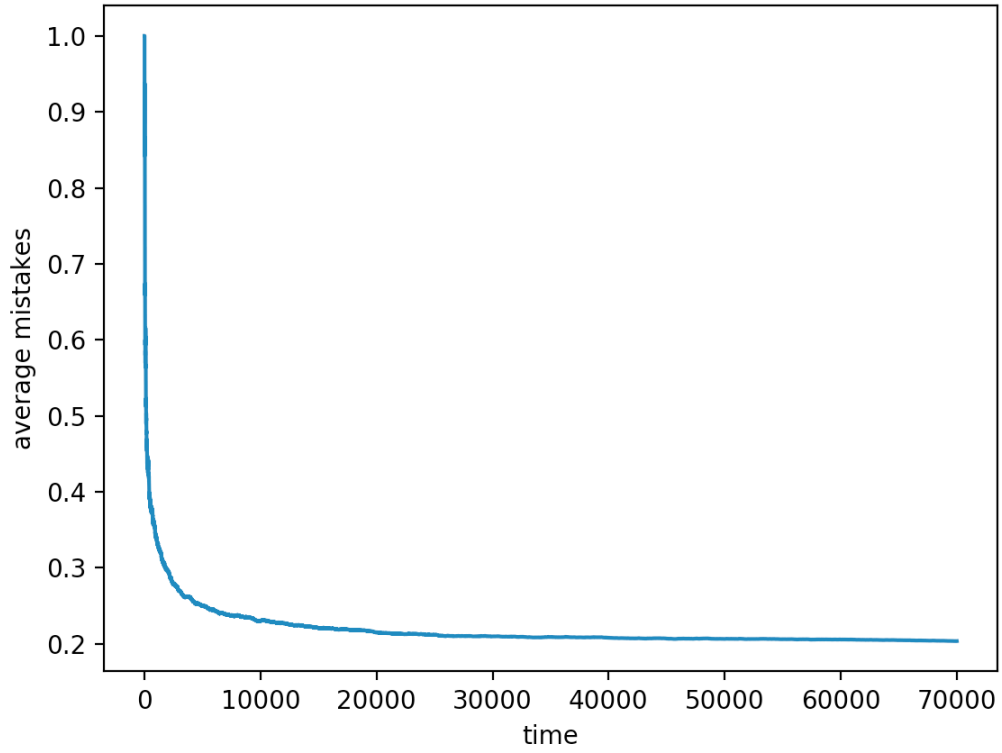
Now if your algorithm misclassifies badly, then obviously the loss cannot be 0, and so you end up with the right-hand term needing to be larger and larger, therefore $W^{y_T}x_T - \max_{\underline{j\epsilon\{1,2,..,k\}}} W^j x_T$ needs to be more and more negative. This means as you make mistakes, loss becomes larger and larger than 1.

By the way if the algorithm were started and initially misclassified, then the loss would be 1 as you would have $1 - (0 - 0)$, which is 1.

So all in all, this hinge loss function makes sense in this context of a multi-class classification problem.

Q2.3 Implementing Perceptron

Below is the output chart of the perceptron's average mistakes over 70,000 cycles.



3 Adaboost (15 Points)

Q3.1 Bounding the weight of a single point

Answer:

Looking at the start of the Adaboost preamble, we are presented with this equation:

$$p_{T+1} = \frac{p_T(m)}{Z_t} \begin{cases} e^{(-\beta_t)} & if\ y_m = h_t(x_m) \\ e^{(\beta_t)} & if\ y_m \neq h_t(x_m) \end{cases}$$

This can be expressed as:

$$p_{T+1} = \frac{p_T(m)}{Z_t} * e^{(-\beta_t h_t y_m)}$$

Shifting the denominator over and then modelling how this changes through time and we get:

$$p_{T+1} = \frac{e^{(-\beta_t h_t y_m)}}{Z_t} * \frac{e^{(-\beta_{t-1} h_{t-1} y_m)}}{Z_{t-1}} * \dots * \frac{e^{(-\beta_1 h_1 y_m)}}{Z_1} * p_1(m)$$

Now, the pseudocode for the algorithm, shows an initialization of $p_1(m)$ as $\frac{1}{M}$ for m = 1, …, M. Therefore $p_1(m)$ can be expressed as $\frac{1}{M}$ and when combining that with all this products, and product of $e^{(x)}$ into natural logs of sums we get:

$$p_{T+1} = \frac{1}{M} \frac{e^{(\sum_{t=1}^{T}(-\beta_t h_t y_m))}}{\prod_{t=1}^{T} Z_t}$$

And since $-\beta_t h_t$ is identical to $f(x_m)$ we get:

$$p_{T+1} = \frac{1}{M} \frac{\left(e^{-y_m f(x_m)}\right)}{\prod_{t=1}^{T} Z_t} \qquad (9)$$

Now we want to find the lower bound. Now if the classifier makes a mistake, the value of $y_m f(x_m)$ is less than or equal to 0. So from equation (9) we get a lower bound when $y_m f(x_m)$ is 0, which thus means that:

$$p_{lower\ bound} = \frac{1}{M} \frac{(e^0)}{\prod_{t=1}^{T} Z_t}$$

$$\therefore p_{lower\ bound} = \frac{1}{M \prod_{t=1}^{T} Z_t}$$

Q3.2 Bounding error using normalizing weights

Answer:

Let's begin with one of the equations provided in the preamble to question 3:

$$\epsilon = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{l}[y_m \neq h_F(x_m)] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

We focus just on:

$$\epsilon = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{l}[y_m \neq h_F(x_m)]$$

We can express this equation as:

$$\epsilon = \frac{1}{M}\sum_{m=1}^{M} e^{-y_m f(x_m)}$$

Now using the equation from the previous question (equation (9)):

$$p_{T+1} = \frac{1}{M}\frac{\left(e^{-y_m f(x_m)}\right)}{\prod_{t=1}^{T} Z_t}$$

We can see that:

$$\epsilon \leq \prod_{t=1}^{T} Z_t * \sum_{m=1}^{M} p_{T+1}$$

We can now recall that the sum of all probabilities is 1 so:

$$\epsilon \leq \prod_{t=1}^{T} Z_t * 1$$

Q3.3 How many iterations should you run AdaBoost

Answer:

We can start off by computing the log of the weight error function. So we compute the log of the following (we want to get rid of the exponents):

$$\epsilon = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{l}[y_m \neq h_F(x_m)] \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

But before we do, we can simplify this equation into:

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

Now after computing the log in order to bring T down from 2 we get:

$$\log(\epsilon) \leq \log\left(2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}\right)$$

The log expression can be split into:

$$\log(\epsilon) \leq \log(2^T) + \log\left(\prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}\right)$$

A log of a product becomes a sum, therefore:

$$\log(\epsilon) \leq T\log(2) + \log\left(\sum_{t=1}^{T} (\,\epsilon_t(1-\epsilon_t)\,)^{0.5}\right)$$

$$\therefore \log(\epsilon) \leq T\log(2) + 0.5\sum_{t=1}^{T} (\,\log(\,\epsilon_t) + log(1-\epsilon_t)\,)$$

Now, referring back to the question, we can see that the weak learner has an error over some data distribution given by:

$$\Pr\left[err_p(h) \leq \frac{1}{2} - \gamma\right] \geq 1-\delta$$

Ideally if it is learning, $err_p(h)$ should be decreasing, and so from the initial baseline of performing randomly and getting better than random, $\frac{1}{2} - \gamma$ must be decreasing. As a result of this, γ must be increasing. Now that we understand that γ must increase, we can make another assumption that allows us to later place all of the Ts together one on side. It is that:

$$0.5 \sum_{t=1}^{T} (\log(\epsilon_t) + log(1 - \epsilon_t)) \approx 0.5T (\log(\epsilon_t) + log(1 - \epsilon_t))$$

We can also remember that $\epsilon = \frac{1}{2} - \gamma$

$$\therefore \log\left(\frac{1}{2} - \gamma\right) \leq T \log(2) + 0.5T (\log\left(\frac{1}{2} - \gamma\right) + log\left(1 - \frac{1}{2} - \gamma\right))$$

$$\therefore \log\left(\frac{1}{2} - \gamma\right) \leq T (\log(2) + 0.5 (\log\left(\frac{1}{2} - \gamma\right) + log\left(1 - \frac{1}{2} - \gamma\right)))$$

$$\therefore \frac{\log\left(\frac{1}{2} - \gamma\right)}{(\log(2) + 0.5 (\log\left(\frac{1}{2} - \gamma\right) + log\left(-\frac{1}{2} - \gamma\right)))} \leq T$$

4. Soft SVM (30 Points)
4. 1 Understanding hyperplanes

4.1.1 What is a hyperplane in 2D? What is a hyperplane in 3D?

Answer:

Generally speaking a hyperplane is a sub-space whose dimension is 1 less than that of its ambient space.

So in the case of a hyperplane in 2D, the hyperplanes of the 2D space are the 1D lines.

In the case of a hyperplane in 3D, the hyperplanes are the 2D planes.

4.1.2. If w = (2, 1, 1), and x = (x1, x2, 1) what does the hyperplane look like? Plot the hyperplane with python, or with an online tool, and provide an image of the plot.

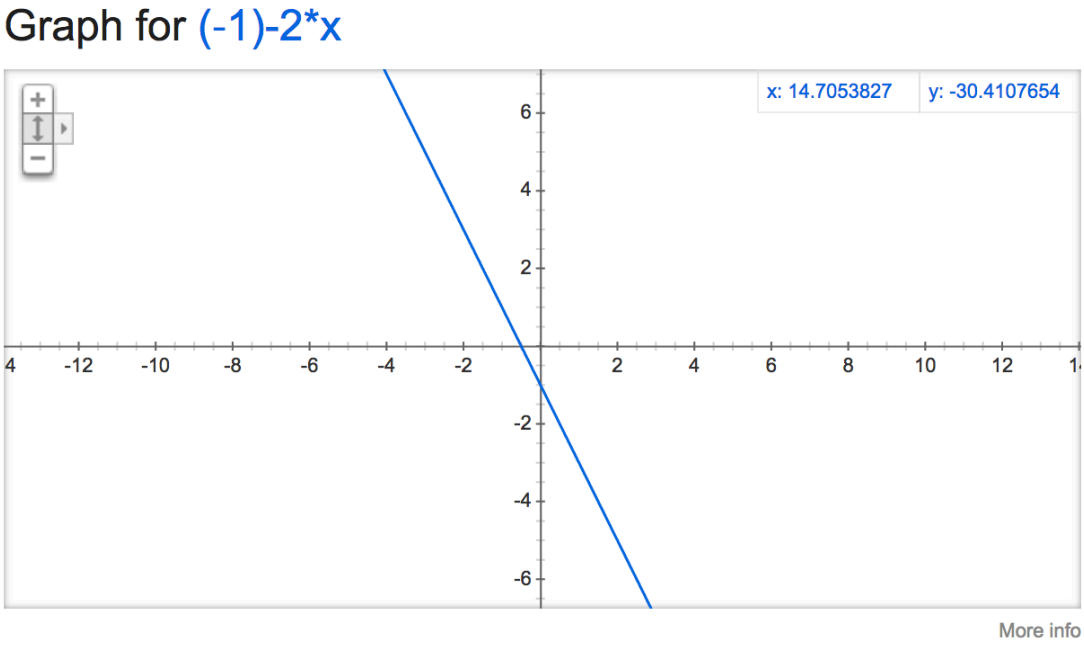
Answer:

The hyperplane can be re-expressed as: $2x + y + 1 = 0 \rightarrow 2x + y = -1$

This is the equation of a line in standard form, and in slope intercept form it is:

$$y = -1 - 2x$$

This is what it looks like this
:



4.1.3. Provide three example inputs x, such that: one is on the hyperplane, and the other two are on either side. Explain how you know where these points land.

Answer:

The equations to keep in mind for the hyperplane are:
$w \cdot x > N$ for positive examples
$w \cdot x < N$ for negative examples
$w \cdot x = N$ for examples on the hyperplane

So this is just a question of dot producting vectors and figuring out if the result is equal to or greater or less than N. In this case, w = (2, 1, 1), and x = (x1, x2, 1).

$$\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = N$$

$$2x_1 + x_2 + 1 = 0$$

$$2x_1 + x_2 = -1$$

This is for on the plane, possible points include:

$$2(1) + (-3) = -1$$

So (1,-3) lies on this hyperplane if (x, y) is identical to (x_1, x_2)

Points that would be positive examples include:

$$2x_1 + x_2 > -1$$

$$2(4) + (2) > -1$$
$$10 > -1$$

So (4,2) lies above the plane (is positive) if (x, y) is identical to (x_1, x_2)

Points that would be negative examples include:

$$2x_1 + x_2 < -1$$

$$2(-3) + (2) < -1$$
$$-4 > -1$$

So (-3,2) lies above the plane (is positive) if (x, y) is identical to (x_1, x_2)

We know these points land where they do because the equations provided in the beginning of this solution show where points stand in relation to the hyperplane.

4.1.4. What is the normal vector of this hyperplane? What is the distance between the hyperplane and the origin (0, 0)?

Answer:

The norm vector is = (2,1)

The distance between the hyperplane and the origin is: $\sqrt{2^2 + 1^1} = \sqrt{5}$

The normalized norm vector (unit length) is:

$$\bar{v} = \frac{b}{||w||} v$$

$$\bar{v} = \frac{1}{\sqrt{5}|} (2,1)$$

4.1.5. If we use instead w = (40, 20), and x = (x1, x2), and a bias term b = 1, do we end up with the same hyperplane, or a different one? Why is this?

Answer:

We end up in a different hyperplane.

The reason goes back to this equation:

$$\bar{v} = \frac{b}{||w||} v$$

What we are changing here is the denominator, as we are keeping the bias term b the same. We now have a distance from the origin of: $\sqrt{40^2 + 20^1} = 47$.

Therefore this is an offset from the origin in perpendicular to the orientation of the previous hyperlane. They thus do not occupy the same hyperplane, but rather a an entirely new hyperplane of matching dimension and orientation (normal vector), just offset from original one.

4.2 Subgradients

4.2.1. Rewrite the hinge loss in English. (Two sentences.)

Answer:

Hinge loss is used for classification losses, whereby a hinge loss of 0 signals that the classification is not correct, and a hinge loss of 1 signals that classification is excellent.

4.2.2. Is the hinge loss convex?

Answer:

Yes the hinge loss is convex. Hinge loss is used to provide a convex upper bound to a non-convex problem.

4.2.3. Is the hinge loss differentiable?

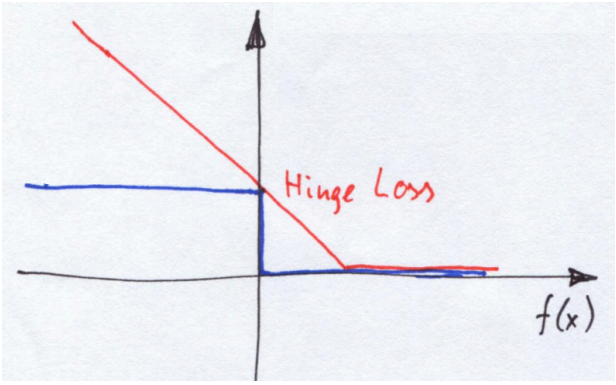
Answer:

No the hinge loss is not differentiable

4.2.4. What are the sub-gradients of the hinge loss?

Answer:

Before I include my answer, I just want to illustrate something:



Hinge loss creates a convex upper bound to step loss signals, which are binary in nature. Now the nature of the shape of a hinge loss is such that its equation can be represented as a piecewise equation, where the hinge loss has a non-

differentiable behavior, and where we have sub-gradients.

So in the case of the picture above, the hinge loss has a section which is flat and horizontal where the gradient is 0, and it has an angled section, where the gradient is equal to derivative of the expression for hinge loss, with respect to weighted x_t. So if hinge loss is:

$$\ell\big(w, (x_t, y_t)\big) = \max\{0, 1 - y_t x_t^T w\}$$

Then the sub-gradient that is not 0 is equal to:

$$\frac{\partial \ell}{\partial w} = -y_t x_t^T$$

Nonetheless 0 is also a sub-gradient.

This image, is an edited one that is can be found on page 5 and 6 of this link from the technical university of Dresden:
http://www1.inf.tu-dresden.de/~ds24/lehre/ml_ws_2013/ml_11_hinge.pdf

4.3 Implementing the Soft SVM

4.3.1 What was the algorithm's accuracy on each class?

Answer:

Below is the output that I get from running the code, if $\lambda = 0.1$

```
Lukas-MBP:Robostats hw2 lukaeerens$ python3 svm.py
read all of the data
shuffled
picked 100975 for train, 25244 for val, out of 126219 total
feature length N = 10
got 811/1000 right on class 1004
got 977/1000 right on class 1100
got 979/1000 right on class 1103
got 958/1000 right on class 1200
got 869/1000 right on class 1400
got 809/1000 right on multiclass
Lukas-MBP:Robostats hw2 lukaeerens$
```

4.3.2 What were your hyperparameter choices? Explain these choices.

Answer:

This λ value was manipulated in order to track how the learning changed. Below is a tabulation of these accuracy scores for each class, as well as for the multiclass accuracy. Note that the score is in terms of % that are correct:

λ \Class	**1004**	**1100**	**1103**	**1200**	**1400**	**Multi**
0.01	81.4	98.3	97.7	36.0	86.6	83.0
0.05	83.7	97.2	98.4	98.9	86.4	80.9
0.1	81.1	97.7	97.9	95.8	86.9	80.9
0.5	83.2	97.8	98.7	35.1	85.2	74.8
1	80.7	97.2	97.9	34.5	86.1	74.7

It seems that there is a gradual improvement in classification accuracy as you decrease λ however for some reason class 1200 has a significantly lower accuracy than the other ones. It is probably responsible for dragging down the multiclass accuracy as it usually lies at just over a third of the accuracy of the other classes.

4.3.3 How long does the algorithm take (in terms of number of points, classes, iterations, feature dimensions, etc.) to train and predict?

Answer:

Number of Points:

Quantity	500	1000	2000
Time (seconds)	0.14	0.27	0.37

Number of Classes:

Quantity	3	5	7
Time (seconds)	0.25	0.27	0.30

Iterations:

Quantity	500	1000	2000
Time (seconds)	0.19	0.27	0.33