COMP9417 - Machine Learning
Tutorial: Classification

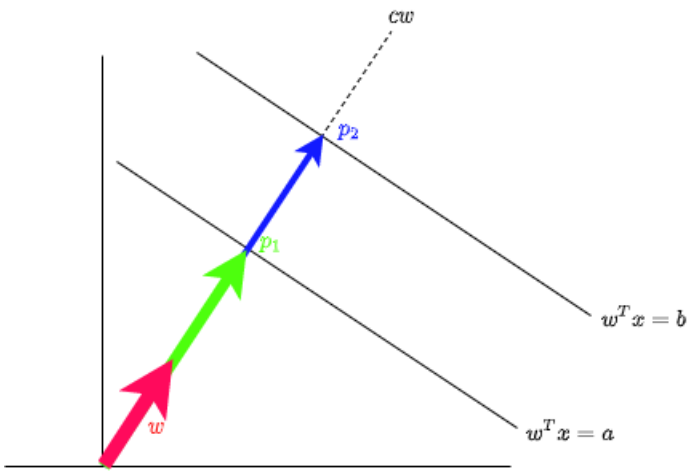
Question 1. Distance between parallel hyperplanes

When constructing linear classifiers, a common calculation that comes up is to compute the distance between two parallel hyperplanes. Consider two parallel hyperplanes: $H_1 = \{x \in \mathbb{R}^n : w^T x = a\}$ and $H_2 = \{x \in \mathbb{R}^n : w^T x = b\}$. Show that the distance between H_1 and H_2 is given by $\frac{|b-a|}{\|w\|_2}$.

Hint: draw a picture.

Solution:

Consider the following graphical depiction of H_1 and H_2:



Both hyperplanes are perpendicular to w, which is depicted in red. The black dotted line is a trace-out of all possible multiples of w by a non-negative constant c, since multiplying a vector by a constant does not change the direction, only the length. Two particular vectors on this line are important here, namely p_1 and p_2, which are the vectors that have the same direction as w and lie on H_1 and H_2 respectively. To compute the distance between H_1 and H_2, we need to just compute the distance between p_1 and p_2.

To do so, we note two pieces of information about p_1: first, $p_1 = dw$ for some constant d, and second, since p_1 is on H_1, we also know that $w^T p_1 = a$. Using these two facts together we can

conclude that

$$\begin{aligned} w^T p_1 = a \text{ and } p_1 = dw &\implies w^T(dw) = a \\ &\implies d = \frac{a}{\|w\|_2^2} \\ &\implies p_1 = \frac{a}{\|w\|_2^2} w. \end{aligned}$$

A similar calculation for p_2 yields

$$p_2 = \frac{b}{\|w\|_2^2} w.$$

Now, the distance between p_1 and p_2 is:

$$\begin{aligned} \|p_2 - p_1\|_2 &= \left\| \frac{b}{\|w\|_2^2} w - \frac{a}{\|w\|_2^2} w \right\|_2 \\ &= \left\| (b-a) \frac{1}{\|w\|_2^2} w \right\|_2 \\ &= |b-a| \frac{1}{\|w\|_2^2} \|w\|_2 \\ &= \frac{|b-a|}{\|w\|_2}. \end{aligned}$$

This equation will come up numerous times throughout the course, make sure you commit it to memory (as well as understand the derivation of course).

Question 2 (Perceptron Training & Capacity)

(a) Consider the following training data:

x_1	x_2	y
-2	-1	-1
2	-1	1
1	1	1
-1	-1	-1
3	2	1

Apply the Perceptron Learning Algorithm with starting values $w_0 = 5$, $w_1 = 1$ and $w_2 = 1$, and a learning rate $\eta = 0.4$. Be sure to cycle through the training data in the same order that they are presented in the table. Present your results in table form:

Iteration	$\langle w, x \rangle$	$y\langle w, x \rangle$	w

Solution:
Running the Perceptron training algorithm results in the following iterations, where highlighted columns signify that the weight vector is changing on that particular iteration. The

correct weights are therefore $w_0 = 3.8, w_1 = 2.6, w_2 = 2.2$. These weights first appear on iteration 9.

Iteration	$\langle w, x \rangle$	$y\langle w, x \rangle$	w
1	2.00	−2.00	$[4.6, 1.8, 1.4]^T$
2	6.80	6.80	$[4.6, 1.8, 1.4]^T$
3	7.80	7.80	$[4.6, 1.8, 1.4]^T$
4	1.40	−1.40	$[4.2, 2.2, 1.8]^T$
5	14.40	14.40	$[4.2, 2.2, 1.8]^T$
6	−2.00	2.00	$[4.2, 2.2, 1.8]^T$
7	6.80	6.80	$[4.2, 2.2, 1.8]^T$
8	8.20	8.20	$[4.2, 2.2, 1.8]^T$
9	0.20	−0.20	$[3.8, 2.6, 2.2]^T$
10	16.00	16.00	$[3.8, 2.6, 2.2]^T$
11	−3.60	3.60	$[3.8, 2.6, 2.2]^T$
12	6.80	6.80	$[3.8, 2.6, 2.2]^T$
13	8.60	8.60	$[3.8, 2.6, 2.2]^T$
14	−1.00	1.00	$[3.8, 2.6, 2.2]^T$

(b) Consider the following three logical functions:

1. $A \wedge \neg B$
2. $\neg A \vee B$
3. $(A \vee B) \wedge (\neg A \vee \neg B)$

Which of these functions can a perceptron learn? Explain.

Solution:
A Perceptron learns a linear classifier, and so it will only be able to learn functions that are linearly separable. Plotting the above functions shows clearly that f_1 and f_2 can be learned, but f_3 cannot.

Question 3. Binary Logistic Regression, two perspectives

Recall from previous weeks that we can view least squares regression as a purely optimisation based problem (minimising MSE), or as a statistical problem (using MLE). We now discuss two perspectives of the Binary Logistic Regression problem. In this problem, we are given a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ where the x_i's represent the feature vectors, just as in linear regression, but the y_i's are now binary. The goal is to model our output as a probability that a particular data point belongs to one of two classes. We will denote this predicted probability by

$$P(y = 1|x) = p(x)$$

and we model it as

$$\hat{p}(x) = \sigma(\hat{w}^T x), \qquad \sigma(z) = \frac{1}{1 + e^{-z}},$$

where $\hat{w}$ is our estimated weight vector. We can then construct a classifier by assigning the class that

has the largest probability, i.e.:

$$\hat{y} = \arg\max_{k=0,1} P(\hat{y} = k|x) = \begin{cases} 1 & \text{if} \quad \sigma(\hat{w}^T x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

note: do not confuse the function $\sigma(z)$ with the parameter σ which typically denotes the standard deviation.

(a) What is the role of the logistic sigmoid function $\sigma()$ in the logistic regression formulation? Why are we not able to simply use linear regression here? (a plot of $\sigma(z)$ may be helpful here).

Solution:
The problem with just modelling $\hat{p}(x) = \hat{w}^T x$ is that $\hat{w}^T x$ is an unbounded quantity, which means that $\hat{w}^T x$ can be any number on the real line. This doesn't make sense since we wish to model $\hat{p}$ as a probability, and probabilities belong to the interval $[0, 1]$. The role of the logistic sigmoid, also referred to as a *squashing function* is to squash the *raw score* $w^T x$ into the desired interval.

(b) We first consider the statistical view of logistic regression. Recall in the statistical view of linear regression, we assumed that $y|x \sim N(x^T \beta^*, \sigma^2)$. Here, we are working with binary valued random variables and so we assume that

$$y|x \sim \text{Bernoulli}(p^*), \qquad p^* = \sigma(x^T w^*)$$

where $p^* = \sigma(x^T w^*)$ is the true unknown probability of a response belonging to class 1, and we assume this is controlled by some true weight vector w^*. Write down the log-likelihood of the data D (as a function of w), and further, write down the MLE objective (but do not try to solve it).

Solution:
The assumption implies that

$$P(y = 1|x) = p^y (1 - y)^{1-y},$$

so we can write down the log-likelihood as

$$\begin{aligned} \ln L(w) &= \ln P(y_1, \ldots, y_n | x_1, \ldots, x_n) \\ &= \ln \left\{ \prod_{i=1}^{n} P(y_i|x_i) \right\} \\ &= \sum_{i=1}^{n} \ln P(y_i|x_i) \\ &= \sum_{i=1}^{n} \ln(p_i^{y_i} (1 - p_i)^{1-y_i}) \\ &= \sum_{i=1}^{n} [y_i \ln p_i + (1 - y_i) \ln(1 - p_i)]. \end{aligned}$$

At this point, we recall that

$$p_i = \sigma(w^T x_i) = \frac{1}{1 + e^{-w^T x_i}},$$

and so

$$\begin{aligned}\ln L(w) &= \sum_{i=1}^{n} [y_i \ln \sigma(w^T x_i) + (1 - y_i) \ln(1 - \sigma(w^T x_i))] \\ &= \sum_{i=1}^{n} \left[y_i \ln \left(\frac{\sigma(w^T x_i)}{1 - \sigma(w^T x_i)} \right) + \ln(1 - \sigma(w^T x_i)) \right].\end{aligned}$$

The MLE optimisation is then

$$\hat{w}_{\text{MLE}} = \arg \max_{w \in \mathbb{R}^p} \ln L(w).$$

Note that we cannot solve this problem by hand as in the linear regression case (you should try this for yourself). This is an example in which we rely solely on numerical methods to find a solution.

(c) An alternative approach to the logistic regression problem is to view it purely from the optimisation perspective. This requires us to pick a loss function and solve for the corresponding minimizer. Write down the MSE objective for logistic regression and discuss whether you think this loss is appropriate.

Solution:
The MSE objective is simply

$$\arg\min_w \frac{1}{n} \|y - \sigma(Xw)\|_2^2$$

where $y = [y_1, \ldots, y_n]^T$ and $\sigma(Xw)$ is the element-wise application of σ to the vector Xw, or we could equivalently write:

$$\arg\min_w \frac{1}{n} \sum_{i=1}^{n} (y_i - \sigma(w^T x_i))^2.$$

This objective makes sense mathematically, but the squared error does not seem like a particularly good way to measure the distance between the resposne y_i and our prediction $\sigma(w^T x_i)$. The reason being that y_i is binary, whereas our prediction is real-valued. There are more technical reasons for not wanting to use the MSE, in particular the MSE is not convex in the parameter vector w for this problem, and this makes it harder to optimize, but this is beyond the scope of the course.

(d) Consider the following problem: you are given two discrete probability distributions, P and Q, and you are asked to quantify how far Q is from P. This is a very common task in statistics and information theory. The most common way to measure the discrepancy between the two is

to compute the Kullback-Liebler (KL) divergence, also known as the relative entropy, which is defined by:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\ln\frac{P(x)}{Q(x)},$$

where we are summing over all of the possible values of the underlying random variable. A good way to think of this is that we have a true distribution P, an estimate Q, and we are trying to figure out how bad our estimate is. Write down the KL divergence between two bernoulli distributions $P = \text{Bernoulli}(p)$ and $Q = \text{Bernoulli}(q)$.

Solution:
A Bernoulli random variable can take on only two values (namely $x = 0, 1$) and so we have:

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= \sum_{x=0}^{1} P(x)\ln\frac{P(x)}{Q(x)} \\ &= P(X=0)\ln\frac{P(X=0)}{Q(X=0)} + P(X=1)\ln\frac{P(X=1)}{Q(X=1)} \\ &= (1-p)\ln\frac{1-p}{1-q} + p\ln\frac{p}{q}. \end{aligned}$$

(e) Continuing with the optimisation based view: In our set-up, one way to quantify the discrepancy between our prediction $\hat{p}_i$ and the true label y_i is to look at the KL divergence between the two bernoulli distributions $P_i = \text{Bernoulli}(y_i)$ and $Q_i = \text{Bernoulli}(\hat{p}_i)$. Use this to write down an appropriate minimization for the logistic regression problem.

Solution:
We can simply define the KL loss as the sum of the KL divergences:

$$\begin{aligned} \mathcal{L}_{\mathrm{KL}}(w) &= \sum_{i=1}^{n} D_{\mathrm{KL}}(P_i\|Q_i) \\ &= \sum_{i=1}^{n} D_{\mathrm{KL}}(\text{Bernoulli}(y_i)\|\text{Bernoulli}(\hat{p}_i)) \\ &= \sum_{i=1}^{n}(1-y_i)\ln\frac{1-y_i}{1-\hat{p}_i} + y_i\ln\frac{y_i}{\hat{p}_i}. \end{aligned}$$

Note that we can expand the individual terms as:

$$\begin{aligned} \ln\frac{1-y_i}{1-\hat{p}_i} + y_i\ln\frac{y_i}{\hat{p}_i} &= (1-y_i)\ln(1-y_i) - (1-y_i)\ln(1-\hat{p}_i) + y_i\ln y_i - y_i\ln\hat{p}_i \\ &\overset{w}{\propto} -(1-y_i)\ln(1-\hat{p}_i) - y_i\ln\hat{p}_i. \end{aligned}$$

In the last step above, we have noted that only terms that involve $\hat{p}_i$ are functions of w, and so we can safely remove all other terms from consideration. This is because we are optimising

with respect to w, and so only need to consider terms that involve w. Plugging this back in shows that

$$\begin{aligned}\mathcal{L}_{\text{KL}}(w) &\overset{w}{\propto} -\sum_{i=1}^{n}[(1-y_i)\ln(1-\hat{p}_i) + y_i \ln \hat{p}_i] \\ &= -\sum_{i=1}^{n}\left[y_i \ln \frac{\hat{p}_i}{1-\hat{p}_i} + \ln(1-\hat{p}_i)\right] \\ &= -\sum_{i=1}^{n}\left[y_i \ln\left(\frac{\sigma(w^T x_i)}{1-\sigma(w^T x_i)}\right) + \ln(1-\sigma(w^T x_i))\right].\end{aligned}$$

(f) In logistic regression (and other binary classification problems), we commonly use the cross-entropy loss, defined by

$$\mathcal{L}_{XE}(a,b) = -a\ln b - (1-a)\ln(1-b).$$

Using your result from the previous part, discuss why the XE loss is a good choice, and draw a connection between the statistical and optimisation views of logistic regression.

Solution:
We can easily see that minimizing the cross entropy loss is equivalent to minimizing the KL loss from the previous question, so the XE loss is a valid objective function. For concreteness, we have

$$\mathcal{L}_{\text{KL}}(w) \overset{w}{\propto} \mathcal{L}_{\text{XE}}(w),$$

and so it follows immediately that

$$\arg\min_w \mathcal{L}_{\text{KL}}(w) = \arg\min_w \mathcal{L}_{\text{XE}}(w),$$

so the XE gives us the same solution as does minimizing the KL divergence. Now, using our result from the previous question and comparing it to our MLE derivation above, we see that

$$\begin{aligned}\hat{w}_{\text{MLE}} &= \arg\max_{w\in\mathbb{R}^p} \ln L(w) \\ &= \arg\max_{w\in\mathbb{R}^p} \sum_{i=1}^{n}\left[y_i \ln\left(\frac{\sigma(w^T x_i)}{1-\sigma(w^T x_i)}\right) + \ln(1-\sigma(w^T x_i))\right] \\ &= \arg\min_{w\in\mathbb{R}^p} -\sum_{i=1}^{n}\left[y_i \ln\left(\frac{\sigma(w^T x_i)}{1-\sigma(w^T x_i)}\right) + \ln(1-\sigma(w^T x_i))\right] \\ &= \arg\min_{w\in\mathbb{R}^p} \mathcal{L}_{\text{KL}}(w) \\ &= \arg\min_{w\in\mathbb{R}^p} \mathcal{L}_{\text{XE}}(w).\end{aligned}$$

So, what we see is that minimizing the XE loss gives us the same solution as would maximizing the log-likelihood. This is similar to what we saw in the linear regression case: using the MSE is equivalent to maximizing the likelihood under the Gaussian assumption. We can think of this as: 'MSE is to Gaussians as XE is to Bernoullis'

Question 4. Numerically solving the logistic regression problem

In the previous problem, we show that in order to solve the logistic regression problem, we must solve the following optimisation:

$$\begin{aligned}\hat{w} &= \arg\min_w \mathcal{L}(w) \\ &= \arg\min_w -\left[\sum_{i=1}^{n} y_i \ln p_i + (1-y_i)\ln(1-p_i)\right],\end{aligned}$$

where

$$p_i = \sigma(w^T x_i) = \frac{1}{1+e^{-w^T x_i}}.$$

Unfortunately in this case, we cannot solve for $\hat{w}$ in closed form. In other words, we cannot simply take derivatives, equate to zero and solve to get a nice solution as in the linear regression case. Instead, we must rely on numerical techniques such as gradient descent. In this question, we will work through and derive the gradient descent updates for the logistic regression problem.

(a) We will need to take derivatives in order to do any form of gradient descent. Show that

$$\sigma'(z) = \sigma(z)(1-\sigma(z)).$$

Then use this result to show that

$$\frac{dp_i}{dw} = p_i(1-p_i)x_i,$$

where $p_i = \sigma(w^T x_i)$.

Solution:
The first result follows by a standard application of chain rule, see HW0 for more details. The second result also follows immediately by the chain rule:

$$\frac{dp_i}{dw} = \frac{d\sigma(w^T x_i)}{dw} = \frac{d\sigma(w^T x_i)}{d(w^T x_i)}\frac{d(w^T x_i)}{dw} = \sigma(w^T x_i)(1-\sigma(w^T x_i))x_i = p_i(1-p_i)x_i.$$

(b) Use the previous result to show that

$$\frac{d\ln p_i}{dw} = (1-p_i)x_i.$$

Solution:
Using the chain rule again gives us:

$$\frac{d\ln p_i}{dw} = \frac{d\ln p_i}{dp_i}\frac{dp_i}{dw} = \frac{1}{p_i}p_i(1-p_i)x_i = (1-p_i)x_i$$

(c) Using the results of the previous parts, compute

$$\frac{d\mathcal{L}(w)}{dw}$$

and write down the gradient descent update for w with step size η.

Solution:

$$\begin{aligned}
\frac{d\mathcal{L}(w)}{dw} &= \frac{d}{dw}\left[-\sum_{i=1}^{n} y_i \ln \hat{p}_i + (1-y_i)\ln(1-\hat{p}_i)\right] \\
&= -\sum_{i=1}^{n}\left[y_i \frac{d}{dw}\ln \hat{p}_i + (1-y_i)\frac{d}{dw}\ln(1-\hat{p}_i)\right] \\
&= -\sum_{i=1}^{n}\left[y_i(1-\hat{p}_i)x_i + (1-y_i)\left(-\frac{1}{1-\hat{p}_i}\hat{p}_i(1-\hat{p}_i)x_i\right)\right] \\
&= -\sum_{i=1}^{n}\left[y_i(1-\hat{p}_i)x_i - (1-y_i)\hat{p}_i x_i\right] \\
&= -\sum_{i=1}^{n}(y_i - \hat{p}_i)x_i.
\end{aligned}$$

Therefore, the gradient descent update at iteration $k+1$ is

$$w^{(k+1)} = w^{(k)} + \eta\sum_{i=1}^{n}(y_i - \hat{p}_i)x_i.$$

(d) A convex function does not have any local minima, and so we are guaranteed to converge to a global minimum when doing gradient descent on a convex function, regardless of our initialisation $w^{(0)}$. Prove that $\mathcal{L}(w)$ is convex.

Solution:
Recall that the Hessian of a function, $H \in \mathbb{R}^{p\times p}$ is just the matrix of second order partial derivatives of the function, and so has (k,l)-th element:

$$\begin{aligned}
H_{kl} &= \frac{\partial^2 \mathcal{L}(w)}{\partial w_k \partial w_l} \\
&= \frac{\partial}{\partial w_k}\frac{\partial \mathcal{L}(w)}{\partial w_l} \\
&= \frac{\partial}{\partial w_k}\left(-\sum_{i=1}^{n}(y_i - \hat{p}_i)x_{il}\right) \\
&= \sum_{i=1}^{n}\hat{p}_i(1-\hat{p}_i)x_{ik}x_{il},
\end{aligned}$$

where x_{il} is the l-th feature of the i-th input vector x_i. From here, it is straight forward to conclude that

$$H = \sum_{i=1}^{n} \hat{p}_i(1 - \hat{p}_i) x_i x_i^T .$$

Now, recall from the previous tutorial that a function is convex if its Hessian is positive semi-definite, i.e. that for any non-zero vector u,

$$u^T H u \geq 0.$$

In our case, we have

$$\begin{aligned} u^T H u &= \sum_{i=1}^{n} \hat{p}_i(1 - \hat{p}_i)(u^T x_i)(u^T x_i) \\ &= \sum_{i=1}^{n} \hat{p}_i(1 - \hat{p}_i)(u^T x_i)^2 \\ &\geq 0, \end{aligned}$$

where the last line holds since $\hat{p}_i \in (0, 1)$ and $(u^T x_i)^2 \geq 0$.