Exercises for Artificial Intelligence

Dr Sean B Holden, 2010-20

1 Learning

1. The purpose of this exercise is to gain some insight into the way in which the parameters of a basic, linear perceptron affect the position and orientation of its decision boundary. Recall that a linear perceptron is based on the function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The perceptron decides that a new input $\mathbf{x}$ is in class 1 if $f(\mathbf{x}) \geq 0$ and decides that the input is in class 2 otherwise. The decision boundary is therefore the collection of all points where $f(\mathbf{x}) = 0$.

It's always easy to find n distinct points where $f(\mathbf{x}) = 0$ because for any $\mathbf{w}$ and b we just need to solve

$$\mathbf{w}^T\mathbf{x}' = -b$$

which is easy using

$$\mathbf{x}^T = \begin{pmatrix} (-b/w_1) & 0 & \cdots & 0 \end{pmatrix}$$
$$\mathbf{x}^T = \begin{pmatrix} 0 & (-b/w_2) & \cdots & 0 \end{pmatrix}$$

and so on. If any of the weights is 0 this is problematic but easy to fix. (I leave it as a warm-up exercise to work out how.) Let $\mathbf{x}'$ and $\mathbf{x}''$ be two points where $f(\mathbf{x}') = 0$ and $f(\mathbf{x}'') = 0$. Let's concentrate on the case where $n = 2$. Consider the vector

$$\mathbf{y} = \mathbf{x}' - \mathbf{x}''.$$

Now take any number $a \in \mathbb{R}$ and look at what happens if we evaluate

$$f(\mathbf{x}' + a\mathbf{y}).$$

We obtain

$$\begin{aligned} f(\mathbf{x}' + a\mathbf{y}) &= \mathbf{w}^T(\mathbf{x}' + a\mathbf{y}) + b \\ &= \mathbf{w}^T\mathbf{x}' + a\mathbf{w}^T\mathbf{y} + b \\ &= f(\mathbf{x}') + a\mathbf{w}^T(\mathbf{x}' - \mathbf{x}'') \\ &= a(\mathbf{w}^T\mathbf{x}' - \mathbf{w}^T\mathbf{x}'') \\ &= a(-b - (-b)) \\ &= 0. \end{aligned}$$

This works for *any* value $a \in \mathbb{R}$, and suggests that the decision boundary is a straight line in $\mathbb{R}^2$ as illustrated in figure 1. (Note however that we haven't yet demonstrated that $f(\mathbf{x}) \neq 0$ if $\mathbf{x}$ is not of the form $\mathbf{x} = \mathbf{x}' + a\mathbf{y}$.)

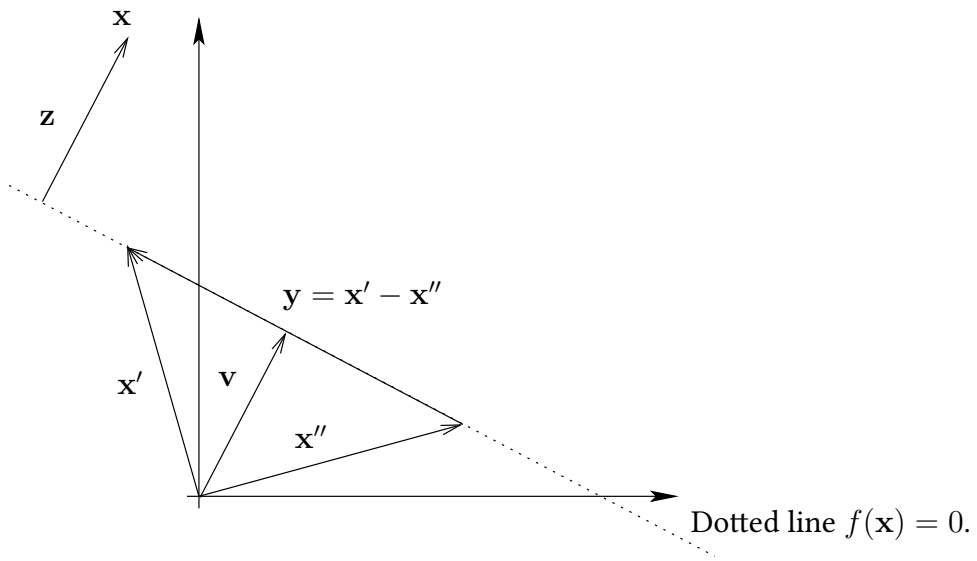

Figure 1: The decision boundary appears to be a straight line.

(a) Prove that the weight vector $\mathbf{w}$ is perpendicular to the line described by $\mathbf{x}' + a\mathbf{y}$; that is, the line corresponding to the set

$$\{\mathbf{x} | \mathbf{x} = \mathbf{x}' + a\mathbf{y} \text{ where } a \in \mathbb{R}\}.$$

(Hint: remember that vectors are perpendicular if their inner product is 0.) Note that this tells us that $\mathbf{w}$ describes the orientation of the decision boundary.

(b) Let $\mathbf{v}$ be the vector from the origin to the line described by $\mathbf{x}' + a\mathbf{y}$ and perpendicular to it as illustrated in figure 1. Prove that

$$||\mathbf{v}|| = \frac{|b|}{||\mathbf{w}||}.$$

Note that this tells us the following: if $||\mathbf{w}|| = 1$ then $|b|$ tells us the distance from the origin to the decision boundary.

(c) Let $\mathbf{x}$ be any point *not* on the line described by $\mathbf{x}' + a\mathbf{y}$. Let $\mathbf{z}$ be the vector from the line to $\mathbf{x}$ and perpendicular to the line as illustrated in figure 1. Prove that

$$||\mathbf{z}|| = \frac{|f(\mathbf{x})|}{||\mathbf{w}||}.$$

This tells us that points not on the line do not obey $f(\mathbf{x}) = 0$ and that the value of $f(\mathbf{x})$ tells us the distance from the decision boundary to $\mathbf{x}$.

(d) Prove that replacing $\mathbf{w}$ with $\mathbf{w}/||\mathbf{w}||$ and b with $b/||\mathbf{w}||$ does not alter the decision boundary.

2. In the application of neural networks to *pattern classification*—where we wish to assign any input vector $\mathbf{x}$ to membership in a specific *class*—it makes sense to attempt to interpret network outputs as probabilities of class membership.

For example, in the medical diagnosis scenario presented in the lectures, where we try to map an input $\mathbf{x}$ to either class A (patient has the disease) or class B (patient is free of the disease) it makes sense to use a network with a single output producing values constrained between 0 and 1 such that the output $h(\mathbf{w}; \mathbf{x})$ of a network using weights $\mathbf{w}$ is interpreted as

$$h(\mathbf{w}; \mathbf{x}) = \Pr(\mathbf{x} \text{ is in class } A).$$

Clearly we also have

$$\Pr(\mathbf{x} \text{ is in class } B) = 1 - h(\mathbf{w}; \mathbf{x})$$

and it follows that training examples should be labelled 1 and 0 for classes A and B respectively.

Say we have a specific training example $(\mathbf{x}', 0)$. What does it tell us about how to choose a good $\mathbf{w}$? Clearly we might want to choose $\mathbf{w}$ to maximise[1]

$$\begin{aligned}\Pr(\text{We see the example } (\mathbf{x}', 0)|\mathbf{w}) &= \Pr(\text{We see the label } 0|\mathbf{w}, \mathbf{x}') \times \Pr(\mathbf{x}') \\ &= \left\{1 - h(\mathbf{w}; \mathbf{x}')\right\} \times \Pr(\mathbf{x}')\end{aligned}$$

where the second step incorporates the assumption that $\mathbf{x}'$ and $\mathbf{w}$ are independent. This quantity is called the *likelihood* of $\mathbf{w}$. Given an entire training sequence

$$\mathbf{s} = ((\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \ldots, (\mathbf{x}_m, c_m))$$

where the labels c_i take values 0 or 1 we can also consider choosing $\mathbf{w}$ to maximise the probability that the entire collection of m input vectors is labelled in the specified manner (the likelihood $\Pr(\mathbf{s}|\mathbf{w})$ of $\mathbf{w}$).

Assuming that the examples in $\mathbf{s}$ are independent, show that in order to achieve this we should choose $\mathbf{w}$ to maximise the expression

$$\sum_{i=1}^{m} c_i \log h(\mathbf{w}; \mathbf{x}_i) + (1 - c_i) \log(1 - h(\mathbf{w}; \mathbf{x}_i)).$$

(Hint: When independence is assumed, $\Pr(A, B) = \Pr(A) \Pr(B)$, and you can maximise an expression equally well by maximising its $\log$.) What does this allow you to conclude about the version of the backpropagation algorithm presented in the lectures?

3. We now return to the case of regression. As in the previous question, the *likelihood* of a hypothesis h can be thought of as the probability of obtaining a training sequence $\mathbf{s}$ given that h is a perfect mapping from attribute vectors to classifications. Assume that $\mathcal{H}$ contains functions $h : X \to \mathbb{R}$ and examples are labelled using a specific target function $f \in \mathcal{H}$ but corrupted by noise, so

$$\mathbf{s} = ((\mathbf{x}_1, o_1), (\mathbf{x}_2, o_2), \ldots, (\mathbf{x}_m, o_m))$$

and

$$o_i = f(\mathbf{x}_i) + e_i$$

[1] The basic result in probability theory being used here is that $\Pr(A, B|C) = \Pr(A|B, C) \Pr(B|C)$. You might want at this point to review the relevant notes.

for $i = 1, 2, \ldots, m$ where e_i denotes noise. If the attribute vectors are fixed, and the e_i are independent and identically distributed with the Gaussian distribution

$$p(e_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(e_i - \mu)^2}{2\sigma^2}\right)$$

where μ is the noise mean and σ^2 the noise variance, then the likelihood of any hypothesis is

$$p(\mathbf{s}|h) = p((o_1, o_2, \ldots, o_m)|h) = \prod_{i=1}^{m} p(o_i|h)$$

where the last step follows because the e_i are independent. Assume in the following that $\mu = 0$.

(a) Show that the mean of o_i is $f(\mathbf{x}_i)$ and the variance of o_i is σ^2.

(b) Show that

$$p(o_i|h) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(o_i - h(\mathbf{x}_i))^2}{2\sigma^2}\right).$$

(Hint: what happens to data having a Gaussian density if you linearly transform it?)

(c) Show that any hypothesis that *maximises* the likelihood is also one that *minimises* the quantity

$$\sum_{i=1}^{m} (o_i - h(\mathbf{x}_i))^2.$$

(d) What does this tell you about the specific example of the backpropagation algorithm given in the lectures?

4. The demonstration of the backpropagation algorithm given in the lectures can be improved. In solving the parity problem what we really want to know is the *probability* that an example should be placed in class one, exactly as described above. Probabilities lie in the interval $[0, 1]$, but the output of the network used in the lectures is unbounded.

- Derive the modification required to the algorithm if the activation function on the output node is changed from $g(x) = x$ to $g(x) = 1/(1+\exp(-x))$. (This is a function commonly used to produce probabilities as outputs, as it has range lying between 0 and 1.)
- Implement the modified algorithm. (Matlab is probably a good language to use.) Apply it to the parity data described in the lectures, and plot the results you obtain.