6.867: Exercises (Week 6)

October 14, 2016

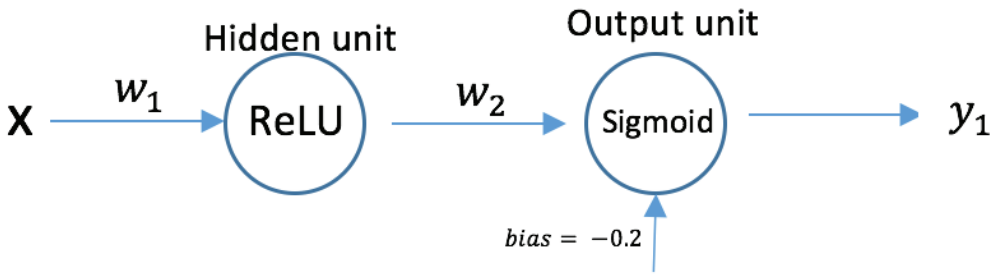
Contents

1 ReLU Backpropagation**

1.1 Single output network

The rectified linear unit (ReLU) is a popular activation function for hidden layers. The activation function is a ramp function $f(z) = \max(0, z)$ where $z = wx$. This has the effect of simply thresholding its input at zero. Unlike the sigmoid, it does not saturate near 1 and is also simpler in gradient computations, resulting in faster convergence of SGD. Furthermore, ReLUs can allow networks to find sparse representations, due to their thresholding characteristic, whereas sigmoids will always generate non-zero values. However, ReLUs can have zero gradient when the activation is negative, blocking the backpropagation of gradients.

Here you use a very small neural network: it has one input unit, taking in a value x, one hidden unit (ReLU), and one output unit (sigmoid). We include a bias term of -0.2 on the sigmoid unit.



We use the following quantities in this problem:

$$z_1 = w_1 x$$
$$a_1 = \text{ReLU}(z_1)$$
$$z_2 = w_2 a_1 - 0.2$$
$$y = \sigma(z_2)$$

The weights are initially $w_1 = \frac{1}{10}$ and $w_2 = -1$.

Let's consider one training example. For that training case, the input value is $x = 2$ (as shown in the diagram), and the target output value $t = 1$. We're using the following loss function:

$$E = \frac{1}{2}(y - t)^2$$

Please supply numeric answers; the numbers in this question have been constructed in such a way that you don't need a calculator. Show your work in case of mis-calculation in earlier steps.

(a) What is the output of the hidden unit for this input?

(b) What is the output of the output unit for this input?

(c) What is the loss, for this training example?

(d) Write out an abstract symbolic expression for derivative of the loss with respect to w_1 as repeated applications of the chain rule. For example, for the derivative of the loss with respect to w_2, we would write $\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial w_2}$.

(e) Write the expression for each partial derivative in the chain rule expansion from the previous part. For example, $\frac{\partial y}{\partial z_2} = y(1-y)$.

(f) What is the derivative of the loss with respect to w_1, for this training example?

(g) What would the update rule for w_1 be? With $\eta =$

(h) If η is large enough, w_1 will update from its current value of 0.1 to a negative value. Assume our new value is $w_1 = -0.1$. What will be the output of the output unit for an input of $x = 2$?

(i) What will happen when we try to update the weight, using this new example, for w_1 for any value of target? Why?

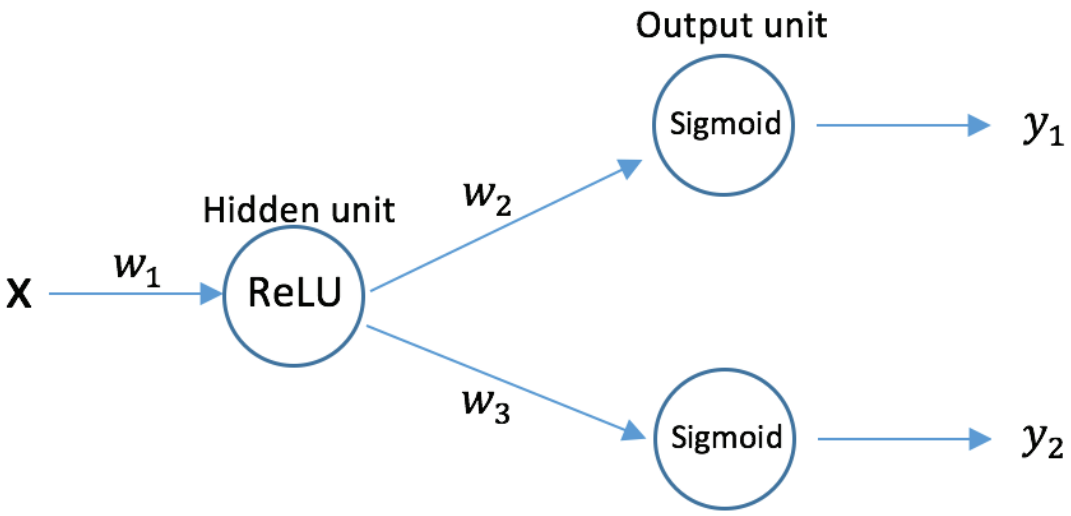
(j) Is it a bad idea to have a ReLU activation at the output layer?

(k) Consider the following activation function:

$$f(z) = \begin{cases} z & \text{if } z > 0 \\ \alpha z & \text{if otherwise.} \end{cases}$$

for some small alpha, e.g. $\alpha = 0.01$, and $z = wx$. Does this address the problem of dying ReLUs?

1.2 Multiple output network



(l)

$$a_1 = \text{ReLU}(0, w_1 x)$$

$$y_1 = \sigma(w_2 a_1)$$

$$y_2 = \sigma(w_3 a_1)$$

Write out an abstract symbolic expression for the derivative of the loss with respect to w_1 for the network above with two output units, as repeated applications of the chain rule.

Multi-output (multi-class) networks are used in many settings such as object recognition, where we are trying to classify an image as being one of K objects. Each of the K possible objects would correspond to an output unit in the network. For this purpose, the sigmoid activation and squared loss are replaced by softmax activation and cross-entropy loss. This is similar to the multi-class logistic regression we saw in the week 4 exercises.

The softmax is given by:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \quad .$$

(m) When $K > 3$, why might sigmoid units be a bad idea?

2 Regularization**

2.1 Weight Decay

We can add L_1 and L_2 penalties to our cost function to regularize our network and prevent overfitting, as we have done with other classification methods. See Neural Networks and Deep Learning by Michael Nielson for derivations and details.

(a) The loss for L_2 regularized case is:

$$L(w) = E(w) + \lambda |w|^2$$

where E(w) is the error. The update for the L_2 regularized case is:

$$w := (1 - \frac{\eta\lambda}{n})w - \eta\frac{\partial E}{\partial w}$$

with regularization. How does this update affect our weights w as we increase λ?

(b) Similarly, the loss for L_1 regularized case is:

$$L(w) = E(w) + \lambda |w|$$

The update for this case is:

$$w := w - (\frac{\eta\lambda}{n})sgn(w) - \eta\frac{\partial E}{\partial w}$$

where $sgn(w)$ is +1 is w is positive and -1 if it is negative. How does this compare to the L_2 update?

2.2 Dropout

Dropout is a regularization technique to reduce overfitting to the training data. A randomly selected portion (eg. 20%) of neurons are ignored during each update cycle of training - these neurons 'drop out' of the network.

During training, without dropout, we compute the activations of a layer l as $a_l = f(z_l)$ Assume we are using a dropout probability of 20%, $a_l = f(z_l)m_l$, where m_l is a mask of the same length with 20% of the entries set to 0 and the others set to 1.

(a) How might this have a regularization affect?

(b) During the feedforward pass without dropout, the inputs to layer $l+1$ are $z_{l+1} = W_{l+1}a_l$. With 20% dropout, how show we change this expression?

2.3 Augmenting with Noisy Training Data

A third way to introduce regularization is to augment the training data with slightly perturbed training examples. We perturb them by applying operations that we might see in the real-world.

(a) Why might this be a good way to improve performance?

(b) How might you do this on MNIST?

3 Convolutional Network Architecture**

Consider the following 1D input and two convolutional filters:

(a) Convolve the first filter with the input.

(b) Convolve the second filter with the input.

(c) What would the output of the convolutional layer look like with both filters?

(d) In the previous part, the spatial size of the output decreased from that of the original input -ie. it had fewer neurons. In some settings, we preserve the size by applying zero padding. What would the output of the convolutional layer look like in this case?

(e) Continuing to use zero padding of size 1, we now use a stride length of 2. What is the output of the convolutional layer now? Compare how this relates to the case with no stride.

(f) If our filters were size 5, rather than size 3, with stride 1, how much padding would you apply to maintain output size?

(g) What is a general expression for the output size (number of neurons) in a convolutional layer, in terms of the filter size (F), stride length (S), amount of padding (P), and the input size (W)?

(h) Consider a 10x10 grayscale image input (no RGB channel). If you made a 1-layer fully connected network, with 1 hidden, how many weight parameters would be required in the hidden layer including bias terms (ignore the output layer)?

(i) Consider a 10x10 grayscale image input (no RGB channel). If you made a 2-layer fully connected network, with 10 and 5 hidden units respectively, how many weight parameters would be required in these hidden layers including bias terms (ignore the output layer)?

(j) With the same 10x10 input image, you now use a convolutional layer with F = 2 (square filters), S = 2 (in both dimensions), and P = 0. How is the output volume of the first convolutional layer with 2 filters, including bias terms?

(k) How many parameters does the above convolutional layer have?

(l) One advantage of convolutional networks is that the number of free parameters can be controlled by parameter sharing for a filter across spatial dimensions. Each filter is replicated across the entire visual field (image). That is, if a filter is used at one spatial coordinate (x_1, y_1), the same weights are also used in a different position (x_2, y_2). Adopting the parameter sharing scheme, how many parameters does the convolutional layer now have?

4 Silly friends*

(a) Like Jody last week, Evelyn sees that you are handling a multi-class problem with 4 classes and suggests that you use as a target value $y^{(i)}$ a vector of two output values, each of which can be 0 or 1, to encode which one of four classes it belongs to.

What is good or bad about Evelyn's approach?

(b) Seeing you working on an implementation of a neural network with sigmoid units, where the inputs are in the range $[0, 1]$, Jean suggests that you initialize the weights randomly in the range $[0, 100]$.

Is Jean's idea good?

5 Activation energy

A bunch of AI students walked into a bar...and promptly started debating alternative activation functions for a neural network intended to do binary classification. The input data is in R^2; the weight vector is 3-dimensional.

The network in fact has just a single unit with two input dimensions.

For each of the suggested activation functions, say:

- Whether it makes sense to train it with the cross-entropy objective function:

$$y \log o + (1 - y) \log(1 - o)$$

where y is the desired output and o is the actual (the output of the network).

- Whether it can represent the same class of separators in the two-dimensional input space as the sigmoidal activation function.
- If it can represent the same class, explain why.
- If it cannot, then draw a separator that can be represented by a sigmoidal activation on a linear combination of the inputs but not by this activation on a linear combination of the inputs **or** a separator that can be represented by this activation function on a linear combination of the inputs, but not by a sigmoidal activation function on a linear combination of the inputs and explain why.

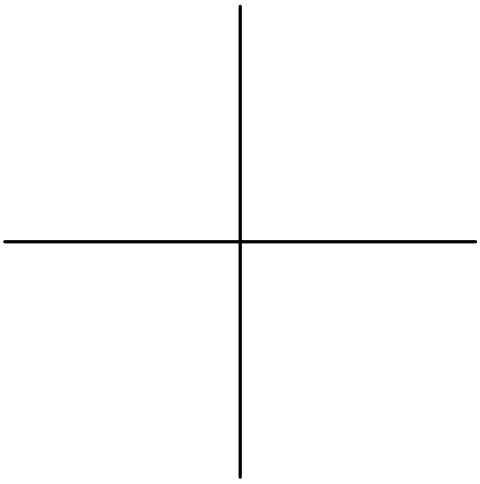
(a)

$$f(a) = \sin(a)$$

Sensible to use cross-entropy: ◯ Yes ◯ No

Same class of separators as sigmoid? ◯ No ◯ No

If yes, explain why. If no, draw separator as explained above.



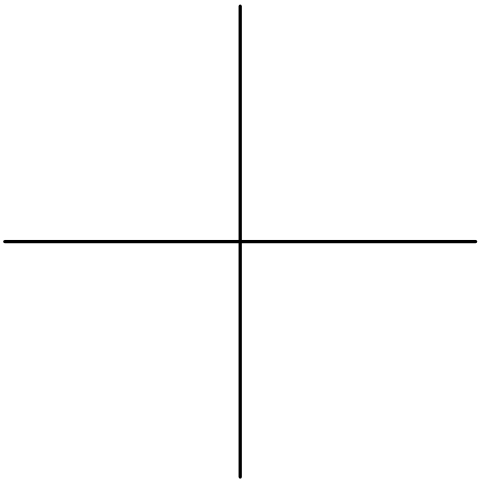
(b)

$$f(a) = \begin{cases} 0 & \text{if } a < 0 \\ 1 & \text{if } a > 1 \\ x & \text{otherwise} \end{cases}$$

Sensible to use cross-entropy: ◯ Yes ◯ No

Same class of separators as sigmoid? ◯ Yes ◯ No

If yes, explain why. If no, draw separator as explained above.



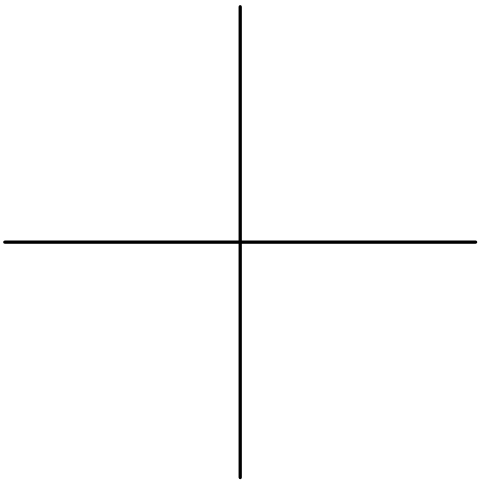
(c)

$$f(a) = e^a$$

Sensible to use cross-entropy: ◯ Yes ◯ No

Same class of separators as sigmoid? ◯ Yes ◯ No

If yes, explain why. If no, draw separator as explained above.



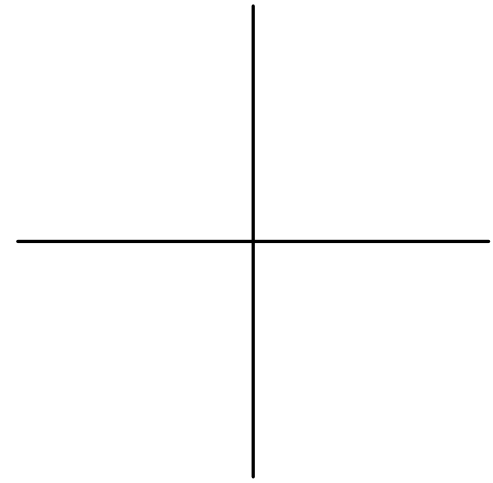
(d)

$$f(a) = a^2$$

Sensible to use cross-entropy: ◯ Yes ◯ No

Same class of separators as sigmoid? ◯ Yes ◯ No

If yes, explain why. If no, draw separator as explained above.



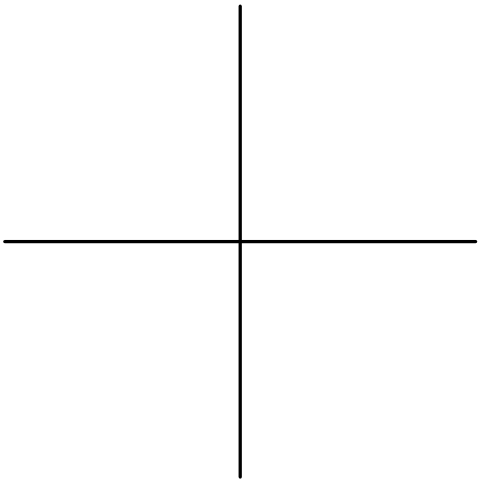
(e)

$$f(a) = a$$

Sensible to use cross-entropy: ◯ Yes ◯ No

Same class of separators as sigmoid? ◯ Yes ◯ No

If yes, explain why. If no, draw separator as explained above.



6 Neural network short answer

1. If you train a neural-net classifier using stochastic gradient descent, and you use a too small learning rate, what can go wrong?
2. How can you determine that you have used too small a learning rate?
3. If you train a neural-net classifier using stochastic gradient descent, and you use a too large learning rate, what can go wrong?
4. How can you determine that you have used too large a learning rate?
5. Is final error on the training set a good measure for finding out whether the learning rate was set too large?
6. Suppose you train a neural-net classifier using stochastic gradient descent, and you measure training error and validation error after N iterations.

   Which of the two errors is a better estimate for the error on unseen future data and why?
7. In a convolutional layer of a neural network, what gives rise to a smaller output dimentionality: a small stride or a large stride?
8. In a convolutional layer of a neural network, can you use 1x1 convolution to learn a linear classifier on top of the inputs? If yes, what format should the input have?
9. For effective training of a neural network, the network should have at least 4 times as many weights as there are training samples? If yes, why? If no, why not?
10. Assume the following (momentum) weight update rule:

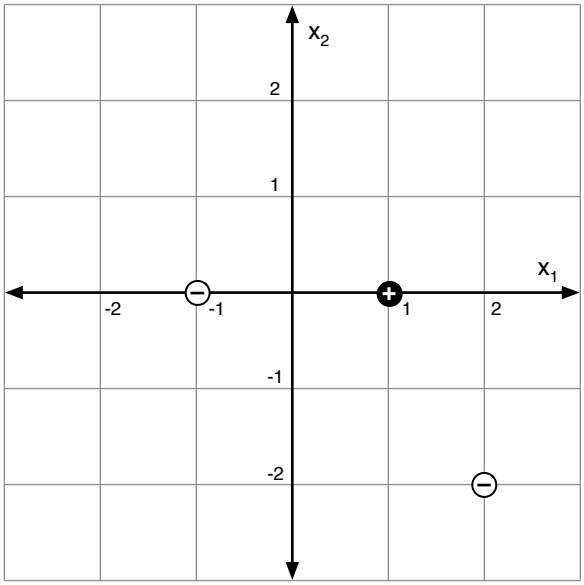
$$m_0 = 0 \tag{6.1}$$

$$m_t = \beta m_{t-1} + (1-\beta)\frac{\partial E}{\partial w} \tag{6.2}$$

$$w_t = w_{t-1} + \lambda m_t \tag{6.3}$$

11. For what value of β do you obtain the standard gradient descent rule?

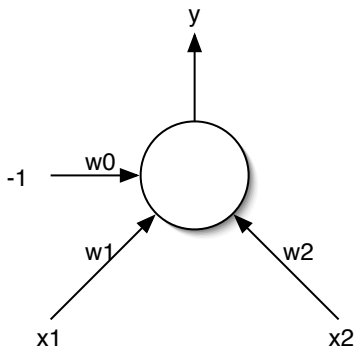
7 Neural Net*



Data points are: Negative: (-1, 0) (2, -2) Positive: (1, 0)

Recall that for neural nets with sigmoidal output units, the negative class is represented by a desired output of 0 and the positive class by a desired output of 1. Hint: Some useful values of the sigmoid $s(z)$ are $s(-1) = 0.27$ and $s(1) = 0.73$.

Assume we have a single sigmoid unit:



Assume that the weights are $w_0 = 0, w_1 = 1, w_2 = 1$. What is the computed y value for each of the points on the diagram above?

(a) $x = (-1, 0)$

(b) $x = (2, -2)$

(c) $x = (1, 0)$

(d) What would be the change in w_2 as determined by backpropagation using a step size (η) of 1.0? Assume the squared loss function. Assume that the input is $x = (2, -2)$ and the initial

weights are as specified above. Show the formula you are using as well as the numerical result.

1. $\Delta w_2 =$

8 Lots of class

(Bishop 5.5) Show that the maximizing likelihood for a multi-class neural network model in which the network outputs have the interpretation $h_k(x, w) = p(y = k \mid x)$ is equivalent to the minimization of the cross-entropy error function

$$E(w) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_k^{(i)} \ln h_k(x^{(i)}, w) \quad .$$

9 Noisy targets

(Bishop 5.4) Consider a binary classification problem in which the target values are $y \in \{0, 1\}$ with a network output $h(x, w)$ that represents $p(y = 1 \mid x)$, and suppose that there is a probability ϵ that the class label on a training data point has been incorrectly set.

(a) Assuming independent and identically distributed data, write down the error function corresponding to the negative log likelihood. Verify that the *cross entropy* error function is obtained when $\epsilon = 0$. Note that this error function makes the model robust to incorrectly labeled data, in contrast to the usual error function.

(b) What is the form of the partial derivative with respect to a single weight in the output layer?

(c) How does the stochastic gradient update rule for $\epsilon = 0.1$ differ from the case when $\epsilon = 0$?

10 MLE

(Bishop 5.2) Show that maximizing the likelihood function under the conditional distribution

$$p(y \mid x, w) = \mathcal{N}(y \mid h(x, w), \beta^{-1}\mathbf{I})$$

for a multi-output neural network is equivalent to minimizing the sum-of-squares error function

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} \|h(x^{(n)}, w) - y^{(n)}\|^2 \quad .$$

11 Not independent

(Bishop 5.3) Consider a regression problem involving multiple target variables in which it is assumed that the distribution of the targets, conditioned on the input vector x is a Gaussian of the form

$$p(y \mid x, w) = \mathcal{N}(y \mid h(x, y), \Sigma)$$

where $h(x, y)$ is the output of a neural network with input vector x and weight vector w and Σ is the covariance of the assumed Gaussian noise on the targets.

Given a set of independent observations $x^{(i)}, y^{(i)}$, write down the error function that must be minimized in order to find the maximum likelihood solution for w. First, assume that Σ is fixed and known.

Second, assume that Σ is also to be determined from the data and write down an expression for the maximum likelihood solution for Σ. Note that the optimizations for w and Σ are now coupled, in contrast to the case of independent target variables.

12 Noisification

This problem is more difficult, but kind of cool.

(Bishop 5.27) Consider a dataset with input-target pairs $(x^{(i)}, t^{(i)})$. Define the expected error as

$$E = \frac{1}{2}\int\int (y(x) - t)^2 p(t|x)p(x)dxdt.$$

Consider training with transformed inputs $z^{(i)} = x^{(i)} + \xi$ where $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Define the expected error of transformed inputs as

$$\tilde{E} = \frac{1}{2}\int\int\int (y(x + \xi) - t)^2 p(z|x)p(x)p(\xi)dxdtd\xi.$$

By following an argument analogous to that of section 5.5.5 in Bishop, show that the expected error of transformed inputs can be written in the form of Tikhonov regularization as

$$\tilde{E} = E + \lambda\Omega,$$

where

$$\Omega = \frac{1}{2}\int \|\nabla h(x, w)\|^2 p(x)dx.$$

13 Convolution

This problem is significantly difficult, and requires reading Bishop. Interesting method that is being used in a lot of computer vision applications. Don't feel you have to understand this.

Consider a neural network, such as the convolutional network discussed in Section 5.5.6 in Bishop, in which multiple weights are constrained to have the same value. Discuss how the standard backpropagation algorithm must be modified in order to ensure that such constraints are satisfied when evaluating the derivatives of an error function with respect to the adjustable parameters in the network.