CS 224N: Assignment 1

RYAN MCMAHON SATURDAY 28TH JANUARY, 2017

Contents

Problem 1: Softmax (10 pts)

1.1 (a) Softmax Invariance to Constant (5 pts)

Prove that softmax is invariant to constant offsets in the input, that is, for any input vector x and any constant c, softmax(x) = softmax($x + c$), where $x + c$ means adding the constant c to every dimension of x. Remember that

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{1.1}$$

Answer:

We can show that softmax(x) = softmax($x + c$) by factoring out c and canceling:

$$\begin{aligned} softmax(x+c)_i &= \frac{e^{x_i+c}}{\sum_j e^{x_j+c}} = \frac{e^{x_i} \times e^c}{e^c \times \sum_j e^{x_j}} \\ &= \frac{e^{x_i} \times \cancel{e^c}}{\cancel{e^c} \times \sum_j e^{x_j}} = softmax(x)_i \end{aligned}$$

1.2 (b) Softmax Coding (5 pts)

Given an input matrix of N rows and D columns, compute the softmax prediction for each row using the optimization in part (a). Write your implementation in `q1_softmax.py`. *You may test by executing* `python q1_softmax.py`.

Note: The provided tests are not exhaustive. Later parts of the assignment will reference this code so it is important to have a correct implementation. Your implementation should also be efficient and vectorized whenever possible (i.e., use numpy matrix operations rather than for loops). A non-vectorized implementation will not receive full credit!

Answer:

See code: ~`/code/q1_softmax.py`.

Problem 2: Neural Network Basics (30 pts)

2.1 (a) Sigmoid Gradient (3 pts)

Derive the gradients of the sigmoid function and show that it can be rewritten as a function of the function value (i.e., in some expression where only (x), but not x, is present). Assume that the input x is a scalar for this question. Recall, the sigmoid function is

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

Answer:

$$\begin{aligned}
\sigma(x) &= \frac{1}{1+e^{-x}} \\
&= \frac{e^x}{1+e^x} \\
\frac{\partial}{\partial x}\sigma(x) &= \frac{e^x \times (1+e^x) - (e^x \times e^x)}{(1+e^x)^2} \\
&= \frac{e^x + \cancel{(e^x \times e^x)} - \cancel{(e^x \times e^x)}}{(1+e^x)^2} \\
&= \frac{e^x}{(1+e^x)^2} = \sigma(x) \times (1-\sigma(x))
\end{aligned}$$

Because $1-\sigma(x) = \sigma(-x)$ we can show that:

$$\begin{aligned}
\frac{\partial}{\partial x}\sigma(x) &= \frac{e^x}{(1+e^x)^2} \\
&= \sigma(x) \times \sigma(-x) \\
&= \frac{e^x}{1+e^x} \times \frac{1}{1+e^{+x}} \\
&= \frac{e^x}{(1+e^x)^2}
\end{aligned}$$

2.2 (b) Softmax Gradient w/ Cross Entropy Loss (3 pts)

Derive the gradient with regard to the inputs of a softmax function when cross entropy loss is used for evaluation, i.e., find the gradients with respect to the softmax input vector $\boldsymbol{\theta}$*, when the prediction is made by* $\hat{\mathbf{y}} = softmax(\boldsymbol{\theta})$*. Remember the cross entropy function is*

$$CE(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \times log(\hat{y}_i) \tag{2.2}$$

where $\mathbf{y}$ *is the one-hot label vector, and* $\hat{\mathbf{y}}$ *is the predicted probability vector for all classes. (Hint: you might want to consider the fact many elements of* $\mathbf{y}$ *are zeros, and assume that only the* $k-th$ *dimension of* $\mathbf{y}$ *is one.)*

Answer:

Let S represent the softmax function:

$$\begin{aligned}
f_i &= e^{\theta_i} \\
g_i &= \sum_{k=1}^{K} e^{\theta_k} \\
S_i &= \frac{f_i}{g_i} \\
\frac{\partial S_i}{\partial \theta_j} &= \frac{f_i' g_i - g_i' f_i}{g_i^2}
\end{aligned}$$

So if $i = j$:

$$\begin{aligned}
f_i' &= f_i; \;\; g_i' = e^{\theta_j} \\
\frac{\partial S_i}{\partial \theta_j} &= \frac{e^{\theta_i} \sum_k e^{\theta_k} - e^{\theta_j} e^{\theta_i}}{(\sum_k e^{\theta_k})^2} \\
&= \frac{e^{\theta_i}}{\sum_k e^{\theta_k}} \times \frac{\sum_k e^{\theta_k} - e^{\theta_j}}{\sum_k e^{\theta_k}} \\
&= S_i \times (1 - S_i)
\end{aligned}$$

And if $i \neq j$:

$$\begin{aligned}
\frac{\partial S_i}{\partial \theta_j} &= \frac{0 - e^{\theta_j} e^{\theta_i}}{(\sum_k e^{\theta_k})^2} \\
&= -\frac{e^{\theta_j}}{\sum_k e^{\theta_k}} \times \frac{e^{\theta_i}}{\sum_k e^{\theta_k}} \\
&= -S_j \times S_i
\end{aligned}$$

We can now use these when operating on our loss function (let L represent the cross entropy function):

$$\begin{aligned}
\frac{\partial L}{\partial \theta_i} &= -\sum_k y_k \frac{\partial log S_k}{\partial \theta_i} \\
&= -\sum_k y_k \frac{1}{S_k}\frac{\partial S_k}{\partial \theta_i} \\
&= -y_i(1 - S_i) - \sum_{k \neq i} y_k \frac{1}{S_k}(-S_k \times S_i) \\
&= -y_i(1 - S_i) + \sum_{k \neq i} y_k S_i \\
&= -y_i + y_i S_i + \sum_{k \neq i} y_k S_i \\
&= S_i(\sum_k y_k) - y_i
\end{aligned}$$

And because we know that $\sum_k y_k = 1$:

$$\frac{\partial L}{\partial \theta_i} = S_i - y_i$$