Backpropagation and Gradient Descent for Training Neural Networks

CS 349-02

April 10, 2017

1 Overview

Given a neural network architecture and labeled training data, we want to find the weights that minimize the loss on the training data.

The loss function varies depending on the output layer and labels.The total loss is the sum of two terms: the data loss and the regularization loss.

$$J = J_{\text{data}} + \frac{\alpha}{2} J_{\text{reg}} \tag{1}$$

Since we generally only regularize the weights and not the biases,

$$J_{\text{reg}} = \|W_2\|^2 + \|W_1\|^2$$

Using gradient descent, we will modify each layer to minimize the total loss. In order to do this, we need to compute the gradient of the total loss in Eq. 1 with respect to each weight matrix or each bias vector.

2 Training Procedure with Gradient Descent

The gradient descent algorithm is similar to what we derived for logistic regression. The only change is that we are updating multiple weight matrices rather than a single weight vector, and since we're doing gradient descent, we update the parameters in the opposite direction of the gradient.

Algorithm 1 Batch Gradient Descent for Single-Hidden-Layer Neural Networks

W_2 = small random matrix ▷ all-zeros may result in 0 gradient with some activations
b_2 = all-zeros
W_1 = small random matrix, b_1 = all-zeros
for $epoch \in [1, 2, \dots maxiter]$ **do**
    $gradW_2$ = all zeros, $gradb_2$ = all zeros
    $gradW_1$ = all zeros, $gradb_1$ = all zeros
    for $x^{(i)}, y^{(i)}$ in training data (i from 1 to N) **do** ▷ Accumulate gradients
        $gradW_2 \leftarrow gradW_2 + \frac{\partial J_{\text{data}}}{\partial W_2}$ ▷ J is the loss for this data-point
        $gradb_2 \leftarrow gradb_2 + \frac{\partial J_{\text{data}}}{\partial b_2}$
        $gradW_1 \leftarrow gradW_1 + \frac{\partial J_{\text{data}}}{\partial W_1}$
        $gradb_1 \leftarrow gradb_1 + \frac{\partial J_{\text{data}}}{\partial b_1}$
    ▷ Now compute gradients of the *average* data loss
    $gradW_2 \leftarrow gradW_2/N$, $gradb_2 \leftarrow gradb_2/N$
    $gradW_1 \leftarrow gradW_1/N$, $gradb_1 \leftarrow gradb_1/N$
    $gradW_2 \leftarrow gradW_2 + \alpha W_2$, $gradW_1 \leftarrow gradW_1 + \alpha W_1$ ▷ Regularization gradients
    $W_2 \leftarrow W_2 - \eta \cdot gradW_2$ ▷ Update the parameters with a gradient descent step
    $b_2 \leftarrow b_2 - \eta \cdot gradb_2$
    $W_1 \leftarrow W_1 - \eta \cdot gradW_1$
    $b_1 \leftarrow b_1 - \eta \cdot gradb_1$
return W_2, b_2, W_1, b_1

3 Equations for Gradient Computations

The missing piece in the above algorithm is how to compute the partial derivatives for the gradient update. This section shows the computations that you should implement in the problem set. Section **??** derives the math behind these computations.

We're given a two-layer network, with an output layer containing C neurons, given by the weight matrix and bias vector W_2 and b_2 and the $softmax$ activation function, and one hidden layer with containing H neurons, given by the weight matrix and bias vector W_1 and b_1, and the $relu$ activation function.

A single data-point x is pushed through the network (the forward pass) with the following computations:

$$s_1 = x.W_1 + b_1 \tag{2}$$

$$a_1 = relu(s_1) \tag{3}$$

$$s_2 = a_1.W_2 + b_2 \tag{4}$$

$$a_2 = softmax(s_2) \tag{5}$$

For a softmax top layer, a_2 is a vector of probabilities, where $a_2[c]$ is the probability that x is classified as c by the neural network.

3.1 Top layer

First, we compute the gradients of the top layer; that is, the gradients of the loss with respect to W_2 (the weights) and b_2 (the biases) of the neurons in the top layer. The quantities we need to compute are $\frac{\partial J_{\text{data}}}{\partial W_2}$ and $\frac{\partial J_{\text{data}}}{\partial b_2}$.

This requires computing an intermediate value, $\frac{\partial J_{\text{data}}}{\partial s_2}$ for a given data point x labeled as y.

Recall that a_2 is a vector of dimensionality C, where C is the number of possible classes.

$$\frac{\partial J_{\text{data}}}{\partial s_2} = \Big[a_2[0] - \mathbb{1}(y=0), a_2[1] - \mathbb{1}(y=1), a_2[2] - \mathbb{1}(y=2), a_2[3] - \mathbb{1}(y=3), \ldots\Big] \tag{6}$$

$\mathbb{1}$ is an indicator variable that becomes 1 when $y = c$ and 0 otherwise.

Note that $\frac{\partial J_{\text{data}}}{\partial s_2}$ is a vector of the same dimensionality as s_2 (the output of the second layer) which is 1 x C dimensions.

We use the above quantity to compute the gradient of the data loss with respect to W_2. See Sec. **??** for the derivation.

$$\frac{\partial J_{\text{data}}}{\partial W_2} = a_1^T . \frac{\partial J_{\text{data}}}{\partial s_2} \tag{7}$$

The gradient of the data loss with respect to W_2 for a single data point x is therefore the dot-product of a_1^T (which has dimensionality H x 1) and $\frac{\partial J_{\text{data}}}{\partial s_2}$ (which has dimensionality 1 x C), giving an H x C matrix.

This is good because it is the same dimensionality as W_2, which is what we need when updating W_2 in the gradient descent training step.

The gradient of the data loss with respect to b_2 is

$$\frac{\partial J_{\text{data}}}{\partial b_2} = \frac{\partial J_{\text{data}}}{\partial s_2} \tag{8}$$

What is dimensionality of this result? What is the dimensionality of b_2? Do they agree?

And that's it for the top layer!

3.2 Hidden layer

We're now going to "backpropagate" the gradients we computed before down to the hidden layer, to estimate how much each neuron in the hidden layer contributes to the loss. The quantities we need to compute are $\frac{\partial J_{\text{data}}}{\partial W_1}$ and $\frac{\partial J_{\text{data}}}{\partial b_1}$.

Just like for the top layer, we first compute an intermediate quantity $\frac{\partial J_{\text{data}}}{\partial s_1}$ with the pseudocode below:

$tmp \leftarrow \frac{\partial J_{\text{data}}}{\partial s_2}.W_2^T$ ▷ tmp has dimensionality 1 x H
for h from 1 to H **do**
    if $a_1[h] \leq 0$ **then**
        tmp[h] = 0 ▷ Comes from the gradient of *relu*. See Sec. **??** for derivation.
$\frac{\partial J_{\text{data}}}{\partial s_1} \leftarrow tmp$

We use the above quantity to compute the gradient of the data loss with respect to W_1.

$$\frac{\partial J_{\text{data}}}{\partial W_1} = x^T . \frac{\partial J_{\text{data}}}{\partial s_1} \tag{9}$$

The gradient of the data loss with respect to W_1 for a single data point x is therefore the dot-product of x^T (which has dimensionality D x 1) and $\frac{\partial J_{\text{data}}}{\partial s_1}$ (which has dimensionality 1 x H), giving an D x H matrix.

This is the same dimensionality as W_1, which is what we need when updating W_1 in the gradient descent training step.

The gradient of the data loss with respect to b_1 is

$$\frac{\partial J_{\text{data}}}{\partial b_1} = \frac{\partial J_{\text{data}}}{\partial s_1} \tag{10}$$

What is dimensionality of this result? What is the dimensionality of b_1? Do they agree?

This is all you need to implement the `backward` method in the problem set.

4 Computing gradients with backpropagation

Skim over this section to derive the math behind the updates we wrote out in the previous section.

4.1 Top Layer

4.1.1 Gradient of data-loss with respect to s_2

We first find the derivative of the loss with respect to s_2. Using the calculus chain rule:

$$\frac{\partial J_{\text{data}}}{\partial s_2} = \frac{\partial J_{\text{data}}}{\partial a_2}\frac{\partial a_2}{\partial s_2} \tag{11}$$

We know from the definition of cross-entropy loss that $J_{\text{data}} = -\log a_2[y]$. After doing some calculus to compute $\frac{\partial a_2}{\partial s_2}$ from the softmax function, and canceling out terms, it turns out that $\frac{\partial J_{\text{data}}}{\partial a_2}\frac{\partial a_2}{\partial s_2}$ is the following vector:

$$\frac{\partial J_{\text{data}}}{\partial s_2} = \left[\frac{\partial J_{\text{data}}}{\partial s_2[0]}, \frac{\partial J_{\text{data}}}{\partial s_2[1]}, \frac{\partial J_{\text{data}}}{\partial s_2[2]}, \dots\right] \tag{12}$$

where

$$\frac{\partial J_{\text{data}}}{\partial s_2[c]} = a_2[c] - \mathbb{1}(y = c) \tag{13}$$

4.1.2 Gradient of data-loss with respect to W_2

We backpropagate the derivative we just computed to get the gradient of the data loss with respect to W_2. For this, we need to compute $\frac{\partial s_2}{\partial W_2}$.

$$\frac{\partial s_2}{\partial W_2} = \frac{\partial}{\partial W_2}(a_1 W_2 + b_2) = a_1^T \tag{14}$$

Finally, we put Step 1 and Step 2 together to compute the gradient of the data loss with respect to W_2:

$$\frac{\partial J_{\text{data}}}{\partial W_2} = \frac{\partial s_2}{\partial W_2} . \frac{\partial J_{\text{data}}}{\partial s_2} \tag{15}$$

4.1.3 Gradient of regularization-loss with respect to W_2

This computation is similar to logistic regression regularization, except that this is a partial derivative with respect to W_2 only:

$$\frac{\partial}{\partial W_2}\frac{\alpha}{2}(\|W_1\|^2 + \|W_2\|^2) = \alpha W_2$$

4.1.4 Gradient of data-loss with respect to b_2

Similarly, compute the gradient of the data loss with respect to b_2.

Just as with W_2, we backpropagate the gradient with respect to s_2 computed in section **??**.

$$\frac{\partial J_{\text{data}}}{\partial b_2} = \frac{\partial s_2}{\partial b_2}.\frac{\partial J_{\text{data}}}{\partial s_2}$$

We already computed the second term of the product in **??**. Computing the first term, using Eq. 4

$$\frac{\partial s_2}{\partial b_2} = \frac{\partial}{\partial b_2}(a_1.W_2 + b_2) = 1$$

Put together the two above terms to get the gradient of the data loss with respect to b_2.

$$\frac{\partial J_{\text{data}}}{\partial b_2} = 1.\frac{\partial J_{\text{data}}}{\partial s_2}$$

4.2 Hidden Layer

4.2.1 Gradient of data-loss with respect to s_1

Using the calculus chain rule again, we'll "backpropagate" the loss down to s_1 (the output of the first layer):

$$\frac{\partial J_{\text{data}}}{\partial s_1} = \frac{\partial J_{\text{data}}}{\partial s_2}\frac{\partial s_2}{\partial s_1}$$

Expanding this by the chain rule:

$$\frac{\partial J_{\text{data}}}{\partial s_1} = \frac{\partial J_{\text{data}}}{\partial s_2}\frac{\partial s_2}{\partial a_1}\frac{\partial a_1}{\partial s_1} \tag{16}$$

The **first term**, $\frac{\partial J_{\text{data}}}{\partial s_2}$, is our old friend that we already calculated for the W_2 and b_2 gradients (Eq. 7). We need not recompute it; just use the stored result.

Compute the **second term** using Eq. 4:

$$\frac{\partial s_2}{\partial a_1} = \frac{\partial}{\partial a_1}(a_1.W_2 + b_2) = W_2^T$$

Let's compute the **third term**. The derivative of a vector with respect to a vector is a matrix of size (in this case) H x H, since both a_1 and s_1 are H dimensional vectors.

$$\frac{\partial a_1}{\partial s_1} = \begin{bmatrix} \frac{\partial a_1[0]}{\partial s_1[0]} & \frac{\partial a_1[0]}{\partial s_1[1]} & \frac{\partial a_1[0]}{\partial s_1[2]} & \frac{\partial a_1[0]}{\partial s_1[3]} & \dots \\ \\ \frac{\partial a_1[1]}{\partial s_1[0]} & \frac{\partial a_1[1]}{\partial s_1[1]} & \frac{\partial a_1[1]}{\partial s_1[2]} & \frac{\partial a_1[1]}{\partial s_1[3]} & \dots \\ \\ \frac{\partial a_1[2]}{\partial s_1[0]} & \frac{\partial a_1[2]}{\partial s_1[1]} & \frac{\partial a_1[2]}{\partial s_1[2]} & \frac{\partial a_1[2]}{\partial s_1[3]} & \dots \end{bmatrix}$$

The above matrix is 0 in all places except the diagonal. At all diagonal indices h, the corresponding entry evaluates to

$$\frac{\partial a_1[h]}{\partial s_1[h]} = \frac{\partial}{\partial s_1[h]} relu(s_1[h]) = \begin{cases} 1 & \text{when } s_1[h] > 0 \\ \\ 0 & \text{otherwise} \end{cases}$$

Put the above three terms together to get $\frac{\partial J_{\text{data}}}{\partial s_1}$, which is the product of the 1 x C matrix $\frac{\partial J_{\text{data}}}{\partial s_2}$, a C x H matrix $\frac{\partial s_2}{\partial a_1} = W_2^T$, and an H x H matrix $\frac{\partial a_1}{\partial s_1}$.

4.2.2 Gradient of data-loss with respect to W_1

Finally, we backpropagate this result down to W_1 to get the gradient of the loss with respect to W_1:

$$\frac{\partial J_{\text{data}}}{\partial W_1} = \frac{\partial s_1}{\partial W_1} \frac{\partial J_{\text{data}}}{\partial s_1}$$

From Eq. **??**,

$$\frac{\partial s_1}{\partial W_1} = \frac{\partial}{\partial W_1}(x.W_1 + b_1) = x^T$$

The data-loss gradient update for W_1 therefore works out to

$$\frac{\partial J_{\text{data}}}{\partial W_1} = x^T . \frac{\partial J_{\text{data}}}{\partial s_1} \tag{17}$$

where $\frac{\partial J_{\text{data}}}{\partial s_1}$ is the result we computed in Eq **??**. This is the dot product of a D x 1 and a 1 x H matrix, giving a result of dimensionality of D x H, which agrees with the dimensionality of W_1.

4.2.3 Gradient of regularization-loss with respect to W_1

Just as we did for W_2, the gradient of regularization loss with respect to W_1 becomes

$$\frac{\partial}{\partial W_1} \frac{\alpha}{2}(\|W_1\|^2 + \|W_2\|^2) = \alpha W_1$$

4.2.4 Gradient of data-loss with respect to b_1

Just as with W_1, we backpropagate the gradient with respect to s_1:

$$\frac{\partial J_{\text{data}}}{\partial b_1} = \frac{\partial s_1}{\partial b_2} . \frac{\partial J_{\text{data}}}{\partial s_1}$$

We already computed the second term of the product in Eq. **??**. Computing the first part, using Eq. **??**

$$\frac{\partial s_1}{\partial b_1} = \frac{\partial}{\partial b_1}(x.W_1 + b_1) = 1$$

Put together the two above terms to get the gradient for updating b_1.

$$\frac{\partial J_{\text{data}}}{\partial b_1} = 1 . \frac{\partial J_{\text{data}}}{\partial s_1}$$