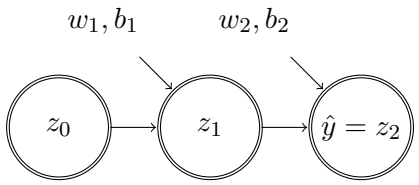
Introduction to Machine Learning	**Fall Semester**
Recitation 8	
Lecturer: Regev Schweiger	*Scribe: Regev Schweiger*

8.1 Backpropagation

We will develop and review the backpropagation algorithm for neural networks. In order to have a concrete example, we will focus on the choice of the sigmoid (i.e., $h(z) = 1/(1+e^{-z})$), with the log loss function (i.e., $\ell(y, \hat{y}) = -y \log \hat{y} - (1-y)\log(1-\hat{y})$), but of course, the derivation holds in general.

8.1.1 Two layers, one node

We will start from the simplest (interesting) example and do it very explicitly. Suppose we are given a sample $x_1, \ldots, x_m$ with labels $y_1, \ldots, y_m \in \{-1, 1\}$. The simplest is two layers, with one node in each:



The function we want to minimize is the loss over all examples:

$$f = \sum_{i=1}^{m} \ell(y_i, z_2(x_i))$$

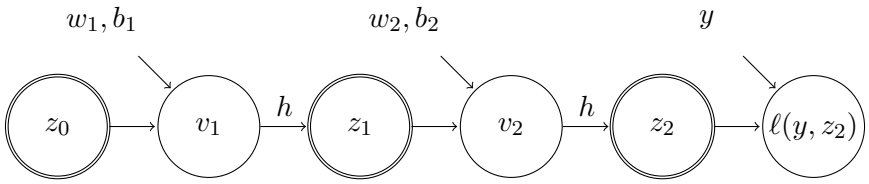
We take a gradient descent approach. At each iteration, we have our current guesses for the best parameter values: $\tilde{w}_1, \tilde{b}_1, \tilde{w}_2, \tilde{b}_2$, and we wish to calculate the gradient *at that point*, e.g:

$$\frac{\partial f}{\partial w_1}\bigg|_{\substack{w_1=\tilde{w}_1, b_1=\tilde{b}_1, \\ w_2=\tilde{w}_2, b_2=\tilde{b}_2}} \quad \text{and} \quad \frac{\partial f}{\partial b_1}\bigg|_{\substack{w_1=\tilde{w}_1, b_1=\tilde{b}_1, \\ w_2=\tilde{w}_2, b_2=\tilde{b}_2}} \quad \text{and} \quad \frac{\partial f}{\partial w_2}\bigg|_{\substack{w_1=\tilde{w}_1, b_1=\tilde{b}_1, \\ w_2=\tilde{w}_2, b_2=\tilde{b}_2}} \quad \text{and} \quad \frac{\partial f}{\partial b_2}\bigg|_{\substack{w_1=\tilde{w}_1, b_1=\tilde{b}_1, \\ w_2=\tilde{w}_2, b_2=\tilde{b}_2}}$$

First, since derivative is linear, we can focus only on the loss over a single sample, and set $y = y_i, z_0 = x_i$. and, e.g.:

$$\frac{\partial f}{\partial w_1} = \sum_{i=1}^{m} \frac{\partial}{\partial w_1} \ell(y, z_2(z_0))$$

We proceed as if there is a single sample. To continue, we will benefit by introducing convenient notation (see also lesson scribe) and drawing it as well. Define by $v_1 = z_0 w_1 + b_1, v_2 = z_1 w_2 + b_2$ the linear combinations used in the calculations of z_1, z_2 (respectively). Namely, $z_1 = h(v_1), z_2 = h(v_2)$. We will add these to the graph above, that will now show the explicit calculation we make. We will also show the loss function, since it is added to the entire calculation.



Derivative w.r.t w_2, b_2

Let first recall the chain rule. Suppose we have three variables/functions x, y, z, and $z = z(y), y = y(x)$, thus $z = z(y(x))$. Say we want to evaluate the derivative of z at a point $\tilde{x}$. Then, the chain rule says:

$$\left.\frac{\partial z}{\partial x}\right|_{x=\tilde{x}} = \left.\frac{\partial z}{\partial y}\right|_{y=y(\tilde{x})} \cdot \left.\frac{\partial y}{\partial x}\right|_{x=\tilde{x}}$$

We want to calculate

$$\left.\frac{\partial}{\partial w_2}\ell(y, z_2(z_0))\right|_{\substack{w_1=\tilde{w}_1, b_1=\tilde{b}_1, \\ w_2=\tilde{w}_2, b_2=\tilde{b}_2}}$$

the dependency on the current point is hidden in z_2. We will denote by $\tilde{v}_i$ the evaluation of v_i on the specific point we are at, e.g., $\tilde{v}_1 = z_0\tilde{w}_1 + \tilde{b}_1$. Similarly, $\tilde{z}_i = h(\tilde{v}_i)$. Following the chain rule and the drawing above, we can see that

$$\left.\frac{\partial}{\partial w_2}\ell(y, z_2(z_0))\right|_{\substack{w_1=\tilde{w}_1, b_1=\tilde{b}_1, \\ w_2=\tilde{w}_2, b_2=\tilde{b}_2}} = \left.\frac{\partial}{\partial z_2}\ell(y, z_2)\right|_{z_2=\tilde{z}_2} \cdot \left.\frac{\partial z_2}{\partial v_2}\right|_{v_2=\tilde{v}_2} \cdot \left.\frac{\partial v_2}{\partial w_2}\right|_{z_1=\tilde{z}_1, w_2=\tilde{w}_2, b_2=\tilde{b}_2}$$

The left expression is the derivative of the loss function wrt to the estimate. In our case, it is the log loss, *evaluated at the current point*, that is:

$$\left.\frac{\partial}{\partial z_2}\ell(y, z_2)\right|_{z_2=\tilde{z}_2} = -y/\tilde{z}_2 + (1-y)/(1-\tilde{z}_2)$$

The middle expression can be solved by the useful identity (exercise) $h'(v) = h(v)(1-h(v))$, to give:

$$\left.\frac{\partial z_2}{\partial v_2}\right|_{v_2=\tilde{v}_2} = h(\tilde{v}_2)(1-h(\tilde{v}_2)) = \tilde{z}_2(1-\tilde{z}_2)$$

The right expression is simply

$$\left.\frac{\partial v_2}{\partial w_2}\right|_{z_1=\tilde{z}_1,w_2=\tilde{w}_2,b_2=\tilde{b}_2} = \tilde{z}_1$$

We can do the same for $\partial/\partial b_2$, we would get the same calculation, with the coefficient 1 instead of z_1 - indeed, we can think about it as another input of constant 1 being input into each one of the nodes.

Derivative w.r.t w_1, b_1

What about the derivative wrt w_1 (and b_1)? Following the chain rule (and the drawing) again,

$$\left.\frac{\partial}{\partial w_1}\ell(y, z_2(z_0))\right|_{\substack{w_1=\tilde{w}_1,b_1=\tilde{b}_1,\\ w_2=\tilde{w}_2,b_2=\tilde{b}_2}}$$
$$= \left.\frac{\partial}{\partial z_2}\ell(y, z_2)\right|_{z_2=\tilde{z}_2} \cdot \left.\frac{\partial z_2}{\partial v_2}\right|_{v_2=\tilde{v}_2} \cdot \left.\frac{\partial v_2}{\partial z_1}\right|_{z_1=\tilde{z}_1,w_2=\tilde{w}_2,b_2=\tilde{b}_2} \cdot \left.\frac{\partial z_1}{\partial v_1}\right|_{v_1=\tilde{v}_1} \cdot \left.\frac{\partial v_1}{\partial w_1}\right|_{w_1=\tilde{w}_1,b_1=}$$

The first two expressions are familiar to us, and in fact we already calculated them! We define

$$\delta_2 = \left.\frac{\partial}{\partial z_2}\ell(y, z_2(z_0))\right|_{\substack{w_1=\tilde{w}_1,b_1=\tilde{b}_1,\\ w_2=\tilde{w}_2,b_2=\tilde{b}_2}}$$

and

$$\delta_1 = \left.\frac{\partial}{\partial z_1}\ell(y, z_2(z_0))\right|_{\substack{w_1=\tilde{w}_1,b_1=\tilde{b}_1,\\ w_2=\tilde{w}_2,b_2=\tilde{b}_2}}$$

Then, we can write

$$\delta_1 = \delta_2 \cdot \left.\frac{\partial z_2}{\partial v_2}\right|_{v_2=\tilde{v}_2} \cdot \left.\frac{\partial v_2}{\partial z_1}\right|_{z_1=\tilde{z}_1,w_2=\tilde{w}_2,b_2=\tilde{b}_2}$$

The last two expressions can be calculated the same way:

$$\left.\frac{\partial z_1}{\partial v_1}\right|_{v_1=\tilde{v}_1} = h(\tilde{v}_1)(1 - h(\tilde{v}_1)) = \tilde{z}_1(1 - \tilde{z}_1)$$

The right expression is simply

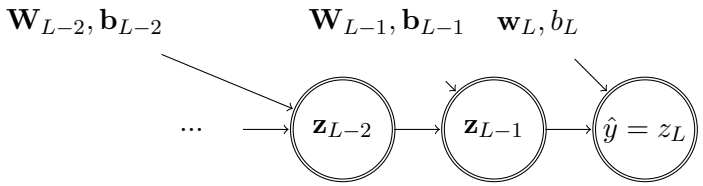
$$\left.\frac{\partial v_1}{\partial w_1}\right|_{w_1=\tilde{w}_1,b_1=\tilde{b}_1} = z_0$$

Summary

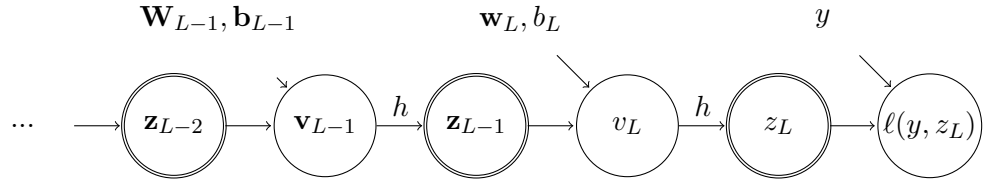
How to calculate the above efficiently? Note that we needed the evaluation of all the functions/variables at the current values of the parameters. We do this using a *forward pass* - Going forward in the graph, starting with $\tilde{v}_1$, using it to calculate $\tilde{z}_1$, then $\tilde{z}_2$, and so forth. Then, we calculate δ_2 as explained above, and use its value to calculate δ_1 - this is the *backward pass* (or, backpropagation). It's easy to extend this logic to any number of layers. Another way to think about it is as a dynamic programming or memoization algorithm - we reuse the same expressions over and over, so we calculate them only once, in the order needed to calculate them.

8.1.2 Many layers, many nodes

We now generalize this to several layers of several nodes. We do this using matrix calculus - see also lesson scribe for a slightly different derivation. In the general case, each layer will now be a vector. The parameters for the transition between layers are matrices and vectors:



With the notation $\mathbf{v}_{t+1} = \mathbf{W}_{t+1}\mathbf{z}_t - \mathbf{b}_{t+1}$, we have:



We can now use matrix calculus to differentiate with simplicity. (It's not new - it's stuff we have all learned in multivariable calculus - Jacobians etc.). We omit the points where the derivatives are evaluated for clarity of presentation - the rationale is the same as before. Denote the i-th row of $\mathbf{W}_t$ by $\mathbf{w}_{t,i}$. Then,

$$\frac{\partial}{\partial \mathbf{w}_{t,i}} \ell(y, z_L(\mathbf{z}_0)) = \frac{\partial}{\partial z_L}\ell(y, z_L) \cdot \frac{\partial z_L}{\partial v_L} \cdot \frac{\partial v_L}{\partial \mathbf{z}_{L-1}} \cdot \frac{\partial \mathbf{z}_{L-1}}{\partial \mathbf{v}_{L-1}} \cdot \frac{\partial \mathbf{v}_{L-1}}{\partial \mathbf{z}_{L-2}} \cdot \ldots \cdot \frac{\partial \mathbf{v}_t}{\partial \mathbf{w}_{t,i}}$$

This gradient is a row vector. We already know the first two expressions, $\frac{\partial}{\partial z_L}\ell(y, z_L)$ and $\frac{\partial z_L}{\partial v_L}$ - they are like before.

The expression $\frac{\partial v_L}{\partial \mathbf{z}_{L-1}}$ is a row vector - it is simply $\tilde{\mathbf{w}}_L$. The expression $\frac{\partial \mathbf{z}_{L-1}}{\partial \mathbf{v}_{L-1}}$ is a matrix - the Jacobian $\mathbf{z}_{L-1}$ as a function of $\mathbf{v}_{L-1}$. It is simple to see its a diagonal matrix with $h'((\tilde{\mathbf{v}}_{L-1})_j)$ on the diagonal j, j. The matrix $\frac{\partial \mathbf{v}_{L-1}}{\partial \mathbf{z}_{L-2}}$ is simply $\tilde{\mathbf{W}}_{L-1}$! So we continue multiplying these matrices, until the final matrix $\frac{\partial \mathbf{v}_t}{\partial \mathbf{w}_{t,i}}$, which is also easy to calculate - only the i-th row is nonzero, and it is $\tilde{z}_t$. It's easy to verify that this gives exactly the same full algorithm as detailed in the lesson scribes. As before, we have a forward pass to calculate all the $\mathbf{v}$-s, and a backward pass to calculate all the $\boldsymbol{\delta}$-s, where $\boldsymbol{\delta}_t = \frac{\partial}{\partial \mathbf{z}_t}\ell(y, z_L(\mathbf{z}_0))$. From this, the calculation of the gradients are as described above and in the lesson.

8.2 Decision Trees

8.2.1 Terminology and Reminder

Assume a binary classification setting (for every training sample, let f be the binary label). We like to decide in each node on the split, i.e., the predicate h to assign to the node. The local parameters are $q = \Pr[f = 1]$, which is the fraction of 1s in the examples reaching the node, $u = \Pr[h = 0]$[1] is the fraction of samples for which $h = 0$ out of the samples reaching the node, $p = \Pr[f = 1|h = 0]$ is the fraction of 1s in the samples reaching the node and having $h = 0$, and $r = \Pr[f = 1|h = 1]$ is the fraction of 1s in the samples reaching the node and having $h = 1$. We have that $q = up + (1 - u)r$. (See Figure 8.1.)

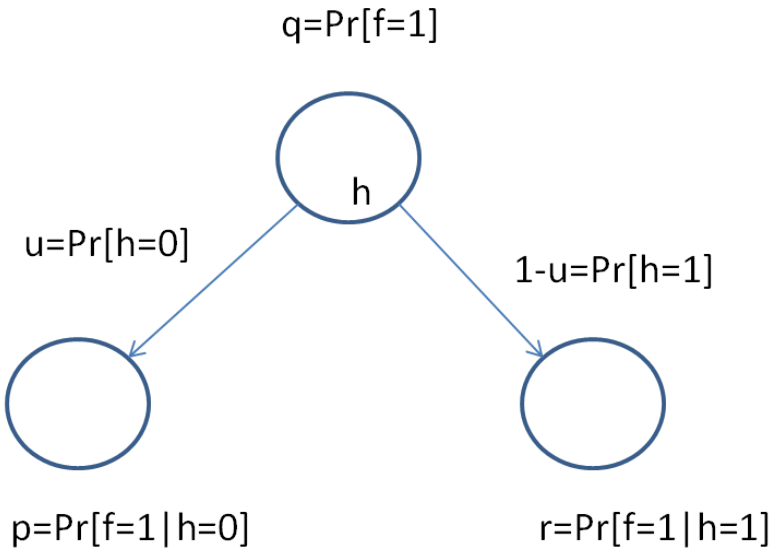


Figure 8.1: The split in a node

Recall the decision-tree algorithm from class: We use a strictly convex node index function $v(\cdot)$[2] that associates a value to a node as a function of the proportion of positively labeled examples in the node (q using our above terminology). Now, by strict convexity of $v(\cdot)$ we have

$$v(q) > u \cdot v(p) + (1 - u)v(r)$$

And at a given node we seek to find a predicate h that splits in a way that mostly reduces the right hand side of the above inequality (the resulting node *potential*).

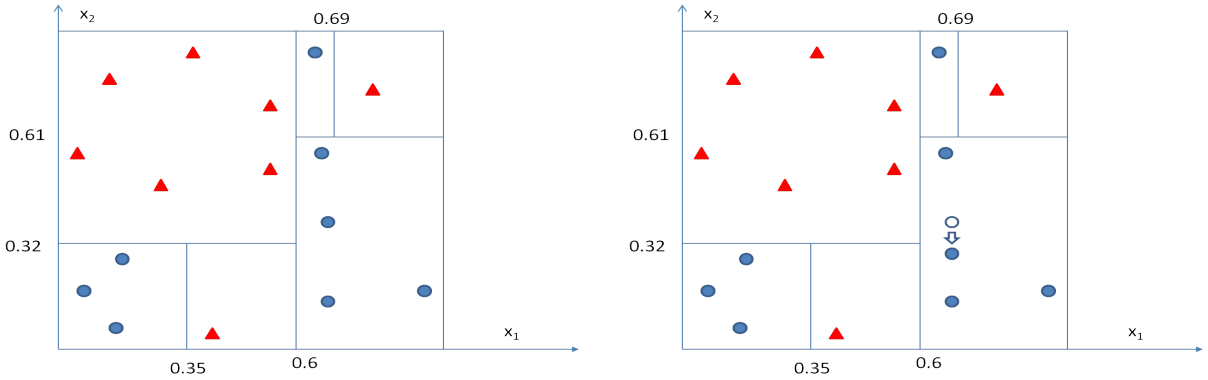
[1] We use $h = 0$ to indicate that the predicate h is *false* and $h = 1$ for the case h is *true*

[2] An example of a split index is $v(p) = -p \log_2 p - (1 - p) \log_2(1 - p)$ which is the binary entropy function. (In class we normalized by multiplying by a half, but this will not make a difference.)

8.2.2 Instability Example

We consider the sample 2-feature binary labeled data in Figure 8.2a [3]. The root's optimal decision stump $h = "x_1 < 0.6"$ reduces the potential [4] from the initial 1 (since the sample contains an equal number of positive and negative samples) to

$$\frac{10}{16}v\left(\frac{7}{10}\right) + \frac{6}{16}v\left(\frac{1}{6}\right) \approx 0.79$$



(a) Original sample set

(b) Slight change in one sample.

Figure 8.2: Example of data for decision tree instability. Triangles are positively labeled and circles are negatively labeled

We continue performing the splits and derive the decision tree of Figure 8.3.

We can now consider what will happen if we slightly modify the location of a single point as follows. (See Figure 8.2b.)

The modified data still has the root split $h = "x_1 < 0.6"$ resulting in the same value ≈ 0.79, but for the root split $h = "x_2 < 0.32"$ we have

$$\frac{7}{16}v\left(\frac{1}{7}\right) + \frac{9}{16}v\left(\frac{7}{9}\right) \approx 0.68 < 0.79$$

This implies that the minor change will change the optimal predicate at the root and might impact the entire tree.

[3] Example from http://www.lsv.uni-saarland.de/pattern_sr_ws0607/psr_0607_Chap10.pdf, slide 30

[4] We use the entropy function throughout.

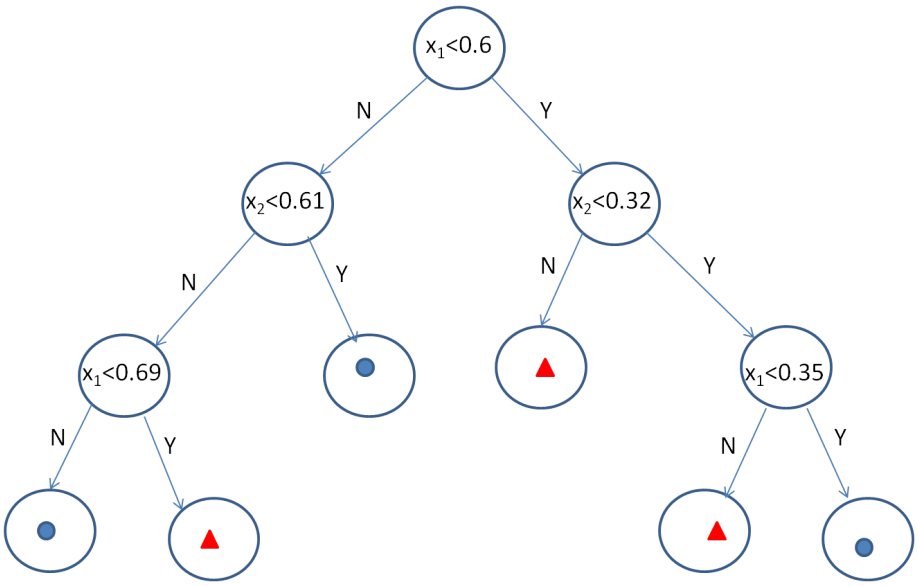


Figure 8.3: The tree that is built