Lab 3

Welcome to the third lab. The first exercise is a simple implementation of gradient descent, the second exercise is a primer on backpropagation with neural networks of 1, 2 and 3 layers, and the third exercise shows the importance of a good weight initialization.

Exercise 1

This exercise is about gradient descent. We will use the function $f(x_1, x_2) = (x_1 - 6)^2 + x_2^2 - x_1 x_2$ as a running example:

1. Use pen and paper to do three iterations of gradient descent:
   - Find the gradient of f;
   - Start from the point $x_1 = x_2 = 6$ and use a step size of $1/2$ for the first step, $1/3$ for the second step and $1/4$ for the third step;
   - What will happen if you keep going?
2. Write a function that performs gradient descent:
   - At the t-th iteration, use a step size of $\alpha t^{-1/2}$
   - Can you find a way to prematurely stop the optimization when you are close to the optimum?

```
func.value = function(x) {
  (x[1] - 6)^2+x[2]^2-x[1]*x[2]
}

func.gradient = function(x) {
  c(
    2*x[1]-x[2]-12,
    -x[1]+2*x[2]
  )
}

func.value(c(6, 6))
```

```
## [1] 0
```

```
func.gradient(c(6, 6))
```

```
## [1] -6  6
```

Does it match what you computed?

```
gradient_descent_optimizer = function(x0, func, grad, max_steps, lrate) {
  x = x0
  v = func(x0)
  for(i in 1:max_steps) {
    lr = lrate / sqrt(i)
    x = x - lr * grad(x)
    vnew = func(x)
    if(v - vnew < 1e-3) {
      break
    }
    v = vnew
  }
  x
}
```

```
gradient_descent_optimizer(c(6, 6), func.value, func.gradient, 10, 0.1)
```

```
## [1] 7.619735 4.380265
```

Play a bit with the starting point and learning rate to get a feel for its behavior; how close can you get to the minimum?

Solution

The gradient of f is:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{vmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{vmatrix} = \begin{vmatrix} 2(x_1 - 6) - x_2 \\ 2x_2 - x_1 \end{vmatrix}$$

For $\mathbf{x} = |6, 6|^T$ we have $f(\mathbf{x}) = 0$ and $\nabla_{\mathbf{x}} f(\mathbf{x}) = |-6, 6|^T$.

Let $\mathbf{x}^{(t)}$ denote the point at the t-th iteration. Then:

t	$\mathbf{x}^{(t)}$	$f(\mathbf{x}^{(t)})$	$\nabla_{\mathbf{x}} f(\mathbf{x})$	$\mathbf{x}^{(t+1)}$
1	$\|6, 6\|$	0	$\|-6, 6\|$	$\|6, 6\| - (1/2) \cdot \|-6, 6\| = \|9, 3\|$
2	$\|9, 3\|$	-9	$\|3, -3\|$	$\|9, 3\| - (1/3) \cdot \|3, -3\| = \|8, 4\|$
3	$\|8, 4\|$	-12	$\|0, 0\|$	$\|8, 4\| - (1/4) \cdot \|0, 0\| = \|8, 4\|$

Where all vectors are intended to be vertical. As the gradient at the last point is zero, nothing will change if we continue to apply this procedure.

Exercise 2

This exercise is about computing gradients with the chain rule, with pen and paper.

1. Show that the derivative of the sigmoid $\sigma(x) = (1 + e^{-x})^{-1}$ is $\sigma(x)(1 - \sigma(x))$
2. Consider the output of a neural network with one layer, $\mathbf{y} = \phi(\mathbf{z}) = \phi(\mathbf{W}\mathbf{x} + \mathbf{b})$ with $\phi(\cdot)$ an arbitrary activation function:
   - Compute the partial derivative of z_i with respect to x_j, w_{jk} and b_j.
   - Compute the partial derivative of y_i wit respect to x_j, w_{jk} and b_j.
3. Consider two layers of a neural network: $\mathbf{y} = \phi_1(\mathbf{W}\phi_2(\mathbf{V}\mathbf{x} + \mathbf{c}) + \mathbf{b})$:
   - Compute the partial derivative of y_i with respect to x_j, v_{jk} and c_j
4. Consider three layers of a neural network $\mathbf{y} = \phi_1(\mathbf{W}\phi_2(\mathbf{V}\phi_3(\mathbf{U}\mathbf{x} + \mathbf{d}) + \mathbf{c}) + \mathbf{b})$:
   - Compute the partial derivative of y_i with respect to x_j, u_{jk} and d_j.

Solution

Question 1

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x}\sigma(x) &= -(1+e^{-x})^{-2} \cdot -e^{-x} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}$$

Question 2

Remember that $z_i = b_i + \sum_j w_{ij}x_j$, which means that:

$$\begin{aligned}
\frac{\partial z_i}{\partial x_j} &= w_{ij} \\
\frac{\partial z_i}{\partial w_{jk}} &= \mathbb{1}[i=j]x_k \\
\frac{\partial z_i}{\partial b_j} &= \mathbb{1}[i=j] \\
\frac{\partial y_i}{\partial z_j} &= \phi'(z_j)\mathbb{1}[i=j] \\
\frac{\partial y_i}{\partial x_j} &= \phi'(z_i)w_{ij} \\
\frac{\partial y_i}{\partial w_{jk}} &= \phi'(z_i)x_k\mathbb{1}[i=j] \\
\frac{\partial y_i}{\partial b_j} &= \phi'(z_i)\mathbb{1}[i=j]
\end{aligned}$$

Question 3

For brevity, let $\phi_1'(\ldots)$ be the derivative of ϕ_1 computed using the value of its argument $\mathbf{W}\phi_2(\mathbf{V}\mathbf{x}+\mathbf{c})+\mathbf{b}$, same for ϕ_2.

$$\begin{aligned}
\frac{\partial y_i}{\partial x_j} &= \phi_1'(\ldots)_i \cdot \sum_k \left(v_{ik} \cdot \phi_2'(\ldots)_k \cdot w_{kj}\right) \\
\frac{\partial y_i}{\partial v_{jk}} &= \phi_1'(\ldots)_i \cdot w_{ik} \cdot \phi_2'(\ldots)_k \cdot x_j \\
\frac{\partial y_i}{\partial c_j} &= \phi_1'(\ldots)_i \cdot v_{ij} \cdot \phi_2'(\ldots)
\end{aligned}$$

Question 4

$$\frac{\partial y_i}{\partial x_j} = \phi_1'(\ldots)_i \cdot \sum_k \left(w_{ik} \cdot \phi_2'(\ldots)_k \cdot \sum_l \left(v_{kl} \cdot \phi_3'(\ldots)_a \cdot u_{lj} \right) \right)$$
$$\frac{\partial y_i}{\partial u_{jk}} = \phi_1'(\ldots)_i \cdot \sum_l \left(w_{il} \cdot \phi_2'(\ldots)_l \cdot v_{lj} \cdot \phi_3'(\ldots)_j \cdot x_k \right)$$
$$\frac{\partial y_i}{\partial d_j} = \phi_1'(\ldots)_i \cdot \sum_k \left(w_{ik} \cdot \phi_2'(\ldots)_k \cdot v_{kj} \cdot \phi_3'(\ldots)_j \right)$$

Exercise 3

This exercise shows that proper initialization of the weights of a neural network is crucial.

1. Show, for a feed-forward neural network with tanh activation on the hidden layers, linear output activation and mean squared error loss function, that the origin in parameters space is a stationary point of the error function.
2. Consider again a feed-forward neural network with tanh activation on the hidden layers, and consider both inputs and weights as i.i.d. random variables with zero mean and a certain variance $Var[x]$ and $Var[w^i]$ (this is the variance of each element of the weight matrix of the i-th layer). Furthermore, assume that all inputs and hidden activations are small enough, so that the tanh is approximately linear. For this exercise, forget about biases. The goal of this exercise is to find the best variance of the weights for a random initialization of a neural network.
   - Show that, for X and Y independent random variables, $Var[XY] = Var[X]E[Y]^2 + Var[Y]E[X]^2 + Var[X]Var[Y]$, where $E[\cdot]$ denotes expectation.
   - Compute the variance of z_j^i, the i-th element of the j-th hidden layer after activation. First, find a recursive formula in terms of the variance of the previous layer, then unroll it to find a closed form in terms of the variance of the inputs, of the weights, and the number of neurons in each layer.
   - Compute the variance of $\partial\mathcal{L}/\partial z_j^i$. Stop at $Var[\partial\mathcal{L}/\partial z_d]$, where d is the number of layers of the network. Again, derive a closed formula.
   - Compute the variance of $\partial\mathcal{L}/\partial w_{ij}^k$ with a closed formula, where w_{ij}^k is the weight connecting the i-th neuron of the k-th layer to the j-th neuron of the $(k+1)$-th layer.
   - As you can see, variances are multiplied together both in the forward pass and in the backward pass. Why is this a problem as the size and number of layers grows?
   - In order to avoid these troubles, we would like to keep the variances constant throughout the network, i.e. have $Var[z_i] = Var[z_j]$ and $Var[\partial\mathcal{L}/\partial z_i] = Var[\partial\mathcal{L}/\partial z_j]$ for all pairs of layers i and j. Use the formulas you derived earlier to find the variance of the weights that satisfies these constraints; as you can see, this is only possible when all layers have the same number of neurons. As a compromise between these two constraints, the variance of every weight should depend on the average number of neurons of the two layers they connect.
   - Finally, assume the weights of layer i are initialized from an uniform distribution in the interval $[-a_i, a_i]$. Find a suitable value for a_i so that $Var[w^i]$ is the same as what you found in the previous point.
   - Congratulations, you have just discovered the Glorot (aka Xavier) initialization, one of the most widely used formulas to initialize weights in deep learning!

Solution

Question 1

Note that the hidden activations of such a network, as well as its outputs, are always zero for every input:

$$z_k^i = \tanh\left(b_{k-1}^i + \sum_j z_{k-1}^j \cdot w_{ij}^{k-1} \right) = \tanh\left(0 + \sum_j z_{k-1}^j \cdot 0 \right) = \tanh(0) = 0$$

Where z_k^i is the activation of the i-th neuron of the k-th layer, b_k^i is the bias of the i-th neuron of the k-th layer, and w_{ij}^k connects neuron j of layer k to neuron i of layer $k+1$. Now consider the derivative of the loss with respect to w_{ij}^k, connecting neuron j of layer k to neuron i of layer $k+1$:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k} = \frac{\partial \mathcal{L}}{\partial z_{k+1}^i} \cdot \tanh'\left(b_k^i + \sum_{l=1}^{n_k} z_k^l \cdot w_{il}^k\right) \cdot z_k^j = 0$$

Because $z_k^j = 0$ for every j and k. Similarly, we can prove:

$$\frac{\partial \mathcal{L}}{\partial z_k^i} = \sum_j w_{ji}^k \cdot \tanh'\left(b_k^j + \sum_l z_k^l \cdot w_{jl}^k\right) \cdot \frac{\partial \mathcal{L}}{\partial z_{k+1}^j} = 0$$

For every i and k, since $w_{ji}^k = 0$. This allows us to show that

$$\frac{\partial \mathcal{L}}{\partial b_k^i} = \frac{\partial \mathcal{L}}{\partial z_{k+1}^i} \cdot \tanh'\left(b_i^k + \sum_{l=1}^{n_k} z_k^l \cdot w_{il}^k\right) = 0$$

Therefore, all gradients are zero, and such a network will never learn anything.

Question 2

First, we prove the formula about the variance of the product of two independent random variables:

$$\begin{aligned}
Var[XY] &= E[(XY)^2] - E[XY]^2 \\
&= E[X^2Y^2] - E[X]^2E[Y]^2 \\
&= E[X^2]E[Y^2] - E[X]^2E[Y]^2 \\
&= E[X^2]E[Y^2] - E[X]^2E[Y]^2 \\
&\quad + (E[X]^2E[Y^2] - E[X]^2E[Y^2]) \\
&\quad + (E[X^2]E[Y]^2 - E[X^2]E[Y]^2) \\
&\quad + (E[X]^2E[Y]^2 - E[X]^2E[Y]^2) \\
&= E[X]^2(E[Y^2] - E[Y]^2) \\
&\quad + E[Y]^2(E[X^2] - E[X]^2) \\
&\quad + (E[X^2] - E[X]^2)(E[Y^2] - E[Y]^2) \\
&= E[X]^2Var[Y] + E[Y]^2Var[X] + Var[X]Var[Y]
\end{aligned}$$

In particular, since we assumed weights and inputs to have zero mean, we will be able to write $Var[XY] = Var[X]Var[Y]$ in the following steps.

The hidden activations are computed as

$$z_j^i = \tanh\left(\sum_{k=1}^{n_{j-1}} z_{j-1}^k \cdot w_{ik}^j\right) \approx \sum_{k=1}^{n_{j-1}} z_{j-1}^k \cdot w_{ik}^j$$

Where the last step follows because we assumed the activations to be small enough and $\tanh'(0) = 1$. The variance is therefore:

$$\begin{aligned}
Var\left[z_j^i\right] &\approx Var\left[\sum_{k=1}^{n_{j-1}} z_{j-1}^k \cdot w_{ik}^j\right] \\
&= \sum_{k=1}^{n_{j-1}} Var\left[z_{j-1}^k \cdot w_{ik}^j\right] \\
&= \sum_{k=1}^{n_{j-1}} Var\left[z_{j-1}^k\right] \cdot Var\left[w_{ik}^j\right] \\
&= n_{j-1} \cdot Var\left[z_{j-1}\right] \cdot Var\left[w^j\right] \\
&= Var[x] \prod_{k=1}^{j} n_{k-1} Var\left[w^k\right]
\end{aligned}$$

Where n_j is the number of neurons in layer j, and n_0 is the dimensionality of the input.

The gradients are computed as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial z_j^i} &= \sum_{k=1}^{n_{j+1}} w_{ki}^{j+1} \cdot \tanh'\left(\sum_{l=1}^{n_{j-1}} z_{j-1}^l \cdot w_{il}^j\right) \cdot \frac{\partial \mathcal{L}}{\partial z_{j+1}^k} \\
&\approx \sum_{k=1}^{n_{j+1}} w_{ki}^{j+1} \cdot \frac{\partial \mathcal{L}}{\partial z_{j+1}^k}
\end{aligned}$$

Where, again, the last step follows because the activations are small enough. Its variance is, then:

$$\begin{aligned}
Var\left[\frac{\partial \mathcal{L}}{\partial z_j^i}\right] &\approx Var\left[\sum_{k=1}^{n_{j+1}} w_{ki}^{j+1} \cdot \frac{\partial \mathcal{L}}{\partial z_{j+1}^k}\right] \\
&= n_{j+1} \cdot Var\left[w^{j+1}\right] \cdot Var\left[\frac{\partial \mathcal{L}}{\partial z_{j+1}}\right] \\
&= Var\left[\frac{\partial \mathcal{L}}{\partial z_d}\right] \cdot \prod_{k=j+1}^{d} n_k Var\left[w^k\right]
\end{aligned}$$

Finally, the gradient of the weights is computed as follows:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k} = \frac{\partial \mathcal{L}}{\partial z_k^i} \cdot \tanh'\left(\sum_{l=1}^{n_{k-1}} z_{k-1}^l \cdot w_{il}^k\right) \cdot z_{k-1}^j \approx \frac{\partial \mathcal{L}}{\partial z_k^i} \cdot z_{k-1}^j$$

Therefore its variance is

$$
\begin{aligned}
Var\left[\frac{\partial \mathcal{L}}{\partial w_{ij}^k}\right] &\approx Var\left[\frac{\partial \mathcal{L}}{\partial z_k^i}\right] \cdot Var\left[z_{k-1}^j\right] \\
&= \left(Var\left[\frac{\partial \mathcal{L}}{\partial z_d}\right] \cdot \prod_{l=k+1}^{d} n_l Var\left[w^l\right]\right) \cdot \left(Var[x] \prod_{l=1}^{k-1} n_{l-1} Var\left[w^l\right]\right) \\
&= Var[x] \cdot \prod_{l=1}^{k-1} n_{l-1} Var\left[w^l\right] \cdot \prod_{l=k+1}^{d} n_l Var\left[w^l\right] \cdot Var\left[\frac{\partial \mathcal{L}}{\partial z_d}\right]
\end{aligned}
$$

As you can see, the variance of the activations and of the gradients depends on:

- The variance of the inputs.
- The variance of the gradient of the loss.
- The variance of the weights of each layer.
- The number of neurons of each layer.

As everything is multiplied together, if we do not control these variances we are going to encounter numerical problems with all but the tiniest of neural network; after several layers, both activations and gradients either explode to huge numbers, or vanish to essentially zero.

We now want to keep the variance of the activations constant, i.e., for every pair of layers i and j:

$$
Var[x] \prod_{k=1}^{i} n_{k-1} Var\left[w^k\right] = Var[x] \prod_{k=1}^{j} n_{k-1} Var\left[w^k\right]
$$

Suppose without loss of generality that $j > i$, then the right-hand side equals the left-hand side multiplied by some more things, therefore equality can only be attained when

$$
n_{k-1} Var\left[w^k\right] = 1 \quad \forall k
$$

A similar reasoning shows that

$$
Var\left[\frac{\partial \mathcal{L}}{\partial z_i}\right] = Var\left[\frac{\partial \mathcal{L}}{\partial z_j}\right] \iff n_k Var\left[w^k\right] = 1 \quad \forall k
$$

These two inequalities can only be satisfied at the same time when $n_k = n_{k-1}$ for every layer k. Following the suggestion and using the average number of neurons, we find that

$$
Var[w^k] = \frac{2}{n_{k-1} + n_k}
$$

An uniform distribution in $[-a_i, a_i]$ has variance $a_i^2/3$, therefore

$$
\frac{a_i^2}{3} = \frac{2}{n_{k-1} + n_k} \iff a_i = \sqrt{\frac{6}{n_{k-1} + n_k}}
$$