Artificial Neural Networks (Gerstner). Solutions for week 5

Error function and Optimization

Exercise 1. Averaging of Stochastic gradients.

We consider stochastic gradient descent in a network with three weights, (w_1, w_2, w_3).

Evaluating the gradient for 100 input patterns (one pattern at a time), we observe the following time series

for w_1: observed gradients are 1.1; 0.9, 1.1; 0.9; 1.1; 0.9; ...

for w_2: observed gradients are 0.1; 0.1; 0.1; 0.1; 0.1; ...

for w_3: observed gradients are 1.1; 0; -0.9; 0; 1.1; 0; -0.9; 0; 1.1; 0; -0.9; ...

a. Calculate the mean gradient $\langle g_k \rangle$ for w_k, $k \in [1, 2, 3]$.

b. Calculate the mean of the squared gradient $\langle g_k^2 \rangle$ for w_1 and w_2 and w_3.

c. Divide the result of (a) by that of (b) so as to calculate $\langle g_k \rangle / \langle g_k^2 \rangle$.

d. You use an an algorithm to update a variable m:
$m(n+1) = \rho m(n) + (1-\rho)x(n) \quad (*)$
where $\rho \in [0, 1)$ and $x(n)$ refers to an observed time series $x(1), x(2), x(3), ...$
Show that, if all values of x are identical [that is, $x(k) = \bar{x}$ for all k], then the algo (*) converges to $m = \bar{x}$.

e. Assume the initial condition $m(0) = 0$. Show that, for $1 - \rho \ll 1$ the algorithm outputs in time step $n + 1$ the value

$$m(n+1) = (1-\rho) \sum_{k=0}^{n} \exp[-(1-\rho)k] \cdot x(n-k)$$

Hint: (i) compare $m(n+1)$ with $m(n)$ and reorder terms. (ii) At the end of your calculation you may approximate $\exp[\epsilon] = 1 + \epsilon$ (which is valid for small $\epsilon \ll 1$).

f. Your friend makes the following statement:
The algo (*) performs a running average of the time series $x(n)$ with an exponentially weighted window that extends roughly over $1/(1-\rho)$ samples. Therefore, if you want to include about 100 samples in the average, you should choose $\rho = 0.99$.
Is your friend's claim correct?

Solution:

a. The gradients correspond to three sequences with periods of 2 (for w_1), 1 (for w_2) and 4 (for w_3). Over 100 samples, they will repeat 50, 100, and 25 times respectively. This gives us

$$\langle g_1 \rangle = \frac{1}{100} \sum_{i=1}^{50} (1.1 + 0.9) = 1 \ ,$$

$$\langle g_2 \rangle = \frac{1}{100} \sum_{i=1}^{100} 0.1 = 0.1 \ ,$$

$$\langle g_3 \rangle = \frac{1}{100} \sum_{i=1}^{25} (1.1 + 0 - 0.9 + 0) = 0.05 \ .$$

b. By the same approach,

$$\langle g_1^2 \rangle = \frac{1}{100} \sum_{i=1}^{50} (1.1^2 + 0.9^2) = 1.01 \ ,$$
$$\langle g_2^2 \rangle = \frac{1}{100} \sum_{i=1}^{100} 0.1^2 = 0.01 \ ,$$
$$\langle g_3^2 \rangle = \frac{1}{100} \sum_{i=1}^{25} (1.1^2 + 0 + (-0.9)^2 + 0^2) = 0.505 \ .$$

c. From the previous two results,

$$\langle g_1 \rangle / \langle g_1^2 \rangle = 0.99 \ ,$$
$$\langle g_2 \rangle / \langle g_2^2 \rangle = 10 \ ,$$
$$\langle g_3 \rangle / \langle g_3^2 \rangle \approx 0.1 \ .$$

d. We can expand the expression for $m(n+1)$ recursively into an expression in terms of all previous data points and the initial mean $m(0)$,

$$\begin{aligned} m(n+1) &= \rho m(n) + (1-\rho)x(n) \\ &= \rho[\rho m(n-1) + (1-\rho)x(n-1)] + (1-\rho)x(n) \\ &= \rho^{n+1} m(0) + (1-\rho) \sum_{k=0}^{n} \rho^k x(n-k) \\ &= \rho^{n+1} m(0) + \bar{x} \cdot (1-\rho) \sum_{k=0}^{n} \rho^k \end{aligned}$$

Taking $n \to \infty$ and $\rho \in [0, 1)$, the first term goes to 0. Recognizing the geometric series in the second term, we have

$$\begin{aligned} m(n+1) &\approx \bar{x} \cdot (1-\rho) \frac{1}{1-\rho} \\ &= \bar{x}. \end{aligned}$$

e. Since $1 - \rho \ll 1$, we approximate

$$\begin{aligned} \exp[-(1-\rho)k] &= \exp(-(1-\rho))^k \\ &\approx (1 + (-(1-\rho)))^k \\ &= \rho^k \end{aligned}$$

From the last solution, taking $m(0) = 0$, we have

$$\begin{aligned} m(n+1) &= (1-\rho) \sum_{k=0}^{n} \rho^k x(n-k) \\ &\approx (1-\rho) \sum_{k=0}^{n} \exp[-(1-\rho)k] \cdot x(n-k) \end{aligned}$$

f. From above, a sample k time steps in the past has a weight of $w_k = (1-\rho)\rho^k$. From (c), we know that $\sum_{k=0}^{n} w_k \approx 1$ for high values of n, in which case we can also ignore $m(0)$.

We can consider the contribution of all samples more than 100 time steps in the past to the current value of the mean, for $\rho = 0.99$:

$$\begin{aligned}\frac{\sum_{k=100}^{n} w_k}{\sum_{k=0}^{n} w_k} &\approx (1-\rho)\sum_{k=100}^{n}\rho^k \\ &\approx 1-(1-\rho)\sum_{k=0}^{99}\rho^k \\ &= 1-(1-\rho)\Big(\frac{1-\rho^{100}}{1-\rho}\Big) \\ &= \rho^{100} \approx 0.366\end{aligned}$$

so samples more than 100 steps in the past will account for roughly one third of m. The value of $\rho^{1/(1-\rho)}$ converges to $\exp(-1) \approx 0.368$ for increasing values of ρ (which follows from the fact that $x \approx -\ln(1-x)$ for small values of x). So, for large enough ρ, the most recent $n = \rho^{1/(1-\rho)}$ samples account for roughly two thirds of the exponential moving average (n is the *time constant*). We can think of this as roughly the number of samples being averaged over, although we should note that samples more than $1/(1-\rho)$ time steps in the past can still make a significant contribution to m (one third).

Exercise 2. Simple Perceptron and Bagging

We have four data points:

Two positive examples $t^1 = t^2 = 1$ at $x^1 = (1,0)^T$ and $x^2 = (0,1)^T$; and Two negative examples $t^3 = t^4 = 0$ at $x^3 = (0,0)^T$ and $x^4 = (1,1)^T$.

a. Draw (with replacement) four times randomly from this data set. What is the probability that you draw each example exactly once?

b. You have generated four new data sets $1 \le k \le 4$ by drawing with replacement from the above set. Each set contains four points. You find that in data set k point k is missing ($1 \le k \le 4$).

You work with the perceptron algorithm with hard gain function $g(a) = 1$ for $a > 0$ and zero otherwise.

Make a graph in the data space (input space) and sketch in the graph a solution that the perceptron algorithm finds for data set $k = 1$. Draw the hyperplane.

c. Sketch in the same graph, a solution (one each) that the perceptron algorithm finds for $k = 2, 3, 4$. Label your proposed solutions with $k = 1...4$.

d. Now you perform bagging. What is the value of the (real-valued) bagged output in each region of the above graph in response to an arbitrary data point x^5. In the above graph, give the regions a different texture and write in each region a number which indicates the amplitude of the bagged response.

e. Now you perform majority voting. How many of the 4 data points are correctly classified?

f. Replace the four points by four Gaussian clusters of 25 data points each (Gaussians centered x^1, x^2, x^3, x^4) with standard deviation $\sigma = 0.1$ each; labels are the same for all points inside one Gaussian cluster.) Repeat the above arguments. Assume that the resampled data set k

has 20 data points from cluster k, 30 data points from another cluster $k' \neq k$, and 25 from the remaining two clusters.

Sketch a plot of this new problem on a separate page and repeat the above arguments (draw the hyperplanes etc, parts b - e). Imagine you generate new data points (from the four Gaussians) for the test set. What's the probability for one of those point of not being correctly clustered after bagging with majority vote?

Solution:

a. You want to pick up each data exactly once, but you don't care about the order in which you pick them. In the first pick, any data is good, in the second pick you have probability 3/4 to pick a data different from the first one, in the third pick you have probability 2/4 to pick a data different from the first two and in the final pick you have probability 1/4 to pick the last data you need. Given that each pick is done independently, the probability of picking 4 different data is $p = 1 \cdot 3/4 \cdot 2/4 \cdot 1/4 = 6/64$. This is also equivalent to $4!/4^4 = 6/64$, where $1/4^4$ is the probability of a specific (ordered) combination of 4 data and the factor 4! is the number of possible permutation (since we don't care about the order).

b. Data set $k = 1$ contains all original data, but x^1. Refer to right hand side of Fig. 1. Since we have chosen g to be an Heavyside function, any hyperplane which correctly separates correctly the points has error $E = 0$, so they are equally good (for this choice of g).

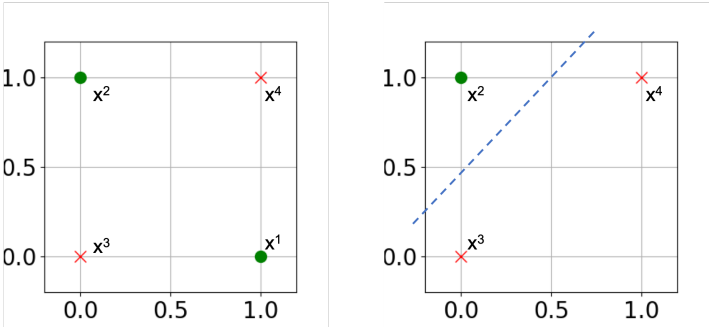

Figure 1: Left side) all 4 data points are plotted. It's the XOR problem and can't be classified with a single layer perceptron. Right side) Data space of the data set for $k = 1$. The large dotted blue line indicates one possible solution (hyperplane) of the perceptron algorithm for this data set.

c. You can see all 4 solutions (hyperplanes) in Fig. 2.

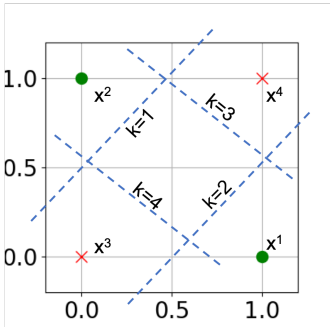

Figure 2

d. You can see the values of $\hat{y}$ in the different regions of the input space in Fig. 3.

e. Just by looking at Fig. 3, we can conclude that all 4 initial data points are correctly classified by majority voting.

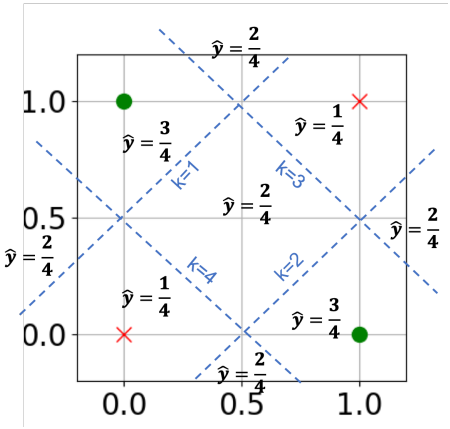

Figure 3

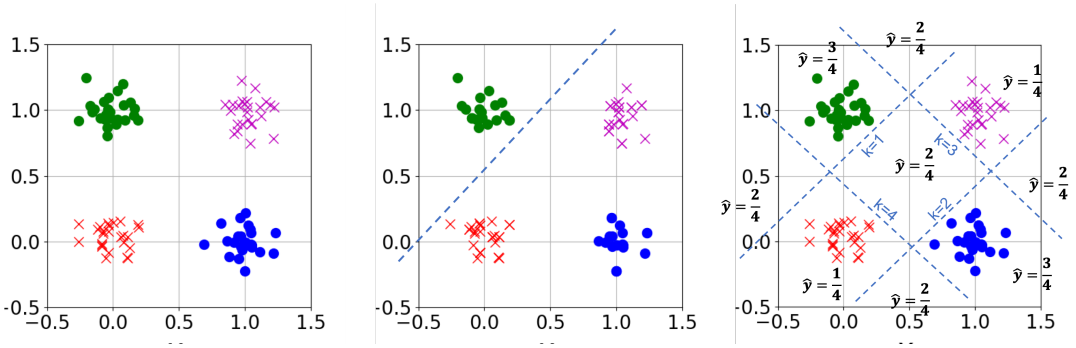

Figure 4: Dots = positive points, crosses= negative points Left) All input data are shown. 25 data points are sample from each of the 4 Gaussian distributions centered on the 4 data points of point a. All point which are sample from the same Gaussian have the same name, so x^1 are in blue, x^2 are in green, x^3 are in red and x^4 are in purple. Center) Data set $k = 1$. We sample with replacement 20 point from x^1, 25 points from x^2, 30 points from x^3 and 25 points from x^4. Right) Analogue of Fig. 2.

f. Fig. 4 (on the right) shows that with the suggested sampling we would still get an hyperplane similar to that of point b. In fact, by placing the hyperplane like in the figure, we have $E = 20$, corresponding to the 20 data points of type x^1, which is the lowest error we can have in this situation. By creating analogously new data set $k = 2, 3, 4$ we get just the analogue of Fig. 2, with the same bagged output and therefore the same outcome of majority voting. Now we generate 100 new data points. How many data point will be classified correctly depends on the Euclidean distance of the hyperplanes from the mean of the cluster. For simplicity, we will consider the error probabilities associated to 1-dimensional Gaussians. In this way we can use the probabilities is Fig. 5. We generate one data point from the Gaussian distribution $N(x^2, \sigma = 0.1)$. If the hyperplane for $k = 1$ is at a distance of 3σ from $\langle x^2 \rangle$, then the probability that a point get mis-classified is $p_{\text{error}} = 1 - 0.9973$. Now we can ask the question: what is the probability that the hyperplane $k = 1$ (drawn with the above method) lies at a distance of 3σ from center of mass of the cluster $\langle x^2 \rangle$? The cluster x^2 contains 25 data points and by construction, they are all on the same side of hyperplane $k = 1$, the hyperplane has probability $p = 0.9973^{25} = 0.93$ of being paced here.

Exercise 3. Bias and variance of gradient estimators

For training data $(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^P, y^P)$ and some loss $E(\boldsymbol{w}) = \frac{1}{P} \sum_\mu \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$, the gradient is given by $\nabla E(\boldsymbol{w}) = \frac{1}{P} \sum_\mu \nabla \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$, with e.g. $\ell(x, y) = \frac{1}{2}(x - y)^2$.

a. In each step of stochastic gradient descent one sample $(\boldsymbol{x}^\mu, y^\mu)$ of the training data is selected.

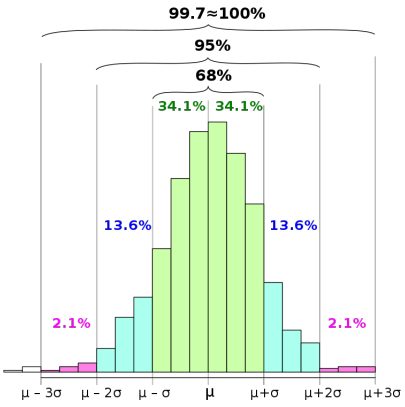

Figure 5: thanks to Wikipedia

Show that $\nabla \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$ is an unbiased estimator of $\nabla E(\boldsymbol{w})$, if each training sample is selected with equal probability. Hint: An estimator $\hat{\theta}$ of a quantity θ is called unbiased, if its expectation $\mathbb{E}\left[\hat{\theta}\right] = \theta$.

b. Instead of single sample stochastic gradient descent it is common practice to use mini-batches. Show that the mini-batch gradient estimator $\frac{1}{M}\sum_{i=1}^{M} \nabla \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^i), y^i)$, with $1 < M < P$, has lower variance than the single sample estimator, if the samples $(\boldsymbol{x}^i, y^i)$ in each mini-batch are sampled uniformly from the training data.

c. How does this exercise link to Ex. 2 of week 1?

Solution:

a. Let μ be a random training sample selected uniformly at random from $\{1, ..., P\}$. Then

$$\begin{aligned} \mathbb{E}\left[\nabla \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)\right] &= \frac{1}{P}\sum_{\mu} \nabla \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu) && (1) \\ &= \nabla E(\boldsymbol{w}) && (2) \end{aligned}$$

b. Let $X_\mu = \nabla \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$ be the gradient from a single training sample. The variance of the mini-batch gradient of size M, $\frac{1}{M}\sum_{i=1}^{M} X_i$ for $\{i, ..., M\}$ selected uniformly at random from $\{1, ..., P\}$ is

$$\mathbb{E}\left[\left(\frac{1}{M}\sum_{i=1}^{M} X_i - \mathbb{E}\left[X\right]\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{M}\sum_{i=1}^{M} (X_i - \mathbb{E}\left[X\right])\right)^2\right] \quad (3)$$

$$= \frac{1}{M^2}\left(\sum_{i=1}^{M} \mathbb{E}\left[(X_i - \mathbb{E}\left[X\right])^2\right] + \sum_{i \neq j} \mathbb{E}\left[(X_i - \mathbb{E}\left[X\right])(X_j - \mathbb{E}\left[X\right])\right]\right) \quad (4)$$

$$= \frac{1}{M}\mathrm{Var}\left[X\right] \quad (5)$$

Exercise 4. ADAM and minibatches.

Suppose that in a project you have already spent some time on optimizing the ADAM parameters ρ_1 and ρ_2 while you ran preliminary tests with a minibatch size of 128 on your computer.

For the final run you get access to a bigger and faster computer that allows you to run minibatches of size 512.

How should you rescale ρ_1 and ρ_2 so as to expect roughly the same behavior of the two machines on the training base?

Hint: For ρ_1 you can directly use the results from Exercise 1. However, for ρ_2 you may want to distinguish between the time series for w_1 and that for w_3. Why? Think of the time series in exercise 1 as the gradients of true stochastic gradient. Then make batches of size 2 and 4, and redo the calculation of the squared gradient. What do you observe?

Solution:

We assume that we need to average over the gradients from the last n samples to get a good approximation to the batch gradient; this averaging happens both within a minibatch (of size s) and across minibatches. When using exponential smoothing as in ADAM, we take $\frac{n}{s} \approx 1/(1-\rho_1)$. To get similar behaviour, we take

$$\begin{aligned}
\frac{n}{s} &\approx \frac{1}{1-\rho_{1,s}} \\
n &\approx \frac{s}{1-\rho_{1,s}} \\
n &\approx \frac{128}{1-\rho_{1,128}} = \frac{512}{1-\rho_{1,512}} \\
1-\rho_{1,512} &= 4(1-\rho_{1,128}) \\
\rho_{1,512} &= 4\rho_{1,128} - 3.
\end{aligned}$$

E.g. if we used $\rho_1 = 0.99$ for minibatch size 128, we can take $\rho_1 = 0.96$ on the new machine with minibatch size 512.

For the squared gradients, we note that averaging *within* a minibatch and averaging *between* minibatches are no longer the same. Within a minibatch, we average over the gradients themselves, while between minibatches, we average over the squared gradients. For instance, taking the w_3 series above with minibatch size 2,

$$\begin{aligned}
r &\approx (0.5(1.1+0))^2(1-\rho)\sum_{k \text{ even}}^{n}\rho^k + (0.5(-0.9+0))^2(1-\rho)\sum_{k \text{ odd}}^{n}\rho^k \\
&\approx (0.5(1.1+0))^2(0.5) + (0.5(-0.9+0))^2(0.5) \\
&= 0.2525
\end{aligned}$$

While for minibatch size 4 we have

$$\begin{aligned}
r &\approx (0.25(1.1+0+(-0.9)+0))^2(1-\rho)\sum_{k=0}^{n}\rho^k \\
&= 0.0025
\end{aligned}$$

which are very different results. For w_1 with minibatch size 2 and 4, we get $r = 1$ in both cases. As a result, there is no straightforward scaling relationship between ρ_2 and the minibatch size.

Exercise 5. Unitwise learning rates

Consider minimizing the *narrow valley* function $E(w_1, w_2) = |w_1| + 75|w_2|$ by gradient descent.

a. Sketch the equipotential lines of E, i.e. the points in the $w_1 - w_2$-plane, where $E(w_1, w_2) = c$ for different values of c.

b. Start at the point $\boldsymbol{w}^{(0)} = (10, 10)$ and make a gradient descent step, i.e. $\boldsymbol{w}^{(1)} = \boldsymbol{w}^{(0)} - \eta(\partial E/\partial w_1, \partial E/\partial w_2)$ with $\eta = 0.1$.

Hint: Use the numeric definition of $\partial|x|/\partial x = sgn(x)$ if $x \neq 0$ and 0 otherwise.

c. Continue gradient descent, i.e .compute $\boldsymbol{w}^{(2)}, \boldsymbol{w}^{(3)}$ and $\boldsymbol{w}^{(4)}$ and draw the points $\boldsymbol{w}^{(0)}, \ldots, \boldsymbol{w}^{(4)}$ in your sketch with the equipotential lines. What do you observe? Can you choose a better value for η such that gradient descent converges faster?

d. Repeat now the gradient descent procedure with different learning rates for the different dimensions, i.e. $\boldsymbol{w}^{(1)} = \boldsymbol{w}^{(0)} - (\eta_1 \partial E/\partial w_1, \eta_2 \partial E/\partial w_2)$ with $\eta_1 = 1$ and $\eta_2 = 1/75$. What do you observe? Can you choose better values for η_1 and η_2 such that gradient descent converges faster?

e. An alternative to individual learning rates is to use momentum, i.e. $\Delta\boldsymbol{w}^{(t+1)} = -\eta(\partial E/\partial w_1, \partial E/\partial w_2) + \alpha\Delta\boldsymbol{w}^{(t)}$ with $\alpha \in [0, 1)$ and $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \Delta\boldsymbol{w}^{(t+1)}$.

Repeat the gradient descent procedure for 3 steps with $\eta = 0.2$ and $\alpha = 0.5$. What do you observe?

f. Assume $\partial E/\partial w_1 = 1$ in all time steps while $\partial E/\partial w_2 = \pm 75$ switches the sign in every time step. Compute $\lim_{t\to\infty} \Delta\boldsymbol{w}^{(t)}$ as a function of η and α. Hint: $\sum_{s=0}^{t} \alpha^s = \frac{1-\alpha^{t+1}}{1-\alpha}$.

g. What do you conclude from this exercise in view of training neural networks by gradient descent with or without momentum?

Solution:

a. In Figure 6 is shown the solution of $|w_2| = \frac{c-|w_1|}{75}$ for three different value of c.

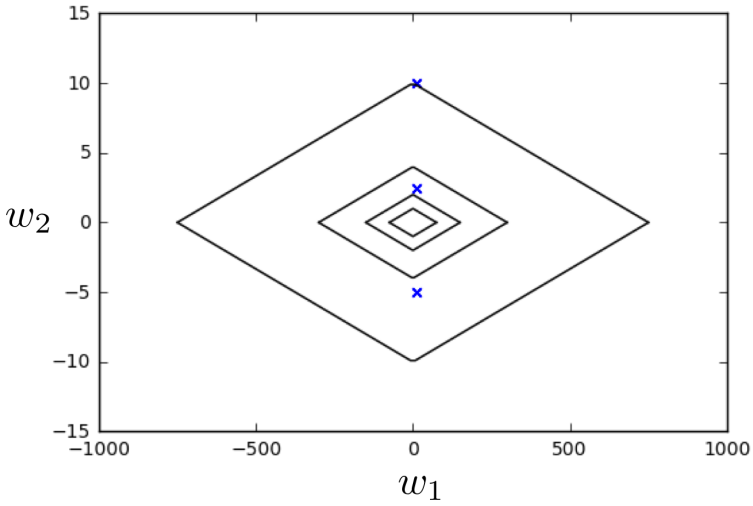

Figure 6: Equipotential lines of the narrow valley function. $c = \{750, 300, 150, 75\}$ from outer to inner lines. The blue cross corresponds to the first iterations of gradient descent.

b.

$$\frac{\partial E}{\partial w_1} = \text{sign}(w_1) \tag{6}$$

$$\frac{\partial E}{\partial w_2} = 75\,\text{sign}(w_2) \tag{7}$$

$$\boldsymbol{w}^{(1)} = (10, 10) - 0.1(1, 75) = (9.9, 2.5) \tag{8}$$

c.

$$\boldsymbol{w}^{(2)} = (9.9, 2.5) - 0.1(1, 75) = (9.8, -5) \tag{9}$$
$$\boldsymbol{w}^{(3)} = (9.8, -5) - 0.1(1, -75) = (9.7, 2.5) \tag{10}$$
$$\boldsymbol{w}^{(4)} = (9.7, 2.5) - 0.1(1, 75) = (9.6, -5) \tag{11}$$

The points are sketched on Figure 6. Because of the big difference in amplitudes of the partial derivatives, finding the correct learning rate is difficult. A smaller learning rate can prevent the oscillations in w_2 but considerably slows down changes in w_1. For faster convergence, the oscillation along the w_2 dimension should be avoided. For this initialization, it can be achieved with a learning rate of $\eta = 10/75$, which makes w_2 to reach the minimum in 1 update step, followed by w_1, 75 steps later.

d.

$$\boldsymbol{w}^{(1)} = (10, 10) - (1, \frac{1}{75})^T(1, 75) = (9, 9) \tag{12}$$
$$\boldsymbol{w}^{(2)} = (9, 9) - (1, \frac{1}{75})^T(1, 75) = (8, 8) \tag{13}$$
$$\boldsymbol{w}^{(3)} = (8, 8) - (1, \frac{1}{75})^T(1, 75) = (7, 7) \tag{14}$$
$$\boldsymbol{w}^{(4)} = (7, 7) - (1, \frac{1}{75})^T(1, 75) = (6, 6) \tag{15}$$

Now the learning rates are scaled with respect to the partial derivatives amplitudes. Therefore, the update steps in both dimensions are equal. $\eta_1 = 10$ and $\eta_2 = 10/75$ allows to reach the minimum after 1 update.

e.

$$\boldsymbol{w}^{(1)} = (10, 10) - 0.2(1, 75) + 0.5(0, 0) = (9.8, -5) \tag{16}$$
$$\boldsymbol{w}^{(2)} = (9.8, -5) - 0.2(1, -75) + 0.5(-0.2, -15) = (9.5, 2.5) \tag{17}$$
$$\boldsymbol{w}^{(3)} = (9.5, 2.5) - 0.2(1, 75) + 0.5(-0.3, 7.5) = (9.15, -8.75) \tag{18}$$

The updates in the first dimension become larger and larger, while the magnitude of the oscillation in the second dimension decreases.

f. With $\eta\partial E/\partial w_1 = \eta$, we find $\Delta w_1^{(t)} = -\eta \sum_{s=0}^{t} \alpha^s = -\eta\frac{1-\alpha^{t+1}}{1-\alpha}$. With $\eta\partial E/\partial w_2 = \pm 75\eta$ we find $\Delta w_2^{(t)} = -75\eta \sum_{s=0}^{t}(-\alpha)^s = -\eta\frac{1-(-\alpha)^{t+1}}{1+\alpha}$. Therefore

$$\lim_{t\to\infty} \Delta\boldsymbol{w}^{(t)} = (-\eta/(1-\alpha), -75\eta/(1+\alpha)) \tag{19}$$

g. Wisely chosen unitwise learning rates can significantly speed up learning. Momentum implements an effective unitwise learning rates. It can speed up learning and dampen oscillations. But generally it does not find the optimal unitwise learning rates.

Exercise 6. Weight space symmetries

Suppose you have found a minimum for some set of weights. Show that in a network with m layers of n neurons each, there are always at least $(n!)^m$ equivalent solutions.

Solution:

Given a solution, and assuming the same activation functions across all neurons in a layer, we can swap any two neurons simply by swapping their input and output weights. There are therefore (n permute n) or $n!$ ways to arrange which neuron has which weights in one layer.

In addition, we can choose combinations across different layers independently. For instance, in a 2–layer network with 3 neurons in each layer, we have 6 arrangements of layer 1 and 6 arrangements of layer 2, and $6 \cdot 6 = 36$ unique combinations of the two layers together. In general, this gives us $(n!)^m$ equivalent solutions.

Exercise 7. Relation of weight decay and early stopping

Suppose that we are close to a minimum at w_1^*, w_2^*. The error function in the neighborhood is given by

$$E = \frac{1}{2}\beta_1(w_1 - w_1^*)^2 + \frac{1}{2}\beta_2(w_2 - w_2^*)^2 \tag{20}$$

a. Show that gradient descent with learning rate γ starting at time zero with weights $w_1(0), w_2(0)$ leads to a new weight after n updates given by

$w_i(n) = w_i^* + (1 - \beta_i\gamma)^n\big(w_i(0) - w_i^*\big)$

b. Suppose that $\beta_2 \gg \beta_1$ (take $\beta_2 = 20\beta_1$). You perform early stopping after n_{stop} steps where $n_{\text{stop}} \approx 1/(5\gamma\beta_1)$.

Show that at n_{stop} we have $w_2 \approx w_2^*$ and $w_1 \approx w_1(0)$.

Hint: $\big(1 + \frac{x}{n}\big)^n \approx \exp(x)$ for large n.

Hence, you may conclude that with an appropriate choice of early stopping, some coordinates have converged and others have not even started convergence.

c. We now consider L2 regularization and work with a modified error function $\tilde{E} = E + \frac{\lambda}{2}\sum_j (w_j)^2$.

Show that the minimum of the error function is at

$w_i = \beta_i w_i^*/(\lambda + \beta_i)$.

d. Consider $\beta_2 \gg \lambda \gg \beta_1$.

Compare the role of λ with the number n_{stop} in early stopping.

Solution:

a.

$$-\frac{\partial E}{\partial w_i} = -\beta_i(w_i - w_i^*) \tag{21}$$

Proof by induction.

- Root: $w_i(1) = w_i(0) - \gamma\beta_i(w_i(0) - w_i^*) = w_i^* + (1 - \gamma\beta_i)(w_i(0) - w_i^*)$
- Induction: Assume $w_i(n) = w_i^* + (1 - \gamma\beta_i)^n(w_i(0) - w_i^*)$

$$\begin{aligned} w_i(n+1) &= w_i(n) - \gamma\beta_i(w_i(n) - w_i^*) \\ &= w_i^* + (1 - \gamma\beta_i)^n(w_i(0) - w_i^*) - \gamma\beta_i\big(w_i^* + (1 - \gamma\beta_i)^n(w_i(0) - w_i^*) - w_i^*\big) \\ &= w_i^* + (1 - \gamma\beta_i)(1 - \gamma\beta_i)^n(w_i(0) - w_i^*) \\ &= w_i^* + (1 - \gamma\beta_i)^{n+1}(w_i(0) - w_i^*) \end{aligned}$$

b. Using the hint with $(1 + x \cdot 5\gamma\beta_1)^{n_{\text{stop}}} = (1 - \beta_i\gamma)^{n_{\text{stop}}}$ and solving for x we find $w_1(n_{\text{stop}}) \approx w_1^* + \exp(-1/5)(w_1(0) - w_1^*) \approx w_1(0)$ and $w_2(n_{\text{stop}}) \approx w_2^* + \exp(-4)(w_2(0) - w_2^*) \approx w_2^*$. ‘

c.

$$\begin{aligned}
\tilde{E} &= \frac{1}{2}\sum_i \beta_i (w_i - w_i^*)^2 + \frac{\lambda}{2} w_i^2 \\
&= \frac{1}{2}\sum_i \left(\beta_i + \lambda\right) w_i^2 - 2\beta_i w_i w_i^* + \beta_i (w_i^*)^2 \\
&= \frac{1}{2}\sum_i \left(\beta_i + \lambda\right) \left(w_i^2 - 2\frac{\beta_i}{\beta_i + \lambda} w_i w_i^* + \frac{\beta_i}{\beta_i + \lambda} (w_i^*)^2 \right) \\
&= \frac{1}{2}\sum_i \left(\beta_i + \lambda\right) \left(w_i - \frac{\beta_i}{\beta_i + \lambda} w_i^* \right)^2 + c\,,
\end{aligned}$$

where c is a constant that does not depend on w_i. Hence, $\tilde{E}$ is minimized for $w_i = \frac{\beta_i}{\beta_i + \lambda} w_i^*$.

d. With $\beta_2 \gg \lambda \gg \beta_1$ the solution is $w_1 = \frac{\beta_1}{\beta_1 + \lambda} w_1^* \approx 0$ and $w_2 = \frac{\beta_2}{\beta_2 + \lambda} w_2^* \approx w_2^*$. If $w_1(0) \approx 0$ we get the same result as with early stopping in b.