OPTIMIZING NEURAL NETWORKS

LECTURE 6
SECTION 2
JUNE 11TH

INSTITUTE FOR APPLIED
COMPUTATIONAL SCIENCE
AT HARVARD UNIVERSITY

UNIVERSITY of
RWANDA

REVIEW OF GRADIENT DESCENT

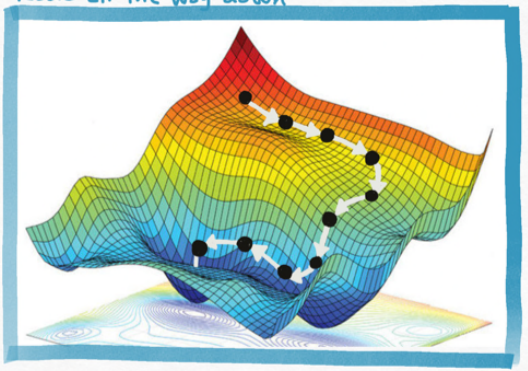
REVIEW OF GRADIENT DESCENT:

When we can't analytically solve for the stationary points of the gradient, we can still exploit the information in the gradient.

The gradient $\nabla_w \ell(w^*)$ at a point w^* is the direction of the steepest increase
The negative gradient $-\nabla_w \ell(w^*)$ is the direction of the steepest decrease.

By following the negative gradient we can eventually find a valley (lowest point).

Following the negative gradient step by step leads all the way down



Gradient Descent Algorithm:

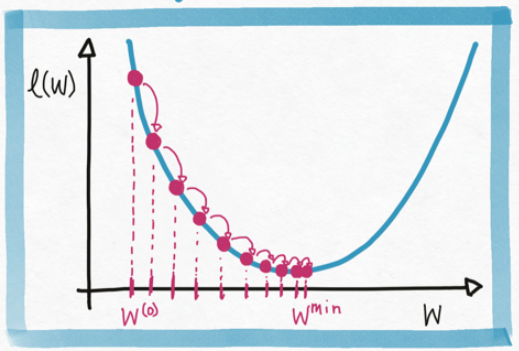
0. start at any point $w^{(0)}$
1. Repeat until stopping condition is met:
   A. Compute the gradient at the current point $w^{(n)}$:
   $$\nabla_w \ell(w^{(n)})$$
   B. Take a step in the negative gradient direction:
   $$w^{(n+1)} = w^{(n)} - \eta \cdot \nabla_w \ell(w^{(n)})$$

DIAGNOSING ISSUES WITH GRADIENT DESCENT

DIAGNOSING GRADIENT DESCENT:

Our choice of the learning rate has a significant impact on the performance of gradient descent.

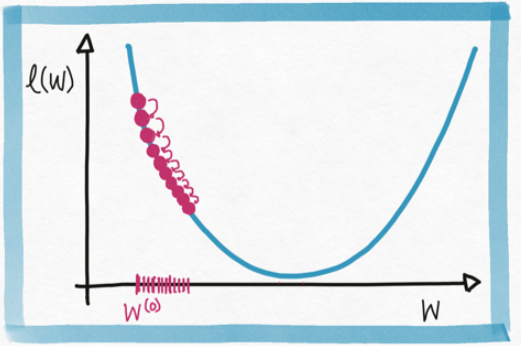
Learning Rate Appropriate



When η is appropriate, the algorithm will eventually find the bottom of a valley, where the gradient is zero.

We say the algorithm converges

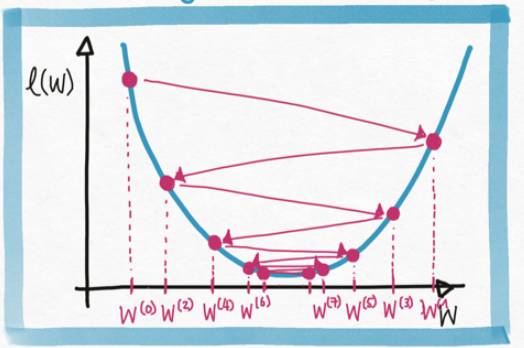
Learning Rate Too Small



When η is too small, the algorithm makes very little progress.

The convergence is too slow.

Learning Rate Too Large



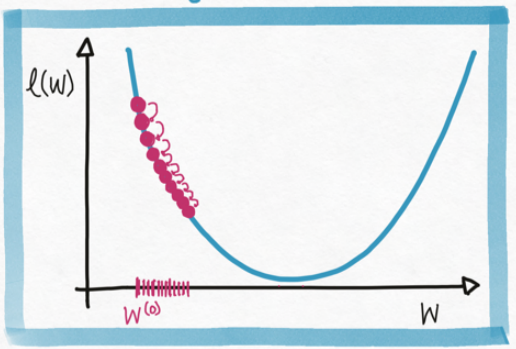
When η is too large, the algorithm may overshoot the optimum value and oscillate.

We may fail to converge.

DIAGNOSING SLOW CONVERGENCE:

How can we tell when gradient descent is converging? A very useful tool is the visualization of the loss function $\ell(w)$ at each step of gradient descent, this is called the trace plot.

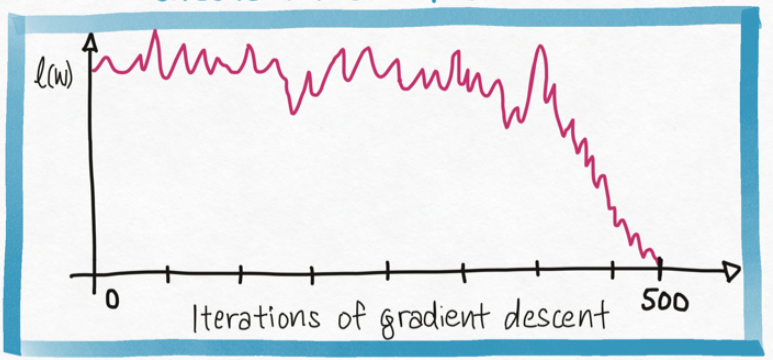
Learning Rate Too Small



When η is too small, the algorithm makes very little progress.

The convergence is too slow.

Gradient Descent Trace Plot

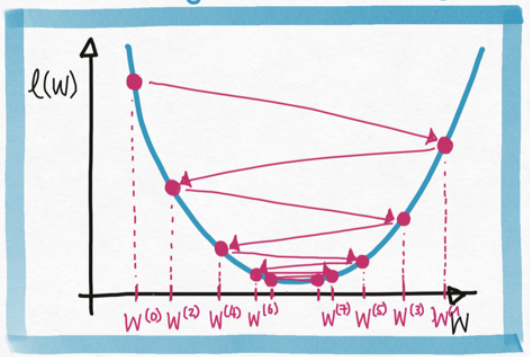


While the loss function is decreasing throughout training, it doesn't look like the descent has stopped, that is, we haven't hit bottom.

DIAGNOSING NON-CONVERGENCE:

How can we tell when gradient descent is converging? A very useful tool is the visualization of the loss function $\ell(w)$ at each step of gradient descent, this is called the trace plot.

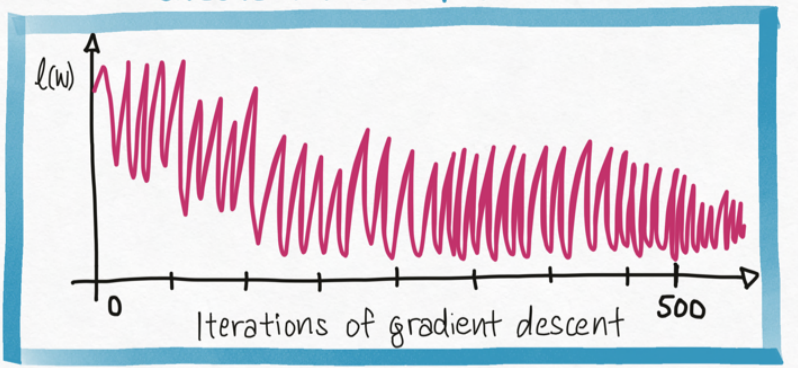
Learning Rate Too Large



When η is too large, the algorithm may overshoot the optimum value and oscillate.

We may fail to converge.

Gradient Descent Trace Plot

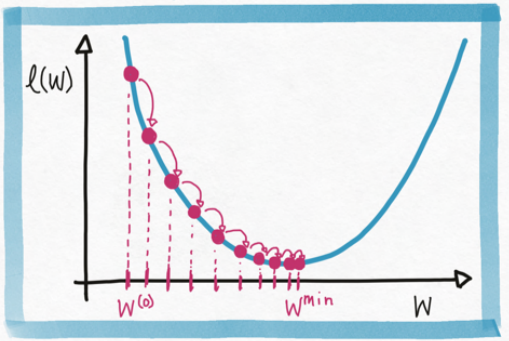


While the loss function decreases some during training, it is mostly oscillating between values rather than converging to zero.

SIGNS OF CONVERGENCE:

How can we tell when gradient descent is converging? A very useful tool is the visualization of the loss function $\ell(w)$ at each step of gradient descent, this is called the trace plot.

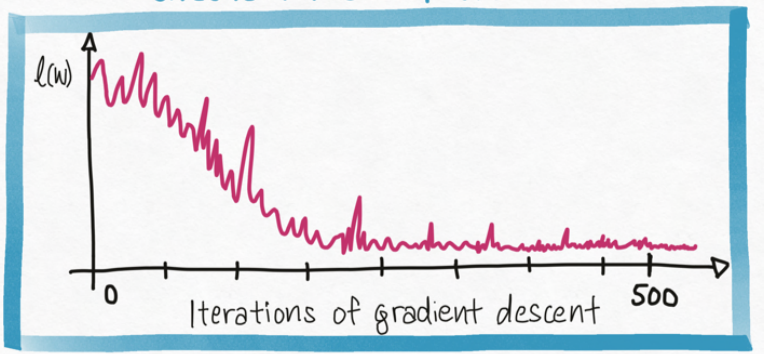
Learning Rate Appropriate



When η is appropriate, the algorithm will eventually find the bottom of a valley, where the gradient is zero.

We say the algorithm converges

Gradient Descent Trace Plot



The loss function has decreased significantly during training. Towards the end of training the loss stabilizes and doesn't look like it can decrease further.

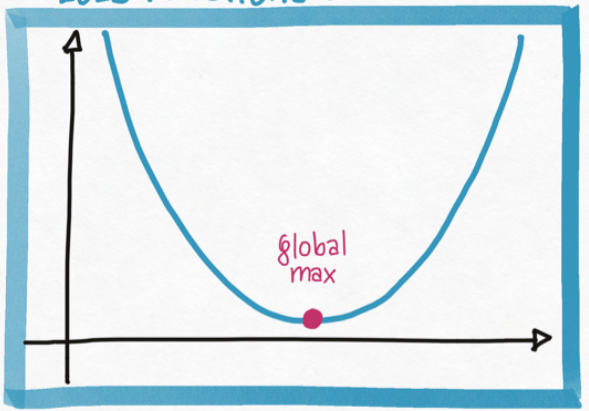
LOCAL VERSUS GLOBAL OPTIMA

LOCAL VS GLOBAL OPTIMA:

If we choose η correctly, then gradient descent will converge to a stationary point. But will this point be a global minima?

If the function we are optimizing is convex then the stationary point will be a global minimum.

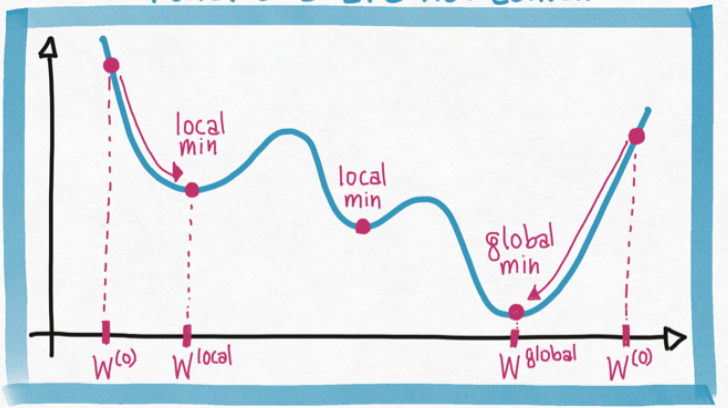
Linear & Polynomial Regression Loss Functions are Convex



Hessian (2nd Derivative) positive semi-definite everywhere.

Every stationary point of the gradient is a global min.

Neural Network Regression Loss Functions are not Convex



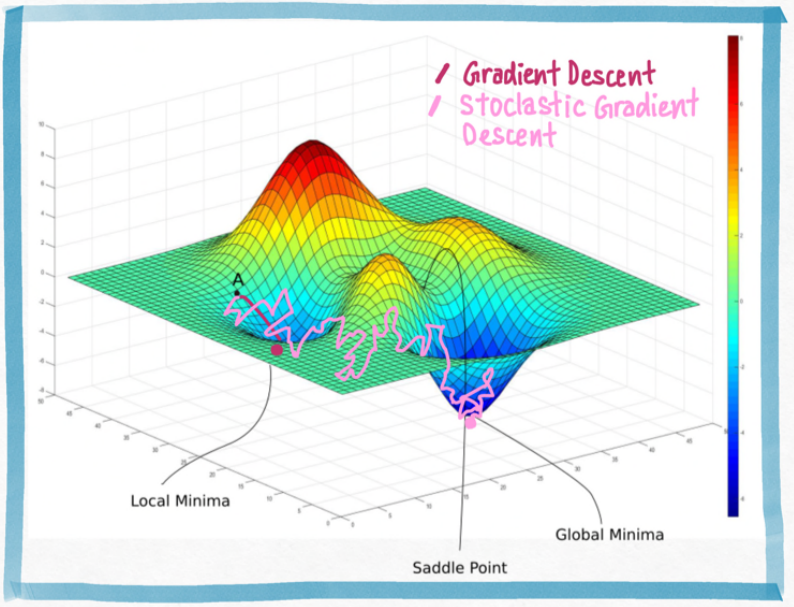
Neural networks with different weights can correspond to the same function.

Most stationary points are local minima but not global optima.

ESCAPING LOCAL MINIMA:

At local minima the gradient is zero and so our update no longer moves us in the parameter space - i.e. we are stuck!

One solution is to add some random noise to the gradient, so that the gradient will unlikely be zero at local optima, and we get a chance to move out of it.



Stochastic Gradient Descent:

instead of computing the gradient of the loss function (e.g. MSE, negative log-likelihood) on the entire dataset, we compute on a small amount of random samples of the training data.

This batch gradient will vary randomly depending on the samples in the batch.