CS5242 : Neural Networks and Deep Learning

Lecture 4: Vanilla Neural Networks
Loss and Optimization

Semester 1 2021/22

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science
National University of Singapore (NUS)

Outline

- Quality of neural network
- Maximizing the quality by gradient ascent
- Learning rate strategy
- Neural network loss
- Minimizing the loss by gradient descent
- Mini-batch/stochastic gradient descent
- Summary

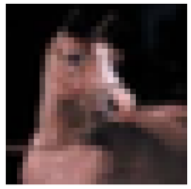
Outline

- Quality of neural network
- Maximizing the quality by gradient ascent
- Learning rate strategy
- Neural network loss
- Minimizing the loss by gradient descent
- Mini-batch/stochastic gradient descent
- Summary

Neural network quality

- Quality of neural networks :
  - Evaluate the confidence of the net to predict the correct class over the whole training set.

- First, assume for simplicity that the training set consists of
  - 4 classes : Horse, Cat, Dog, Deer
  - 5 labelled pictures



HORSE

CAT

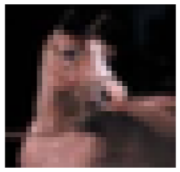
DOG

DEER

DOG

Neural network quality

Training set


HORSE

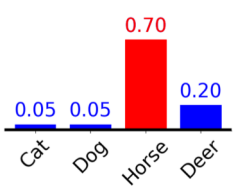

CAT

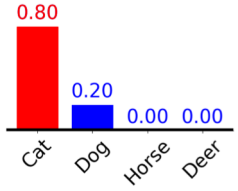

DOG

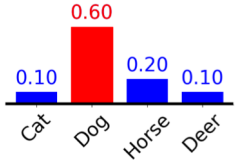

DEER

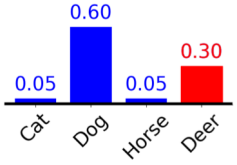

DOG

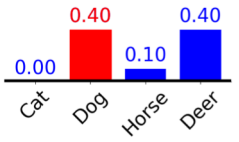
Network prediction











Quality of the network

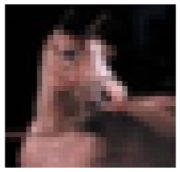
$$Q_{\text{mean}} = (0.70 \times 0.80 \times 0.60 \times 0.30 \times 0.40)^{1/5} = 0.53$$

Geometric mean

In average, the network gives a 53% confidence score to the correct class

Good NN quality

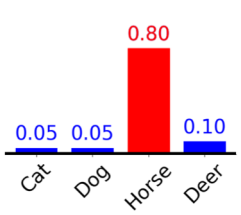
Training set

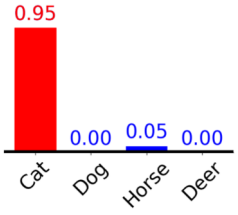


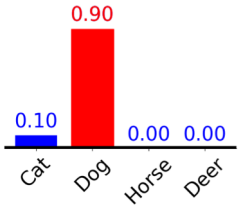


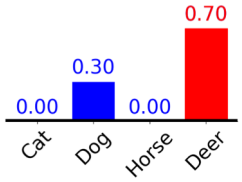
HORSE | CAT | DOG | DEER | DOG

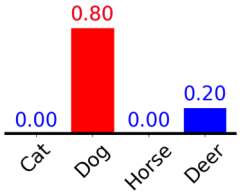
Network prediction











Quality of the network

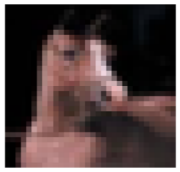
$$Q_{\text{mean}} = (0.80 \times 0.95 \times 0.90 \times 0.70 \times 0.80)^{1/5} = 0.83$$

In average, the network gives a 83% confidence score to the correct class

Optimal classification: $Q_{\text{mean}} \approx 1.00$

Bad NN quality

Training set



HORSE

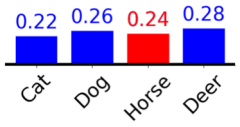
CAT

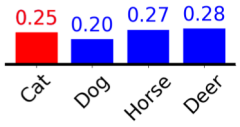
DOG

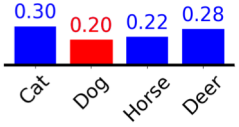
DEER

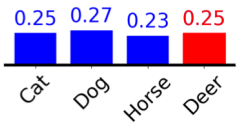
DOG

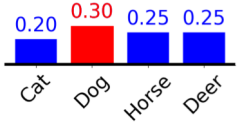
Network prediction











Quality of the network

$$Q_{\text{mean}} = (0.24 \times 0.25 \times 0.20 \times 0.25 \times 0.30)^{1/5} = 0.24$$

In average, the network gives a 24% confidence score to the correct class

Random classification: $Q_{\text{mean}} \approx 0.25 \approx 1/\text{K}$

Neural network quality

The i^{th} picture of the training set

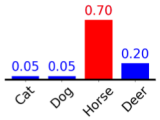
$\mathbf{x}^{(i)}$

The class index of the i^{th} picture

$\text{cl}(i)$

$\mathbf{p}^{(i)}$

Network output for the i^{th} picture

$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
				
3	1	2	4	2
 $= \begin{bmatrix} 0.05 \\ 0.05 \\ 0.70 \\ 0.20 \end{bmatrix}$	$\begin{bmatrix} 0.80 \\ 0.20 \\ 0.00 \\ 0.00 \end{bmatrix}$	$\begin{bmatrix} 0.05 \\ 0.60 \\ 0.20 \\ 0.10 \end{bmatrix}$	$\begin{bmatrix} 0.05 \\ 0.60 \\ 0.05 \\ 0.30 \end{bmatrix}$	$\begin{bmatrix} 0.00 \\ 0.40 \\ 0.10 \\ 0.40 \end{bmatrix}$

$$
\begin{aligned}
Q_{\text{mean}} &= (0.70 \times 0.80 \times 0.60 \times 0.30 \times 0.40)^{1/5} \\
&= \Big((\text{entry cl}(1) \text{ of } \mathbf{p}^{(1)}) \times (\text{entry cl}(2) \text{ of } \mathbf{p}^{(2)}) \times \cdots \times (\text{entry cl}(5) \text{ of } \mathbf{p}^{(5)}) \Big)^{1/5} \\
&= \left(\prod_{i=1}^{5} \text{entry cl}(i) \text{ of vector } \mathbf{p}^{(i)} \right)^{1/5}
\end{aligned}
$$

Neural network quality

- With a training set containing N labeled pictures :

The class index of the i^{th} picture (integer)

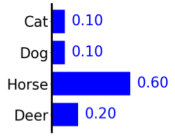
$$Q_{\text{mean}} = \left(\prod_{i=1}^{N} \text{entry } \text{cl}(i) \text{ of vector } \mathbf{p}^{(i)} \right)^{1/N}$$

How confident the network is that the i^{th} picture belongs to the class $\text{cl}(i)$

Network output for the i^{th} picture $\mathbf{x}^{(i)}$:

$$\mathbf{p}^{(i)} = \text{Softmax}(W\mathbf{x}^{(i)} + \mathbf{b})$$

Vector with 4 entries

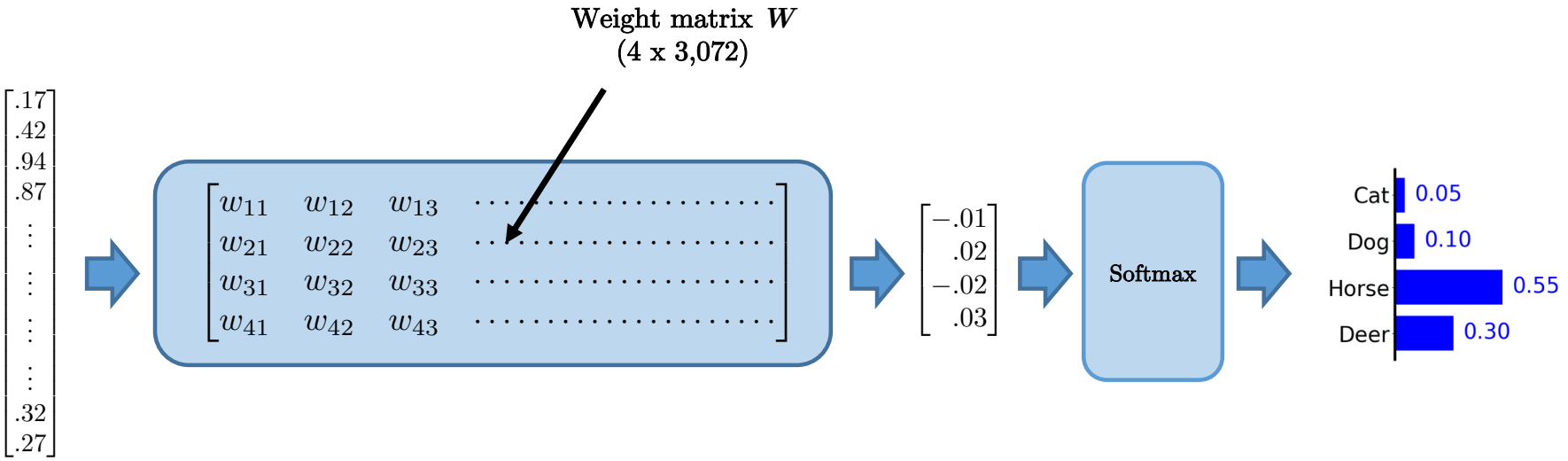


Outline

- Quality of neural network
- **Maximizing the quality by gradient ascent**
- Learning rate strategy
- Neural network loss
- Minimizing the loss by gradient descent
- Mini-batch/stochastic gradient descent
- Summary

Maximizing the quality of neural networks

- Learning = Maximizing the quality Q_{mean} of the network over its parameters W.
- Maximizing Q_{mean} will be done by gradient ascent.
- Let us consider a vanilla one-layer NN :



Quality function

- The weight matrix W contains the 4 templates of size 3,072 :

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots \\ w_{21} & w_{22} & w_{23} & \cdots \\ w_{31} & w_{32} & w_{33} & \cdots \\ w_{41} & w_{42} & w_{43} & \cdots \end{bmatrix}$$

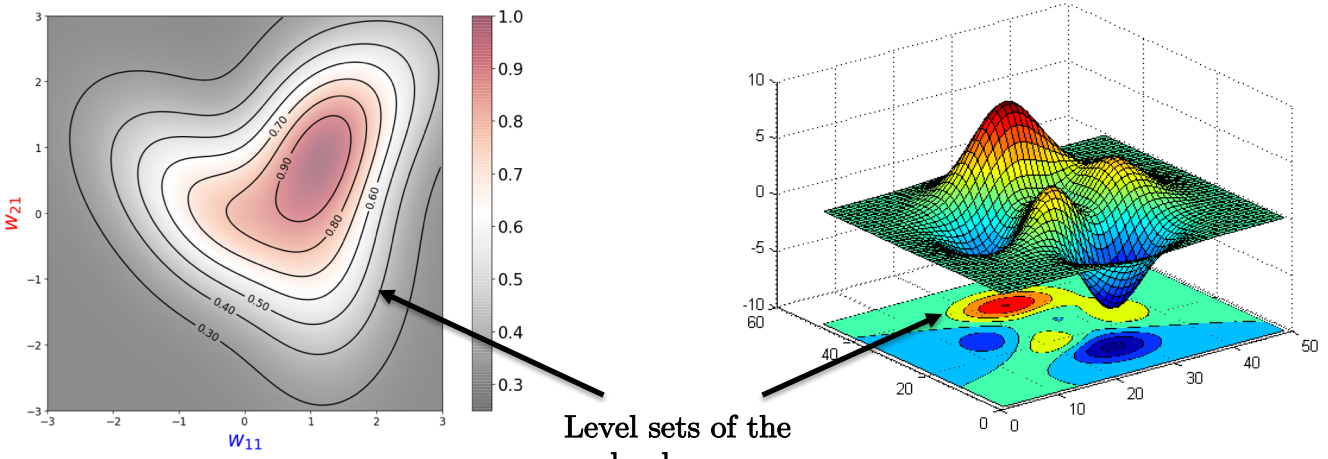
- The Quality Q_{mean} is a function of $\underbrace{12{,}288}_{4 \text{ x } 3{,}072}$ parameters/weights :

$$\underbrace{Q_{\text{mean}}(w_{11}, w_{12}, w_{13}, \ldots \; ; \; w_{21}, w_{22}, w_{23}, \ldots \; ; \; w_{31}, w_{32}, w_{33}, \ldots \; ; \; w_{41}, w_{42}, w_{43}, \ldots)}_{Q_{\text{mean}}(W)}$$

Landscape of quality function

- For each configuration W of the network, there exists a value $Q_{mean}(W)$.
  - All the possible configurations W provide the landscape of Q_{mean}.
- As Q_{mean} has 12k variables, it is not possible to visualize this landscape.
- Let's assume that Q_{mean} has only 2 parameters for visualization purpose :

$Q_{\text{mean}}(w_{11}, w_{21})$



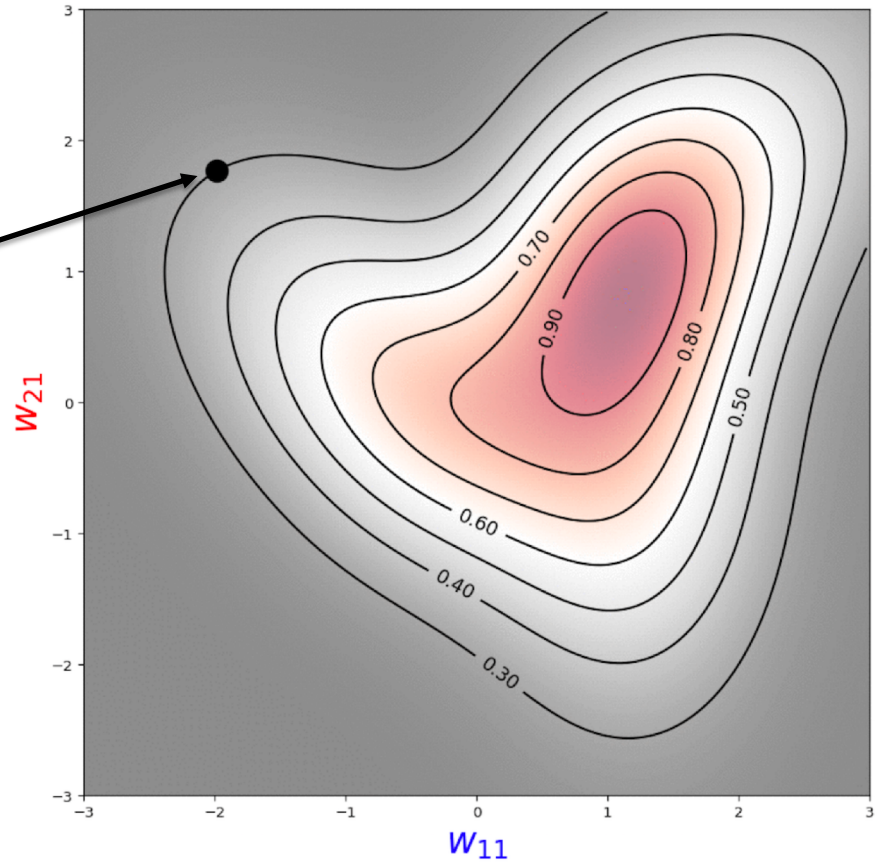
Level sets of the landscape

Maximizing the quality

- Gradient ascent algorithm:
  - Start randomly.
  - Move in the direction of the steepest ascent.
  - Stop when reaching the maximum.

Initialization

Start randomly



Network Quality:

Q_{mean} = 0.25

Network Parameters:

w_{11} = -2.20
w_{12} = 2.20

Update scheme

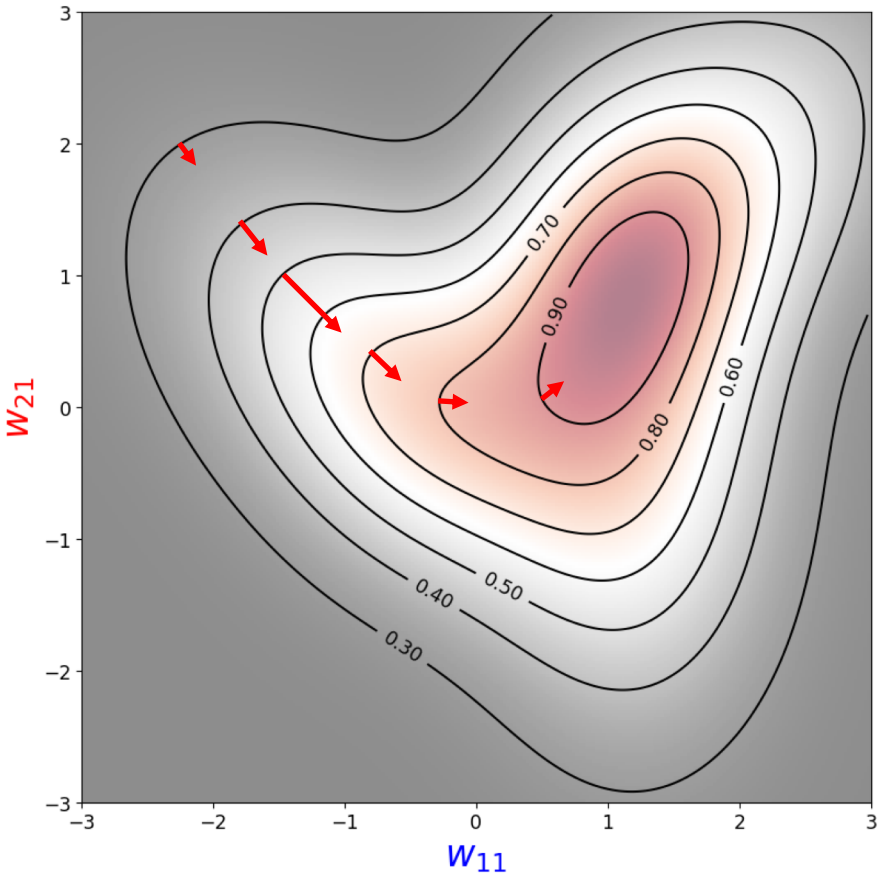
- Gradient ascent:

  Follow the gradient!

- The parameters are updated according to the equation:

$$\begin{bmatrix} w_{11} \\ w_{21} \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{21} \end{bmatrix} + \mathrm{lr} \begin{bmatrix} \frac{\partial Q}{\partial w_{11}} \\ \frac{\partial Q}{\partial w_{21}} \end{bmatrix}$$

- So the displacements are proportional to the gradient.
- The gradient is:
  - Perpendicular to the level sets.
  - Its magnitude is proportional to the slope of the landscape.

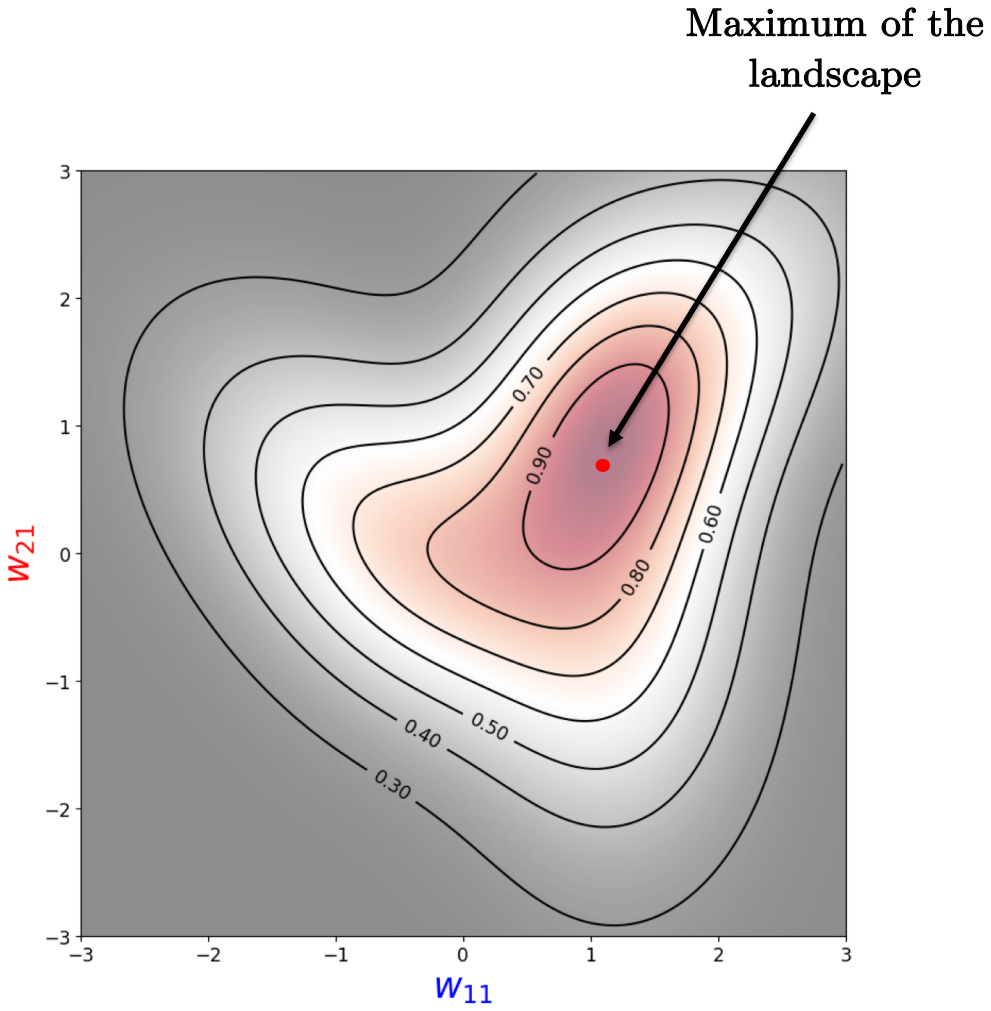


Maximum

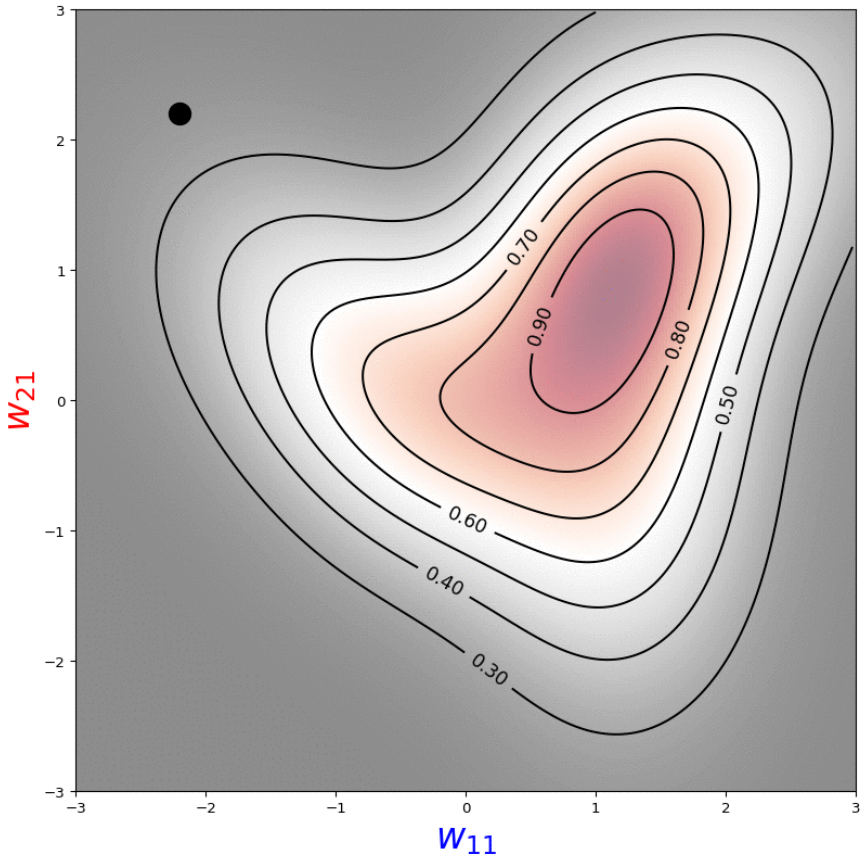
- At the maximum:
  - We reach the top of the mountain, which is perfectly flat.
  - We stop moving since the magnitude of the gradient is zero.

At the maximum location: $w_{11} = 1.1$, $w_{21} = 0.8$

We have: $\begin{bmatrix} \frac{\partial Q}{\partial w_{11}} \\ \frac{\partial Q}{\partial w_{21}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$



Gradient ascent in action



Network Quality:

$Q_{mean} = 0.25$

Network Parameters:

$w_{11} = -2.20$
$w_{12} = 2.20$

High-dimensional quality function

- The previous Q_{mean} function has 2 variables.
  - The gradient vector ∇Q_{mean} has thus 2 elements.

$$Q_{\text{mean}}(w_{11}, w_{21}) \qquad \nabla Q_{\text{mean}} = \begin{bmatrix} \frac{\partial Q}{\partial w_{11}} \\ \\ \frac{\partial Q}{\partial w_{21}} \end{bmatrix}$$

- But the original quality Q_{mean} is a function of 12,288 parameters/weights.
  - The gradient vector ∇Q_{mean} has consequently 12,288 elements :

$$Q_{\text{mean}}(w_{11}, w_{12}, w_{13}, \ldots \; ; \; w_{21}, w_{22}, w_{23}, \ldots \; ; \; w_{31}, w_{32}, w_{33}, \ldots \; ; \; w_{41}, w_{42}, w_{43}, \ldots)$$

$$\nabla Q_{\text{mean}} = \begin{bmatrix} \partial Q/\partial w_{11} \\ \partial Q/\partial w_{12} \\ \partial Q/\partial w_{13} \\ \vdots \\ \partial Q/\partial w_{21} \\ \partial Q/\partial w_{22} \\ \partial Q/\partial w_{23} \\ \vdots \\ \partial Q/\partial w_{31} \\ \partial Q/\partial w_{32} \\ \partial Q/\partial w_{33} \\ \vdots \\ \partial Q/\partial w_{41} \\ \partial Q/\partial w_{42} \\ \partial Q/\partial w_{43} \\ \vdots \end{bmatrix}$$

High-dimensional update formula

$$w_{ij} = w_{ij} + \text{lr}\,\frac{\partial Q}{\partial w_{ij}} \quad \Rightarrow \quad \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ \vdots \\ w_{21} \\ w_{22} \\ w_{23} \\ \vdots \\ w_{31} \\ w_{32} \\ w_{33} \\ \vdots \\ w_{41} \\ w_{42} \\ w_{43} \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ \vdots \\ w_{21} \\ w_{22} \\ w_{23} \\ \vdots \\ w_{31} \\ w_{32} \\ w_{33} \\ \vdots \\ w_{41} \\ w_{42} \\ w_{43} \\ \vdots \end{bmatrix} + \text{lr} \begin{bmatrix} \partial Q/\partial w_{11} \\ \partial Q/\partial w_{12} \\ \partial Q/\partial w_{13} \\ \vdots \\ \partial Q/\partial w_{21} \\ \partial Q/\partial w_{22} \\ \partial Q/\partial w_{23} \\ \vdots \\ \partial Q/\partial w_{31} \\ \partial Q/\partial w_{32} \\ \partial Q/\partial w_{33} \\ \vdots \\ \partial Q/\partial w_{41} \\ \partial Q/\partial w_{42} \\ \partial Q/\partial w_{43} \\ \vdots \end{bmatrix}$$

Matrix representation

- Scalar form of the gradient ascent equation:

$$w_{ij} \quad = \quad w_{ij} \quad + \quad \text{lr} \quad \frac{\partial Q}{\partial w_{ij}}$$

- Matrix form of the gradient ascent equation:

$$\underbrace{\begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots \\ w_{21} & w_{22} & w_{23} & \dots \\ w_{31} & w_{32} & w_{33} & \dots \\ w_{41} & w_{42} & w_{43} & \dots \end{bmatrix}}_{W} =$$

$$\underbrace{\begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots \\ w_{21} & w_{22} & w_{23} & \dots \\ w_{31} & w_{32} & w_{33} & \dots \\ w_{41} & w_{42} & w_{43} & \dots \end{bmatrix}}_{W} + \text{lr} \underbrace{\begin{bmatrix} \partial Q/\partial w_{11} & \partial Q/\partial w_{12} & \partial Q/\partial w_{13} & \dots \\ \partial Q/\partial w_{21} & \partial Q/\partial w_{22} & \partial Q/\partial w_{23} & \dots \\ \partial Q/\partial w_{31} & \partial Q/\partial w_{32} & \partial Q/\partial w_{33} & \dots \\ \partial Q/\partial w_{41} & \partial Q/\partial w_{42} & \partial Q/\partial w_{43} & \dots \end{bmatrix}}_{\text{Gradient reshaped as a matrix } \frac{\partial Q}{\partial W}}$$

Outline

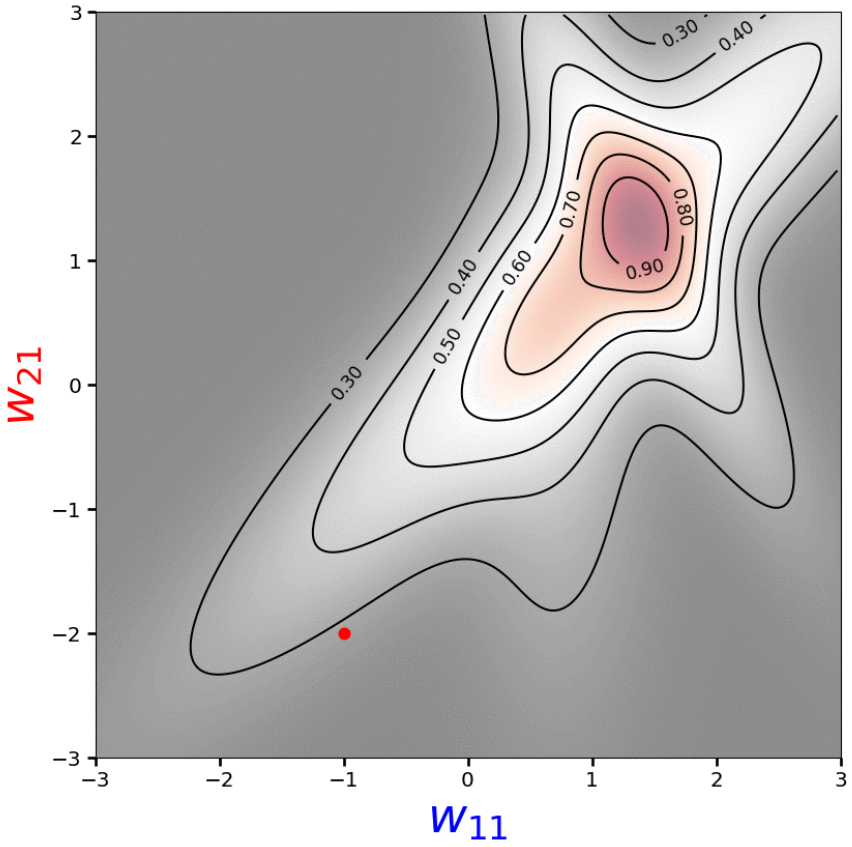
- Quality of neural network
- Maximizing the quality by gradient ascent
- **Learning rate strategy**
- Neural network loss
- Minimizing the loss by gradient descent
- Mini-batch/stochastic gradient descent
- Summary

Learning rate

- It is important to carefully choose the learning rate:
  - If the learning rate is too big, the algorithm will diverge.
  - If the learning rate is too small, it will take too long to reach the maximum.

Learning rate too large

- $\mathrm{lr} = 5.0$



Iteration 1

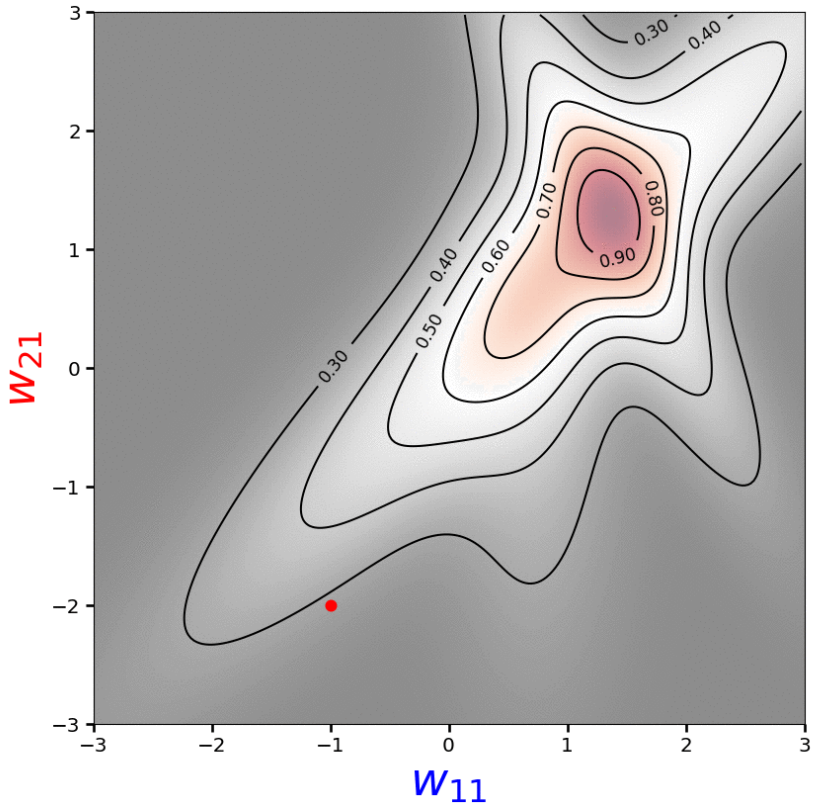
Learning rate:
lr = 5.0

Network Quality:
$Q_{mean} = 0.28$

Network Parameters:
$w_{11} = -1.00$
$w_{12} = -2.00$

Learning rate too small

- lr=0.625
  - 8 times smaller than lr=5.0



Iteration 1

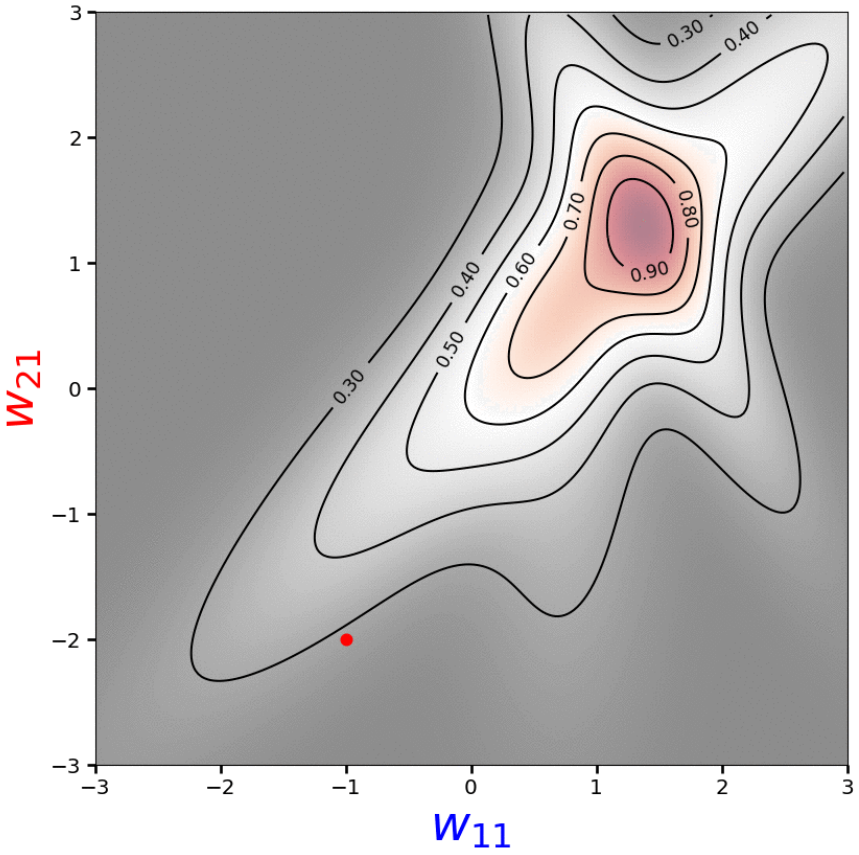
Learning rate:
lr = 0.625

Network Quality:
Q_{mean} = 0.28

Network Parameters:
w_{11} = -1.00
w_{12} = -2.00

Learning rate schedule

- Best way to go !
  - Make a learning rate schedule :
    - Start with lr=2.5.
    - Divide by 2 every 10 iterations.



Iteration 1

Learning rate:
lr = 2.500

Network Quality:
Q_{mean} = 0.28

Network Parameters:
w_{11} = -1.00
w_{12} = -2.00

Outline

- Quality of neural network
- Maximizing the quality by gradient ascent
- Learning rate strategy
- **Neural network loss**
- Minimizing the loss by gradient descent
- Mini-batch/stochastic gradient descent
- Summary

Optimizing product is slow

- Quality function is a product of functions:

$$Q_{\text{mean}} = \left(\prod_{i=1}^{N} \text{entry cl}(i) \text{ of vector } \mathbf{p}^{(i)} \right)^{1/N}$$

- Gradient of a product is computationally expensive (e.g. derivate of a product of 50,000 terms)

$$w_{ij} \quad = \quad w_{ij} \quad + \quad \text{lr} \quad \frac{\partial Q}{\partial w_{ij}}$$

- It is much cheaper to compute the gradient of a sum.
  - How? The log function over a product provides a sum.

New loss

- News loss = - log of neural network quality

$$\mathcal{L} = \text{ - } \log Q_{\text{mean}} = \text{ - } \log \left(\prod_{i=1}^{N} \text{entry cl}(i) \text{ of vector } \mathbf{p}^{(i)} \right)^{1/N}$$

$$= \text{ - } \frac{1}{N} \log \left(\prod_{i=1}^{N} \text{entry cl}(i) \text{ of vector } \mathbf{p}^{(i)} \right)$$

$$= \text{ - } \frac{1}{N} \sum_{i=1}^{N} \log \left(\text{entry cl}(i) \text{ of vector } \mathbf{p}^{(i)} \right)$$

This function is much easier to differentiate.

Cross-entropy loss

- The new loss is called cross-entropy loss.
- Cross-entropy is a standard function in information theory.
- Cross-entropy is a function of W :

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} -\log \Big(\text{entry cl}(i) \text{ of vector } \mathbf{p}^{(i)} \Big)$$

$$\log \Big(\text{entry 2 of vector } \begin{bmatrix} p_1^{(i)} \\ p_2^{(i)} \\ p_3^{(i)} \\ p_4^{(i)} \end{bmatrix} \Big) = \log \Big(p_2^{(i)} \Big)$$

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} -\text{entry cl}(i) \text{ of vector } \log \Big(\mathbf{p}^{(i)} \Big)$$

$$\mathcal{L}(w_{11}, w_{12}, \ldots) = \frac{1}{N} \sum_{i=1}^{N} -\text{entry cl}(i) \text{ of vector } \text{LogSoftmax}(W \mathbf{x}^{(i)})$$

Function of
12,288 parameters.

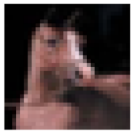
Cross-entropy loss

$$\mathcal{L}(w_{11}, w_{12}, \ldots) = \overbrace{\frac{1}{N} \sum_{i=1}^{N} \underbrace{-\text{entry cl}(i) \text{ of vector } \log\left(\mathbf{p}^{(i)}\right)}_{\ell^{(i)}(w_{11}, w_{12}, \ldots)}}^{\text{Average loss over all data points}}$$

Loss of the i^{th} data point =

- log of the probability given to the correct class

Cross-entropy loss

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$\mathbf{x}^{(i)}$					
$\text{cl}^{(i)}$	3	1	2	4	2
$\mathbf{p}^{(i)}$	$\begin{bmatrix}0.05\\0.05\\0.70\\0.20\end{bmatrix}$	$\begin{bmatrix}0.80\\0.20\\0.00\\0.00\end{bmatrix}$	$\begin{bmatrix}0.05\\0.60\\0.20\\0.10\end{bmatrix}$	$\begin{bmatrix}0.05\\0.60\\0.05\\0.30\end{bmatrix}$	$\begin{bmatrix}0.00\\0.40\\0.10\\0.40\end{bmatrix}$

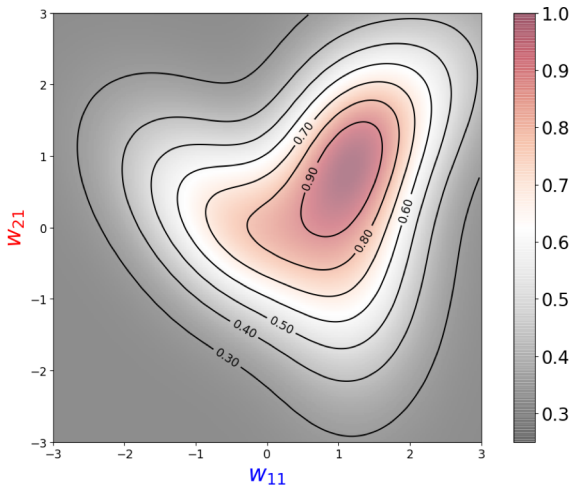
$$\text{Average Loss} = \frac{1}{5}\sum_{1}^{5} -\text{entry cl}(i) \text{ of vector } \log\left(\mathbf{p}^{(i)}\right)$$

$$= \frac{1}{5}\Big(-\log(0.70) - \log(0.80) - \log(0.60) - \log(0.30) - \log(0.40)\Big)$$

$$= \frac{1}{5}\Big(\ 0.37 \ + \ 0.22 \ + \ 0.51 \ + \ 1.20 \ + \ 0.92 \ \Big) \ = \ 0.68$$

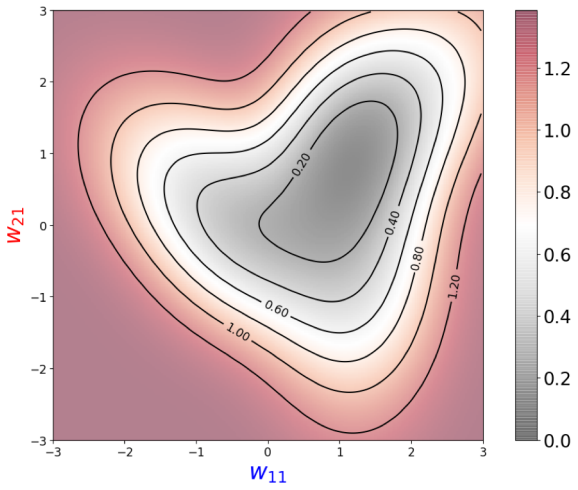
Range of values

Q_{mean}



$0.25 \leq Q_{\text{mean}} \leq 1$

$\mathcal{L} = -\log(Q_{\text{mean}})$



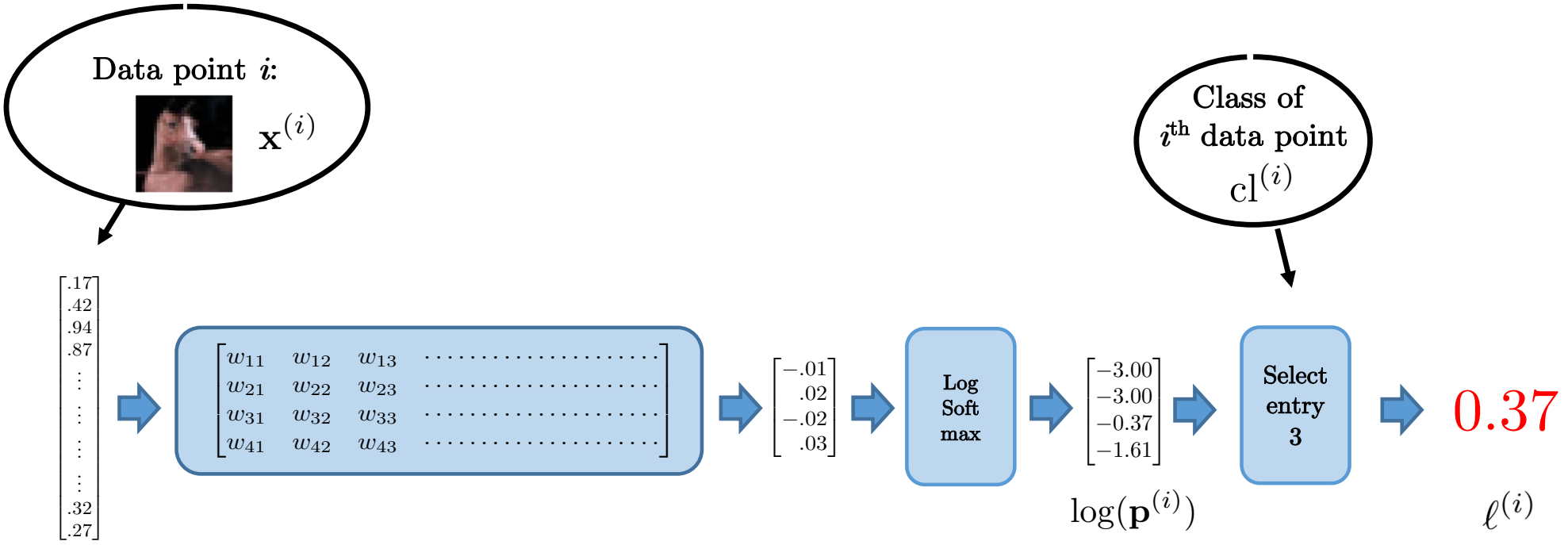
$0 \leq \mathcal{L} \leq 1.4$

- log(1)

- log(0.25)

We will want L to be as close to 0 as possible.

Representing the network and the loss



Outline

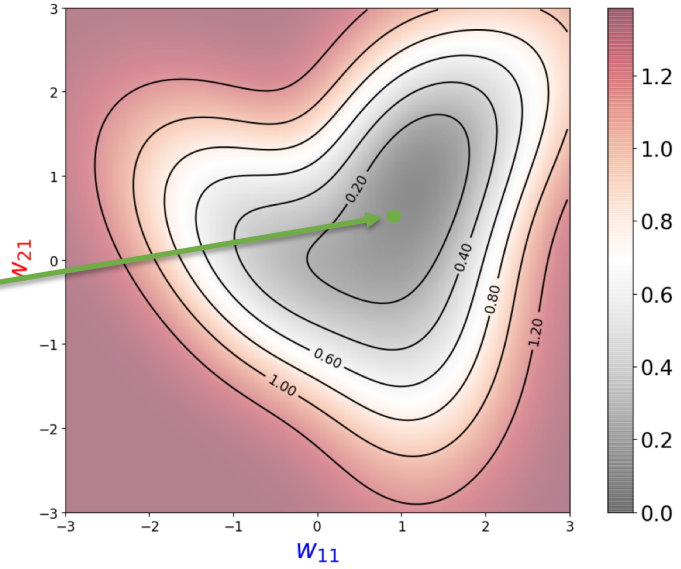
- Quality of neural network
- Maximizing the quality by gradient ascent
- Learning rate strategy
- Neural network loss
- **Minimizing the loss by gradient descent**
- Mini-batch/stochastic gradient descent
- Summary

Loss

- The total loss over the whole training set is:

$$\mathcal{L}(w_{11}, w_{12}, \ldots) = \frac{1}{N} \sum_{i=1}^{N} \ell^{(i)}(w_{11}, w_{12}, \ldots)$$

We want to tune the internal parameters of the network in order to minimize the total loss (ideally make it equal to 0)



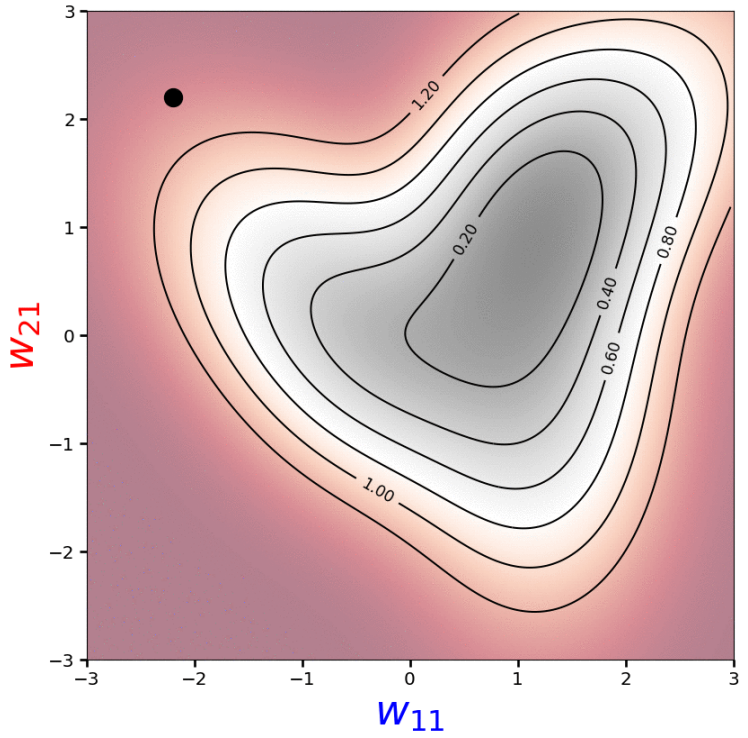
Loss minimization

- The total loss is minimized by gradient descent:

$$w_{rs} = w_{rs} - \mathrm{lr}\,\frac{\partial \mathcal{L}}{\partial w_{rs}}$$

Minimization

$$\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} - \mathrm{lr} \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{11}} \\ \frac{\partial \mathcal{L}}{\partial w_{12}} \end{bmatrix}$$



$\mathcal{L}$ = 1.39

Network Parameters:
w_{11} = -2.20
w_{12} = 2.20

Matrix representation

- Matrix form of the gradient descent equation:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots \\ w_{21} & w_{22} & w_{23} & \dots \\ w_{31} & w_{32} & w_{33} & \dots \\ w_{41} & w_{42} & w_{43} & \dots \end{bmatrix} =$$

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots \\ w_{21} & w_{22} & w_{23} & \dots \\ w_{31} & w_{32} & w_{33} & \dots \\ w_{41} & w_{42} & w_{43} & \dots \end{bmatrix} - \text{lr} \begin{bmatrix} \partial\mathcal{L}/\partial w_{11} & \partial\mathcal{L}/\partial w_{12} & \partial\mathcal{L}/\partial w_{13} & \dots \\ \partial\mathcal{L}/\partial w_{21} & \partial\mathcal{L}/\partial w_{22} & \partial\mathcal{L}/\partial w_{23} & \dots \\ \partial\mathcal{L}/\partial w_{31} & \partial\mathcal{L}/\partial w_{32} & \partial\mathcal{L}/\partial w_{33} & \dots \\ \partial\mathcal{L}/\partial w_{41} & \partial\mathcal{L}/\partial w_{42} & \partial\mathcal{L}/\partial w_{43} & \dots \end{bmatrix}$$

$$W = W - \text{lr} \frac{\partial \mathcal{L}}{\partial W}$$

Outline

- Quality of neural network
- Maximizing the quality by gradient ascent
- Learning rate strategy
- Neural network loss
- Minimizing the loss by gradient descent
- **Mini-batch/stochastic gradient descent**
- Summary

Gradient descent

- Loss:

$$\mathcal{L}(w_{11}, w_{12}, \ldots) = \frac{1}{N} \sum_{i=1}^{N} \ell^{(i)}(w_{11}, w_{12}, \ldots)$$

(Average) loss

Loss of data point $x^{(i)}$

- (Full batch) gradient descent:

$$W = W - \text{lr} \, \frac{\partial \mathcal{L}}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \underbrace{\frac{1}{N} \sum_{i=1}^{N} \frac{\partial \ell^{(i)}}{\partial W}}$$

The gradient matrix $\frac{\partial \mathcal{L}}{\partial W}$ is the average of 50,000 gradient matrices $\frac{\partial \ell^{(i)}}{\partial W}$!

Mini-batch gradient descent

- We can be more efficient by computing the average of only m gradient matrices.
- Mini-batch gradient descent:

$$W = W - \text{lr} \frac{\partial \mathcal{L}}{\partial W}$$

$$\frac{\partial \mathcal{L}}{\partial W} \approx \underbrace{\frac{1}{m} \sum_{i \in B} \frac{\partial \ell^{(i)}}{\partial W}}$$

Random approximation of the gradient $\frac{\partial \mathcal{L}}{\partial W}$ using a batch B of m indices chosen at random:

$$B = \{i_1, i_2, i_3, \ldots, i_m\}$$

Mini-batch gradient descent

- Advantages of mini-batch gradient descent:
  - The gradient is quickly computed: No need to go through the whole training set (which can be huge) to update the weights.
  - Datasets can be redundant : Many data are similar, a subset of the data will give almost the same gradient than using all data points.

$$W = W - \text{lr}\ \frac{\partial \mathcal{L}}{\partial W}$$

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial W} = \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} \dfrac{\partial \ell^{(i)}}{\partial W} & \text{Mini-batch gradient} \\ \dfrac{\partial \mathcal{L}}{\partial W} \approx \dfrac{1}{m} \displaystyle\sum_{i \in B} \dfrac{\partial \ell^{(i)}}{\partial W} & \text{Full batch gradient} \end{cases}$$

Mini-batch gradient descent

- Batch size :
  - No batch (single data): batch size is 1. This is called stochastic gradient descent (SGD). It approximates the gradient by averaging over 1 data point. Best for generalization but not as fast as mini-batch GD.
  - Mini-batch: batch size is 200. This is called mini-batch GD. It approximates the gradient by averaging over 200 data point. Good for generalization and fastest GD.
  - Full batch (all training data): batch size is 60,000. This is called gradient descent (GD). It provides the exact gradient by averaging over all data points. Weak for generalization and slowest GD.

Connection to template matching

- Mini-batch GD = template update formula ! (No proof given)
  - The mini-batch gradient of the cross-entropy loss turns out to be exactly the same formula for template matching. (No coincidence)

$$\frac{\partial \mathcal{L}^B}{\partial W} = -\frac{1}{m} EX^T$$

- The template update scheme reduces to Mini-batch GD. (See next slide)
- When we did template matching update, we actually minimized the cross-entropy loss by Mini-batch GD.

Connection to template matching

$$\begin{bmatrix} .17 & .12 \\ .42 & .32 \\ .94 & .74 \\ .87 & .65 \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \\ \vdots & \vdots \\ .32 & .42 \\ .27 & .18 \end{bmatrix}$$

$$\begin{bmatrix} -.91 & .75 & .76 & \cdots & -.83 & -.93 \\ .89 & -.75 & -.67 & \cdots & +.54 & .79 \\ -.83 & -.86 & .25 & \cdots & .64 & -.88 \\ -.73 & -.78 & -.89 & \cdots & .97 & -.56 \end{bmatrix}$$

$$\begin{bmatrix} .02 & -.01 \\ -.01 & .03 \\ .01 & .01 \\ .01 & -.02 \end{bmatrix}$$

Soft max

$$\begin{bmatrix} 0.30 & 0.25 \\ 0.20 & 0.30 \\ 0.20 & 0.26 \\ 0.25 & 0.19 \end{bmatrix}$$

$$\begin{bmatrix} 0.00 & 1.00 \\ 0.00 & 0.00 \\ 1.00 & 0.00 \\ 0.00 & 0.00 \end{bmatrix}$$

$$\begin{bmatrix} -0.30 & +0.75 \\ -0.20 & -0.30 \\ +0.80 & -0.26 \\ -0.25 & -0.19 \end{bmatrix}$$

$$\frac{1}{2} \begin{bmatrix} -0.30 & +0.75 \\ -0.20 & -0.30 \\ +0.80 & -0.26 \\ -0.25 & -0.19 \end{bmatrix} \begin{bmatrix} .17 & .42 & .94 & .87 & \cdots & \cdots & \cdots & \cdots & .32 & .27 \\ .12 & .32 & .74 & .65 & \cdots & \cdots & \cdots & \cdots & .42 & .18 \end{bmatrix}$$

$$W = W - \text{lr} \frac{\partial \mathcal{L}^B}{\partial W}$$

$$\frac{\partial \mathcal{L}^B}{\partial W} = -\frac{1}{m} EX^T$$

Outline

- Quality of neural network
- Maximizing the quality by gradient ascent
- Learning rate strategy
- Neural network loss
- Minimizing the loss by gradient descent
- Mini-batch/stochastic gradient descent
- **Summary**

Summary

Cross-entropy loss : $$\mathcal{L}(w_{11}, w_{12}, \ldots) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\ell^{(i)}(w_{11}, w_{12}, \ldots)}$$

Each data point has a loss describing how well it has been classified by the network.

Gradient Descent : $$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \ell^{(i)}}{\partial W}$$

Mini-batch Gradient Descent : $$\frac{\partial \mathcal{L}^B}{\partial W} = \underbrace{\frac{1}{m} \sum_{i \in B} \frac{\partial \ell^{(i)}}{\partial W}}$$

Random approximation of the gradient using a batch B

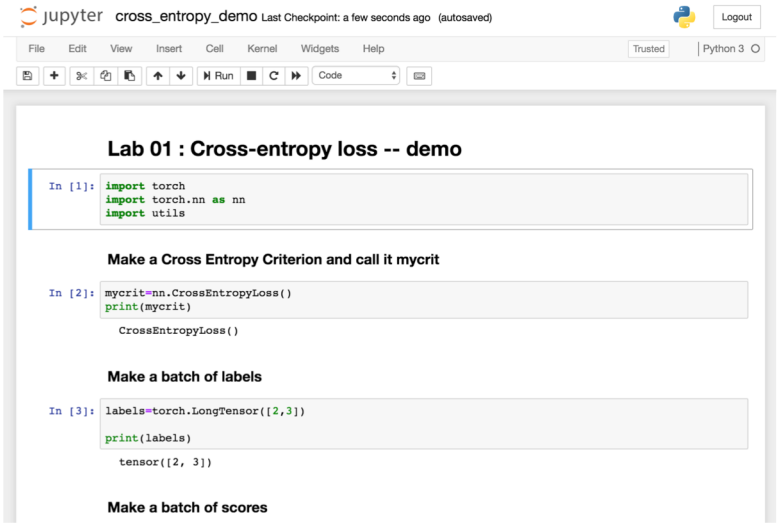
$$B = \{i_1, i_2, i_3, \ldots, i_m\}$$

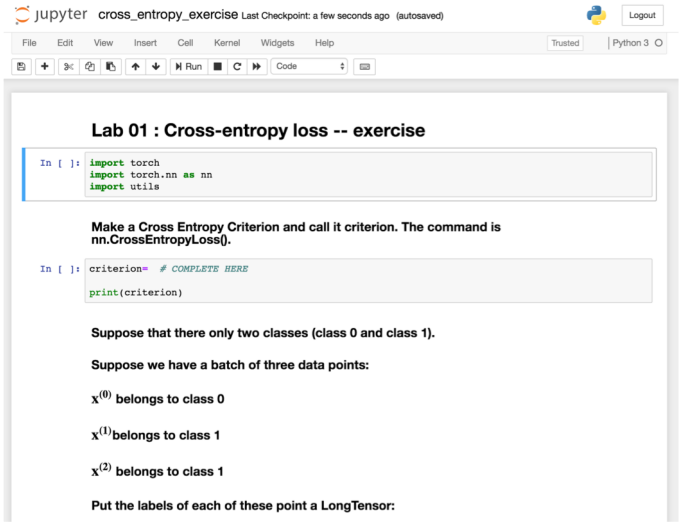
Doing Mini-batch GD is exactly the same thing than doing template matching with mini-batch.

Doing GD is exactly the same thing than doing template matching with full batch.

Lab 01

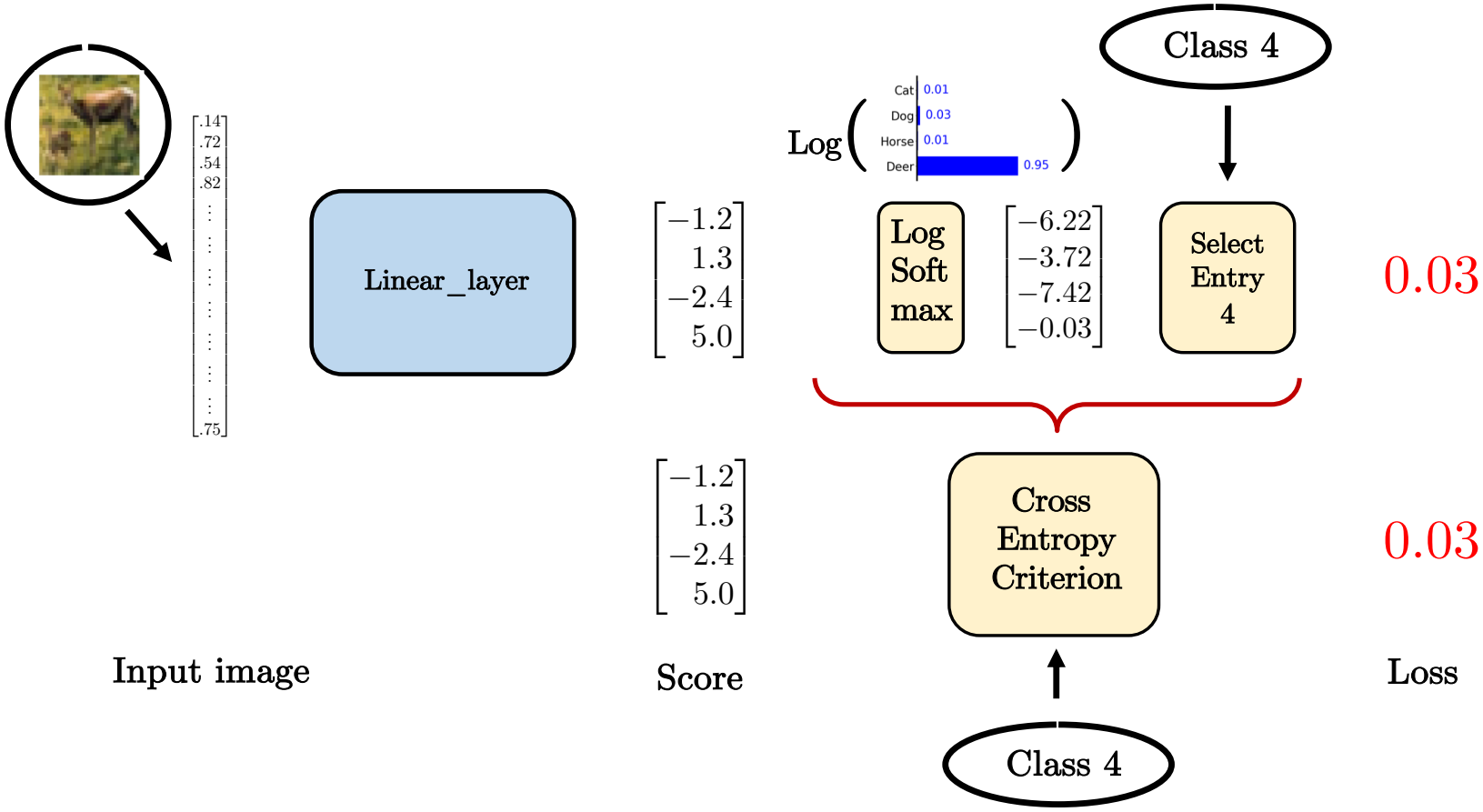
- Cross-entropy loss:





Lab 01

- **Function nn.CrossEntropyLoss() :**



Questions?