Computer vision and deep learning

Lecture 2

Today’s agenda

- Image classifiers
- Loss functions
- Evaluating a classifier

Image classification I

$\longrightarrow$ Classifier $\longrightarrow$ CAT*

- prediction

X_i - image sample to be classifier

y_i - ground truth

* assuming that we have a set of labels L = {cat, shark, ship …}

Image source: https://katzenworld.co.uk/2019/06/30/mews-funny-cat-picture-collection/

Image classification II

- Unlike other computer science problems:
  - Find the maximum value in an array
  - Sort an array
  - Find the shortest distance between two nodes in a graph
- **No obvious way** to implement an algorithm for image classification

Classification challenges

View point variations
Occlusions
Scale variation
Deformation
Lightning conditions
Background clutter
Intra-class variation

Linear classification

Linear classification

- Define a function which maps image pixels to class scores

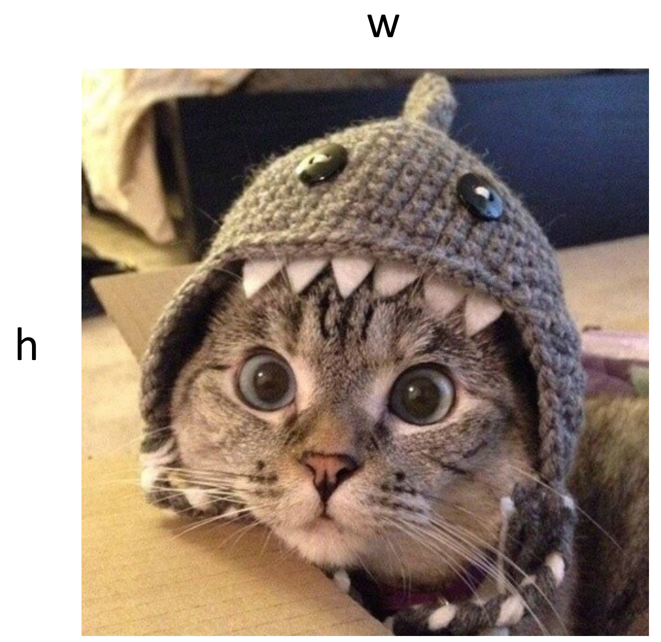
$$f(x_i, W, b) = W x_i + b$$

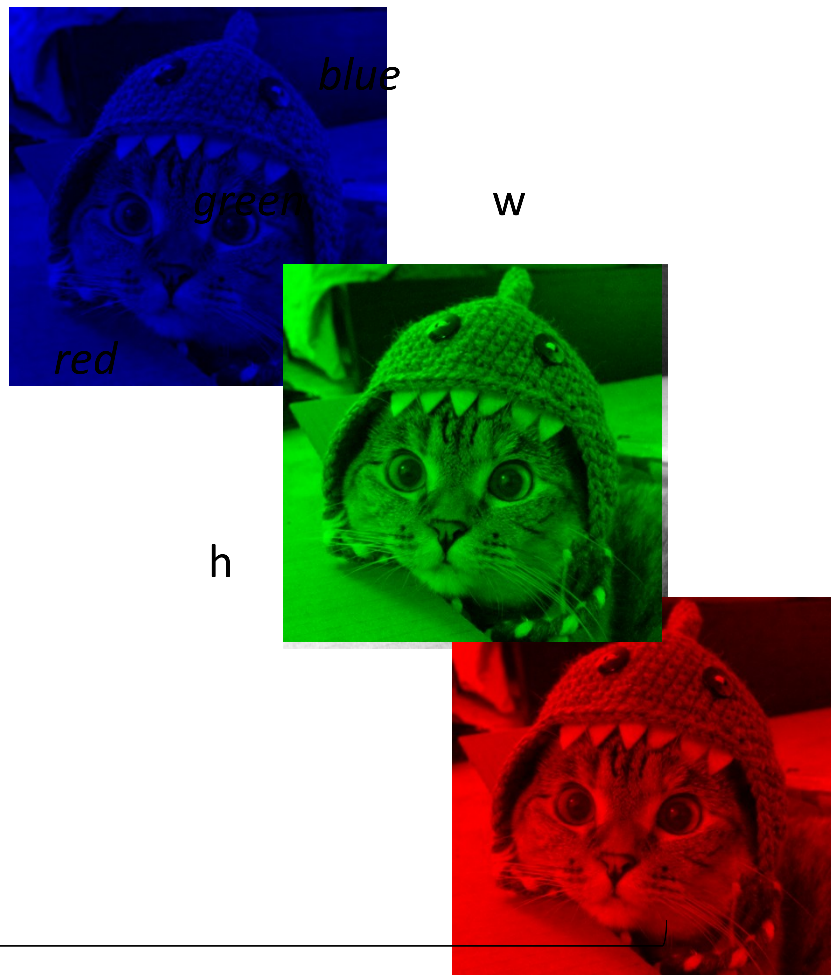
weights

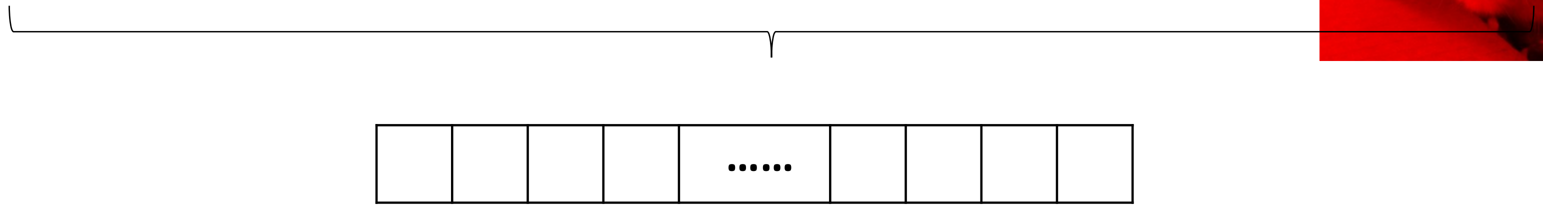
bias

$$\xrightarrow{\quad} \text{Classifier} \xrightarrow{\quad} \text{N class scores}$$

Image representation







Flatten the image to a 1D array

w*h*3 values between [0, 255]

Linear classification

CIFAR 10 example:

- 3 channel images, 32x32 pixels $x_i \in \mathbb{R}^D$
- N = 10 class scores

$$f\colon \mathbb{R}^D \rightarrow \mathbb{R}^N \quad , f(x_i, W, b) = W x_i + b$$

$\longrightarrow$ Classifier $\longrightarrow$ N class scores

N = 10

D = 32x32x3 = 3072

Linear classification

CIFAR 10 example:

- 3 channel images, 32x32 pixels, x_i size [3072 x 1]
- N = 10 class scores

$x_i \in \mathbb{R}^D$

??? [D x 1]

$$f(x_i, W, b) = \underbrace{W}_{} \boxed{\;} x_i + b$$

[3072 x 1]

$f: \mathbb{R}^D \rightarrow \mathbb{R}^N$

$\longrightarrow$ Classifier $\longrightarrow$

N class scores

N = 10

D = 32x32x3 = 3072

Linear classification

CIFAR 10 example:

- 3 channel images, 32x32 pixels
- N = 10 class scores

$x_i \in \mathbb{R}^D$

[D x 1]

size: N x D

$$f(x_i, W, b) = \boxed{W} x_i + b$$

10 x 3072

[3072 x 1]

$f: \mathbb{R}^D \to \mathbb{R}^N$

$\longrightarrow$ Classifier $\longrightarrow$ N class scores

N = 10

D = 32x32x3 = 3072

Linear classification

CIFAR 10 example:

- 3 channel images, 32x32 pixels
- N = 10 class scores

$x_i \in \mathbb{R}^D$

[D x 1]

size: N x D

size: N x 1

$$f(x_i, W, b) = \boxed{W} x_i \boxed{+b}$$

$f: \mathbb{R}^D \rightarrow \mathbb{R}^N$

10 x 3072

10 x 1

[3072 x 1]

$\longrightarrow$ Classifier $\longrightarrow$

N class scores

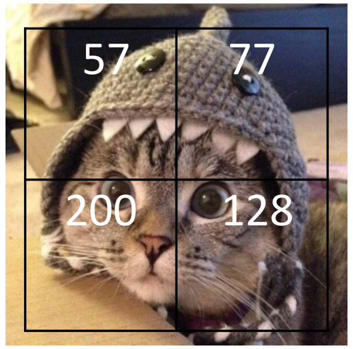
N = 10

D = 32x32x3 = 3072

Linear classification- simplified example

$$\overset{\text{weights } W}{\begin{bmatrix} 0.7 & -0.3 & 2.5 & 1 \\ -0.3 & 0.5 & 3.1 & -3 \\ 0 & 0.25 & 0.4 & 0.1 \end{bmatrix}} \cdot \overset{X}{\begin{bmatrix} 57 \\ 77 \\ 200 \\ 128 \end{bmatrix}} + \overset{\text{bias } b}{\begin{bmatrix} 1.7 \\ -2 \\ 3 \end{bmatrix}} = \begin{array}{cl} \begin{bmatrix} 646.5 \\ 255.4 \\ 115.05 \end{bmatrix} & \begin{array}{l} \text{cat score} \\ \text{boat score} \\ \text{shark score} \end{array} \end{array}$$

Flattened input image



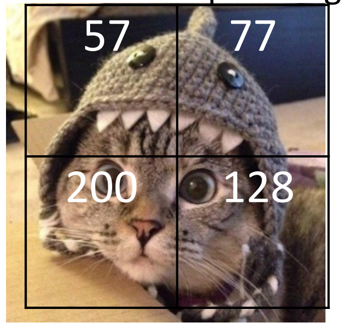
Example adapted from: https://cs231n.github.io/

Linear classification – implementation hints

The bias trick

$$\overset{\text{weights } W}{\begin{bmatrix} 0.7 & -0.3 & 2.5 & 1 & \blacksquare \\ -0.3 & 0.5 & 3.1 & -3 & \blacksquare \\ 0 & 0.25 & 0.4 & 0.1 & \blacksquare \end{bmatrix}} \cdot \overset{x}{\begin{bmatrix} 57 \\ 77 \\ 200 \\ 128 \end{bmatrix}} + \overset{\text{bias } b}{\begin{bmatrix} 1.7 \\ -2 \\ 3 \end{bmatrix}} = \begin{array}{ll} \begin{bmatrix} 646.5 \\ 255.4 \\ 115.05 \end{bmatrix} & \begin{array}{l} \text{cat score} \\ \text{boat score} \\ \text{shark score} \end{array} \end{array}$$

Flattened input image



Example adapted from: https://cs231n.github.io/

Linear classification – implementation hints

The bias trick

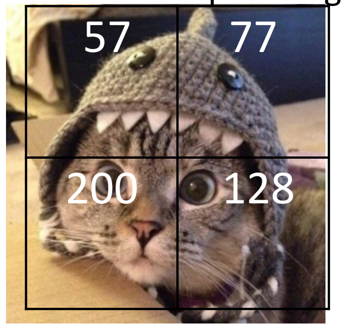
weights W

x

$$\begin{bmatrix} 0.7 & -0.3 & 2.5 & 1 & \boxed{1.7} \\ -0.3 & 0.5 & 3.1 & -3 & \boxed{-2} \\ 0 & 0.25 & 0.4 & 0.1 & \boxed{3} \end{bmatrix} \cdot \begin{bmatrix} 57 \\ 77 \\ 200 \\ 128 \\ \boxed{1} \end{bmatrix} = \begin{array}{|c|l} \hline 646.5 & \text{cat score} \\ \hline 255.4 & \text{boat score} \\ \hline 115.05 & \text{shark score} \\ \hline \end{array}$$

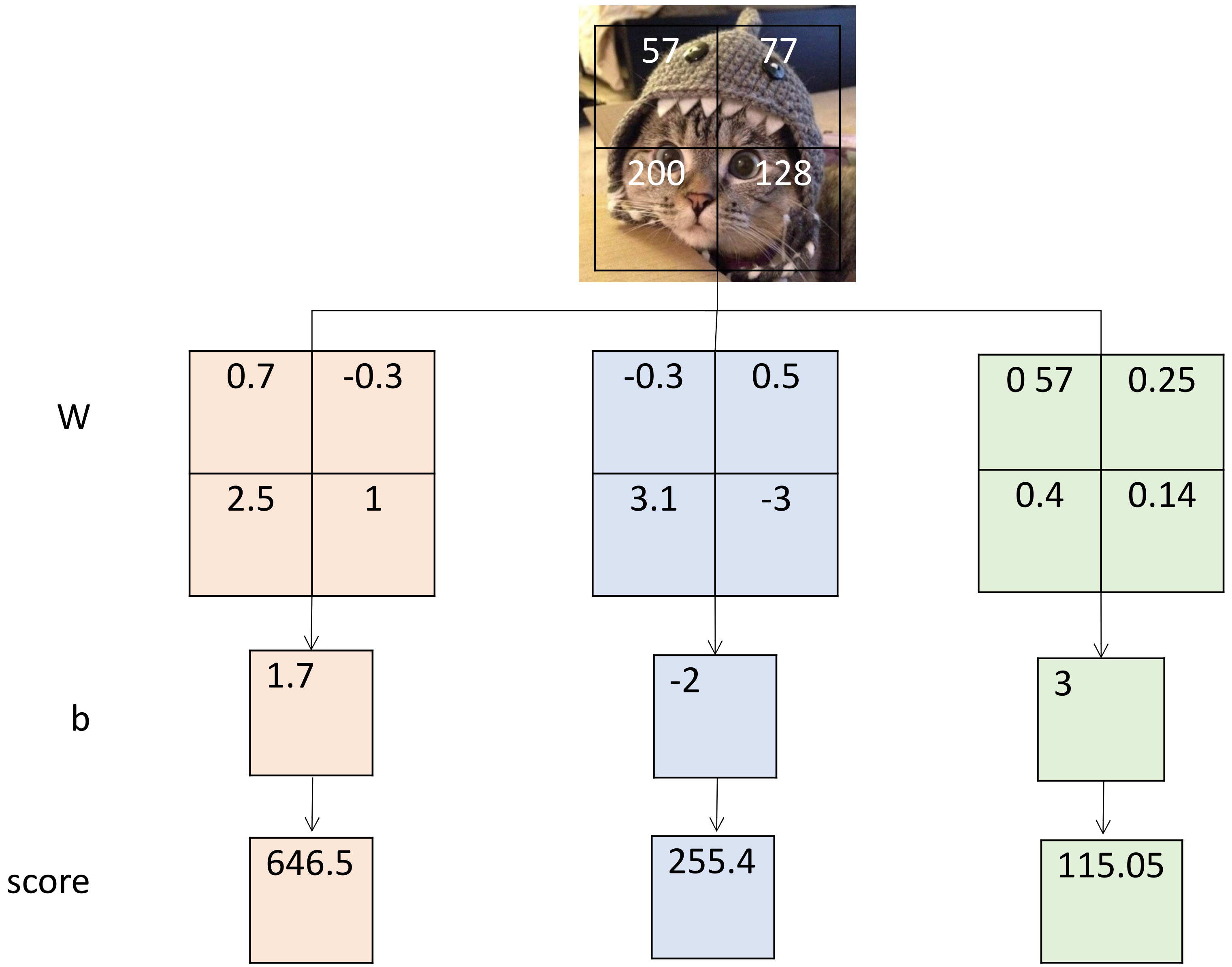
bias b

Flattened input image



Example adapted from: https://cs231n.github.io/

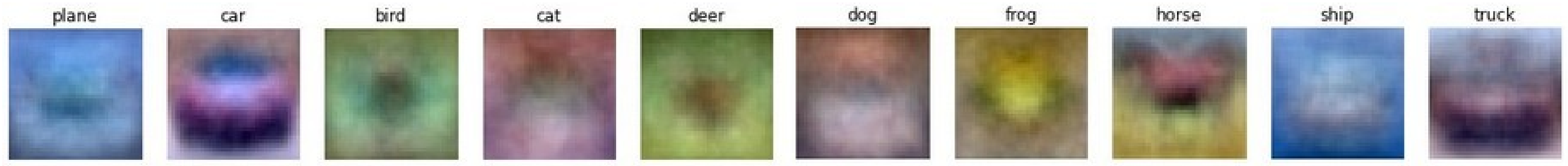
Linear classification- simplified example



0.7	-0.3	2.5	1
-0.3	0.5	3.1	-3
0	0.25	0.4	0.1

Example adapted from: https://cs231n.github.io/

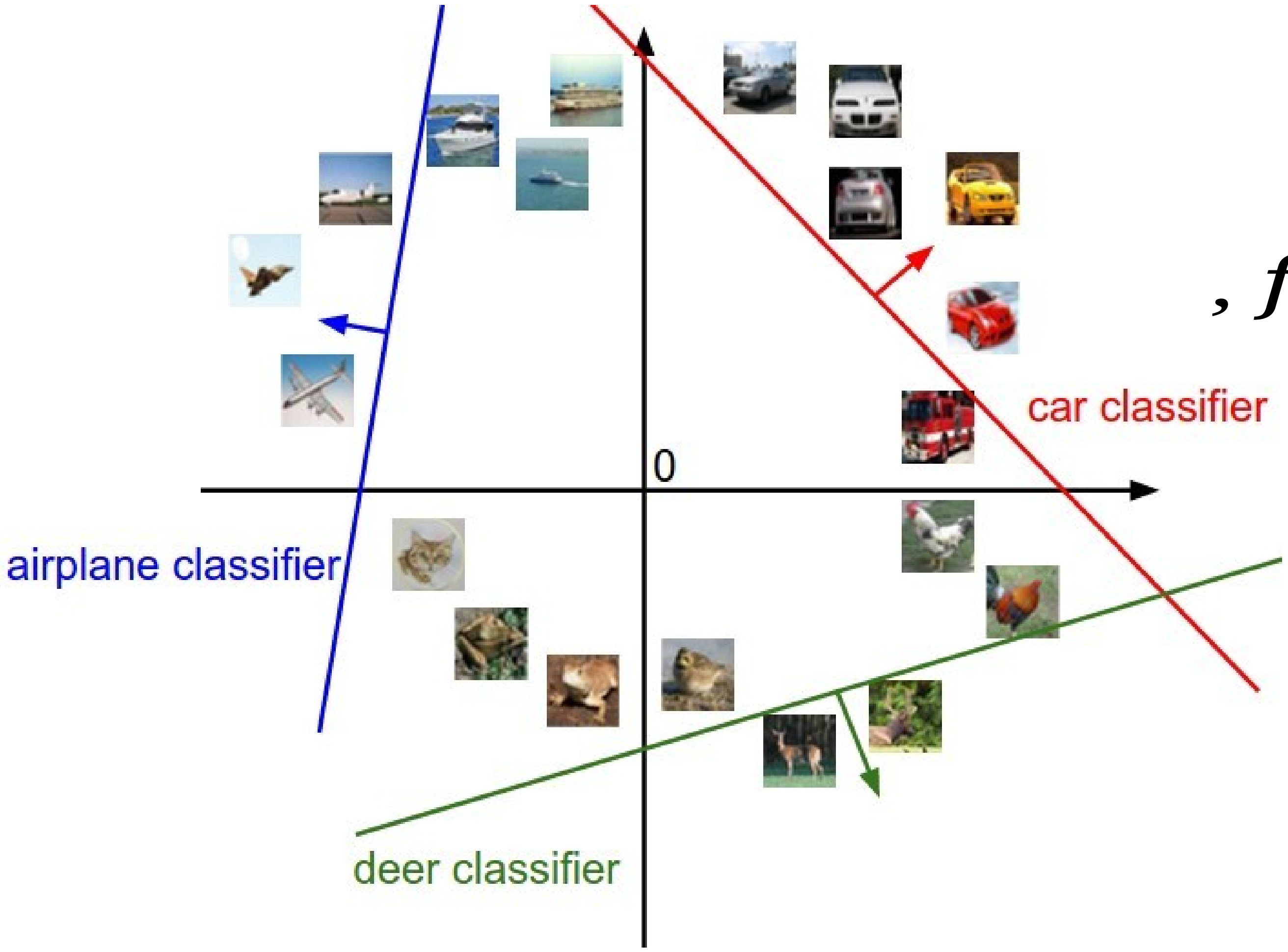
Linear classification – visual interpretation



- Linear classification can be seen as a *template matching* process, where the weights are learned
- Each row of the weights matrix is a *prototype* (template) for one of the classes

Image source: https://cs231n.github.io/

Linear classification – geometric interpretation



Hyper-plane cutting up the space

$$, f(x_i, W, b) = W x_i + b$$

Image source: https://cs231n.github.io/

Linear classification

$$f(x_i, W, b) = W x_i + b$$

- Input x_i → fixed
- N: number of classes →fixed
- W – weights → we have control over their setting
- b – bias → we have control over their setting
  - b doesn’t actually interact with the data
- *f* – ***score function*** *- map raw pixels to class scores*

Goal: set W and b such that the correct class has the a higher value than in incorrect ones

Next steps

- Define a **loss function *L***
  - ***L* -** measures the quality of W and b based on how well the predicted scores agree with the ground truth labels
  - ***L* –** how happy are we with the scores across the training data? How good is our (current) classifier?
- Find an efficient way to determine the parameters (W and b) such that the loss function is minimized

Loss function

$$L = \frac{1}{N} \sum_{i}^{N} L_i \left(f \left(x_i , W \right) , y_i \right)$$

- N – number of training examples
- - ith image in our training dataset
- - ground truth label for the ith image in our training set (i.e. cat, boat or shark)

Example – hinge loss or SVM loss

- = - the prediction of the jth image in our training dataset
- m – margin

$$L_i = \sum_{j != y_i}^{\square} max\left(0, s_j - s_{y_i} + m\right)$$

Example – hinge loss

- = - the prediction of the jth image in our training dataset
- m – margin

$$L_i = \sum_{j != y_i}^{\square} max\left(0, s_j - s_{y_i} + m\right)$$

$max\left(0, s_j - s_{y_i} + m\right)$

```
if
    return 0
else:
    return
```

Example – hinge loss

- = - the prediction of the jth image in our training dataset
- m – margin

$$L_i = \sum_{j != y_i} max\left(0, s_j - s_{y_i} + m\right)$$

$$¿ \sum_{j != y_i} \begin{cases} 0, s_{y_i} \geq s_j + m \\ ¿ m + s_j - s_{y_i}, otherwise \end{cases}$$

```
$max\left(0, s_j - s_{y_i} + m\right)$
        if
            return 0
        else:
            return
```

Example – hinge loss

- = - the prediction of the jth image in our training dataset
- m – margin

$$L_i = \sum_{j != y_i}^{❑} max\left(0, s_j - s_{y_i} + m\right)$$

$$¿ \sum_{j != y_i} \begin{cases} 0, s_{y_i} \geq s_j + m \\ ¿\, m + s_j - s_{y_i}, otherwise \end{cases}$$

```
$max\left(0, s_j - s_{y_i} + m\right)$
            if
                return 0
            else:
                return
```

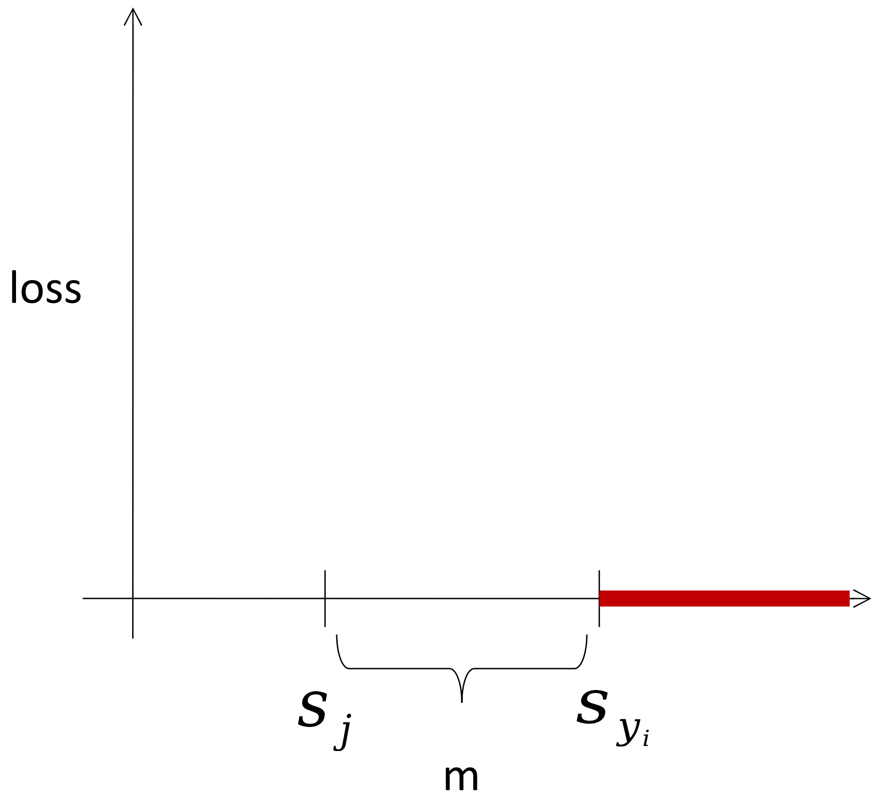
Intuition: we are “happy” if the score of the correct class if larger, by a margin of at least m, than the score of the other classes

Example – hinge loss

- = - the prediction of the jth image in our training dataset
- m – margin

$$L_i = \sum_{j != y_i} \max\left(0, s_j - s_{y_i} + m\right)$$

$$¿ \sum_{j != y_i} \begin{cases} 0, s_{y_i} \geq s_j + m \\ ¿ m + s_j - s_{y_i}, otherwise \end{cases}$$

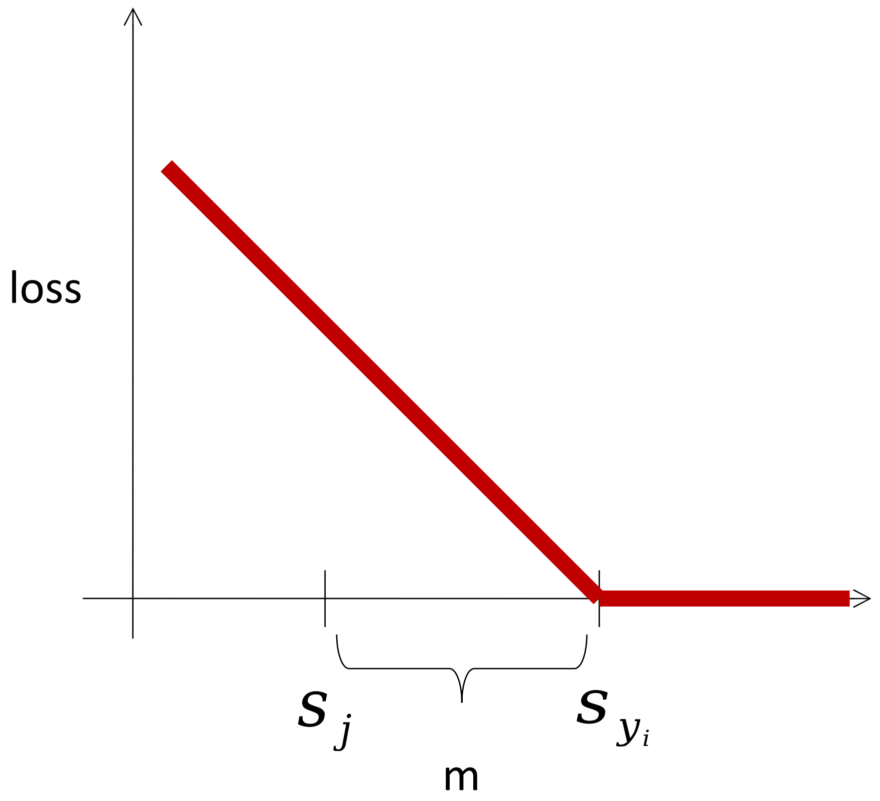


Example – hinge loss

- = - the prediction of the jth image in our training dataset
- m – margin

$$L_i = \sum_{j!=y_i} max\left(0, s_j - s_{y_i} + m\right)$$

$$¿ \sum_{j!=y_i} \begin{cases} 0, s_{y_i} \geq s_j + m \\ ¿\, s_j - s_{y_i}, otherwise \end{cases}$$



Example – hinge loss

Class			
cat	3.2	0.1	0.3
boat	-5	14	8
shark	10	5	14
Hinge loss			

$$L_{Hi} = \sum_{j != y_i} max\left(0, s_j - s_{y_i} + m\right)$$

Let's set m = 1

For image 1

=max(0, - 5 – 3.2 + 1) + max(0, 10 – 3.2 + 1)
= 0 + 7.8
= 7.8

Example – hinge loss

Class			
cat	3.2	0.1	0.3
boat	-5	14	8
shark	10	5	14
Hinge loss	**7.8**		

$$L_{Hi} = \sum_{j != y_i} max\left(0, s_j - s_{y_i} + m\right)$$

Let's set m = 1

For image 1

=max(0, - 5 – 3.2 + 1) + max(0, 10 – 3.2 + 1)
= 0 + 7.8
= 7.8

Example – hinge loss

Class			
cat	3.2	0.1	0.3
boat	-5	14	8
shark	10	5	14
Hinge loss	**7.8**	**0**	**0**

$$L_{Hi} = \sum_{j != y_i} max\left(0, s_j - s_{y_i} + m\right)$$

Let's set m = 1

For image 2

= max(0, 0.1 – 14 + 1) + max(0, 5 – 14 + 1)
= 0 + 0
= 0

For image 3

= max(0, 0.3 – 14 + 1) + max(0, 8 – 14 + 1)
= 0 + 0
= 0

Total loss L

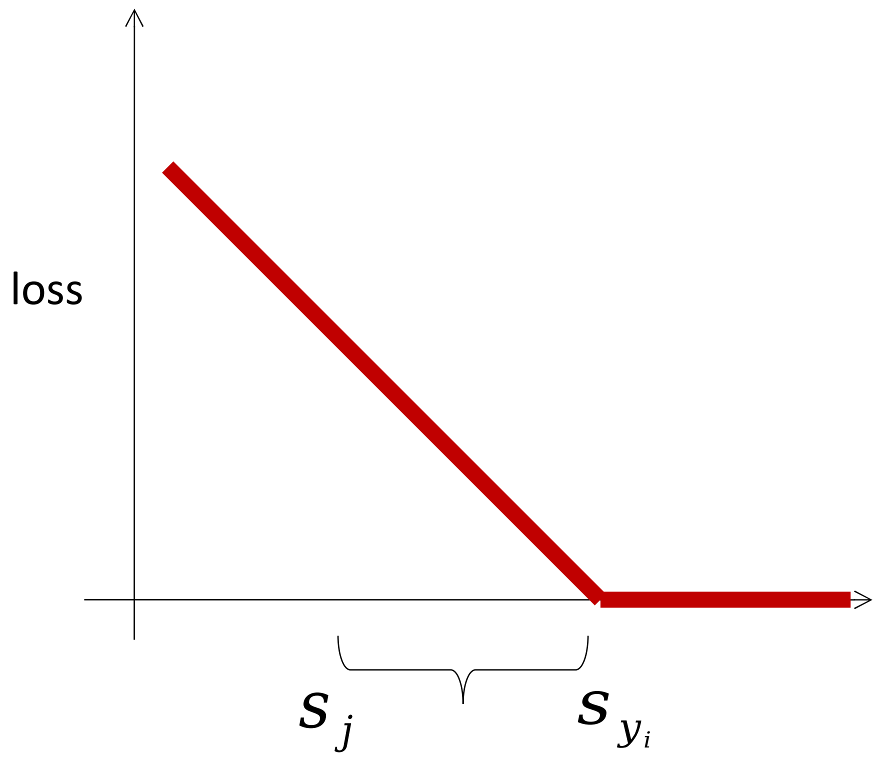
$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

L = (7.8 + 0 + 0)/3 = 2.6

Hinge loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

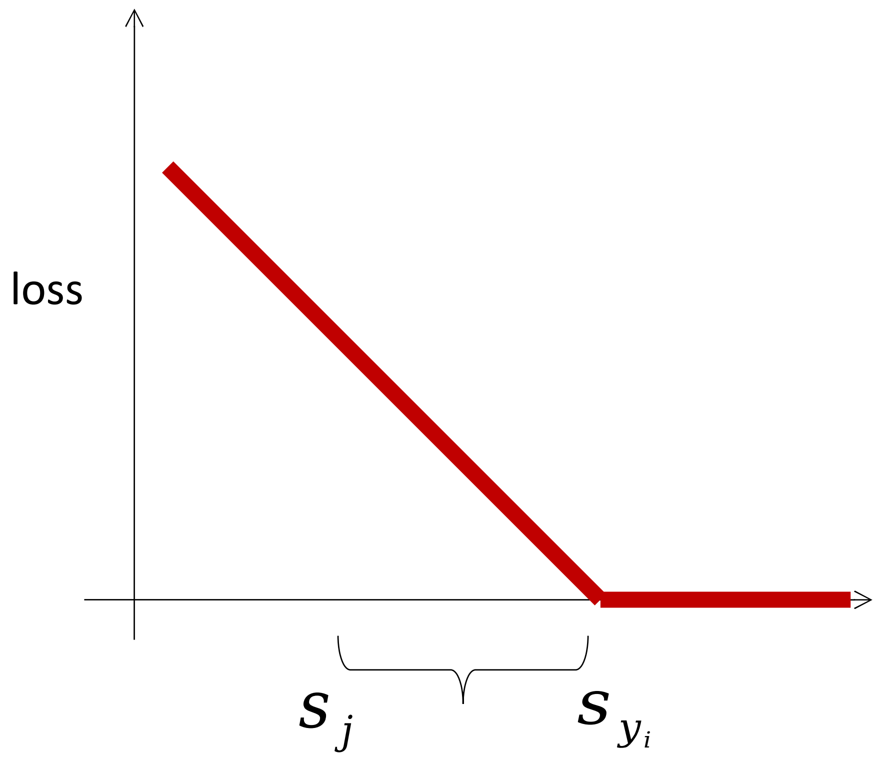
- What is the min value and the max value of hinge loss?



Hinge loss

$$L_{Hi} = \sum_{j != y_i} max\left(0, s_j - s_{y_i} + 1\right)$$

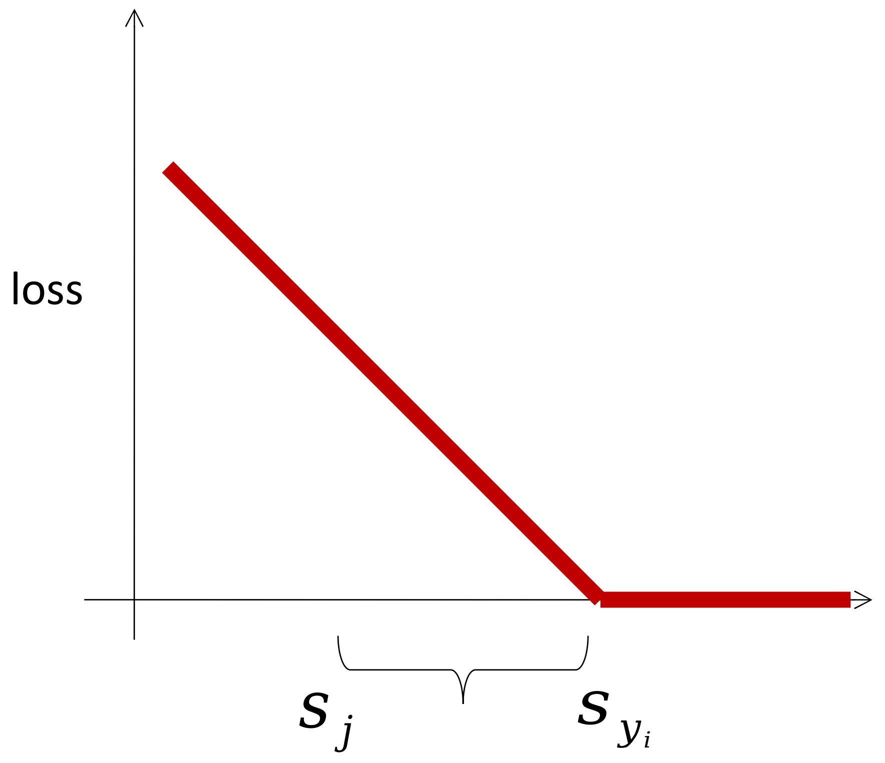
- What is the min value and the max value of hinge loss?
  - min is 0: the correct score is much larger (>=m) than the incorrect scores
  - max is infinity
- In the beginning of the training (W ≈ 0) so all scores ≈ 0. What is the value of L ?



Hinge loss

$$L_{Hi} = \sum_{j \,!= y_i} max\left(0, s_j - s_{y_i} + 1\right)$$

- What is the min value and the max value of hinge loss?
- In the beginning of the training (W ≈ 0) so all scores ≈ 0. What is the value of L ?
  - C – 1



Regularization

- Suppose that you found some values for the parameters W such that the loss function is 0. Do you think there is a single setting for the parameters W, such that the loss is 0?

Regularization

- Suppose that you found some values for the parameters W such that the loss function is 0. Do you think there is a single setting for the parameters W, such that the loss is 0?
- W, 2W, 3W, αW
- extend the loss function with a **regularization penalty** to discourage large weights

Regularization

$$L = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i}_{\text{Data loss}} + \overbrace{\lambda R(W)}^{\text{Regularization loss}}$$

- Prevent overfitting (the model will perform ”too” well on the training data)
- Occam’s razor: William of Ockham (c. 1287–1347)
- Make the model simpler

Occam’s razor: "the simplest explanation is most likely the right one"

Image source: https://ro.pinterest.com/pin/192669690286459858/?nic_v2=1a57hjLPe

Regularization

- L2 regularization

x = [2, 0.1, 0.1, 0.5]

W1 = [0.2, 0.8, 0.2, 0.1] → R(W) = 0.73

W2 = [1, 0, 0, 0] → R(W) = 1

W1· X = 2

W2· X = 2

Regularization

- L2 regularization

$x = [2, 0.1, 0.1, 0.5]$

$W1 = [0.2, 0.8, 0.2, 0.1] \rightarrow R(W) = 0.73$ 👍

$W2 = [1, 0, 0, 0] \rightarrow R(W) = 1$

$W1 \cdot X = 2$

$W2 \cdot X = 2$

Prefers to “spread out” the weights such that each element has the same influence on the prediction

Regularization

- L1 regularization

$x = [2, 0.1, 0.1, 0.5]$
$W1 = [0.2, 0.8, 0.2, 0.1] \rightarrow R(W) = 1.4$
$W2 = [1, 0, 0, 0] \rightarrow R(W) = 1$

$W1 \cdot X = 2$
$W2 \cdot X = 2$

Regularization

- L1 regularization

x = [2, 0.1, 0.1, 0.5]
W1 = [0.2, 0.8, 0.2, 0.1] → R(W) = 1.4
W2 = [1, 0, 0, 0] → R(W) = 1

W1· X = 2
W2· X = 2

- Prefers more sparse weight matrices, most of the values in W should be close to 0, except for some values where the weights are allowed to deviate
- Feature selection

Example – softmax classifier

Class	
cat	3.2
boat	-5
shark	10

So far, we don't have interpretation for the outputs of the score function

What if we would like to determine the probability of each class?

Example – softmax classifier

So far, we don't have interpretation for the outputs of the score function

What if we would like to determine the probability of each class?

Class	
cat	3.2
boat	-5
shark	10

s

Softmax function

$$P(Y=c|X=x_i \text{¿} = \frac{e^{s_c}}{\sum_j e^{s_j}}$$

Example – softmax classifier

So far, we don't have interpretation for the outputs of the score function

What if we would like to determine the probability of each class?

Class	
cat	3.2
boat	-5
shark	10

s

Softmax function

$$P(Y=c|X=x_i\text{¿}=\frac{e^{s_c}}{\sum_j e^{s_j}}$$

normalize

Class	
cat	3.2
boat	-5
shark	10

s =
[
3.2
-5
10
]

=
[
24.5325
0.0067
22026.4658
]

= 22026.4658

$$\frac{e^{s_c}}{\sum_j e^{s_j}} = ¿$$

[
0.0011125
0.0000003
0.9988872
]

softmax function

0 – cat class label, 1 – boat, 2 – shark class

softmax function

[
0.0011125
0.0000003
0.9988872
]

vs

hardmax function, one hot encoding

[
1
0
0
]

- prediction for class j

- prediction for class j

Intuition:

- The loss = should be as small as possible
  → the correct probability class should be as big as possible

[
1
0
0
]

[
0.0011125
0.0000003
0.9988872
]

- prediction for class j

Intuition:

- The loss = should be as small as possible
  → the correct probability class should be as big as possible

This loss function just looks at the correct class from the ground truth and tries to make the predicted probability of that class as big as possible

$$L = \frac{1}{N} \sum_{i}^{N} L_i$$

- What is the minimum and maximum value of ?

$$L = \frac{1}{N} \sum_{i}^{N} L_i$$

- What is the minimum and maximum value of ?
  - Min 0
  - Max infinity

$$L = \frac{1}{N} \sum_{i}^{N} L_i$$

- What is the minimum and maximum value of the ?
  - Min 0
  - Max infinity

Optimization

How can we find the best value for W?

Gradient descent

Follow the slope!

Follow the slope

- In 1D the derivative of a function is:
  - Slope of the “line”

$$f'(a) = \frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Follow the slope

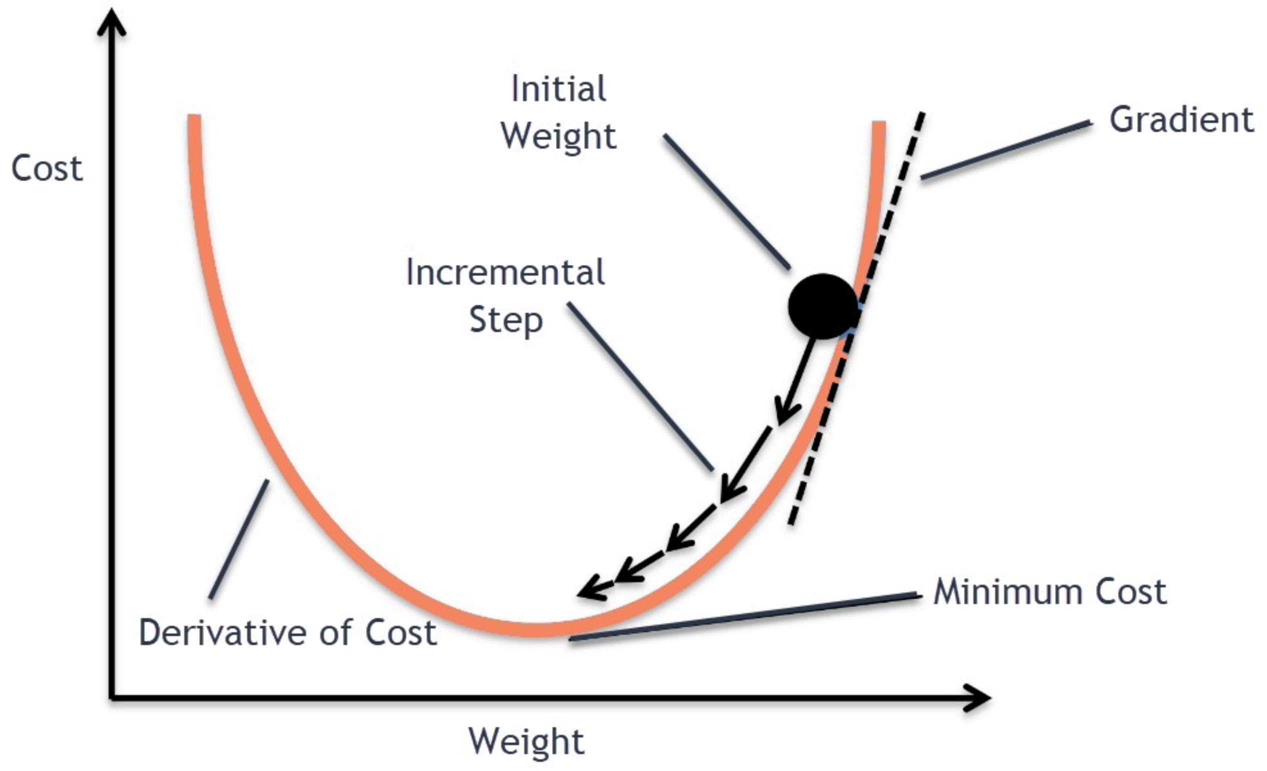


Image source: https://medium.com/analytics-vidhya/stochastic-gradient-descent-1ab661fabf89

Follow the slope

- In N dimensions, the gradient is a vector containing the partial derivative along each dimensions

Numerical gradient

W	[0.3, 0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,…] **loss 1.2345**
gradient	?

Numerical gradient

W [0.3,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,...]

loss 1.2345

W + h

h = 0.0001

[0.3001,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,...]

loss 1.2331

= 0.3 + h

= 0.3 + 0.0001

$$f'(a) = \frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

gradient

?

Numerical gradient

W [0.3,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,...]

loss 1.2345

W + h
h = 0.0001

[0.3001,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,...]

= 0.3 + h

= 0.3 + 0.0001

loss 1.2331

gradient [(1.2345 - 1.2331)/0.0001, ...]

Numerical gradient

W [0.3,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,...]

loss 1.2345

W + h
h = 0.0001

[0.3001,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,...]

loss 1.2331

= 0.3 + h

= 0.3 + 0.0001

gradient [23.5, ...]

Numerical gradient

W [0.3,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,…]

loss 1.2345

W + h
h = 0.0001

[0.3, 0.2401, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,…]

loss 1.2345

= 0.24 + h

gradient [23.5, …]

Numerical gradient

W [0.3,0.24, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,…]

loss 1.2345

W + h
h = 0.0001

[0.3001,0.2401, -1.9, 0.22, 0.3, 0.9, -1.7, 1.5, 0.29,…]

loss 1.2345

= 0.24 + h

gradient [23.5, (1.2345 – 1.2345)/0.0001, ….]

Analytical gradient

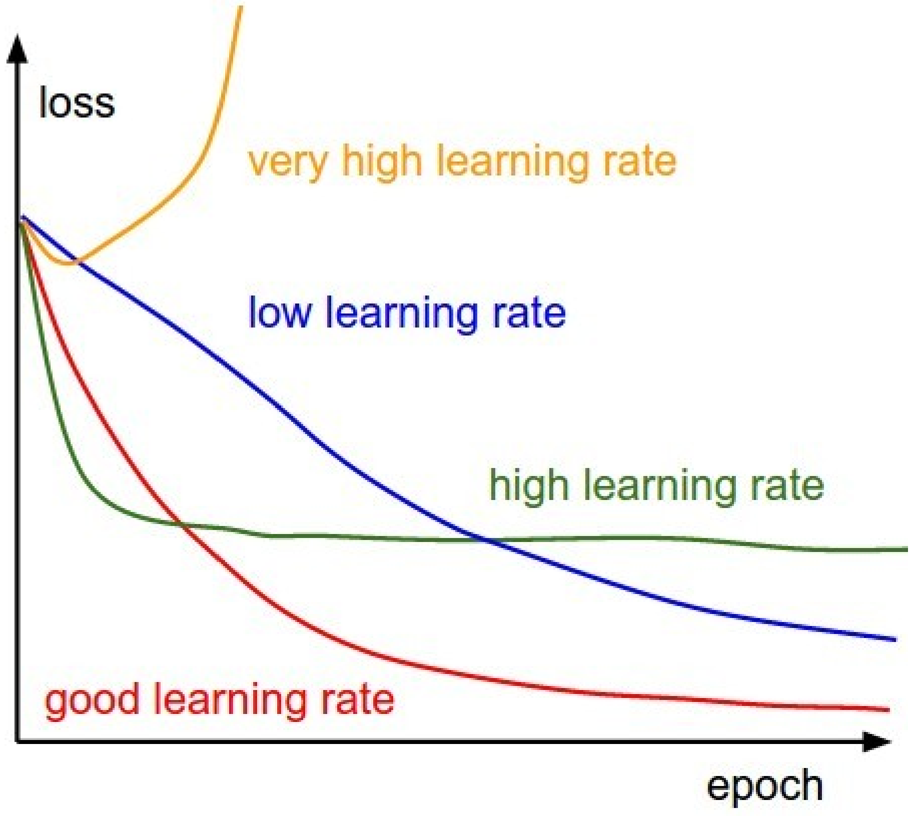
- Numerical gradient: slow and approximate
- **Analytical gradient.** Given:

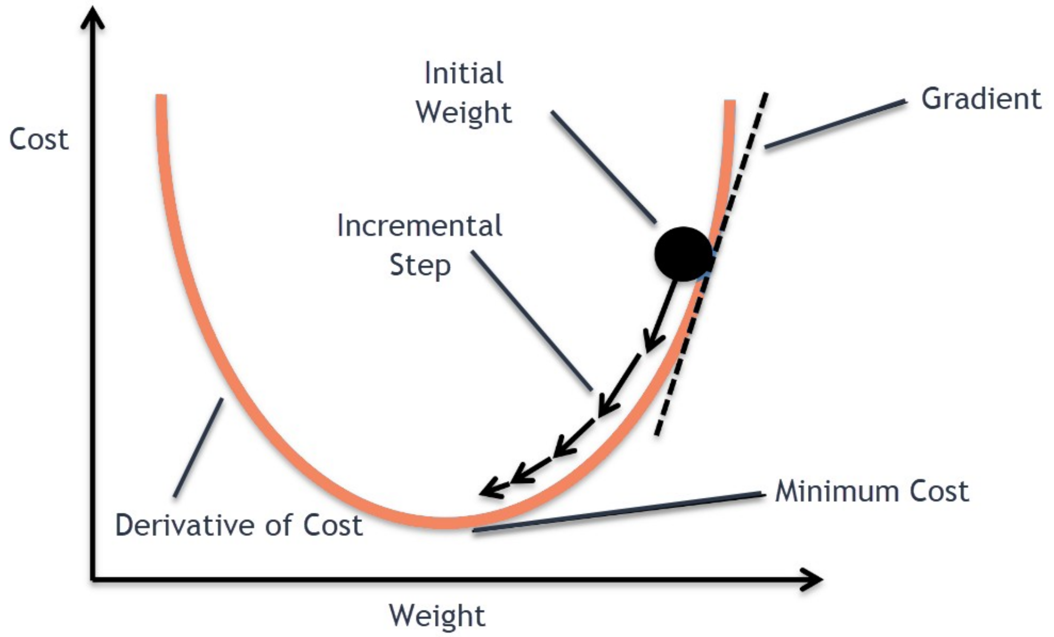
$$L = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W) \qquad L_i = \sum_{j != y_i}^{\square} max\,(0, s_j - s_{y_i} + m)$$

, we need to compute

and we can use calculus for this!

```
while True:
    W_gradient = compute_gradient(loss_func, data, W)
    # update parameters
    W += -lr*W_gradient
```





Example adapted from: https://cs231n.github.io/

Mini-batch gradient descent

Gradient descent is infeasible to compute if we have a large dataset

- Update the weights using a **mini-batch** of examples
- Common sizes for the batch: 32, 64, 128

```
while True:
    batch = sample_batch(data)
    W_gradient = compute_gradient(loss_func, batch, W)
    # update parameters
    W += -lr*W_gradient
```

What did we learn today?

- Score function
- Loss function
  - Hinge loss
  - Softmax loss
- Regularization
- Optimization
  - Gradient descent
  - Mini batch gradient descent

Recommended resources

- https://cs231n.github.io/linear-classify/#intro
- Loss functions (Andrew Ng`s great explanations):
  - https://www.youtube.com/watch?v=LLux1SW--oM
  - https://www.youtube.com/watch?v=ueO_Ph0Pyqk
- www.coursera.org – Deep Learning specialization: Neural Networks and Deep Learning (week 1 and 2)