Loss Function

Kietikul Jearanaitanakij

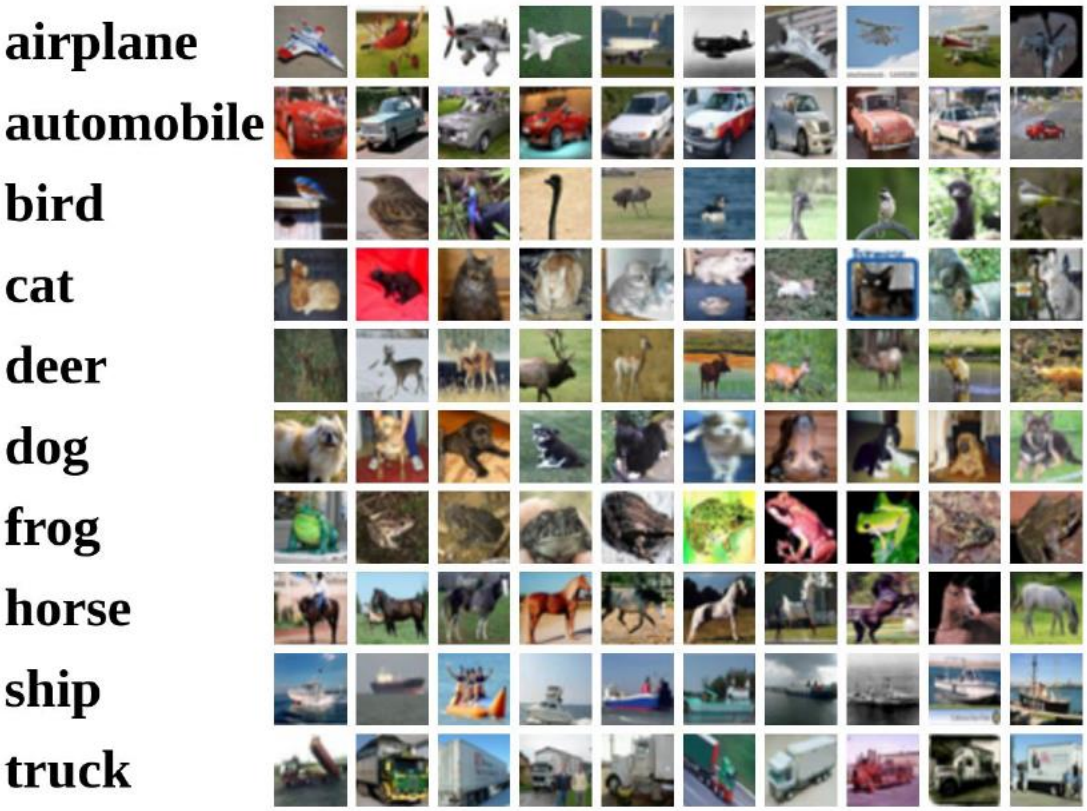
Department of Computer Engineering, KMITL

(Slides are adapted from cs231n @Stanford University)

Linear Classification (Revisited)

Suppose we want to use the linear classifier to classify images in CIFAR10.

CIFAR10



The CIFAR-10 is the labeled subsets of the 80 million tiny images dataset. (collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton)

50,000 training images
each image is **32x32x3**

10,000 test images.

10 classes.

Parametric Approach: Linear Classifier

Input image

$$f(x,W) = Wx$$

weight

Image

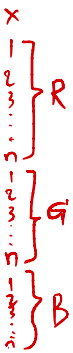
Array of **32x32x3** numbers
(3072 numbers total)

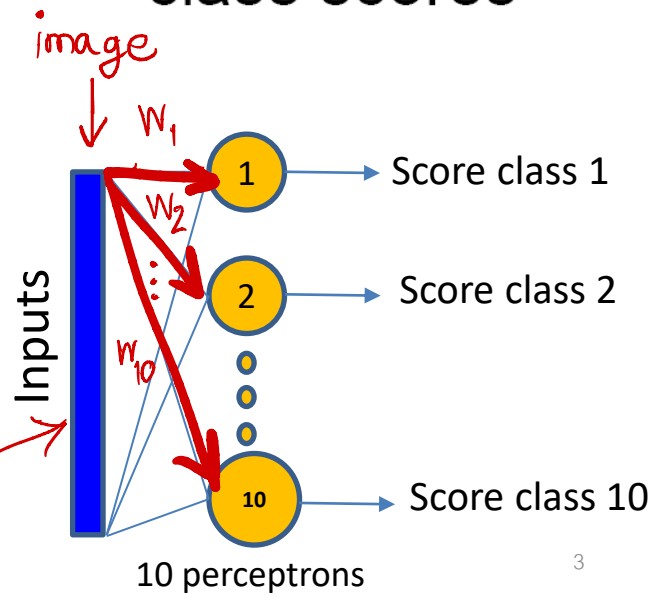
$f(\mathbf{x},\mathbf{W})$

10 numbers giving class scores

W

parameters or weights





Parametric Approach: Linear Classifier

$$f(x, W) = Wx$$

3072x1

10x1

10x3072

Image

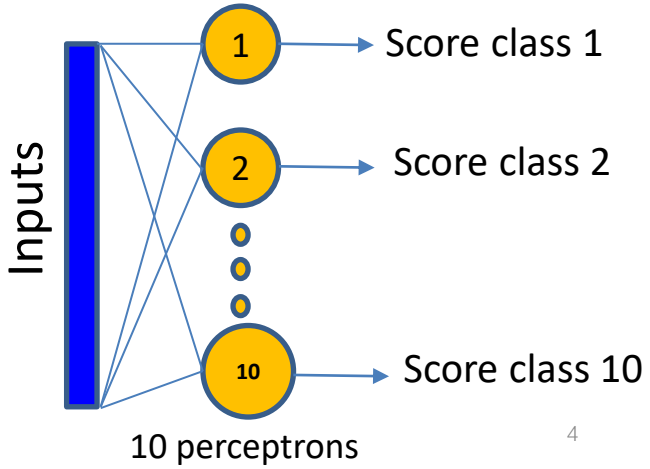
Array of **32x32x3** numbers
(3072 numbers total)

f(**x**,**W**)

10 numbers giving class scores

W

parameters or weights



Parametric Approach: Linear Classifier

$$f(x, W) = Wx + b$$

3072x1 (x)

10x1 (b)

10x1 (f(x,W))

10x3072 (W)

Image

Array of **32x32x3** numbers
(3072 numbers total)

3072 dimensions

f(**x**,**W**)

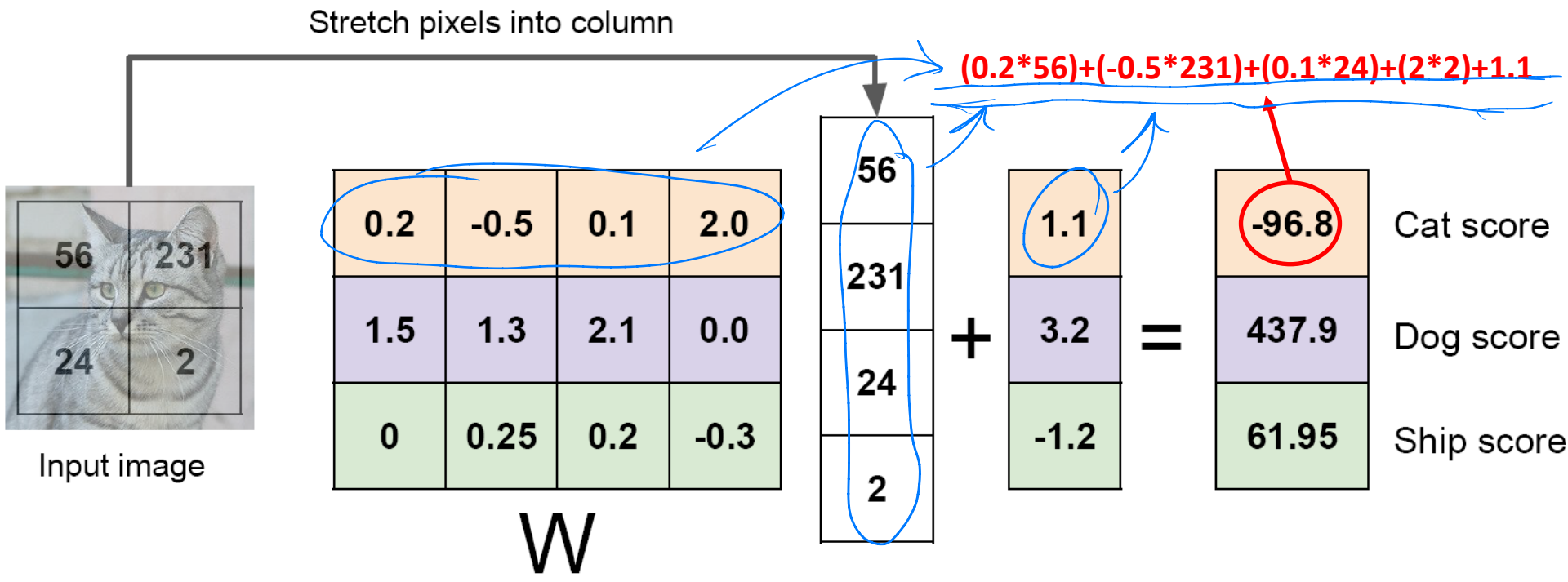
10 numbers giving class scores

W

parameters or weights

f(x,W) is called a score function.

Simplified scenario: 4 input features, 3 output classes.



Interpreting a Linear Classifier

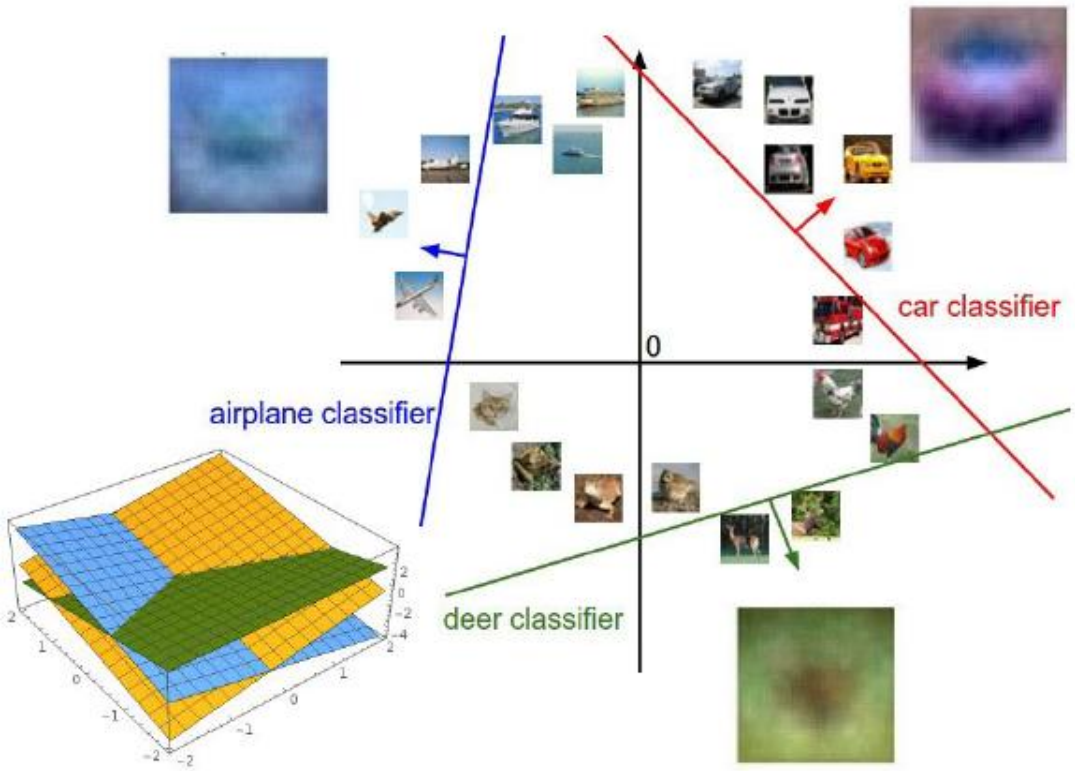
$$f(x,W) = Wx + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

เอา W ที่ผ่านการ train แล้ว มา ทำเป็น image

32×32×3

Interpreting a Linear Classifier



Plot created using Wolfram Cloud

$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

Score function : f(x,W) = Wx + b



airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Score function does not tell how good our classifier is.
We need a **loss function**.

Loss function

SVM Loss (Hinge)

Cross-entropy Loss

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

A **loss function** tells how good our current classifier is

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Target class (0-9)

Where x_i is image and
y_i is (integer) label

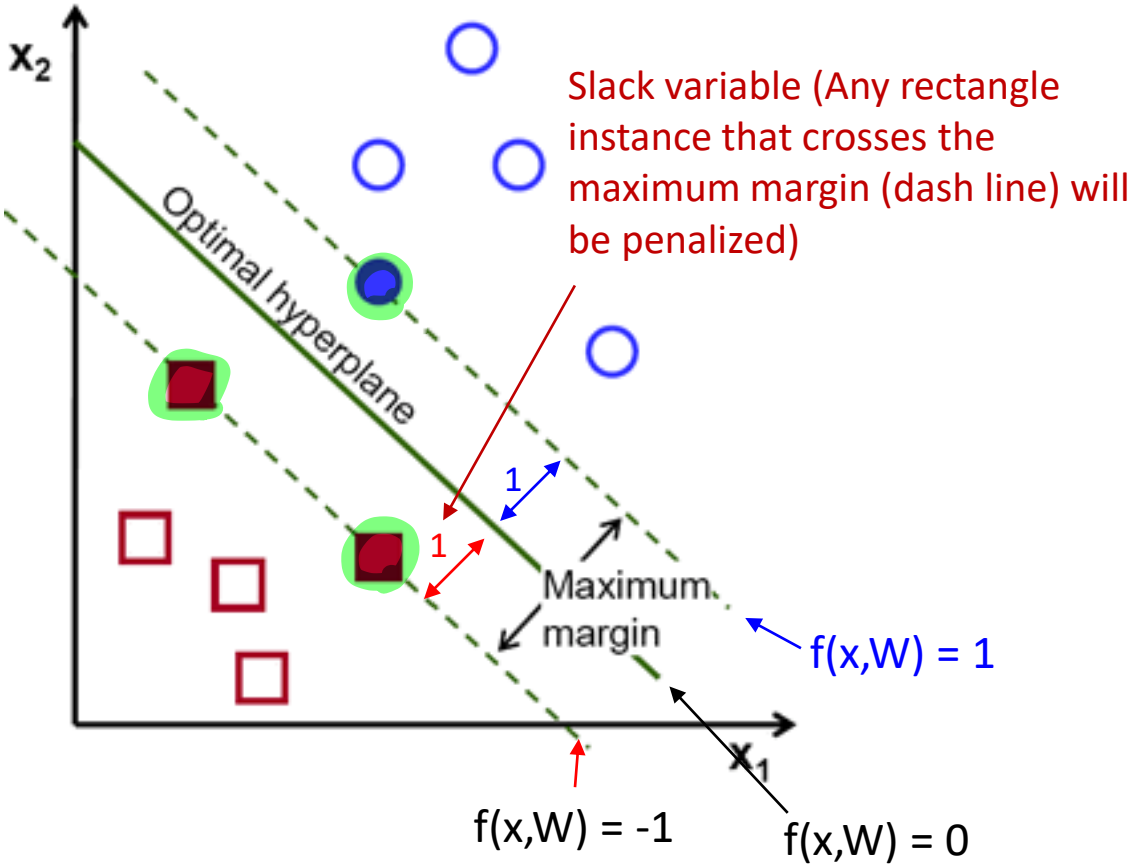
Loss over the dataset is a sum of loss over examples:

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

Score, Target, # of classes, class

SVM Loss

- Review: SVM (Support Vector Machine)

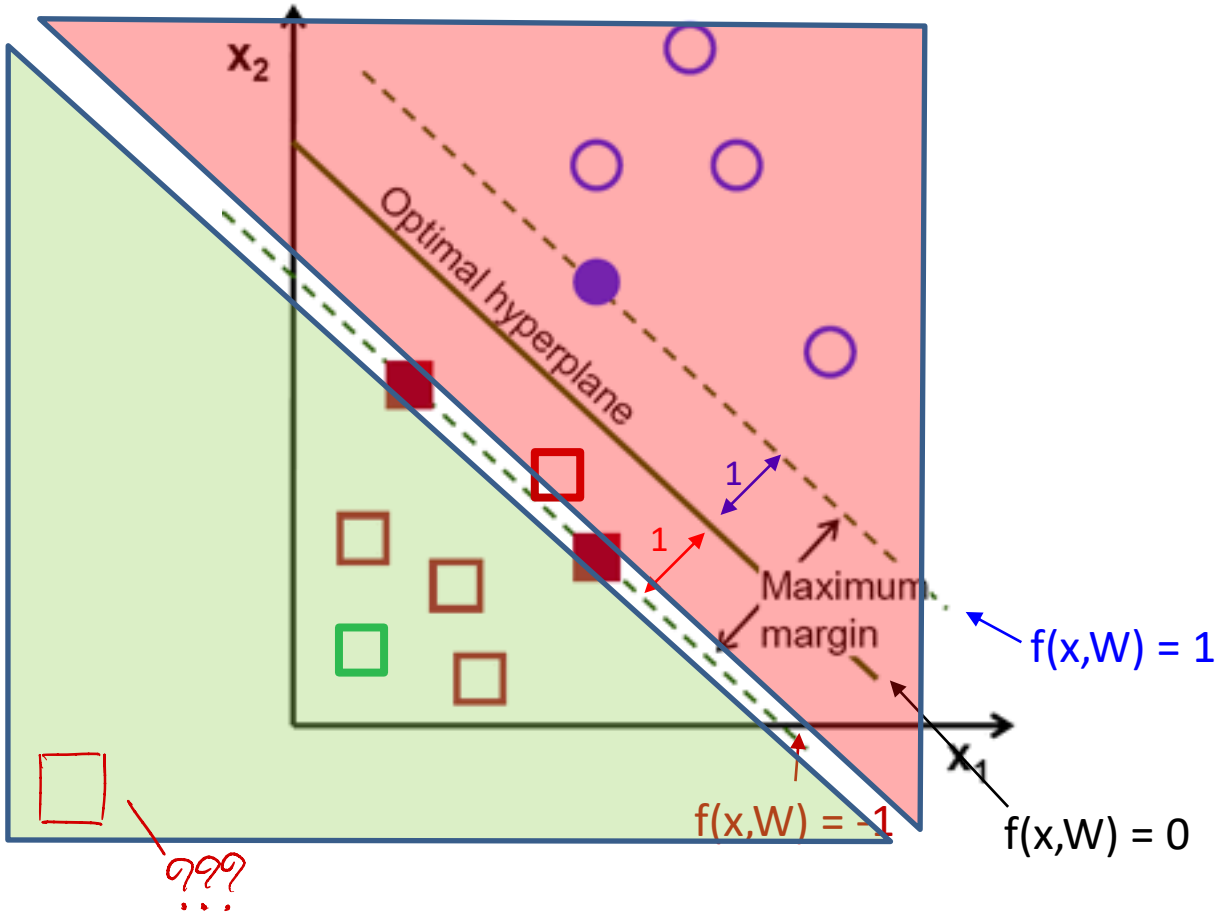


SVM Loss

- Review: SVM (Support Vector Machine)

Penalty = 0

Penalty > 0



Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



	s_{y_i}	s_j	s_j
cat	**3.2**	1.3	2.2
car	s_j 5.1	s_{y_i} **4.9**	s_j 2.5
frog	s_j -1.7	s_j 2.0	s_{y_i} **-3.1**

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

Score of other class

Score of the target class

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

= max(0, 2.2 - (-3.1) + 1)
+max(0, 2.5 - (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Multiclass SVM loss:

Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Loss over full dataset is average:

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

L = (2.9 + 0 + 12.9)/3
= **5.27**

Multiclass SVM Loss: Example code

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

```
def L_i_vectorized(x, y, W):
  scores = W.dot(x)
  margins = np.maximum(0, scores - scores[y] + 1)
  margins[y] = 0
  loss_i = np.sum(margins)
  return loss_i
```

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a W such that L = 0. Is this W unique?

No : $2 \cdot W \rightarrow L = 0$ อยู่ดี

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

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

Before:

= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

With W twice as large:

= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0
= 0

Smaller values of W are preferred.

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

SVM

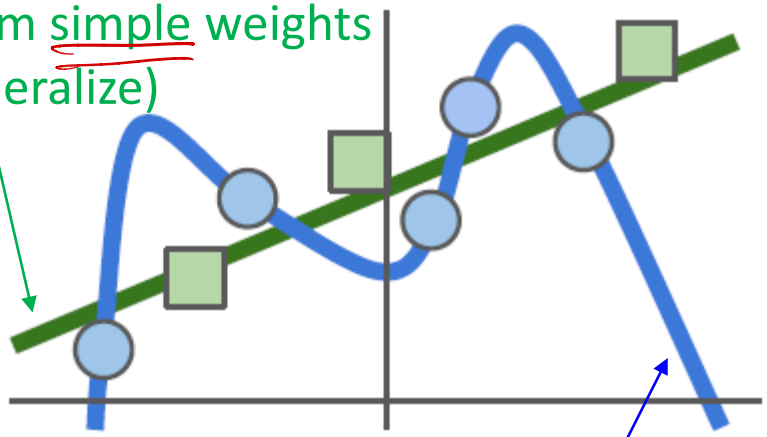
Data loss: Model predictions should match training data

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

Data loss: Model predictions should match training data

Regularization: Model should be "simple", so it works on test data

Model from simple weights (more generalize)



Occam's Razor:
"Among competing hypotheses, the simplest is the best"
William of Ockham, 1285 - 1347

Model from complex weights (tends to overfitting)

Regularization

λ = regularization strength (hyperparameter)

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i}\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

L2 regularization $R(W) = \sum_k \sum_l W_{k,l}^2$ มีปัญหากับ $0 < w < 1$

L1 regularization $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2) $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Softmax Classifier

(Cross - entropy loss)

- Score function assigns the real values onto all possible classes.
- The class which has the highest score represents the class of the given image.
- However, score of each class cannot directly compare to other classes since there is no proportion of the scores among all classes.
- We can handle this problem with Softmax function.

Softmax Classifier

Score function $s = f(x_i; W)$

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

where

$$s = f(x_i; W)$$

cat **3.2** s_k

car 5.1

frog -1.7

Softmax function

$$P(Y = \text{cat} \mid X = x_i) = \frac{e^{3.2}}{e^{3.2} + e^{5.1} + e^{-1.7}} = \text{Prob. cat}$$

$$P(Y = \text{car} \mid X = x_i) = \text{Prob. car}$$

$$P(Y = \text{Frog} \mid X = x_i) = \text{Prob. Frog}$$

1.0

Softmax Classifier

cat **3.2**

car 5.1

frog -1.7

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ where $$s = f(x_i; W)$$

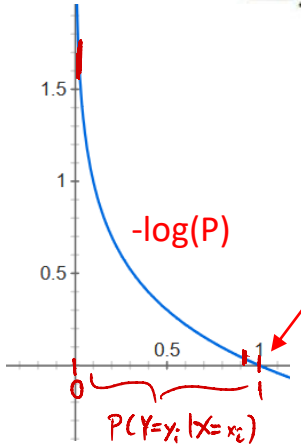
Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i)$$ cross-entropy loss

---

in summary: $$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$ ← Target class

Log likelihood



If P is approaching 1, loss will be zero.

Example: Softmax Classifier

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

cat	**3.2**
car	5.1
frog	-1.7

Score (S)

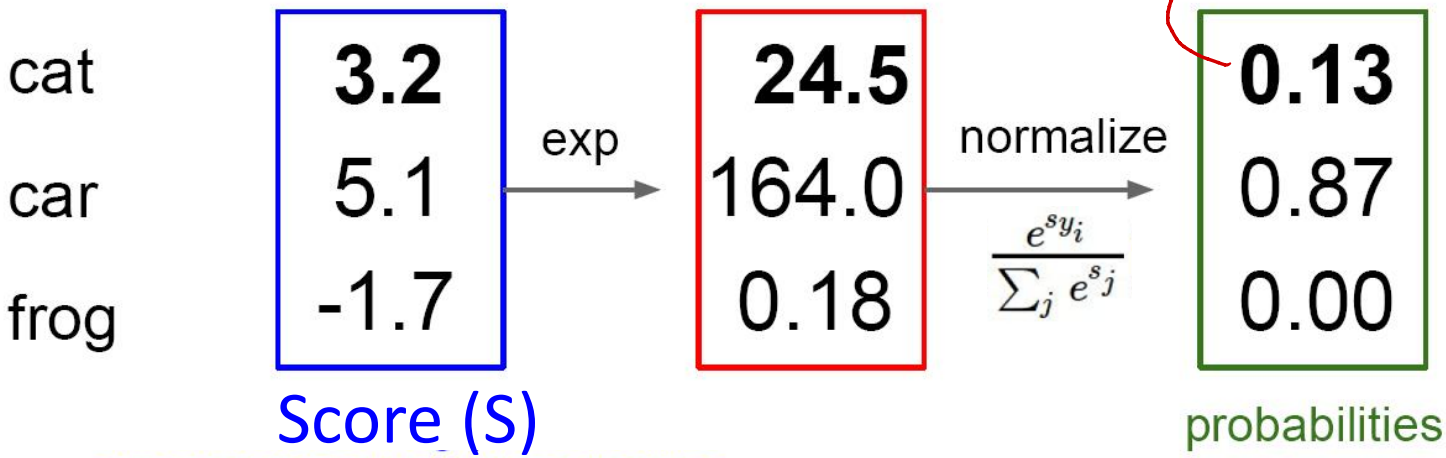
Example: Softmax Classifier

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

Example: Softmax Classifier

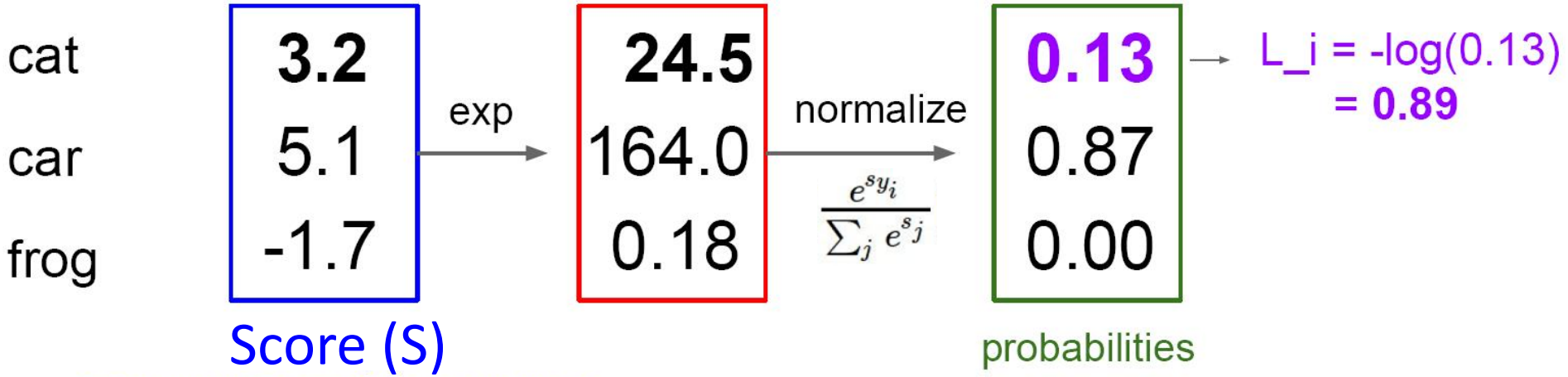
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

24.5 / 24.5 + 164 + 0.18



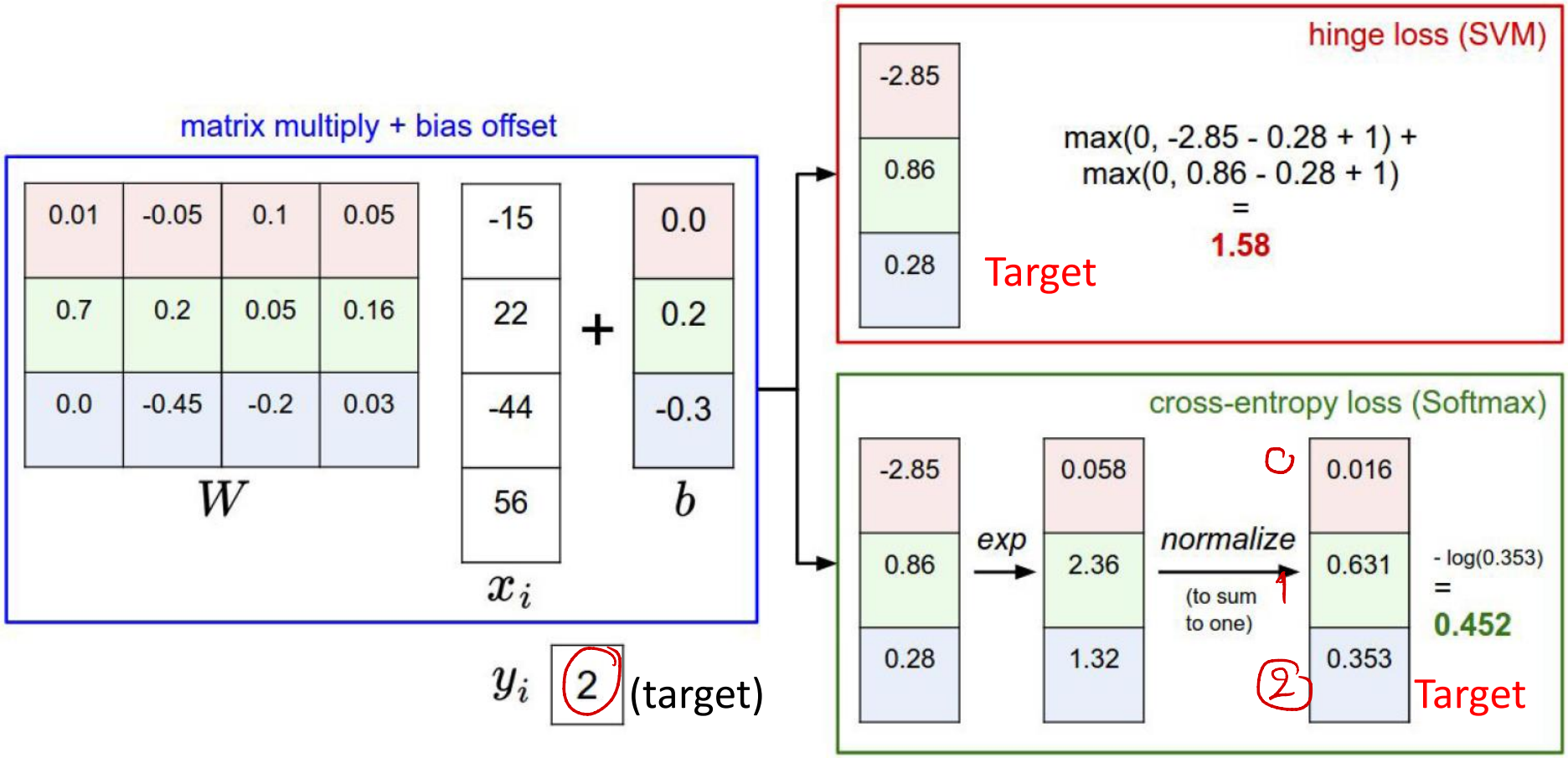
Example: Softmax Classifier

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$



Softmax loss คือค่า loss เฉพาะ target class เท่านั้น

Example: Hinge loss (SVM) VS Cross-entropy loss



Recap

How do we find the best W?

- We have some dataset of (x,y)
- We have a **score function:** $s = f(x; W) \overset{\text{e.g.}}{=} Wx$
- We have a **loss function**:

Softmax

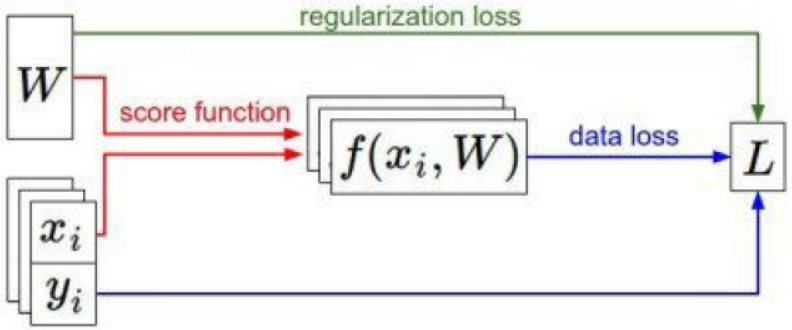
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

SVM

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

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$ Full loss

ตอน train จริงใช้ตัวนี้หา loss



Optimization

Optimization strategies

What do we need to optimize?

=> f(x;**W**) = **W**x

In order to minimize f, we need to optimize **weights**.

Strategies

1. Random search => Simple but low accuracy
2. Follow the slope

Partial derivative

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

} Gradient

We need to find the partial derivatives of all w_i.

Numerical approach

มองแบบ คณิตศาสตร์

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

Numerical approach

current W:	**W + h** (first dim):	**gradient dW:**
[0.34, -1.11, 0.78, 0.12, 0.55, 2.81, -3.1, -1.5, 0.33,…]	[0.34 + **0.0001**, -1.11, 0.78, 0.12, 0.55, 2.81, -3.1, -1.5, 0.33,…]	[?, ?, ?, ?, ?, ?, ?, ?, ?,…]
loss 1.25347	**loss 1.25322**	

Numerical approach

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25322

gradient dW:

slope

[**-2.5**,
?,
?,

(1.25322 - 1.25347)/0.0001
= -2.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,…]

Numerical approach

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,…]

Numerical approach

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (third dim):

[0.34,
-1.11,
0.78 + **0.0001**,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

gradient dW:

[-2.5,
0.6,
0,
?,
?,
?,…]

(1.25347 - 1.25347)/0.0001
= 0

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

This is silly. The loss is just a function of W:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \sum_k W_k^2$$

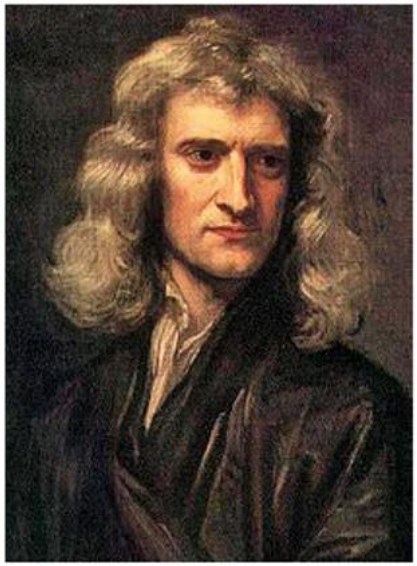
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$s = f(x; W) = Wx$$

want $\nabla_W L$

We can use calculus to compute the gradient (analytic approach).

This image is in the public domain

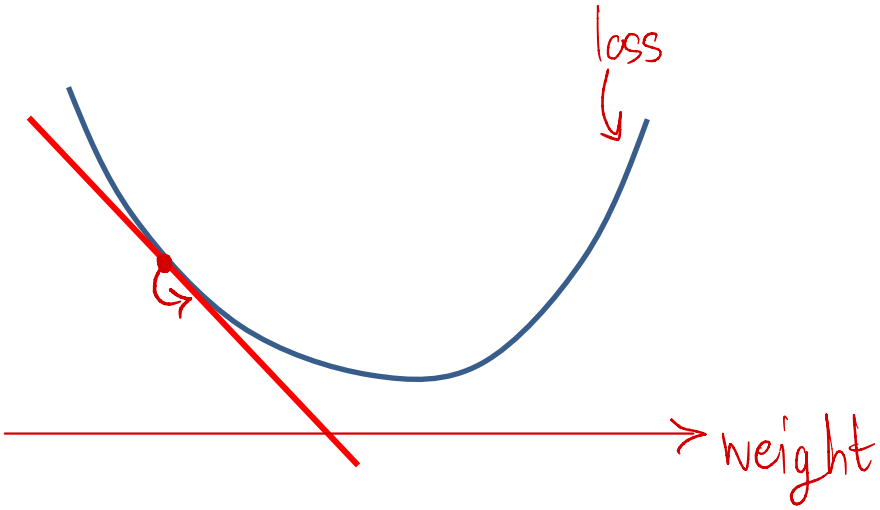


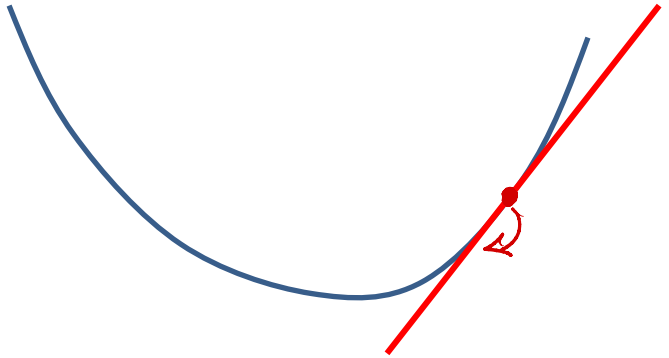
This image is in the public domain

Who? Isaac Newton
Gottfried Leibniz.

$$\frac{\partial f(w)}{\partial w_i} = \lim_{h \to 0} \frac{f(w_i + h) - f(wi)}{h}$$

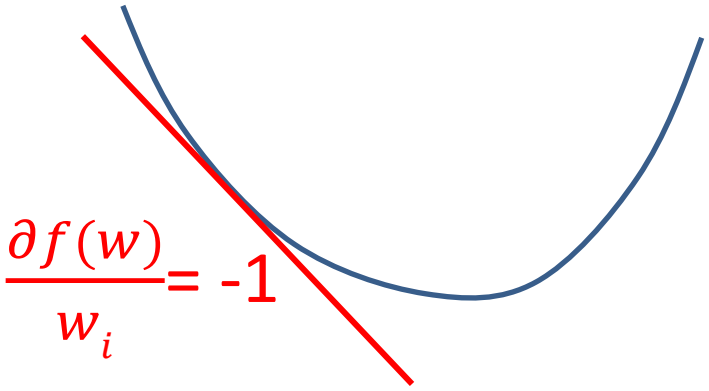
Gradient is actually the slope of the function corresponding to the small change (+h) of w_i.





Negative slope, e.g. -1, -1.5

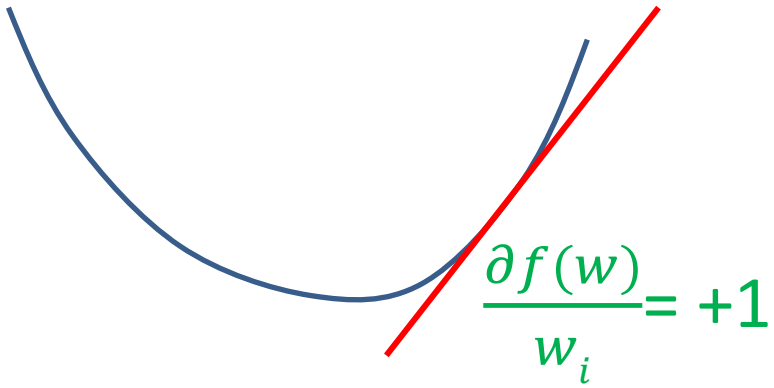
Positive slope, e.g. +1.2, +0.5



Negative slope means:
When we increase w_i by +h, we get smaller value of f(w).
This is good, so we follow the gradient.

$$w_i = w_i + stepsize.\left|\frac{\partial f(w)}{w_i}\right|$$

new old α Gradient



Positive slope means:
When we increase w_i by +h, we get larger value of f(w).
This is bad, so we follow the negative gradient.

$$w_i = w_i + stepsize.\left(-\left|\frac{\partial f(w)}{w_i}\right|\right)$$