

FACULTY OF MATHEMATICS,  
PHYSICS AND INFORMATICS  
Comenius University  
Bratislava

Neural Networks for Computer Vision

# Lecture 3: Loss Functions and Optimization

Ing. Viktor Kocur, PhD., doc. RNDr. Zuzana Černeková, PhD.

28.9.2023

# Acknowledgment



The majority of slides are directly adopted from slides for CS231n<sup>1</sup> course at Stanford University!

<sup>1</sup>Stanford CS231n lecture slides. <http://cs231n.stanford.edu/slides/>

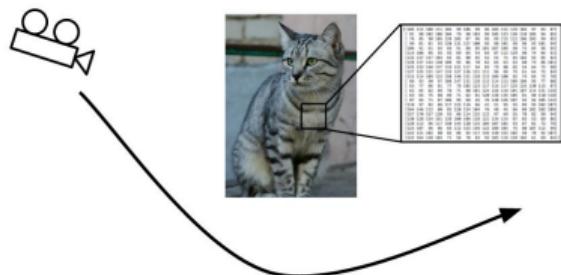
# Contents



- Recap
- Linear Classifier
- Optimization

# Recall from last time: Challenges of recognition

Viewpoint



Illumination



Deformation



Occlusion



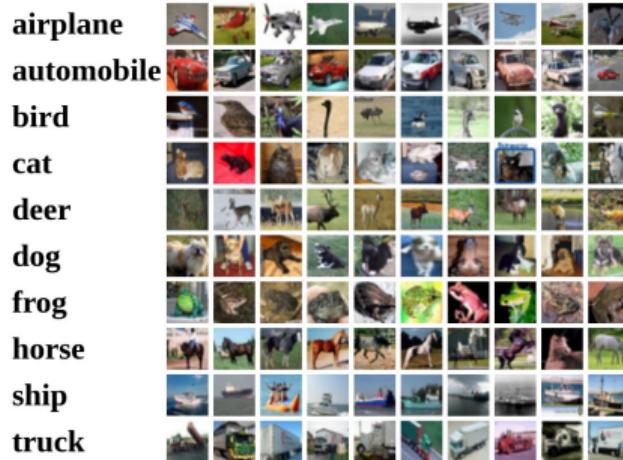
Clutter



Intraclass Variation



# Recall from last time: data-driven approach, kNN



train

test

train

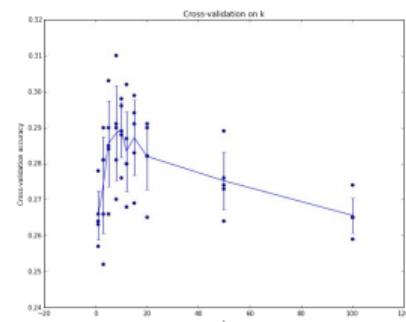
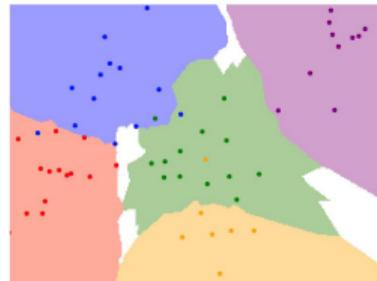
validation

test

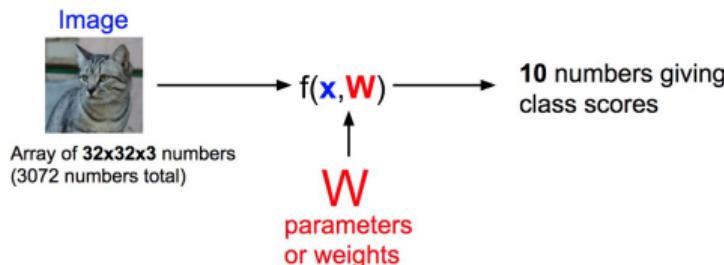
1-NN classifier



5-NN classifier



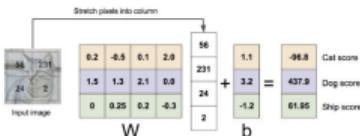
# Recall from last time: Linear Classifier



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

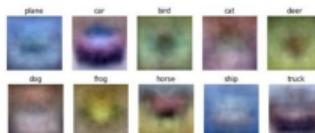
## Algebraic Viewpoint

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



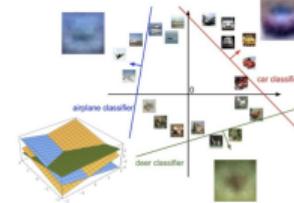
## Visual Viewpoint

One template per class



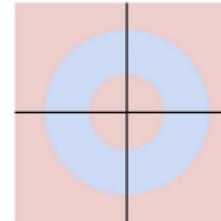
## Geometric Viewpoint

Hyperplanes cutting up space



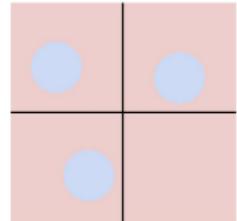
**Class 1:**  
 $1 \leq L_2 \text{ norm} \leq 2$

**Class 2:**  
Everything else



**Class 1:**  
Three modes

**Class 2:**  
Everything else



# Recall from last time: Linear Classifier



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

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function.  
**(optimization)**

Cat image by Nikita is licensed under CC-BY 2.0; Car image is CC0 1.0 public domain; Frog image is in the public domain

Suppose: 3 training examples, 3 classes.

With some  $W$  the scores  $f(x, W) = Wx$  are:

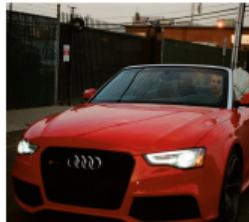


cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

Suppose: 3 training examples, 3 classes.

With some  $W$  the scores  $f(x, W) = Wx$  are:

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

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

Given a dataset of examples

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

Where  $x_i$  s image and  
 $y_i$  s (integer) label

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



cat	<b>3.2</b>	1.3	2.2
car	<b>5.1</b>	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

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

Given a dataset of examples

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

Where  $x_i$  s image and  
 $y_i$  s (integer) label

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

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

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



cat	<b>3.2</b>	1.3	2.2
car	<b>5.1</b>	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

### 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:

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

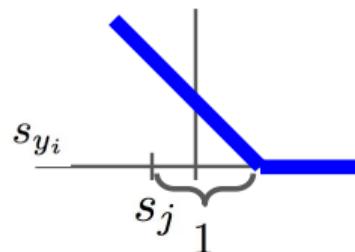
Suppose: 3 training examples, 3 classes.  
 With some  $W$  the scores  $f(x, W) = Wx$  are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

### Multiclass SVM loss:

“Hinge loss”



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

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

### 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:

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

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



cat	<b>3.2</b>	1.3	2.2
car	<b>5.1</b>	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>		

### 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:

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

$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	2.9	<b>0</b>	

## 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:

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

$$\begin{aligned}
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	2.9	0	<b>12.9</b>

### 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:

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

$$\begin{aligned} &= \max(0, 2.2 - (-3.1) + 1) \\ &\quad + \max(0, 2.5 - (-3.1) + 1) \\ &= \max(0, 6.3) + \max(0, 6.6) \\ &= 6.3 + 6.6 \\ &= 12.9 \end{aligned}$$

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	<b>0</b>	<b>12.9</b>

### 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:

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

$$\begin{aligned} L &= (2.9 + 0 + 12.9)/3 \\ &= \mathbf{5.27} \end{aligned}$$

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	<b>0</b>	<b>12.9</b>

### 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:

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

Q: What happens to loss if car scores change a bit?

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	0	<b>12.9</b>

### 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:

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

Q2: what is the min/max possible loss?

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	<b>0</b>	<b>12.9</b>

### 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:

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

Q3: At initialization  $W$  is small so all  $s \approx 0$ .  
What is the loss?

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	0	<b>12.9</b>

### 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:

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

Q4: What if the sum was over all classes?  
(including  $j = y_i$ )

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	<b>0</b>	<b>12.9</b>

### 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:

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

Q5: What if we used mean instead of sum?

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>	<b>0</b>	<b>12.9</b>

### 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:

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

**Q6: What if we used**

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

# 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?

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

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

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

**No!  $2W$  is also has  $L = 0!$**

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



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	2.9		<b>0</b>

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

**Before:**

$$\begin{aligned}
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

**With  $W$  twice as large:**

$$\begin{aligned}
 &= \max(0, 2.6 - 9.8 + 1) \\
 &\quad + \max(0, 4.0 - 9.8 + 1) \\
 &= \max(0, -6.2) + \max(0, -4.8) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

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

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

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

**No!  $2W$  is also has  $L = 0$ !**

**How do we choose between  $W$  and  $2W$ ?**

# Regularization

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

**Data loss:** Model predictions  
should match training data

# Regularization

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

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too well* on training data

# Regularization

$\lambda$  = regularization strength  
(hyperparameter)

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

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too well* on training data

# Regularization

$\lambda$  = regularization strength  
(hyperparameter)

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

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too well* on training data

## Simple examples

L2 regularization:  $R(W) = \sum_k \sum_l W_{k,l}^2$

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

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

# Regularization

$\lambda$  = regularization strength  
(hyperparameter)

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

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too well* on training data

## Simple examples

L2 regularization:  $R(W) = \sum_k \sum_l W_{k,l}^2$

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

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

## More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

# Regularization

$\lambda$  = regularization strength  
(hyperparameter)

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

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too well* on training data

Why regularize?

- Express preferences over weights
- Make the model *simple* so it works on test data
- Improve optimization by adding curvature

## Regularization: Expressing Preferences

$$x = [1, 1, 1, 1]$$

L2 Regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

# Regularization: Expressing Preferences

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

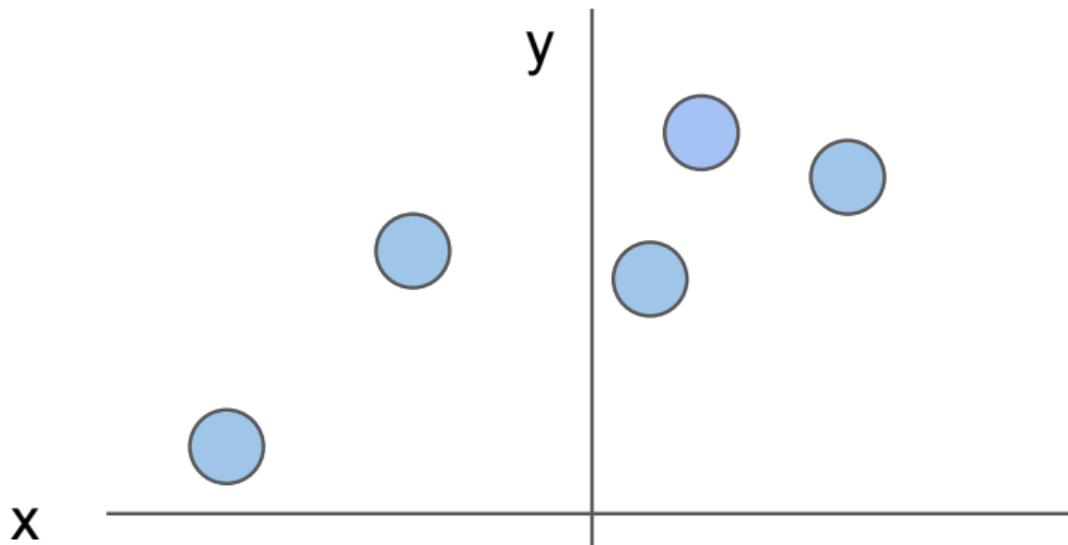
L2 Regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

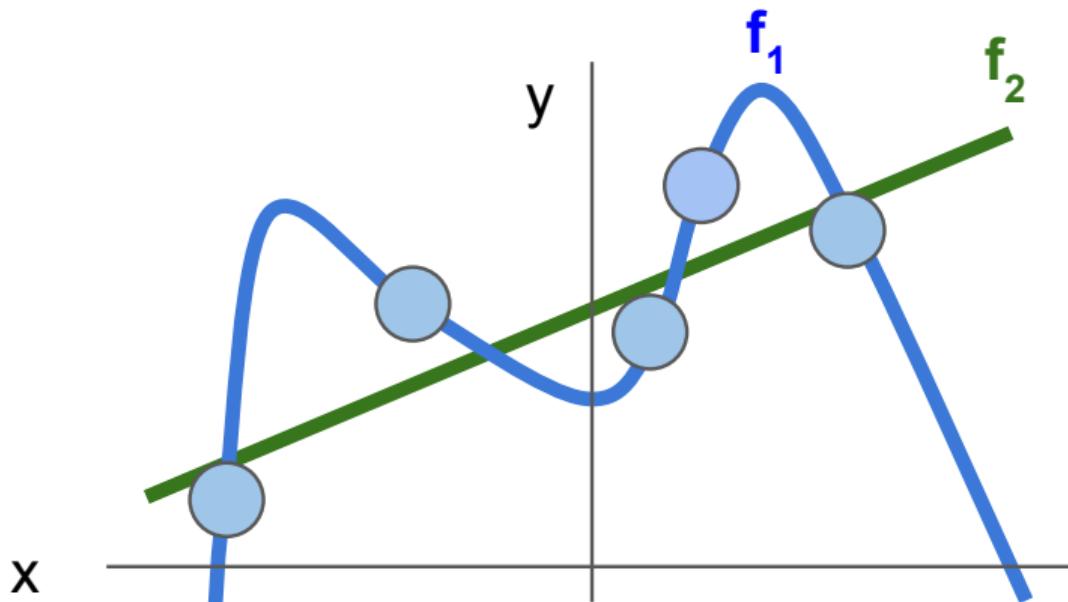
L2 regularization likes to  
“spread out” the weights

$$w_1^T x = w_2^T x = 1$$

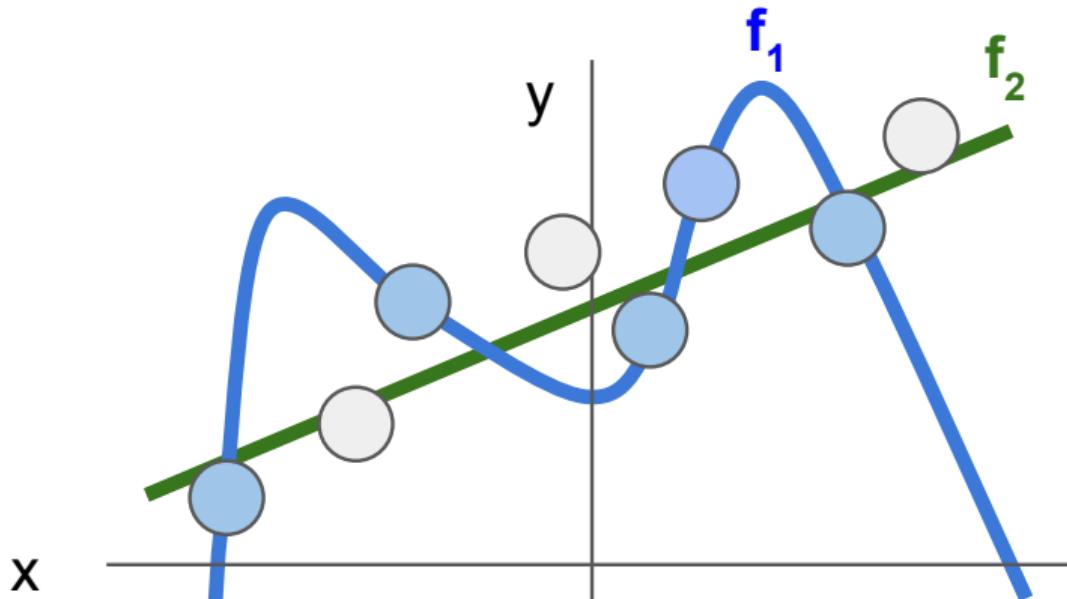
# Regularization: Prefer Simpler Models



# Regularization: Prefer Simpler Models



# Regularization: Prefer Simpler Models



Regularization pushes against fitting the data  
too well so we don't fit noise in the data

## Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

Probabilities  
must be  $\geq 0$

cat	<b>3.2</b>
car	5.1
frog	-1.7

$\xrightarrow{\text{exp}}$

<b>24.5</b>
164.0
0.18

unnormalized  
probabilities

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat	<b>3.2</b>
car	5.1
frog	-1.7

$\exp$

<b>24.5</b>
164.0
0.18

unnormalized  
probabilities

normalize

<b>0.13</b>
0.87
0.00

probabilities

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2
5.1
-1.7

exp

24.5
164.0
0.18

normalize

0.13
0.87
0.00

Unnormalized  
log-probabilities / logits

unnormalized  
probabilities

probabilities

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2
5.1
-1.7

exp

24.5
164.0
0.18

normalize

0.13
0.87
0.00

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

$$\rightarrow L_i = -\log(0.13) \\ = 2.04$$

Unnormalized  
log-probabilities / logits

unnormalized  
probabilities

probabilities

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2
5.1
-1.7

exp

24.5
164.0
0.18

normalize

0.13
0.87
0.00

Unnormalized  
log-probabilities / logits

unnormalized  
probabilities

probabilities

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

$$\rightarrow L_i = -\log(0.13) \\ = 2.04$$

**Maximum Likelihood Estimation**  
Choose weights to maximize the likelihood of the observed data  
(See CS 229 for details)

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2
5.1
-1.7

Unnormalized  
log-probabilities / logits

exp

24.5
164.0
0.18

unnormalized  
probabilities

normalize

0.13
0.87
0.00

probabilities

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

compare

1.00
0.00
0.00

Correct  
probs

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2
5.1
-1.7

Unnormalized  
log-probabilities / logits

exp

24.5
164.0
0.18

unnormalized  
probabilities

normalize

0.13
0.87
0.00

probabilities

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

compare

1.00

Kullback–Leibler  
divergence

$$D_{KL}(P||Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

0.00

0.00

Correct  
probs

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax Function

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2
5.1
-1.7

exp

24.5
164.0
0.18

normalize

0.13
0.87
0.00

compare

1.00

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P\|Q)$$

0.00

0.00

Unnormalized  
log-probabilities / logits

unnormalized  
probabilities

probabilities

Correct  
probs

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

Maximize probability of correct class

Putting it all together:

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

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

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

Maximize probability of correct class

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

Putting it all together:

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

cat	<b>3.2</b>
car	5.1
frog	-1.7

Q: What is the min/max possible loss  $L_i$ ?

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

Maximize probability of correct class

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

Putting it all together:

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

cat	<b>3.2</b>
car	5.1
frog	-1.7

Q: What is the min/max possible loss  $L_i$ ?  
A: min 0, max infinity

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

Maximize probability of correct class

Putting it all together:

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

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

cat	<b>3.2</b>
car	5.1
frog	-1.7

Q2: At initialization all  $s$  will be approximately equal; what is the loss?

# Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



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

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

Softmax  
Function

Maximize probability of correct class

Putting it all together:

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

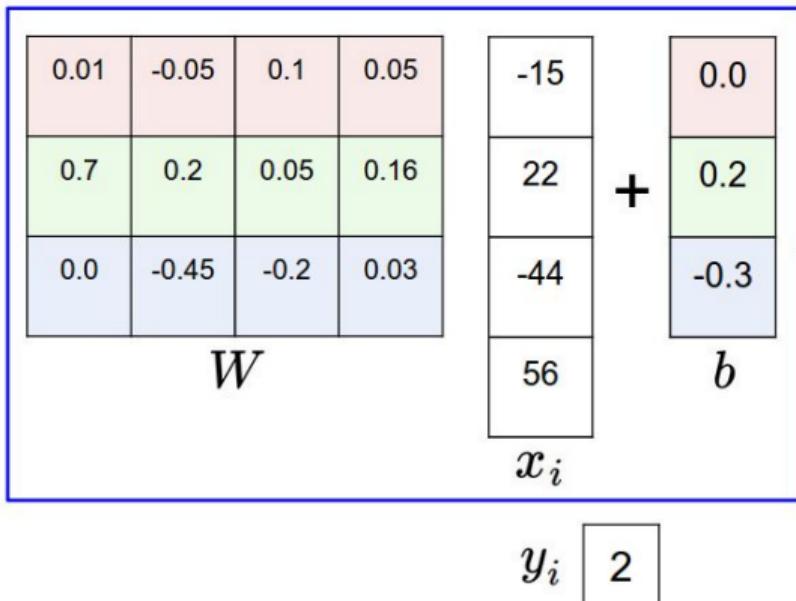
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

cat	<b>3.2</b>
car	5.1
frog	-1.7

Q2: At initialization all  $s$  will be approximately equal; what is the loss?  
A:  $\log(C)$ , eg  $\log(10) \approx 2.3$

# Softmax vs. SVM

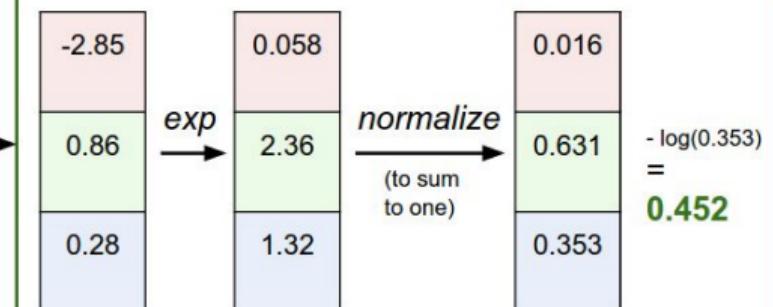
matrix multiply + bias offset



hinge loss (SVM)

$$\begin{aligned} & \max(0, -2.85 - 0.28 + 1) + \\ & \max(0, 0.86 - 0.28 + 1) \\ = & 1.58 \end{aligned}$$

cross-entropy loss (Softmax)



# Softmax vs. SVM

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

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

# Softmax vs. SVM

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

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

---

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and  $y_i = 0$

Q: Suppose I take a datapoint and I jiggle a bit (changing its score slightly). What happens to the loss in both cases?

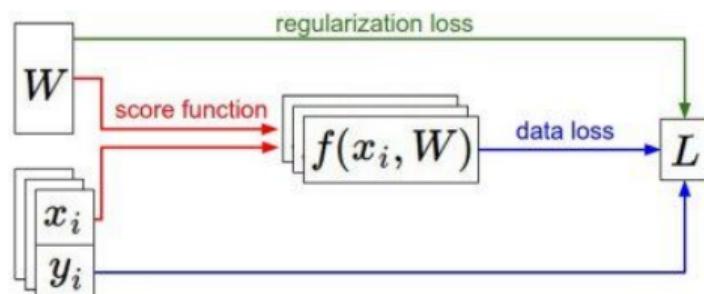
# Recap

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

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \quad \text{Softmax}$$

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

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



# Recap

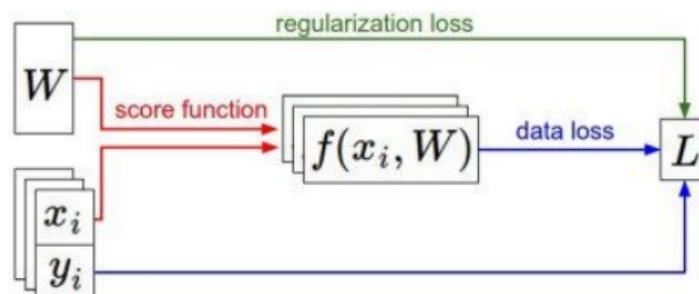
How do we find the best  $W$ ?

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

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \quad \text{Softmax}$$

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

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



# Optimization



Finding the best  $W$  is an optimization task:

$$W = \arg \min_w \frac{1}{N} \sum_{i=1}^N L_i(\mathbf{x}_i, \mathbf{y}_i, W) + \lambda R(W) \quad (1)$$

How can we find the minimum?

# Optimization



Finding the best  $W$  is an optimization task:

$$W = \arg \min_W \frac{1}{N} \sum_{i=1}^N L_i(\mathbf{x}_i, \mathbf{y}_i, W) + \lambda R(W) \quad (1)$$

How can we find the minimum?

We can use gradient descent!

$$W_{i,j}^{n+1} = W_{i,j}^n - \eta \frac{\partial L}{\partial w_i} \quad (2)$$

# Gradient



Partial derivative and gradient definition:

$$\frac{\partial f}{\partial x_i} = \lim_{h \rightarrow 0} \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i, \dots, x_n)}{h} \quad (3)$$

$$\text{grad}(f) = \nabla f = \nabla_{\mathbf{x}} f = \frac{\partial f}{\partial \mathbf{x}} = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right) \quad (4)$$

# Numerical gradient computation



$$\frac{\partial f}{\partial x_i}(\mathbf{x}) \approx \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i, \dots, x_n)}{h} \quad (5)$$

$$\approx \frac{f(x_1, x_2, \dots, x_i + h, \dots, x_n) - f(x_1, x_2, \dots, x_i - h, \dots, x_n)}{2h} \quad (6)$$

- Slow
- Approximate
- Easy to compute once  $f$  is implemented

# Analytical gradient



- Fast
- Accurate
- Requires more involved implementation

In practice we derive analytical gradient, but we check our implementation by using numerical approximations!

# Numerical gradient example



**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 gradient example



**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:**

[?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

# Numerical gradient example



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:

[-2.5,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

$$\frac{(1.25322 - 1.25347)}{0.0001} = -2.5$$

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

?,  
?,...]

# Numerical gradient example



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,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

# Numerical gradient example



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,  
?,  
?,  
?

$$\frac{(1.25353 - 1.25347)}{0.0001} = 0.6$$

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

?,...]

# Numerical gradient example



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,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

# Numerical gradient example



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,  
?,  
?

$$\frac{(1.25347 - 1.25347)}{0.0001} = 0$$

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

,...,]

# Numerical gradient example



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,  
,  
,

## Numeric Gradient

- Slow! Need to loop over all dimensions
- Approximate

,...,]

# Analytical gradient example



**current W:**

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

**loss 1.25347**

$dW = \dots$   
(some function  
data and  $W$ )

**gradient  $dW$ :**

[-2.5,  
0.6,  
0,  
0.2,  
0.7,  
-0.5,  
1.1,  
1.3,  
-2.1,...]



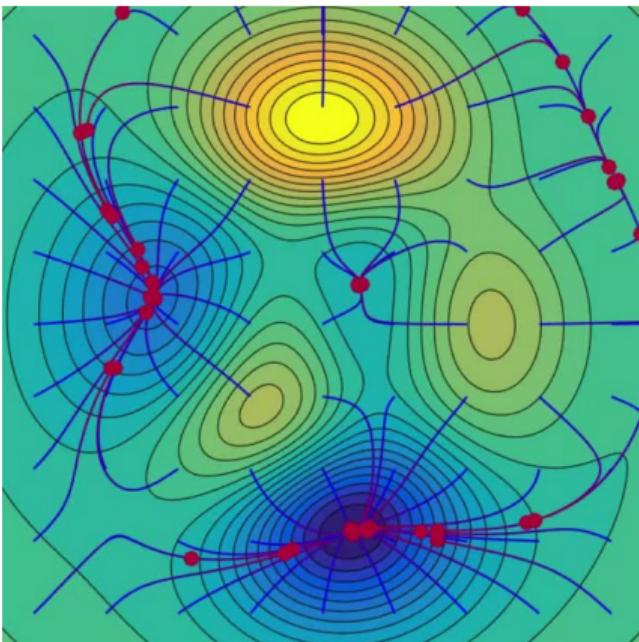
# Gradient descent



$$W_{i,j}^{n+1} = W_{i,j}^n - \eta \frac{\partial L}{\partial w_i} \quad (7)$$

```
W = initialize_weights()
for i in range(num_steps):
    grad_W = compute_grad(W, training_data)
    W = W - eta * grad_W
```

# Gradient Descent Animation



[https://en.wikipedia.org/wiki/File:Gradient\\_Descent\\_in\\_2D.webm](https://en.wikipedia.org/wiki/File:Gradient_Descent_in_2D.webm)

# Stochastic Gradient Descent



Full loss:

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(\mathbf{x}_i, \mathbf{y}_i, W) + \lambda R(W) \quad (8)$$

$$\nabla_w L(W) = \frac{1}{N} \sum_{i=1}^N \nabla_w L_i(\mathbf{x}_i, \mathbf{y}_i, W) + \lambda \nabla_w R(W) \quad (9)$$

- Expensive computation - both in time and space!
- We can use approximations!

# Stochastic Gradient Descent



Batch loss:

$$L(W) \approx \frac{1}{M} \sum_{i \in B} L_i(\mathbf{x}_i, \mathbf{y}_i, W) + \lambda R(W) \quad (10)$$

$$\nabla_W L(W) \approx \frac{1}{M} \sum_{i \in B} \nabla_W L_i(\mathbf{x}_i, \mathbf{y}_i, W) + \lambda \nabla_W R(W) \quad (11)$$

- We choose  $M = 16, 32, 64\dots$
- $B$  is a random subset from the training data with size  $M$
- Significantly faster than using all of the data!

```
W = initialize_weights()
for i in range(num_steps):
    batch_data = sample_data(training_data, batch_size)
    grad_W = compute_grad(W, batch_data)
    W = W - eta * grad_W
```

## SGD remarks



Provided that we have a magical computer that can compute vectorized operations such that computing gradient for one sample takes the same amount of memory and time as computing it for  $n$  samples. Would SGD be useless compared to standard GD with all samples?



Provided that we have a magical computer that can compute vectorized operations such that computing gradient for one sample takes the same amount of memory and time as computing it for  $n$  samples. Would SGD be useless compared to standard GD with all samples?

Depends on the problem, but generally SGD can outperform GD with all samples due to local optima and saddle points! E.g. points where  $\nabla_w L = 0$ .

## SGD remarks



Provided that we have a magical computer that can compute vectorized operations such that computing gradient for one sample takes the same amount of memory and time as computing it for  $n$  samples. Would SGD be useless compared to standard GD with all samples?

Depends on the problem, but generally SGD can outperform GD with all samples due to local optima and saddle points! E.g. points where  $\nabla_w L = 0$ .

In future lectures you will learn about some improvements to the SGD algorithm which make convergence faster. Nevertheless even pure SGD is still often used, especially if you have budget to tune the learning rate schedule!