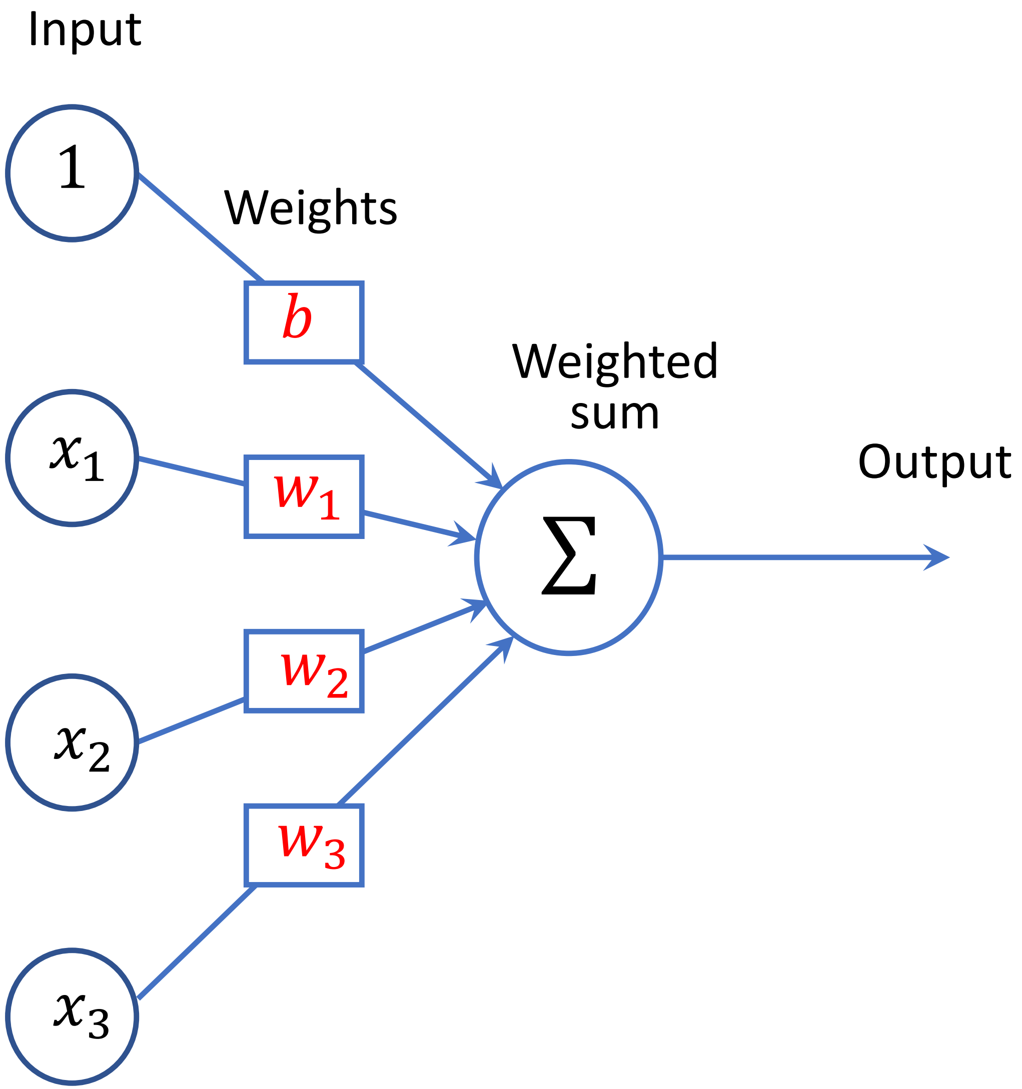
Previous Lecture: Recap

- Course Information
- Introduction to Deep Learning
- Deep Learning Basics
  - Linear Regression
  - Loss Function
  - Gradient Descent
  - Regularization

Previous Lecture: Recap



Previous Lecture: Recap

UNIVERSITY OF OULU

Given training data: $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$

Linear Hypothesis:

$$f(\boldsymbol{x}) = b + \boldsymbol{w}^T\boldsymbol{x}$$

Loss Function

$$L(\boldsymbol{w}, b) = \frac{1}{2N}\sum_{i=1}^{N}\left(y - \left(b + \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)\right)^2$$

A set of functions

Model

$f_1, f_2 \dots$

Goodness of function f

Training Data

Pick the "Best" Function

How?

Gradient Descent

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w}, b}{\operatorname{argmin}} L(\boldsymbol{w}, b)$$

$$= \underset{\boldsymbol{w}, b}{\operatorname{argmin}} \frac{1}{2N}\sum_{i=1}^{N}\left(y - \left(b + \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)\right)^2$$

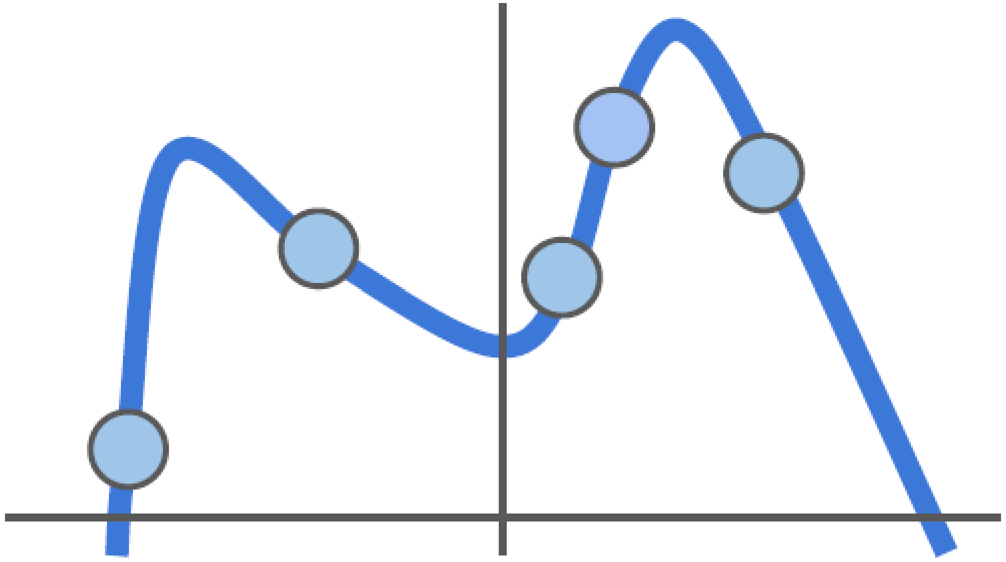
Regularization

$$f(\boldsymbol{x}) = b + \boldsymbol{w}^T\boldsymbol{x}$$

$$L(\boldsymbol{w}, b) = \underbrace{\sum_i \left(y^{(i)} - \left(b + \boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)\right)^2}_{} + \underbrace{\lambda \sum_j w_j^2}_{}$$

Data loss: Model predictions should match training data.

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



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

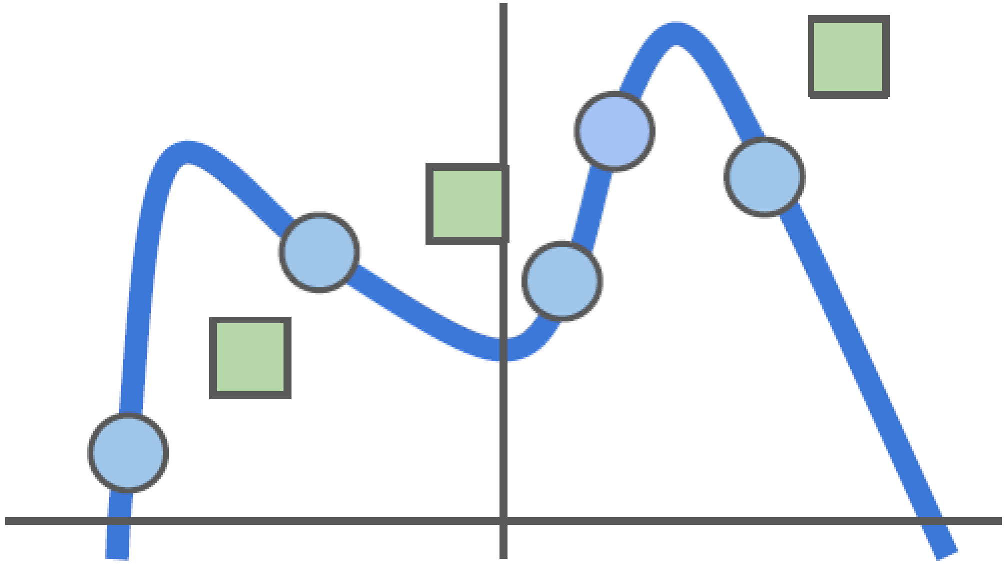
Regularization

$$f(\boldsymbol{x}) = b + \boldsymbol{w}^T\boldsymbol{x}$$

$$L(\boldsymbol{w}, b) = \underbrace{\sum_i \left(y^{(i)} - (b + \boldsymbol{w}^T\boldsymbol{x}^{(i)})\right)^2}_{} + \lambda \underbrace{\sum_j w_j^2}_{}$$

Data loss: Model predictions should match training data.

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



Occam's Razor:

"Among competing hypotheses, ***the simplest is the best****"*

William of Ockham, 1285–1347

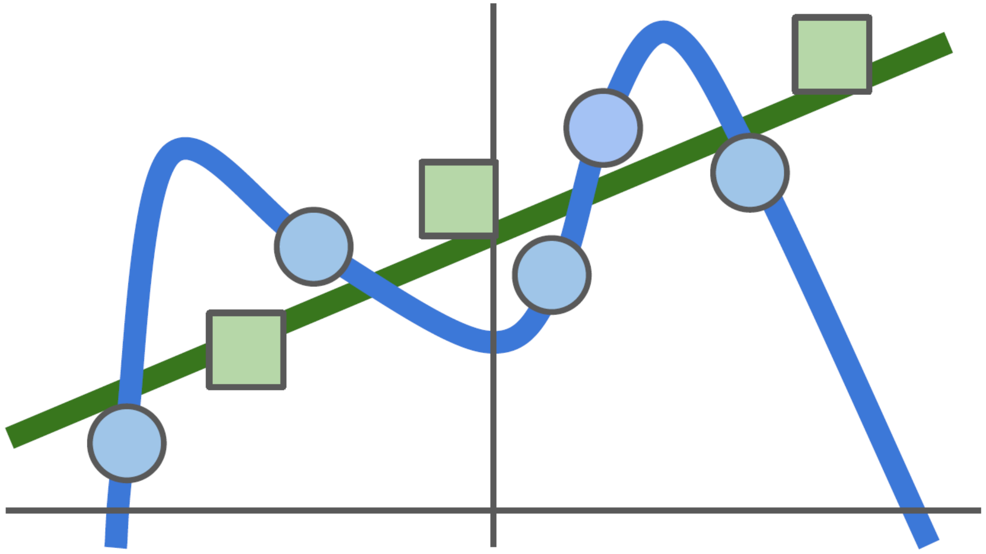
Regularization

$$f(\boldsymbol{x}) = b + \boldsymbol{w}^T\boldsymbol{x}$$

$$L(\boldsymbol{w}, b) = \underbrace{\sum_i \left(y^{(i)} - (b + \boldsymbol{w}^T\boldsymbol{x}^{(i)})\right)^2}_{} + \lambda \underbrace{\sum_j w_j^2}_{}$$

Data loss: Model predictions should match training data.

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



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

Today: Lecture 2

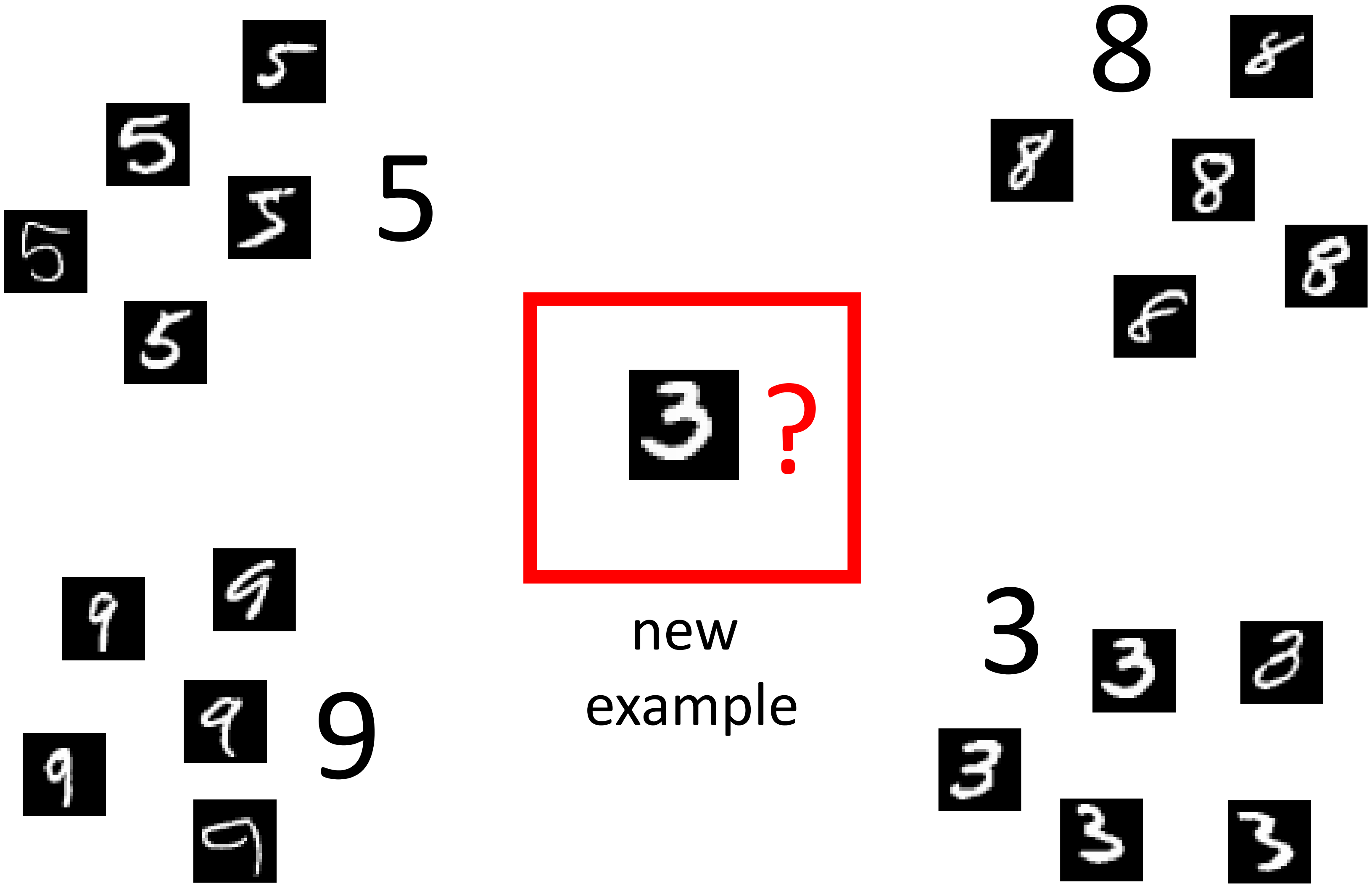
Linear Classifier

More about Lost Functions

Tips for Gradient Descent

Logistic Regression (Selflearn)

Supervised Classification



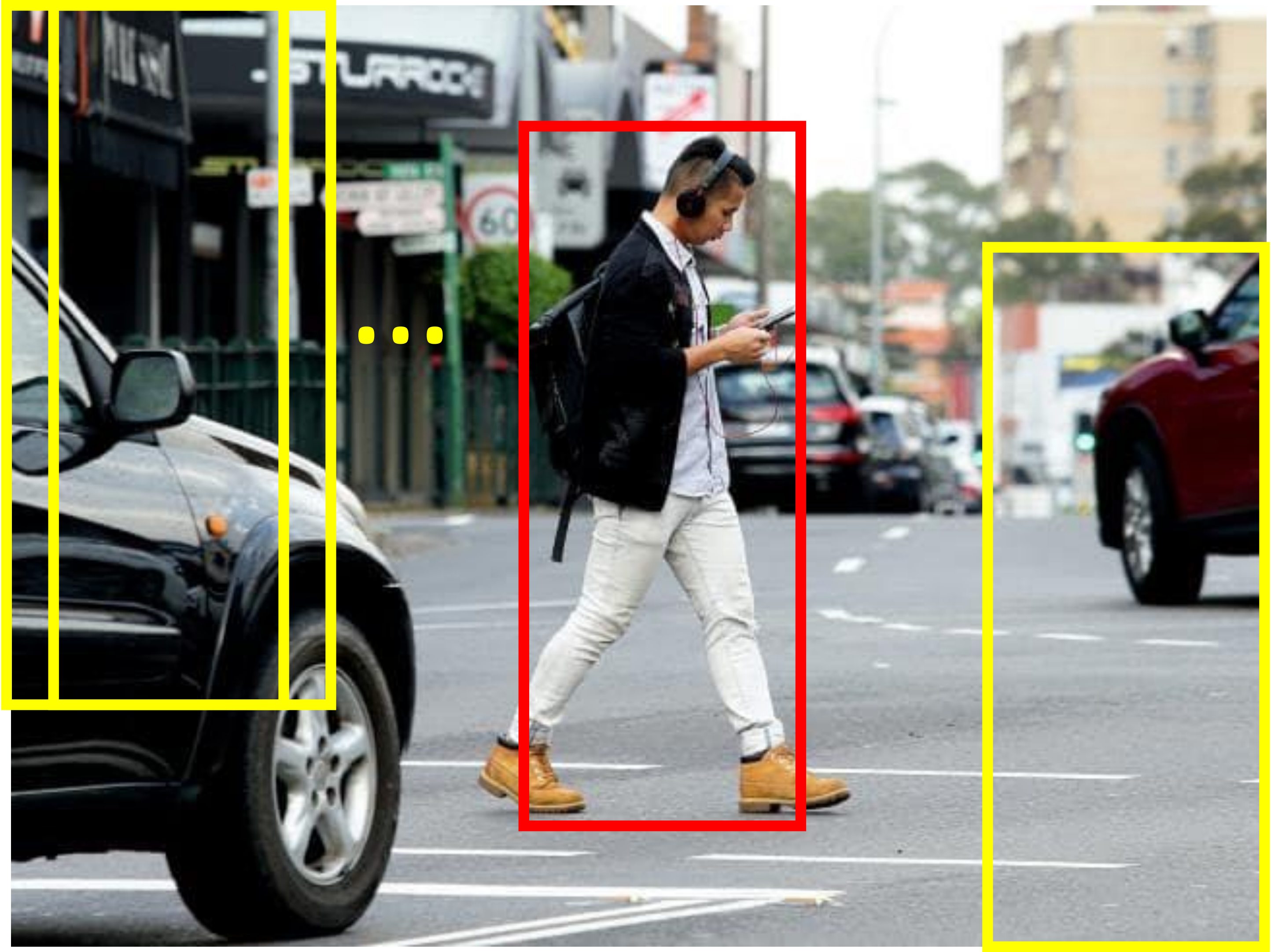
Classification: object classification

ImageNet Challenge: classification of 1000 object category

Classification: face recognition

Pedestrian Detection

Pedestrian Detection



Classification

- Email: Spam / Not Spam?
- Pedestrian Detection: Pedestrian / Not Pedestrian?
- Tumor: Malignant / Benign ?

$y \in \{0,1\}$

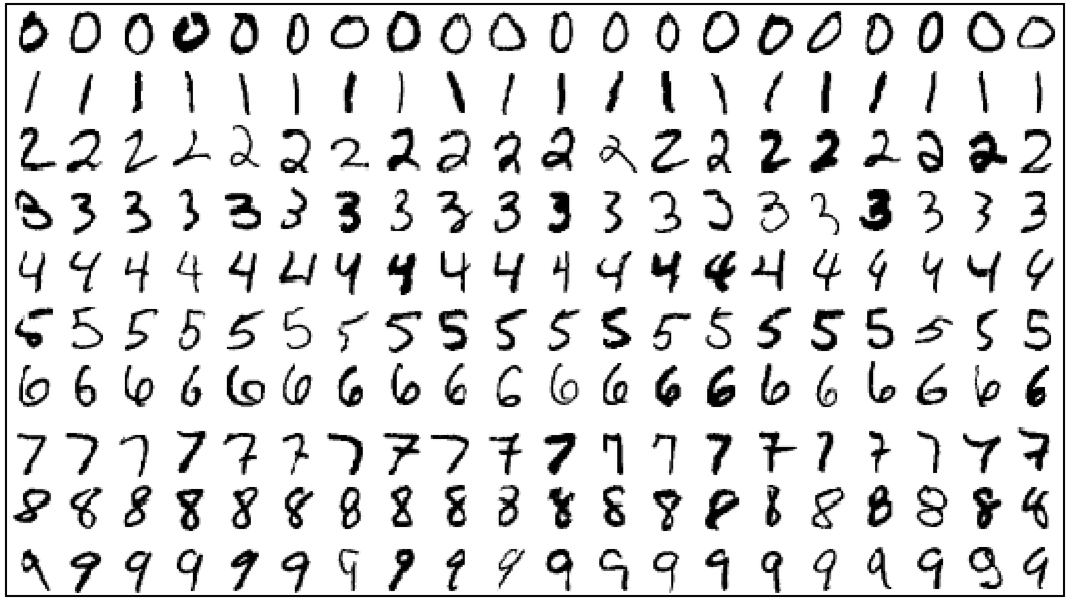
0: "Negative Class" (*e.g.* Not Pedestrian)
1: "Positive Class" (*e.g.* Pedestrian)

Values 0 and 1 are somewhat arbitrary.

$y \in \{-1,1\}$

0: "Negative Class" (*e.g.* Not Pedestrian)
1: "Positive Class" (*e.g.* Pedestrian)

Linear Classifier



Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

$\boldsymbol{x}^{(i)}$: an image;

$y^{(i)}$: the label (integer);

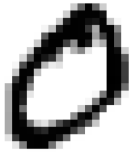
e.g. $y^{(i)} \in \{1, 2, \ldots, 10\}$

$\mathbb{R}^{10\times 1024}$

$\mathbb{R}^{10\times 1}$ $\mathbb{R}^{1024\times 1}$

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

image



$$\longrightarrow f(\boldsymbol{x}, \boldsymbol{W}) \longrightarrow$$

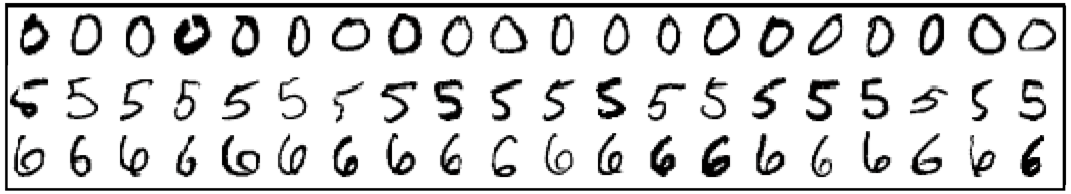
10 number
giving class scores

parameters
or weights

A array of 32 × 32 values
(1024 numbers in total)

Linear Classifier

UNIVERSITY OF OULU



Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

$\boldsymbol{x}^{(i)}$: an image;

$y^{(i)}$: the label (integer);

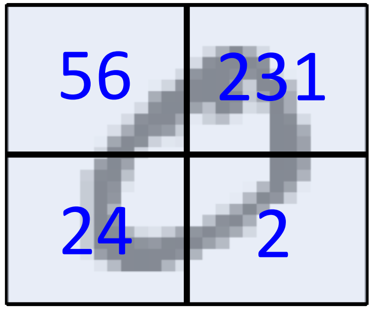
e.g. $y^{(i)} \in \{1,2,3\}$

"zero" score

"five" score

"six" score

Rearrange pixel values into a vector

$f\Bigg($ 

$, \boldsymbol{W}\Bigg)$

$$= \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0.0 & 0.25 & 0.2 & -0.3 \end{bmatrix} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix}$$

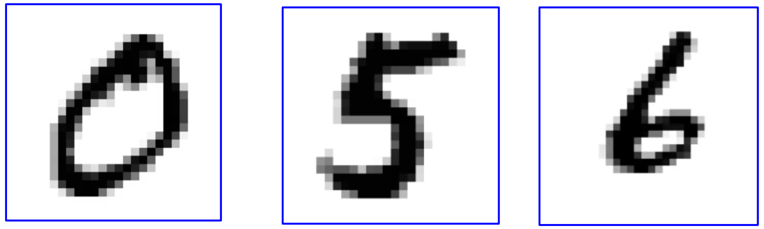
$\boldsymbol{W}$

Example with an image of 4 pixels, and 3 classes

image

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores $f(\boldsymbol{x},\boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:



"zero"	**3.2**	1.3	2.2
"five"	5.1	**4.9**	2.5
"six"	−1.7	2.0	**−3.1**

Given a dataset of examples:

$$\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$$

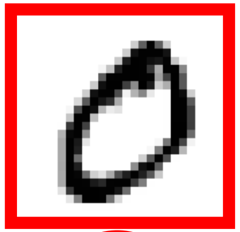
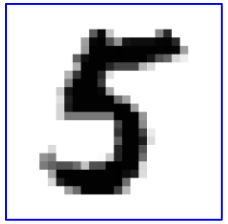
$\boldsymbol{x}^{(i)}$: an image;
$y^{(i)}$: the label (integer);
e.g. $y^{(i)} \in \{1,2,3\}$

A **loss function** tells how good our current model is.
Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N}\sum_{i} l^{(i)}\left(f\left(\boldsymbol{x}^{(i)}, \boldsymbol{W}\right), y^{(i)}\right)$$

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores
$f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:


		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

Multiclass SVM loss:

Given an example $(\boldsymbol{x}^{(i)}, y^{(i)})$ (omit superscript (i))

$\boldsymbol{x}^{(i)}$: an image;

$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \epsilon \{1,2,3\}$

and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$.

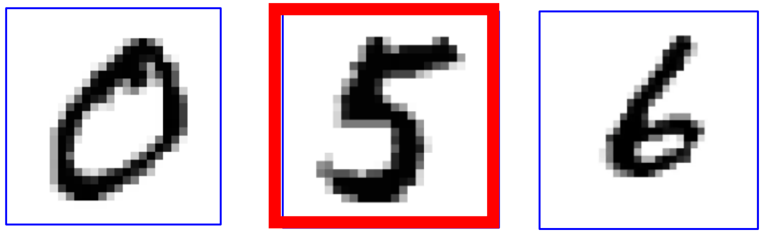
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

$$l^{(1)} = max\left(0, s_2^{(1)} - s_1^{(1)} + 1\right) + max\left(0, s_3^{(1)} - s_1^{(1)} + 1\right)$$
$$= max(0, 5.1 - 3.2 + 1) + max(0, -1.7 - 3.2 + 1)$$
$$= max(0, 2.9) + max(0, -3.9) = 2.9 + 0 = 2.9$$

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores
$f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:



		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

$l^{(1)} = 2.9$

Multiclass SVM loss:

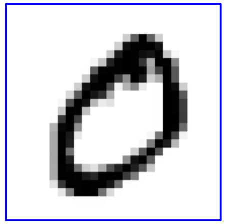
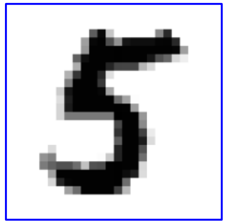
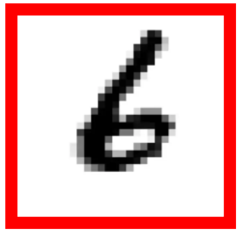
Given an example $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$
(omit superscript (i))
$\boldsymbol{x}^{(i)}$: an image;
$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \in \{1,2,3\}$
and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f\left(\boldsymbol{x}^{(i)}, \boldsymbol{W}\right)$.
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

$$l^{(2)} = max\left(0, s_1^{(2)} - s_2^{(2)} + 1\right) + max\left(0, s_3^{(2)} - s_2^{(2)} + 1\right)$$
$$= max(0, 1.3 - 4.9 + 1) + max(0, 2.0 - 4.9 + 1)$$
$$= max(0, -2.6) + max(0, -1.9) = 0 + 0 = 0$$

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores
$f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:

		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
				
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

$l^{(1)} = 2.9$

$l^{(2)} = 0$

Multiclass SVM loss:

Given an example $(\boldsymbol{x}^{(i)}, y^{(i)})$
(omit superscript (i))
$\boldsymbol{x}^{(i)}$: an image;
$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \in \{1,2,3\}$
and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$.
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

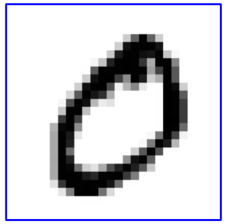
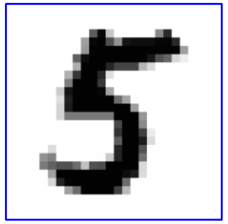
$$l^{(3)} = max\left(0, s_1^{(3)} - s_3^{(3)} + 1\right) + max\left(0, s_2^{(3)} - s_3^{(3)} + 1\right)$$

$$= max(0, 2.2 + 3.1 + 1) + max(0, 2.5 + 3.1 + 1)$$

$$= max(0, 6.3) + max(0, 6.6) \qquad = 6.3 + 6.6 = 12.9$$

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores $f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:

		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
				
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

$l^{(1)} = 2.9$

$l^{(2)} = 0$

$l^{(3)} = 13.9$

Multiclass SVM loss:

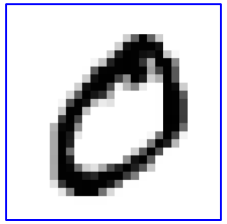
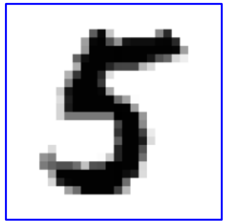
Given an example $(\boldsymbol{x}^{(i)}, y^{(i)})$
(omit superscript (i))
$\boldsymbol{x}^{(i)}$: an image;
$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \in \{1,2,3\}$
and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$.
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

$$L = \frac{l^{(1)} + l^{(2)} + l^{(3)}}{3} = \frac{2.9+0+12.9}{3} = 5.27$$

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores
$f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:

		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
				
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

Multiclass SVM loss:

Given an example $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$ (omit superscript (i))

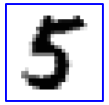
$\boldsymbol{x}^{(i)}$: an image;

$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \in \{1,2,3\}$

and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f\left(\boldsymbol{x}^{(i)}, \boldsymbol{W}\right)$.

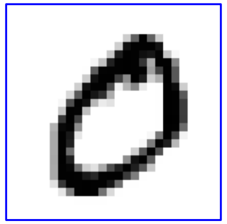
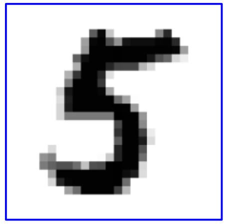
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

Q1: What happens to loss if  scores change a bit?

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores
$f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:

		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
				
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

Multiclass SVM loss:

Given an example $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)$ (omit superscript (i))

$\boldsymbol{x}^{(i)}$: an image;

$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \epsilon\{1,2,3\}$

and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f\left(\boldsymbol{x}^{(i)}, \boldsymbol{W}\right)$.

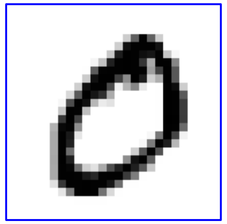
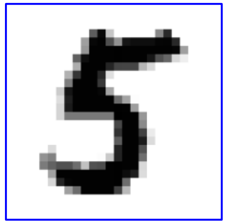
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

Q2: What is the min/max values of the loss?

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores $f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:

		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
				
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

Multiclass SVM loss:

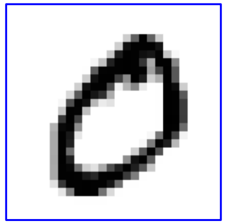
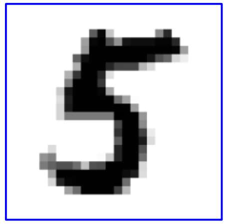
Given an example $(\boldsymbol{x}^{(i)}, y^{(i)})$
(omit superscript (i))
$\boldsymbol{x}^{(i)}$: an image;
$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \in \{1,2,3\}$
and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$.
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

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

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores $f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:

		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
				
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

Multiclass SVM loss:

Given an example $(\boldsymbol{x}^{(i)}, y^{(i)})$ (omit superscript (i))

$\boldsymbol{x}^{(i)}$: an image;

$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \epsilon\{1,2,3\}$

and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$.

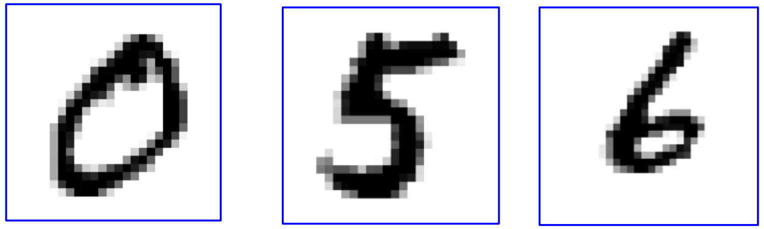
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

Q5: What if we use mean instead of sum?

Linear Classifier

Suppose: 3 training examples, 3classes.
With some $\boldsymbol{W}$ the scores $f(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ are:



		$\boldsymbol{x}^{(1)}$	$\boldsymbol{x}^{(2)}$	$\boldsymbol{x}^{(3)}$
$s_1^{(i)}$	"zero"	**3.2**	1.3	2.2
$s_2^{(i)}$	"five"	5.1	**4.9**	2.5
$s_3^{(i)}$	"six"	−1.7	2.0	**−3.1**

Multiclass SVM loss:

Given an example $(\boldsymbol{x}^{(i)}, y^{(i)})$ (omit superscript (i))
$\boldsymbol{x}^{(i)}$: an image;
$y^{(i)}$: the label (integer); *e.g.* $y^{(i)} \in \{1,2,3\}$
and using the shorthand for the scores vector: $\boldsymbol{s}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$.
The SVM loss has the form:

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

Q: What if we use $l^{(i)} = \sum_{k \neq y^{(i)}} (\max(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + +1))^2$

Regularization

$$s^{(i)} = f(x^{(i)}, W)$$

$$L(\boldsymbol{w}, b) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k\neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right) + \lambda R(\boldsymbol{W})$$

Data loss: Model predictions should match training data.

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

In common use:

L2 regularization $R(\boldsymbol{W}) = \Sigma_k \Sigma_n W_{kn}^2$

L1 regularization $R(\boldsymbol{W}) = \Sigma_k \Sigma_n |W_{kn}|$

Elastic net (L1 + L2) $R(\boldsymbol{W}) = \Sigma_k \Sigma_n (\beta W_{kn}^2 + |W_{kn}|)$

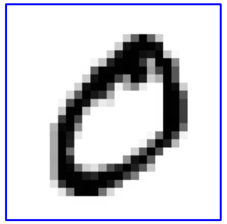
Max norm regularization (might see later)

Dropout (will see later)

Fancier: Batch normalization, stochastic depth

Softmax Classifier (Multinomial Logistic Regression)

Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$



scores = unnormalized log probabilities of the classes.

$$P(Y = k | X = \boldsymbol{x}^{(i)}) = \frac{\exp(z_k^{(i)})}{\sum_j \exp(z_j^{(i)})} \quad \text{where} \quad \boldsymbol{z}^{(i)} = f(\boldsymbol{x}^{(i)}, \boldsymbol{W})$$

"zero"	**3.2**
"five"	5.1
"six"	−1.7

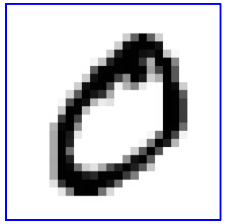
Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

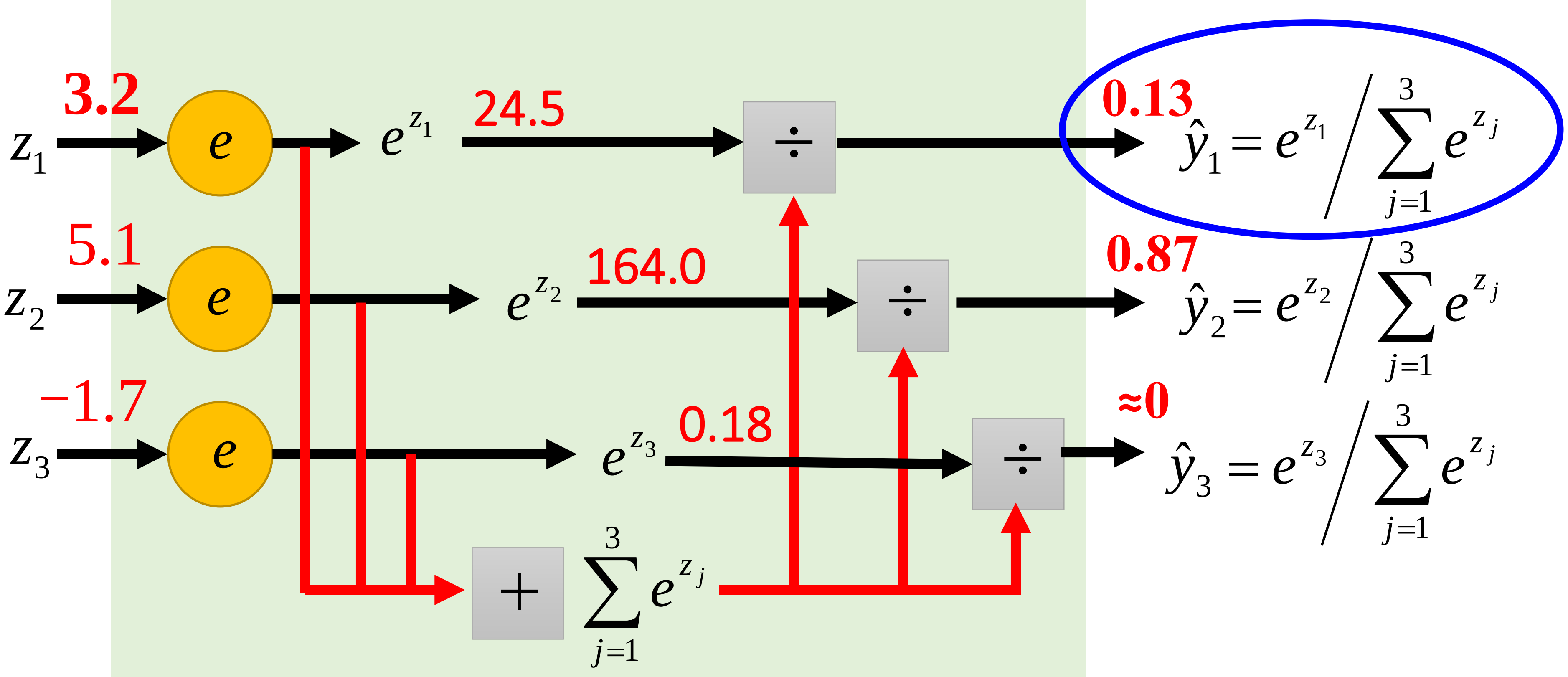
$$l^{(i)} = -\log P(Y = y^{(i)} | X = \boldsymbol{x}^{(i)})$$

$$= -\log\left(\frac{\exp(z_k^{(i)})}{\sum_j \exp(z_j^{(i)})}\right)$$

Softmax Classifier (Multinomial Logistic Regression)

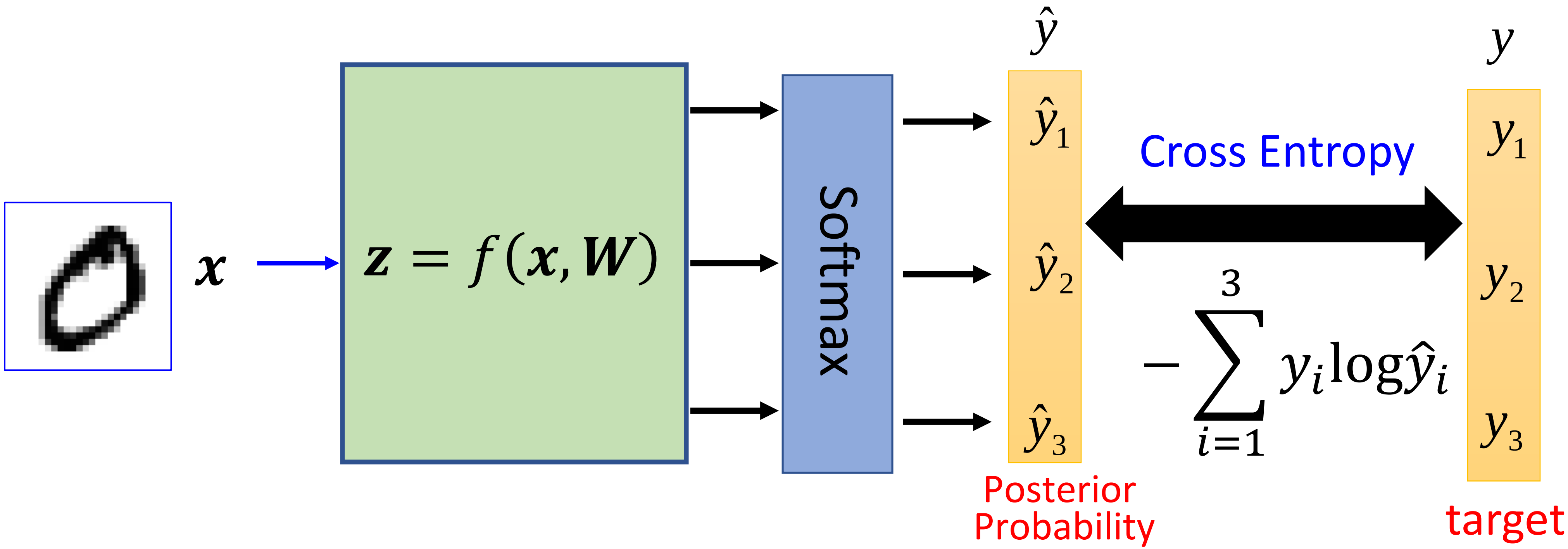
$$l^{(i)} = -\log\left(\exp\left(z^{(i)}_{y^{(i)}}\right) / \sum_j \exp\left(z^{(i)}_j\right)\right)$$





$$l^{(i)} = -\log(0.13) = 0.89$$

Softmax Classifier (Multinomial Logistic Regression)



If $x \in$ class 1

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$-\log \hat{y}_1$$

If $x \in$ class 2

$$y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$-\log \hat{y}_2$$

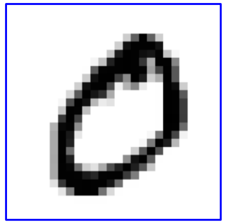
If $x \in$ class 3

$$y = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

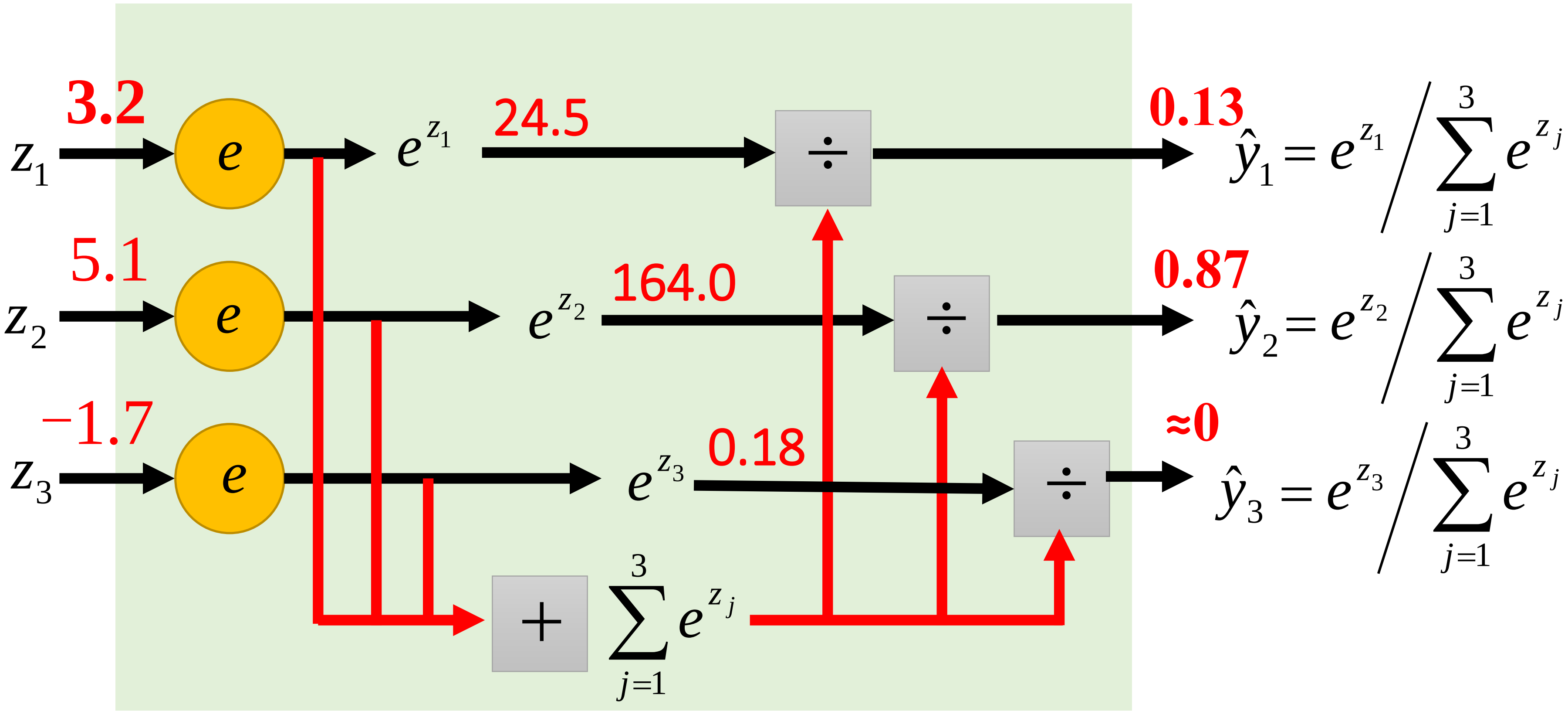
$$-\log \hat{y}_3$$

Softmax Classifier (Multinomial Logistic Regression)

$$l^{(i)} = -\log\left(\exp\left(z_{y^{(i)}}^{(i)}\right) / \sum_j \exp\left(z_j^{(i)}\right)\right)$$



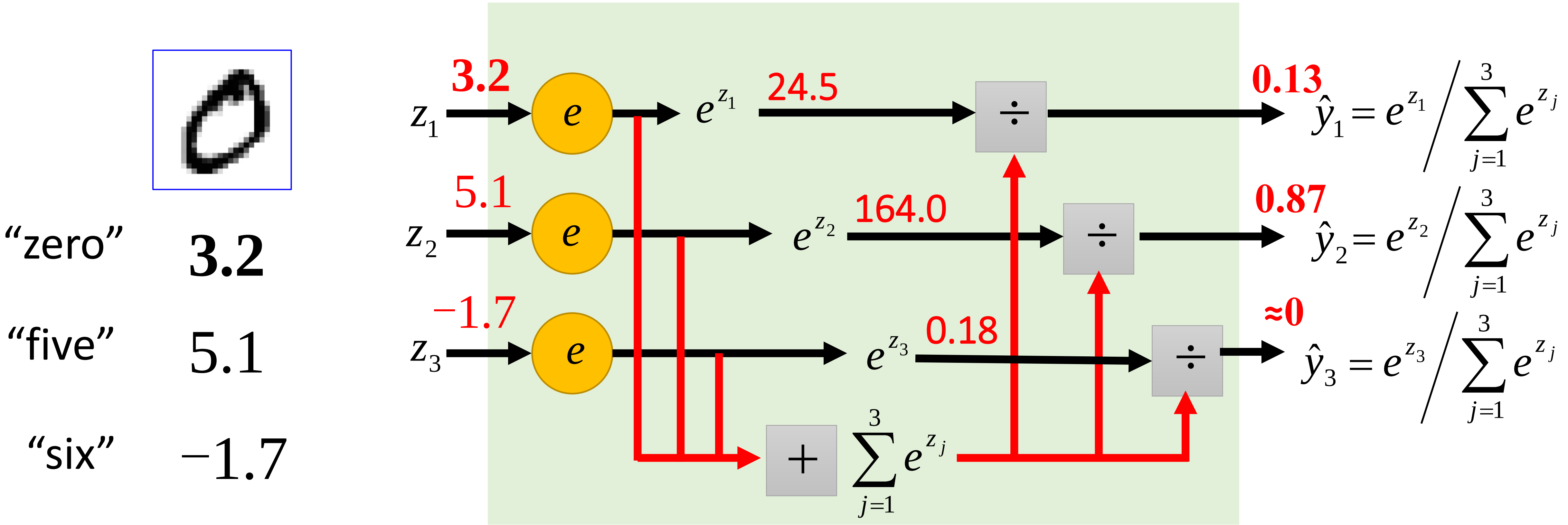
"zero"	**3.2**
"five"	5.1
"six"	−1.7



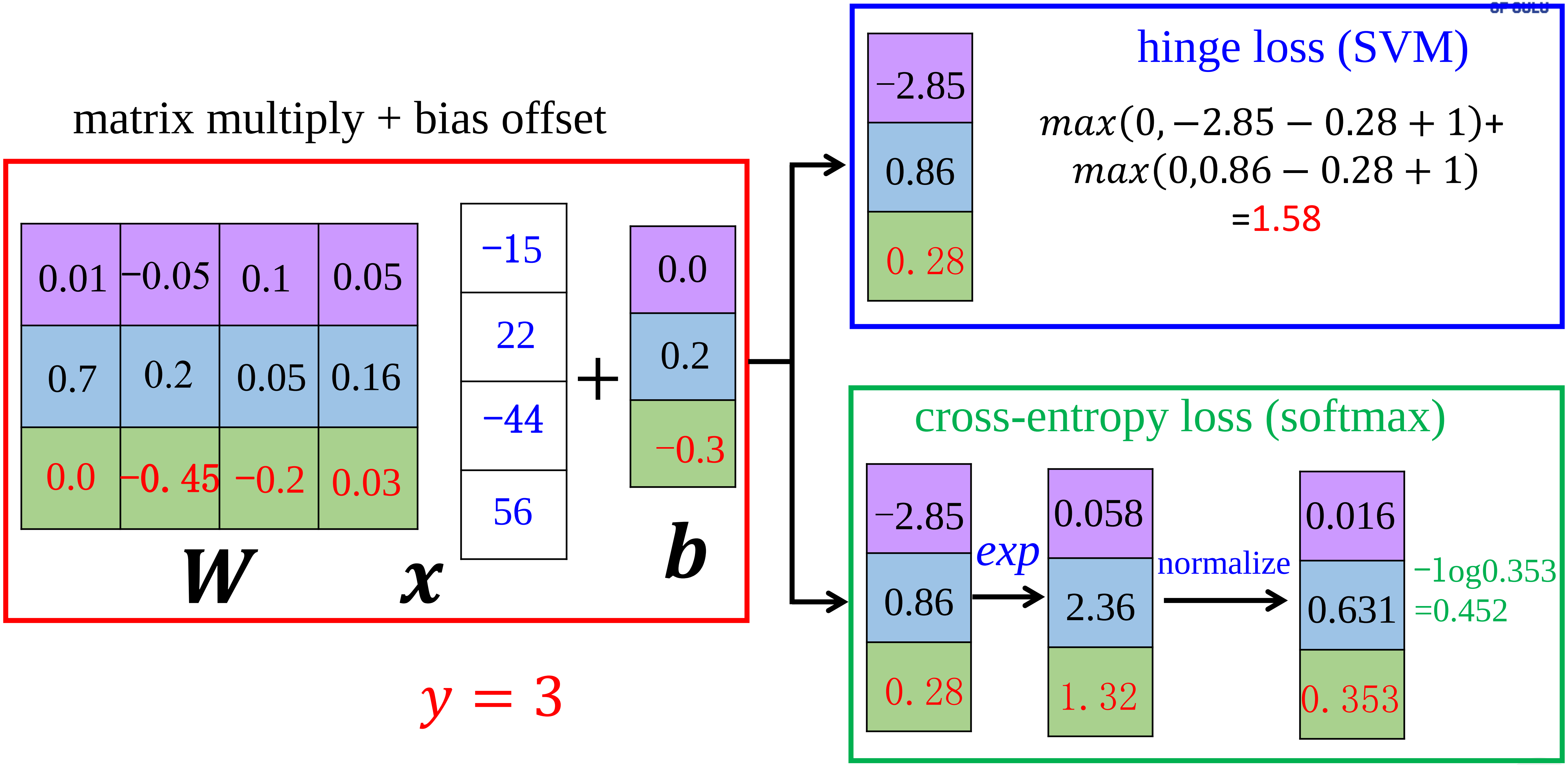
Q1: What is the min/max possible loss $l^{(i)}$?

Softmax Classifier (Multinomial Logistic Regression)

$$l^{(i)} = -\log\left(\exp\left(z^{(i)}_{y^{(i)}}\right) / \sum_j \exp\left(z^{(i)}_j\right)\right)$$



Q2: Usually at initialization $\boldsymbol{W}$ is small so all $z \approx 0$.
What is the loss?


matrix multiply + bias offset
0.01
−0.05
0.1
0.05
0.7
0.2
0.05
0.16
0.0
−0.45
−0.2
0.03
W
−15
22
−44
56
x
+
0.0
0.2
−0.3
b
y = 3
hinge loss (SVM)
−2.85
0.86
0.28
max(0, −2.85 − 0.28 + 1)+
max(0,0.86 − 0.28 + 1)
=1.58
cross-entropy loss (softmax)
−2.85
0.86
0.28
exp
0.058
2.36
1.32
normalize
0.016
0.631
0.353
−log0.353
=0.452

In Summary

Given training data: $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

UNIVERSITY OF OULU

Loss Functions

$$l^{(i)} = -\log\left(\exp\left(z_k^{(i)}\right) / \sum_j \exp(z_j^{(i)})\right)$$ softmax

$$l^{(i)} = \sum_{k \neq y^{(i)}} max\left(0, s_k^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$ hinge loss

$$L = \frac{1}{N}\sum_{i=1}^{N} l^{(i)} + \lambda R(\boldsymbol{W})$$

$e.g.\ \ f(\boldsymbol{x}) = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{x}$

A set of functions

Model

$f_1, f_2 \dots$

Goodness of function f

Training Data

Pick the "Best" Function

How?

Gradient Descent

Minimize loss function $L(\boldsymbol{W})$ $\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} L(\boldsymbol{W})$

Tips for Gradient Descent

Review: Gradient Descent

In step 3, we have to solve the following optimization problem:

$$\theta^* = \arg\min_{\theta} L(\theta)$$

L : loss function θ: parameters

Suppose that θ has two variables $\{\theta_1, \theta_2\}$

Randomly start at $\theta^{(0)} = \begin{bmatrix} \theta_1^{(0)} \\ \theta_2^{(0)} \end{bmatrix}$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta_1)/\partial \theta_1 \\ \partial L(\theta_2)/\partial \theta_2 \end{bmatrix}$$

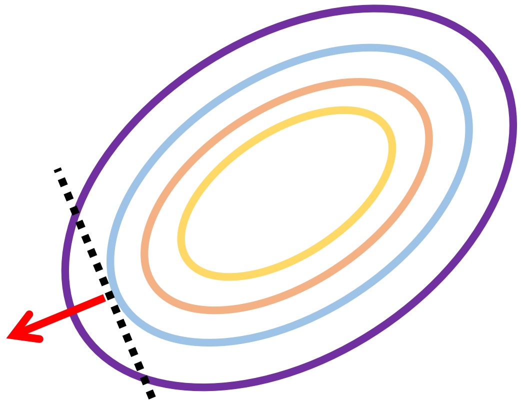
$$\begin{bmatrix} \theta_1^{(1)} \\ \theta_2^{(1)} \end{bmatrix} = \begin{bmatrix} \theta_1^{(0)} \\ \theta_2^{(0)} \end{bmatrix} - \eta \begin{bmatrix} \partial L\left(\theta_1^{(0)}\right)/\partial \theta_1 \\ \partial L\left(\theta_2^{(0)}\right)/\partial \theta_2 \end{bmatrix} \Rightarrow \theta^{(1)} = \theta^{(0)} - \eta \nabla L\left(\theta^{(0)}\right)$$

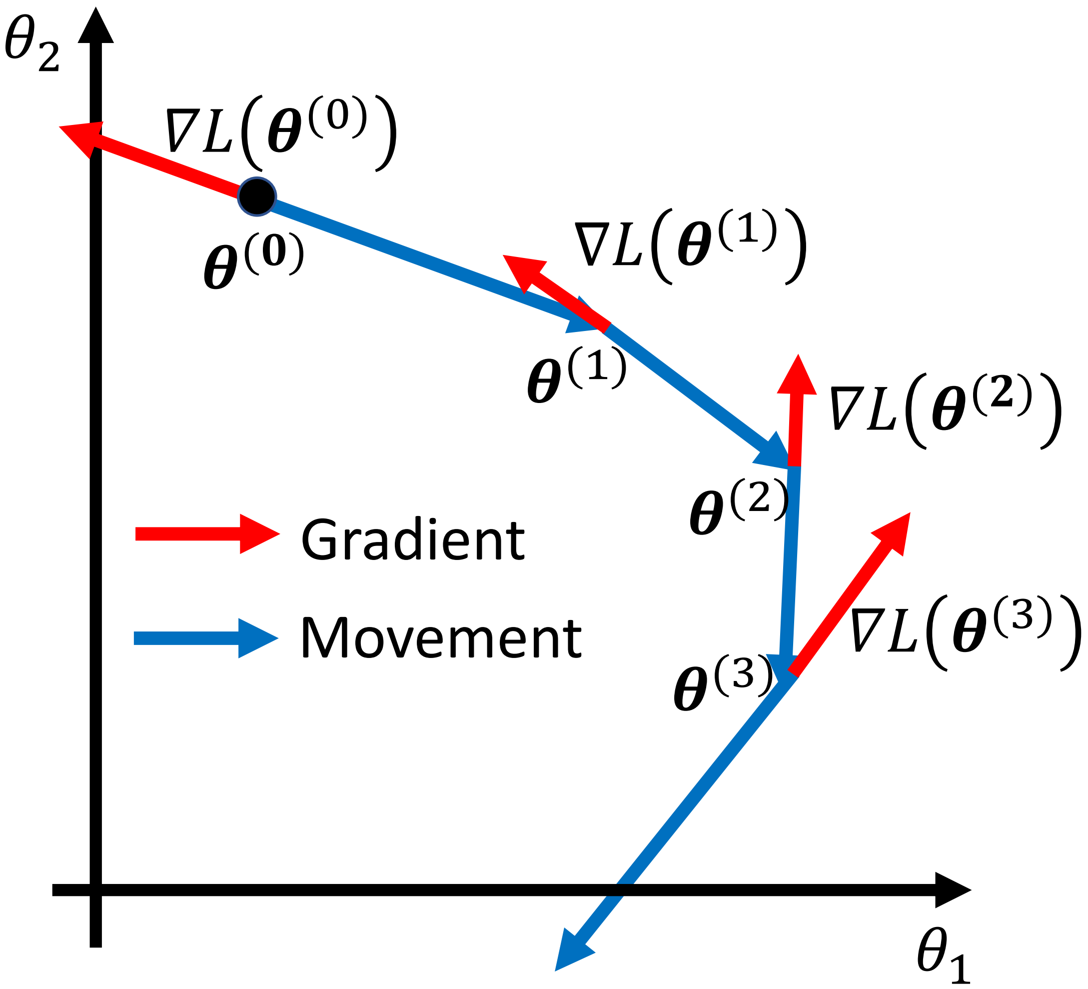
$$\begin{bmatrix} \theta_1^{(2)} \\ \theta_2^{(2)} \end{bmatrix} = \begin{bmatrix} \theta_1^{(1)} \\ \theta_2^{(1)} \end{bmatrix} - \eta \begin{bmatrix} \partial L\left(\theta_1^{(1)}\right)/\partial \theta_1 \\ \partial L\left(\theta_2^{(1)}\right)/\partial \theta_2 \end{bmatrix} \Rightarrow \theta^{(2)} = \theta^{(1)} - \eta \nabla L\left(\theta^{(1)}\right)$$

……

Review: Gradient Descent

Gradient: the normal direction of the contour of loss function





Start at position $\boldsymbol{\theta}^{(0)}$

Compute gradient at $\boldsymbol{\theta}^{(0)}$

Move to $\boldsymbol{\theta}^{(1)} = \boldsymbol{\theta}^{(0)} - \eta\nabla L(\boldsymbol{\theta}^{(0)})$

Compute gradient at $\boldsymbol{\theta}^{(1)}$

Move to $\boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} - \eta\nabla L(\boldsymbol{\theta}^{(1)})$

$\vdots$

Gradient Descent

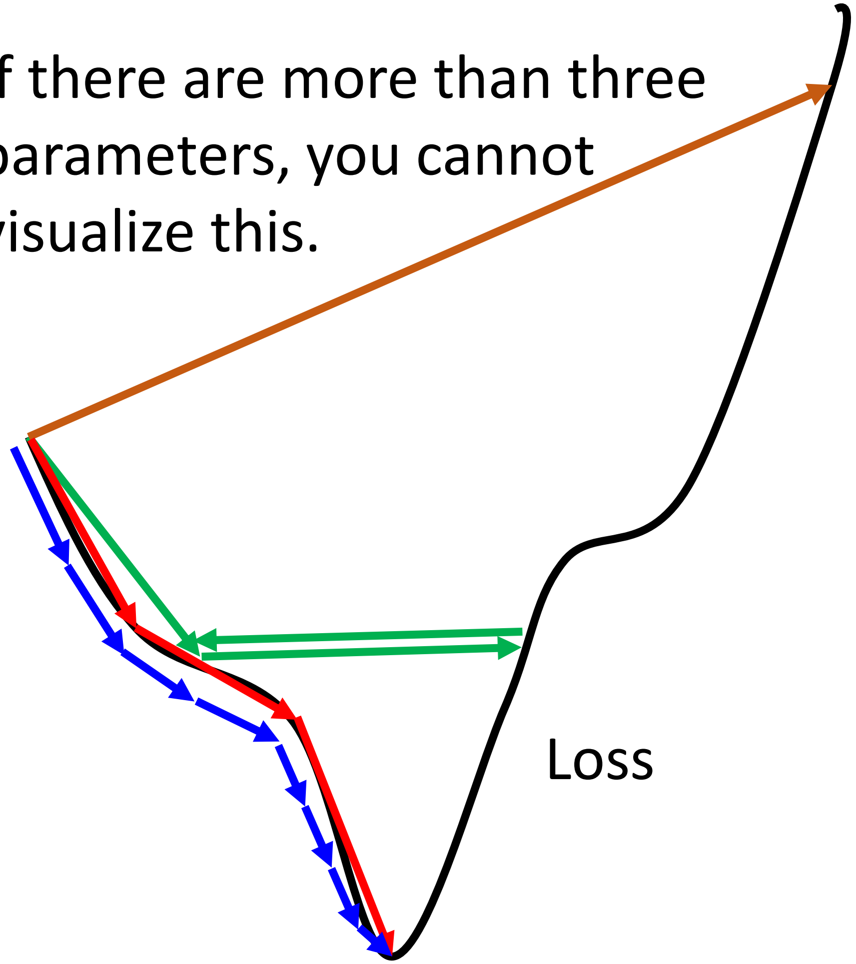
Tip 1: Tuning your learning rates

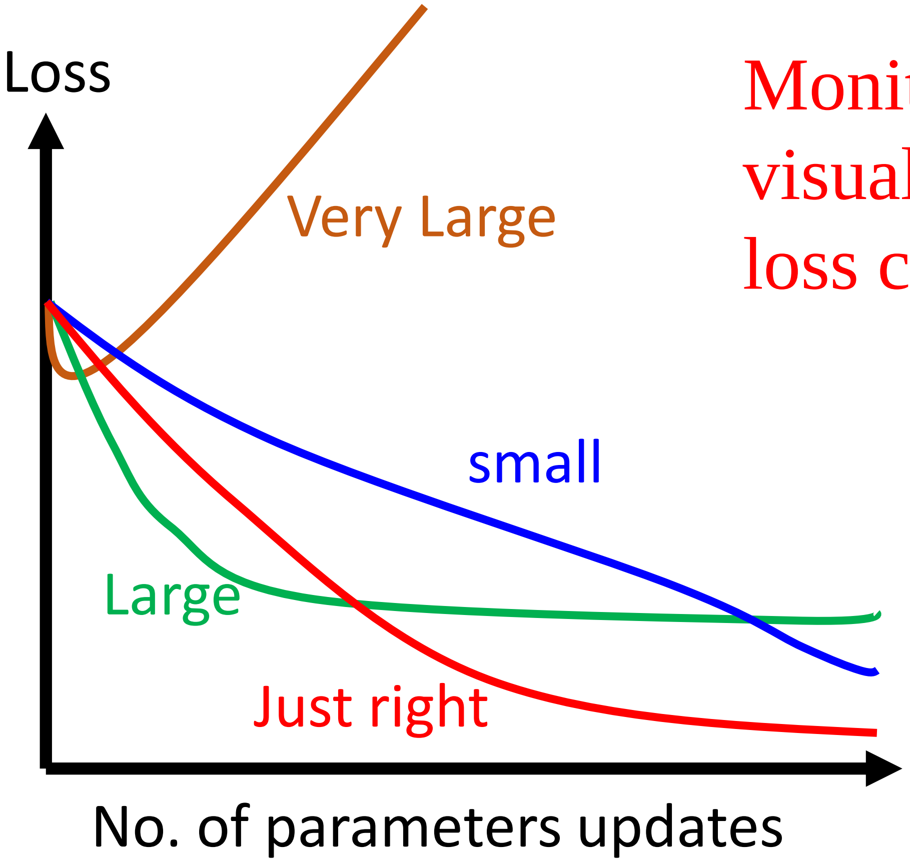
Learning Rate

$$\theta^{(i)} = \theta^{(i-1)} - \eta \nabla L(\theta^{(i-1)})$$

Set the learning rate η carefully

If there are more than three parameters, you cannot visualize this.





Monitor and visualize the loss curve!

But you can always visualize this.

Adaptive Learning Rates

- Popular and Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate to make sure it converges to the minimum.
  - *e.g.* $1/t$ decay: $\eta^{(t)} = \eta/\sqrt{t+1}$
- Learning rate **cannot be "one size fits all"**
  - Giving different parameters different learning rates

Adagrad

- Divide the learning rate of each parameter by the ***root mean square of all its previous derivatives***

Vanilla Gradient descent

$$w^{(t+1)} \leftarrow w^{(t)} - \eta^{(t)} g^{(t)}$$

w is one parameters

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}|_{w=w^{(t)}}$$

Adagrad

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta^{(t)}}{\sigma^{(t)}} g^{(t)}$$

$\sigma^{(t)}$: ***root mean square*** of all the previous derivatives of parameter *w* up to iteration *t*.

Parameter dependent

Adagrad

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}\Big|_{w=w^{(t)}}$$

$\sigma^{(t)}$: ***root mean square*** of all the previous derivatives of parameter w up to iteration t.

$$w^{(1)} \leftarrow w^{(0)} - \frac{\eta^{(0)}}{\sigma^{(0)}} g^{(0)} \qquad \sigma^{(0)} = \sqrt{(g^{(0)})^2}$$

$$w^{(2)} \leftarrow w^{(1)} - \frac{\eta^{(1)}}{\sigma^{(1)}} g^{(1)} \qquad \sigma^{(1)} = \sqrt{\frac{1}{2}[(g^{(0)})^2 + (g^{(1)})^2]}$$

$$w^{(3)} \leftarrow w^{(2)} - \frac{\eta^{(2)}}{\sigma^{(2)}} g^{(2)} \qquad \sigma^{(2)} = \sqrt{\frac{1}{3}[(g^{(0)})^2 + (g^{(1)})^2 + (g^{(2)})^2]}$$

$$\vdots$$

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta^{(t)}}{\sigma^{(t)}} g^{(t)} \qquad \sigma^{(t)} = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^{(i)})^2}$$

UNIVERSITY OF OULU

Adagrad

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}|_{w=w^{(t)}}$$

- Divide the learning rate of each parameter by the ***root mean square of all its previous derivatives***

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta^{(t)}}{\sigma^{(t)}} g^{(t)}$$

$$\eta^{(t)} = \frac{\eta}{\sqrt{\cancel{t+1}}}$$

$$\sigma^{(t)} = \sqrt{\frac{1}{\cancel{t+1}} \sum_{i=0}^{t} (g^{(i)})^2}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g^{(i)})^2}} g^{(t)}$$

- Other adaptive learning rate methods: Adadelta, Adam,…

Contradiction?

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}\Big|_{w=w^{(t)}}$$

Vanilla Gradient descent

$$w^{(t+1)} \leftarrow w^{(t)} - \eta^{(t)} \boxed{g^{(t)}}$$

Larger gradient, larger step

Adagrad

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta}{\sqrt{\underline{\sum_{i=0}^{t} (g^{(i)})^2}}} \boxed{g^{(t)}}$$

Larger gradient, larger step

Larger gradient, smaller step

Gradient Descent

- Tip 2: Stochastic Gradient Descent

Make the training faster

Stochastic Gradient Descent

$$l^{(i)} = -\log\left(\exp\left(z_k^{(i)}\right) / \sum_j \exp(z_j^{(i)})\right)$$

$$L = \frac{1}{N} \boxed{\sum_{i=1}^{N}} l^{(i)}$$

Loss is the summation over all training examples

- ***Gradient Descent***

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla L(\theta^{(i)})$$

- ***Stochastic Gradient Descent (SGD)***

Faster!

Randomly pick one example $\boldsymbol{x}^{(i)}$ to update parameters

$$\boxed{l^{(i)}} = -\log\left(\exp\left(z_k^{(i)}\right) / \sum_j \exp(z_j^{(i)})\right)$$

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla L(\theta^{(i)})$$

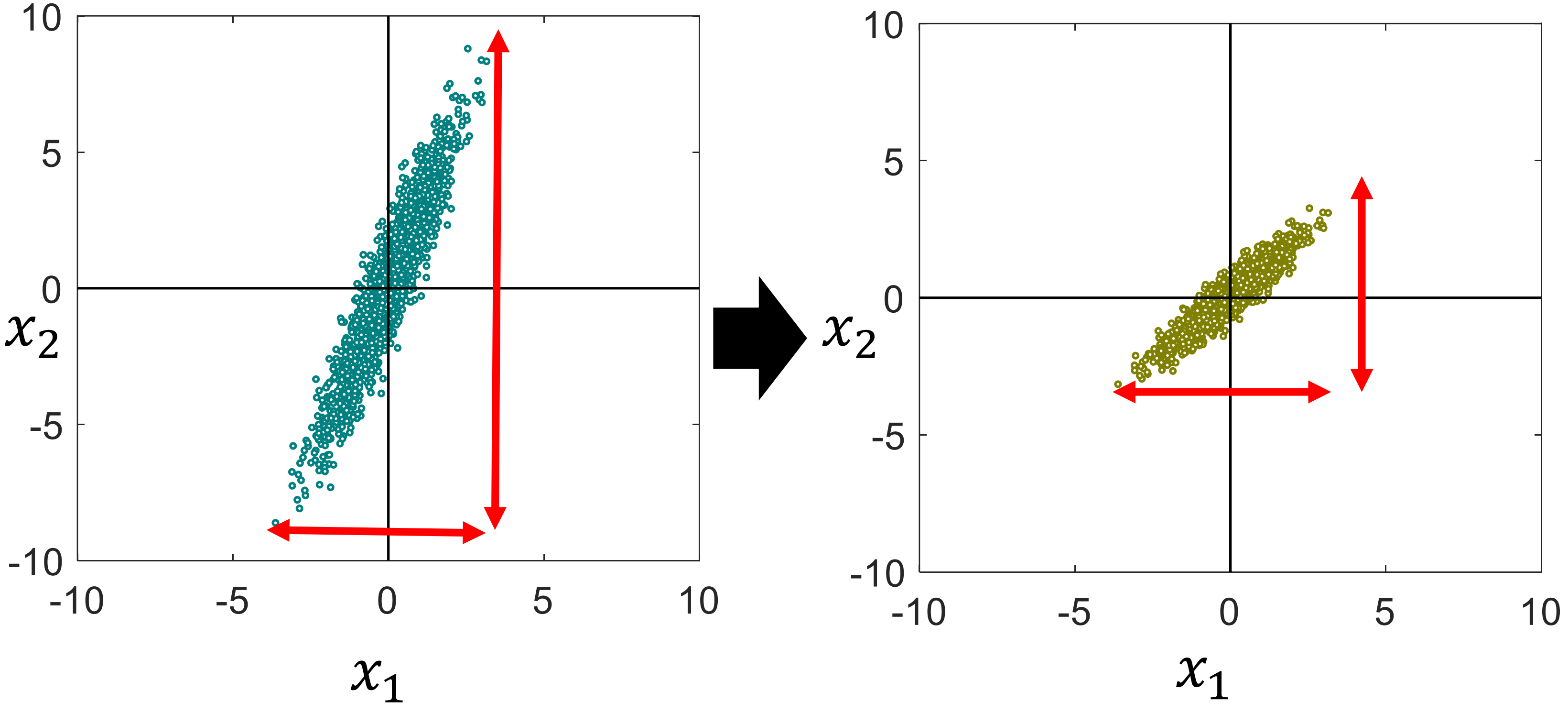
Loss for only one example, *i.e.* i^{th} sample

Gradient Descent

- Tip 3: Feature Scaling

Feature Scaling/Feature Normalization

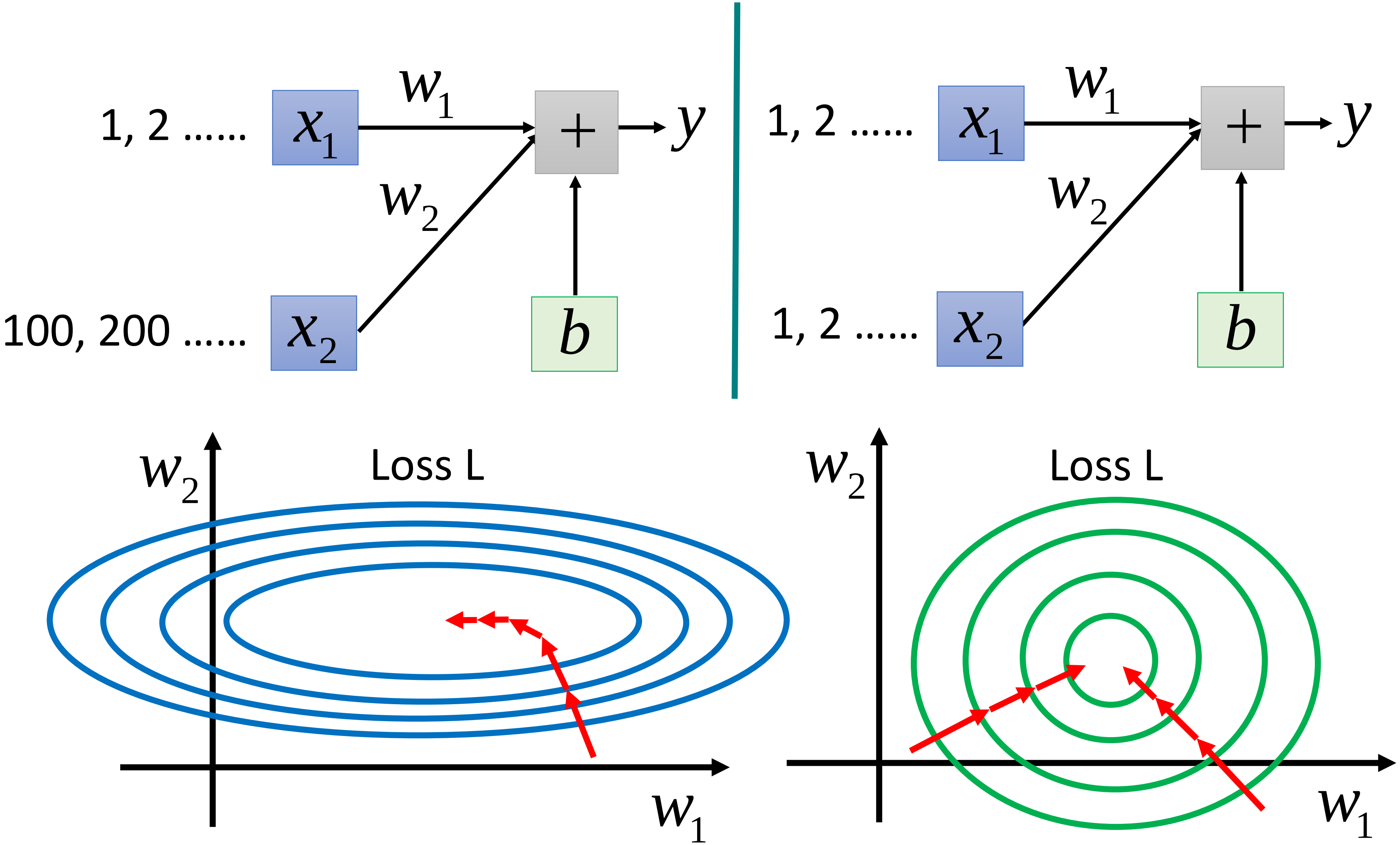
$$y = b + w_1 x_1 + w_2 x_2$$



Make different features have the same scaling

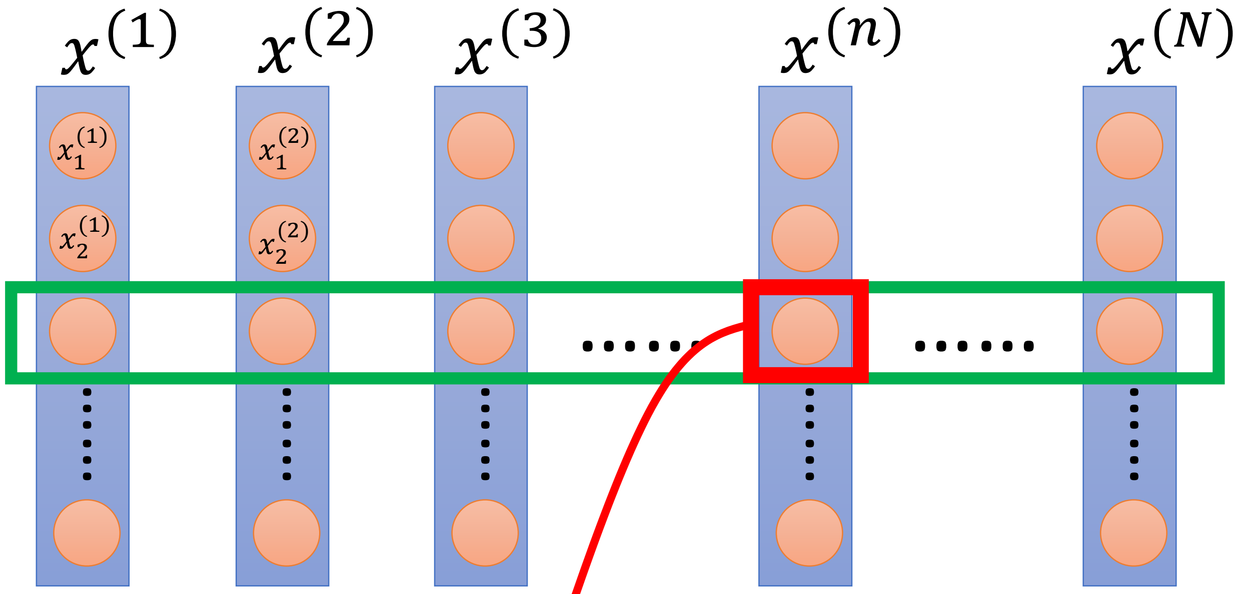
Feature Scaling

$$y = b + w_1 x_1 + w_2 x_2$$



Feature Scaling

$$m_i = \frac{1}{N}\sum_{i=1}^{N} x_i^{(n)}$$



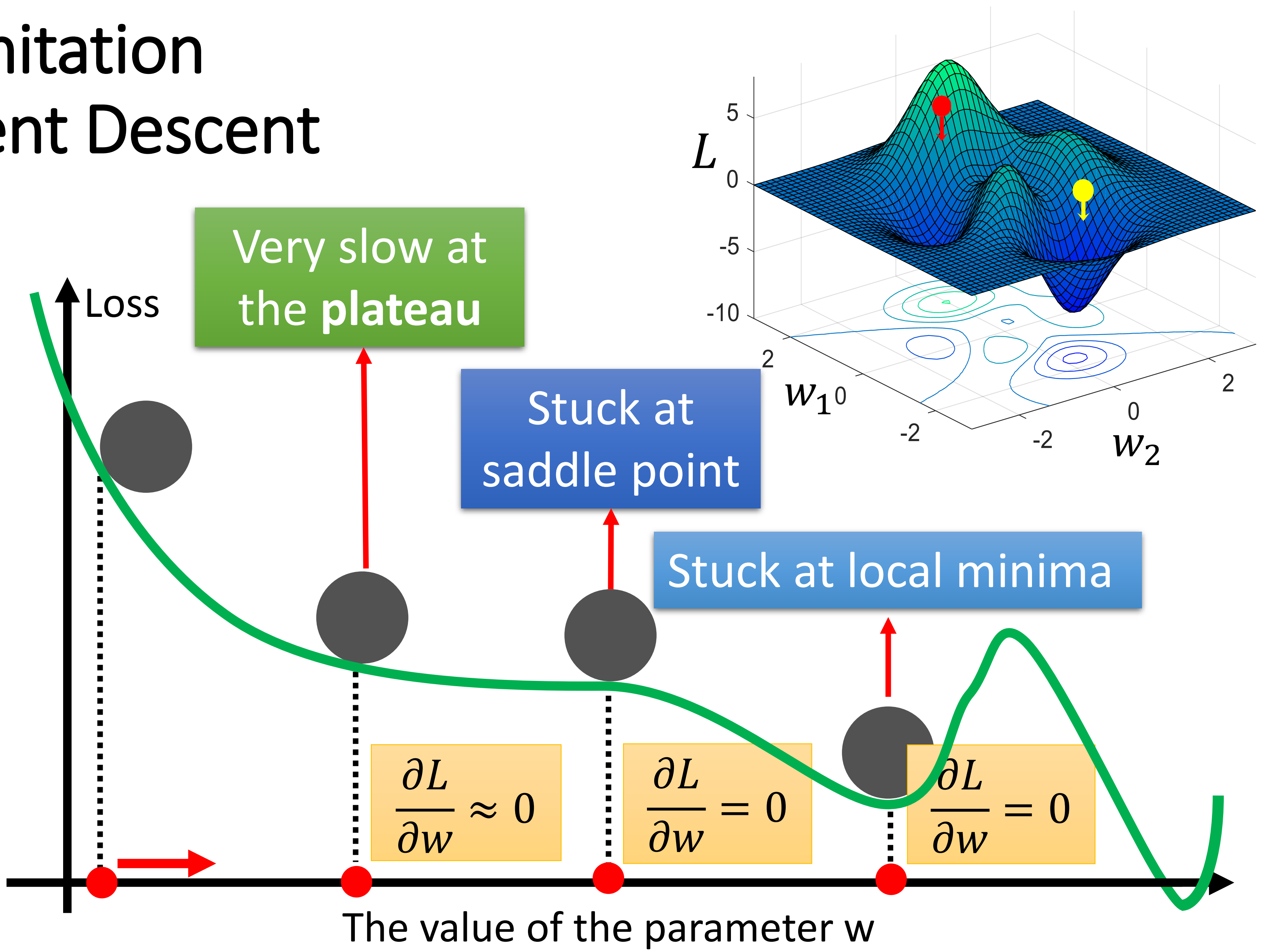
For each dimension i:

- mean: m_i
- Standard deviation: σ_i

$$x_i^{(n)} \leftarrow \frac{x_i^{(n)} - m_i}{\sigma_i}$$

Make each feature component zero mean and unit standard deviation.

More Limitation of Gradient Descent



An overview of gradient descent optimization algorithms

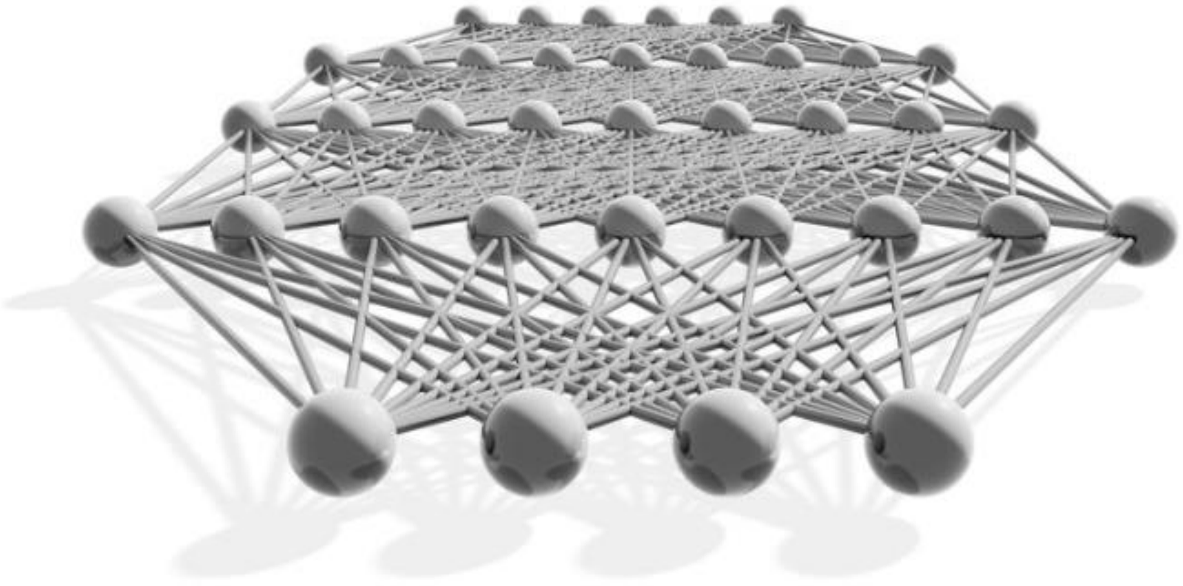
- https://arxiv.org/pdf/1609.04747.pdf

51

Summary and Next Lecture

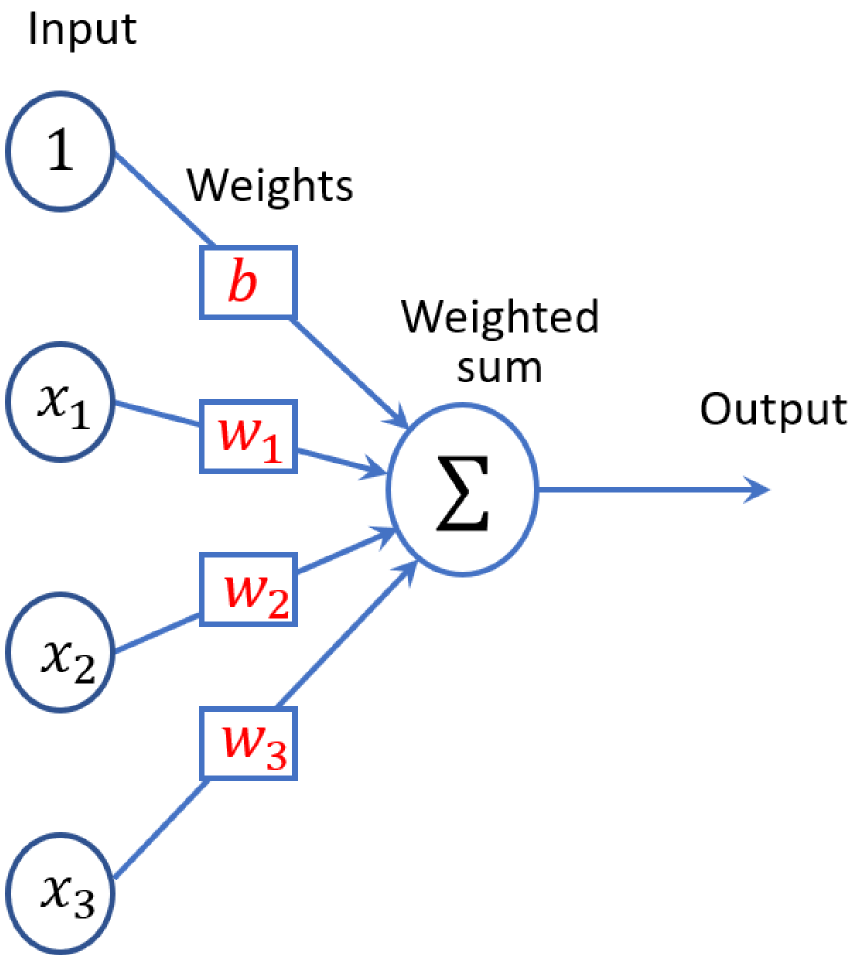
(This Wednesday, 04 November)

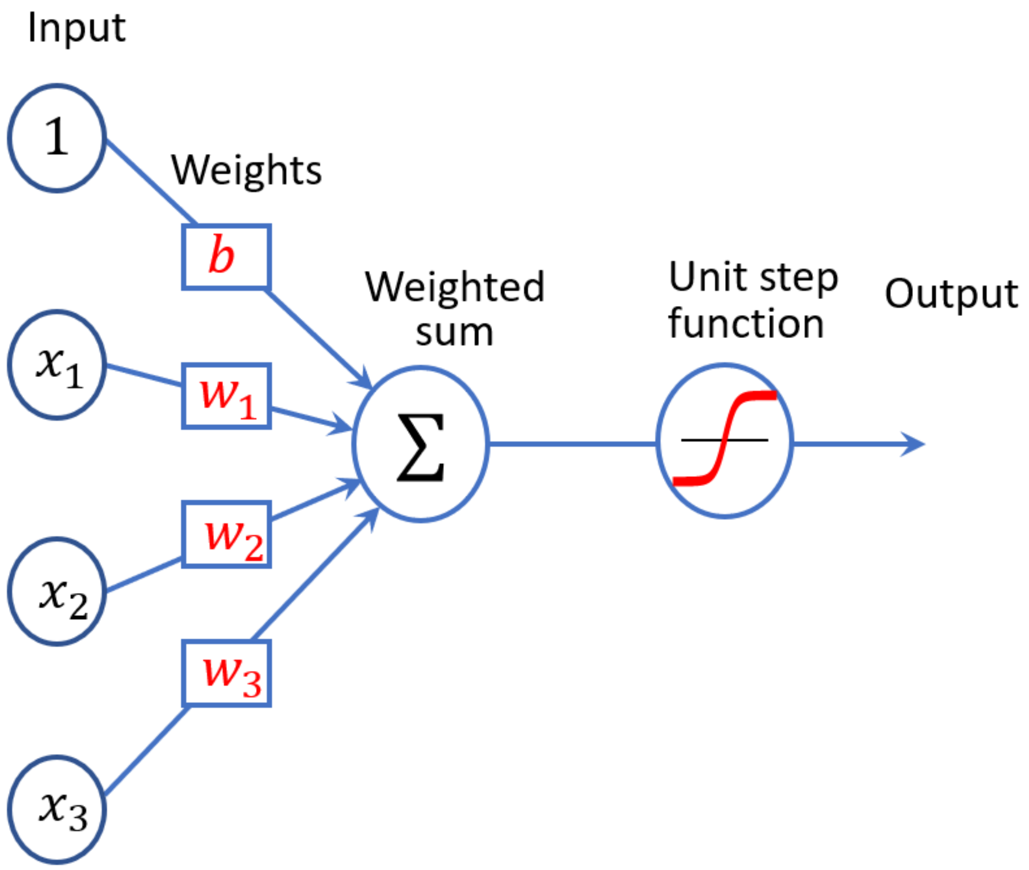
- Neural Networks
- Multilayer Neural Networks
- Backpropagation



Selflearn

Linear Regression vs Logistic Regression





Logistic Regression Model

Want $0 \le f_{w,b}(x) \le 1$

$$f_{w,b}(x) = \boldsymbol{w}^T\boldsymbol{x} + b?$$

$$f_{w,b}(x) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$$

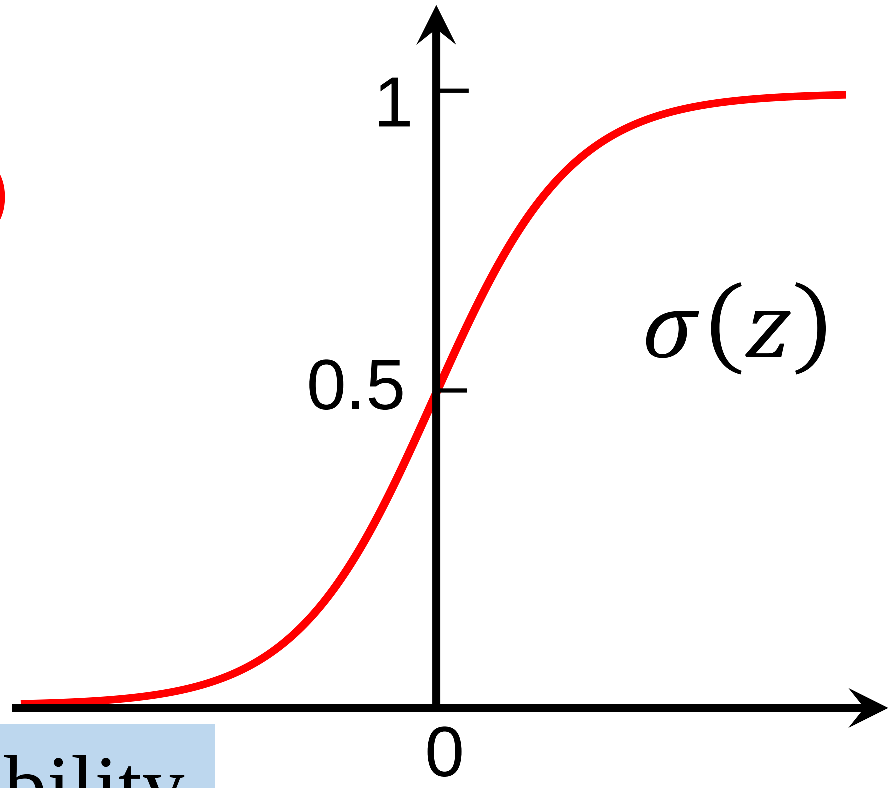
$$\sigma(z) = \frac{1}{1 + exp(-z)}$$

Sigmoid function

Logistic function

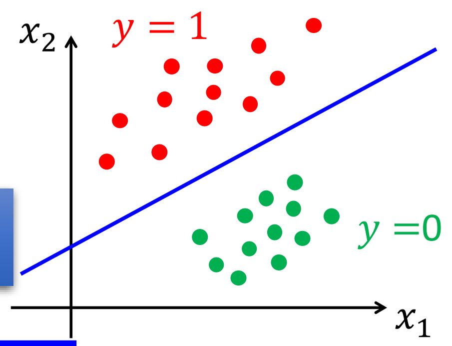
$\sigma(z) \ge 0.5$, class 1

$\sigma(z) < 0.5$, class 2



$\sigma(z)$ means posterior Probability

Step 1: Function Set



- Function set: Including all different w and b

$$
\begin{cases} P_{w,b}(C_1|x) \geq 0.5, & \text{class 1} \\ P_{w,b}(C_1|x) < 0.5, & \text{class 2} \end{cases}
$$

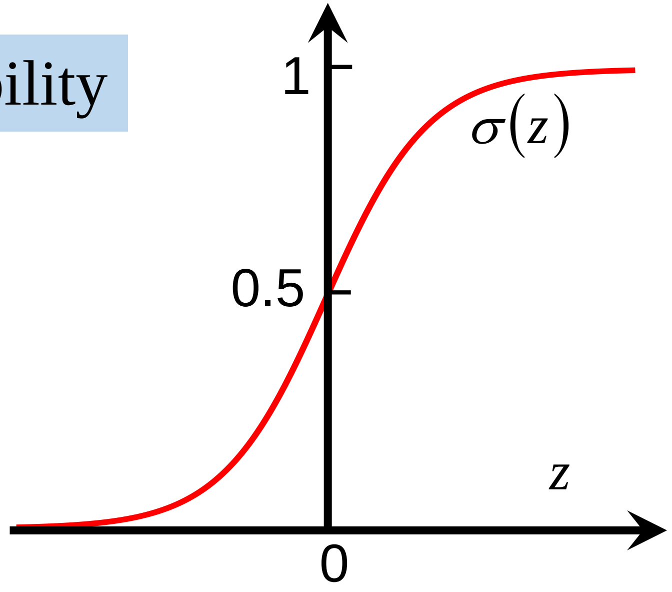
$P_{w,b}(C_1|x) = \sigma(z)$ Posterior Probability

- Hypothesis

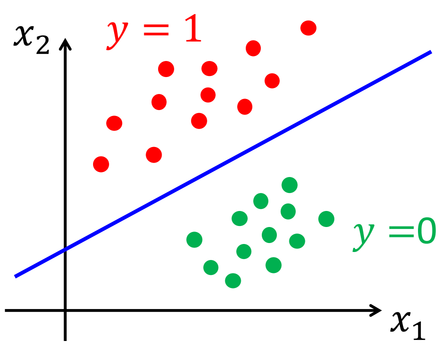
sigmoid function (logistic function)

$$z = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



Step 1: Function Set



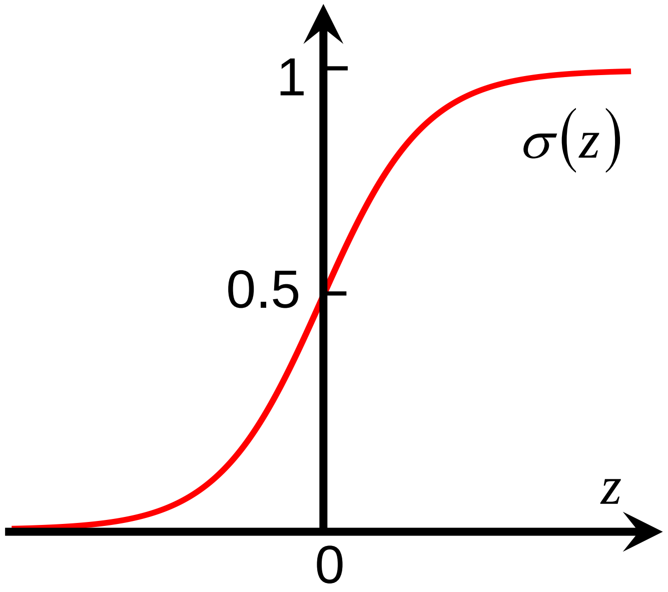
Function set: Including all different w and b

$$
\begin{cases} z \geq 0 & \text{class 1} \\ z < 0 & \text{class 2} \end{cases}
$$

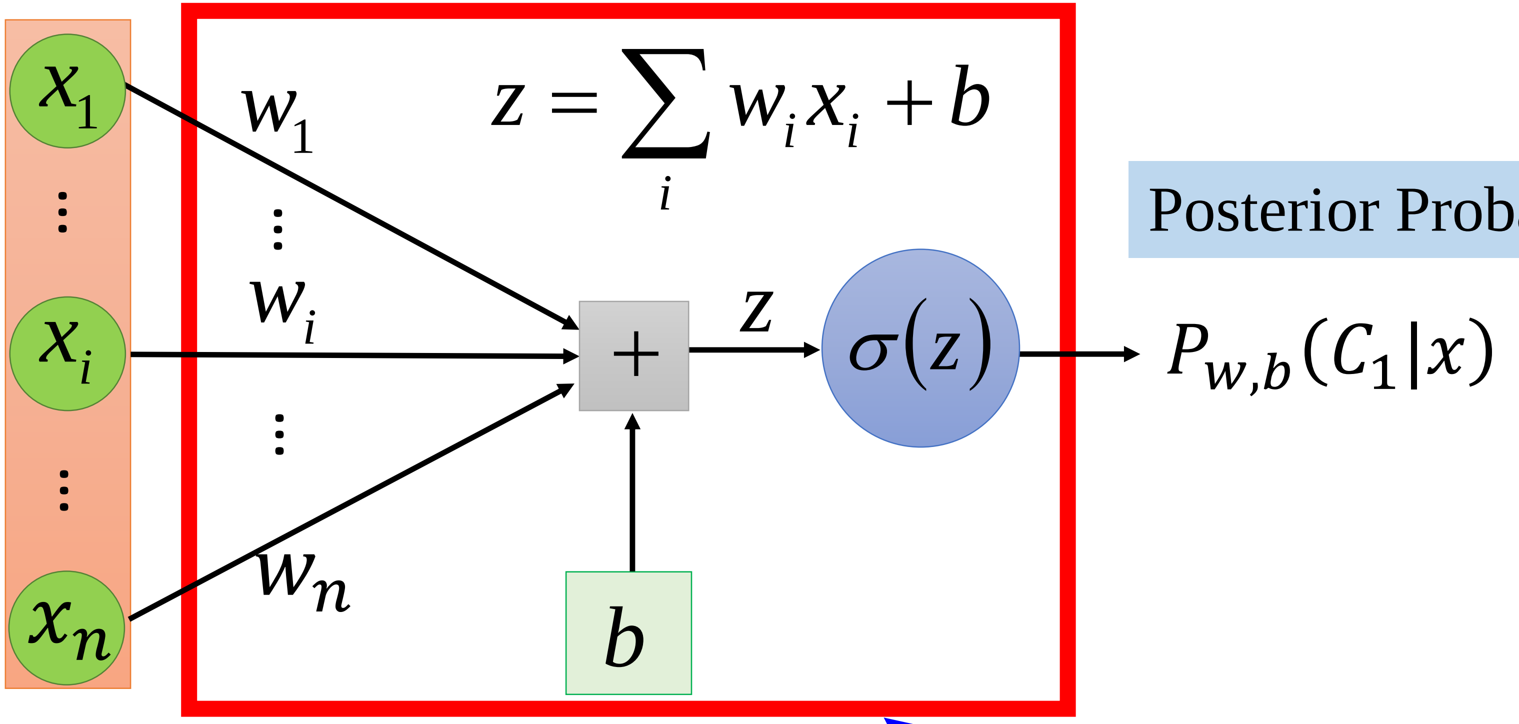
$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = \boldsymbol{w}^T\boldsymbol{x} + b \quad = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



Step 1: Function Set

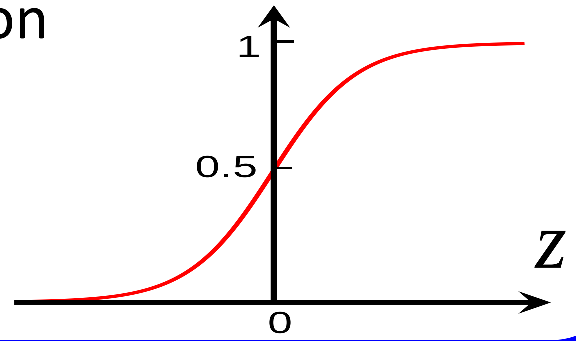


$\sigma(z)$

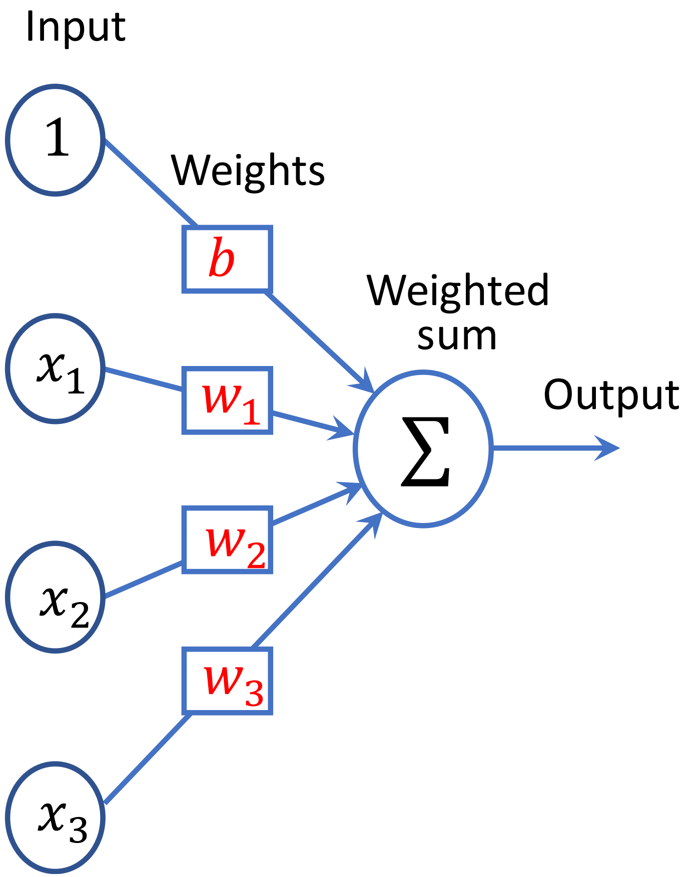
Logistic Regression

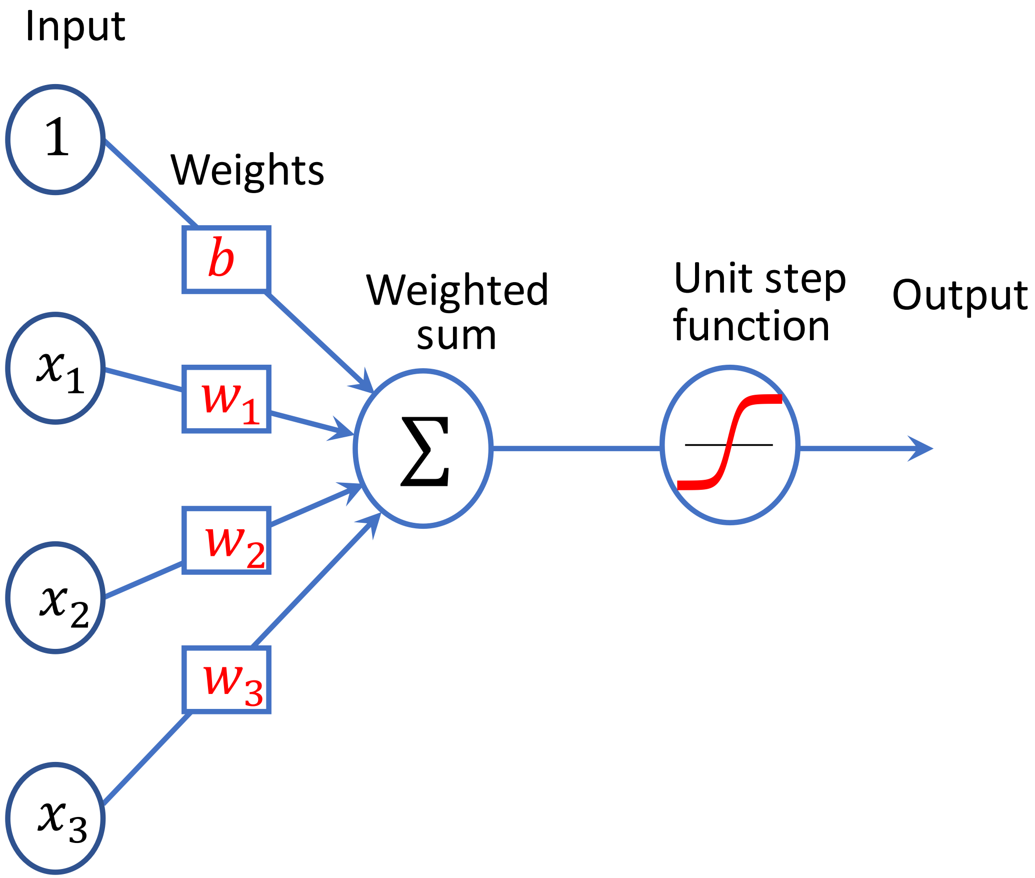
Sigmoid Function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Linear Regression (Prediction) vs. Logistic Regression (Classification)





Step 2: Goodness of a Function

Training Data	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	……	$x^{(N)}$
	C_1	C_1	C_2		C_1

$$P_{w,b}(C_1|x) = \frac{1}{1+exp(-z)}$$

$$z = \boldsymbol{w}^T\boldsymbol{x} + b$$

Assume the data is generated based on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Given a set of ***w*** and b, what is its probability of generating the data?

$$L(w,b) = f_{w,b}\left(x^{(1)}\right)f_{w,b}\left(x^{(2)}\right)\left(1 - f_{w,b}\left(x^{(3)}\right)\right)\cdots f_{w,b}\left(x^{(N)}\right)$$

The most likely w* and b* is the one with the largest $L(w,b)$.

$$w^*, b^* = \arg\max_{w,b} L(w,b)$$

$$x^{(1)} \quad x^{(2)} \quad x^{(3)} \quad \cdots\cdots$$
$$C_1 \quad C_1 \quad C_2$$

$$x^{(1)} \quad x^{(2)} \quad x^{(3)} \quad \cdots\cdots$$
$$y^{(1)} = 1 \quad y^{(2)} = 1 \quad y^{(3)} = 0$$

$y^{(k)}$: 1 for class 1, 0 for class 2

$$L(w,b) = f_{w,b}(x^{(1)}) f_{w,b}(x^{(2)}) \left(1 - f_{w,b}(x^{(3)})\right) \cdots$$

$$\boxed{w^*, b^* = arg \max_{w,b} L(w,b)} = \boxed{w^*, b^* = arg \min_{w,b} -lnL(w,b)}$$

$$-lnL(w,b)$$
$$= -lnf_{w,b}(x^{(1)}) \rightarrow -\left[1\, lnf(x^{(1)}) + 0\, ln\left(1 - f(x^{(1)})\right)\right]$$
$$-lnf_{w,b}(x^{(2)}) \rightarrow -\left[1\, lnf(x^{(2)}) + 0\, ln\left(1 - f(x^{(2)})\right)\right]$$
$$-ln\left(1 - f_{w,b}(x^{(3)})\right) \rightarrow -\left[0\, lnf(x^{(3)}) + 1\, ln\left(1 - f(x^{(3)})\right)\right]$$
$$\vdots$$

$$x^{(1)} \quad x^{(2)} \quad x^{(3)} \quad \cdots\cdots$$
$$C_1 \quad C_1 \quad C_2$$

$$x^{(1)} \quad x^{(2)} \quad x^{(3)} \quad \cdots\cdots$$
$$y^{(1)} = 1 \quad y^{(2)} = 1 \quad y^{(3)} = 0$$

$y^{(k)}$: 1 for class 1, 0 for class 2

$$L(w,b) = f_{w,b}\left(x^{(1)}\right) f_{w,b}\left(x^{(2)}\right) \left(1 - f_{w,b}\left(x^{(3)}\right)\right) \cdots$$

$$\boxed{w^*, b^* = arg \max_{w,b} L(w,b)} = \boxed{w^*, b^* = arg \min_{w,b} -lnL(w,b)}$$

$$-lnL(w,b)$$
$$= -lnf_{w,b}\left(x^{(1)}\right) \Rightarrow -\left[y^{(1)} lnf\left(x^{(1)}\right) + \left(1 - y^{(1)}\right) ln\left(1 - f\left(x^{(1)}\right)\right)\right]$$
$$-lnf_{w,b}\left(x^{(2)}\right) \Rightarrow -\left[y^{(2)} lnf\left(x^{(2)}\right) + \left(1 - y^{(2)}\right) ln\left(1 - f\left(x^{(2)}\right)\right)\right]$$
$$-ln\left(1 - f_{w,b}\left(x^{(3)}\right)\right) \Rightarrow -\left[y^{(3)} lnf\left(x^{(3)}\right) + \left(1 - y^{(3)}\right) ln\left(1 - f\left(x^{(3)}\right)\right)\right]$$
$$\vdots$$

$$\| $$

$$\sum_{k=1}^{N} -\left[y^{(k)} lnf_{w,b}\left(x^{(k)}\right) + \left(1 - y^{(k)}\right) ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right)\right]$$

UNIVERSITY OF OULU

Step 2: Goodness of a Function

$$L(w,b) = f_{w,b}\left(x^{(1)}\right) f_{w,b}\left(x^{(2)}\right)\left(1 - f_{w,b}\left(x^{(3)}\right)\right) \cdots f_{w,b}\left(x^{(N)}\right)$$

$$-lnL(w,b) = lnf_{w,b}\left(x^{(1)}\right) + lnf_{w,b}\left(x^{(2)}\right) + ln\left(1 - f_{w,b}\left(x^{(3)}\right)\right) \cdots$$

$y^{(k)}$: 1 for class 1, 0 for class 2

$$= \sum_{k=1}^{N} \underline{-\left[y^{(k)} lnf_{w,b}\left(x^{(k)}\right) + \left(1 - y^{(k)}\right) ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right)\right]}$$

Cross entropy between two Bernoulli distribution

Distribution p:

$p(x=1) = y^{(n)}$

$p(x=0) = 1 - y^{(n)}$

cross entropy

Distribution q:

$q(x=1) = f\left(x^{(n)}\right)$

$q(x=0) = 1 - f\left(x^{(n)}\right)$

$$H(p,q) = -\sum_{x} p(x) ln\left(q(x)\right)$$

Step 2: Goodness of a Function

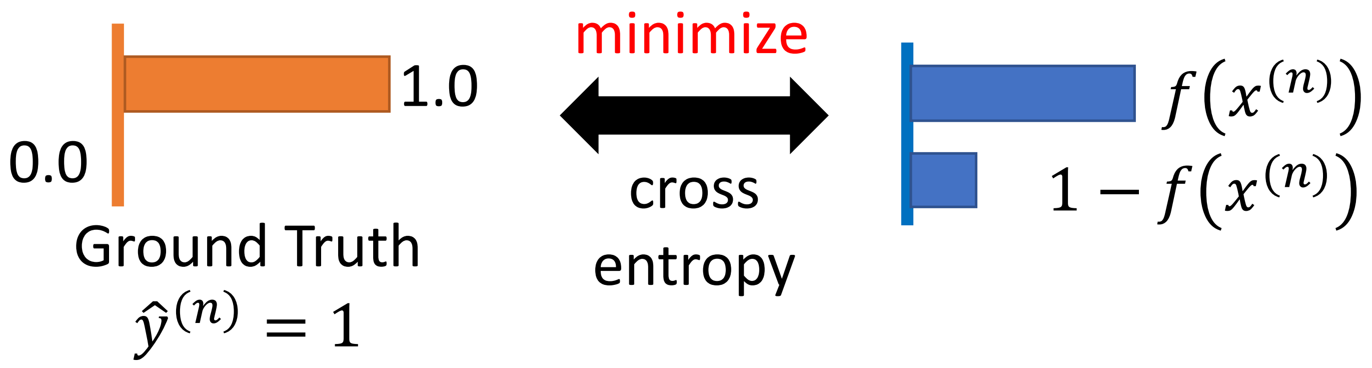
$$L(w,b) = f_{w,b}\left(x^{(1)}\right) f_{w,b}\left(x^{(2)}\right)\left(1 - f_{w,b}\left(x^{(3)}\right)\right) \cdots f_{w,b}\left(x^{(N)}\right)$$

$$-lnL(w,b) = lnf_{w,b}\left(x^{(1)}\right) + lnf_{w,b}\left(x^{(2)}\right) + ln\left(1 - f_{w,b}\left(x^{(3)}\right)\right) \cdots$$

$y^{(k)}$: 1 for class 1, 0 for class 2

$$= \sum_{k=1}^{N} -\left[y^{(k)} lnf_{w,b}\left(x^{(k)}\right) + \left(1 - y^{(k)}\right) ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right)\right]$$

Minimize cross entropy between two Bernoulli distribution



Step 3: Find the best function

chain rule

$$\left(1-f_{w,b}\left(x^{(k)}\right)\right)x_i^{(k)}$$

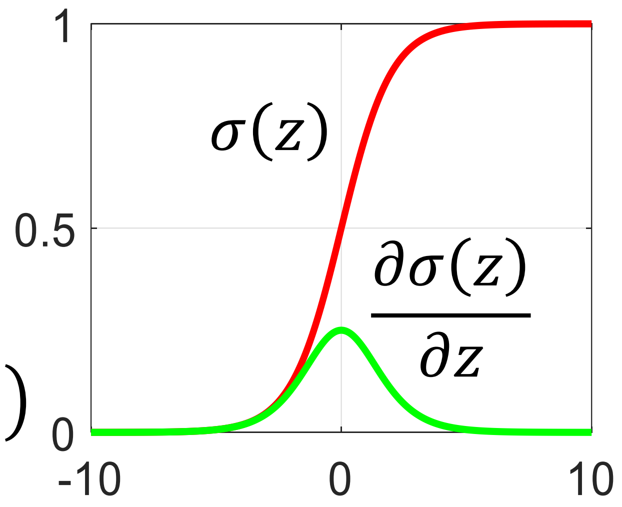
$$\boxed{\frac{d\ln x}{dx}=\frac{1}{x}}$$

$$\frac{-\ln L(w,b)}{\partial w_i}=\sum_n -\left[y^{(k)}\boxed{\frac{\ln f_{w,b}\left(x^{(n)}\right)}{\partial w_i}}+\left(1-y^{(k)}\right)\frac{\ln\left(1-f_{w,b}\left(x^{(k)}\right)\right)}{\partial w_i}\right]$$

$$\frac{\partial \ln f_{w,b}(x)}{\partial w_i}=\frac{\partial \ln f_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i}$$

$$\boxed{\frac{\partial z}{\partial w_i}=x_i}$$

$$\frac{\partial \ln\sigma(z)}{\partial z}=\frac{1}{\sigma(z)}\frac{\partial\sigma(z)}{\partial z}=\frac{1}{\cancel{\sigma(z)}}\cancel{\sigma(z)}\left(1-\sigma(z)\right)$$



$$f_{w,b}(x)=\sigma(z)=\frac{1}{1+exp(-z)}$$

$$z=\boldsymbol{w}^T\boldsymbol{x}+b=\sum_i w_i x_i+b$$

Step 3: Find the best function

$$\left(1 - f_{w,b}\left(x^{(k)}\right)\right) x_i^{(k)} \qquad -f_{w,b}\left(x^{(k)}\right) x_i^{(k)}$$

$$\frac{-lnL(w,b)}{\partial w_i} = \sum_k -\left[y^{(k)} \boxed{\frac{lnf_{w,b}\left(x^{(k)}\right)}{\partial w_i}} + \left(1 - y^{(k)}\right) \boxed{\frac{ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right)}{\partial w_i}}\right]$$

$$\frac{\partial ln\left(1 - f_{w,b}(x)\right)}{\partial w_i} = \frac{\partial ln\left(1 - f_{w,b}(x)\right)}{\partial z} \frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln\left(1 - \sigma(z)\right)}{\partial z} = -\frac{1}{1 - \sigma(z)} \frac{\partial \sigma(z)}{\partial z} = -\frac{1}{\cancel{1 - \sigma(z)}} \sigma(z) \cancel{\left(1 - \sigma(z)\right)}$$

$$f_{w,b}(x) = \sigma(z) = \frac{1}{1+exp(-z)}$$

$$z = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

$$\left(1 - f_{w,b}\left(x^{(k)}\right)\right) x_i^{(k)} \qquad -f_{w,b}\left(x^{(k)}\right) x_i^{(k)}$$

$$\frac{-\ln L(w,b)}{\partial w_i} = \sum_k -\left[y^{(k)} \boxed{\frac{\ln f_{w,b}\left(x^{(k)}\right)}{\partial w_i}} + \left(1 - y^{(k)}\right) \boxed{\frac{\ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right)}{\partial w_i}} \right]$$

$$= \sum_n -\left[y^{(k)} \left(1 - f_{w,b}\left(x^{(k)}\right)\right) x_i^{(k)} - \left(1 - y^{(k)}\right) f_{w,b}\left(x^{(k)}\right) x_i^{(k)} \right]$$

$$= \sum_k -\left[y^{(k)} - \cancel{y^{(k)} f_{w,b}\left(x^{(k)}\right)} - f_{w,b}\left(x^{(k)}\right) + \cancel{y^{(k)} f_{w,b}\left(x^{(k)}\right)} \right] x_i^{(k)}$$

$$= \sum_k -\left(y^{(k)} - f_{w,b}\left(x^{(k)}\right) \right) x_i^{(k)}$$

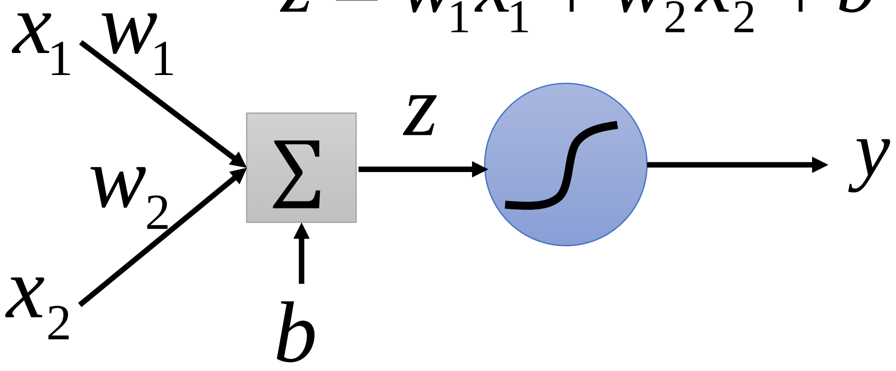
Larger difference, larger update

$$w_i \leftarrow w_i - \eta \sum_k -\left(\underline{y^{(k)} - f_{w,b}\left(x^{(k)}\right)} \right) x_i^{(k)}$$

$$P_{w,b}(C_1|x) = f_{w,b}(x) = \sigma(z)$$

Limitation of Logistic Regression

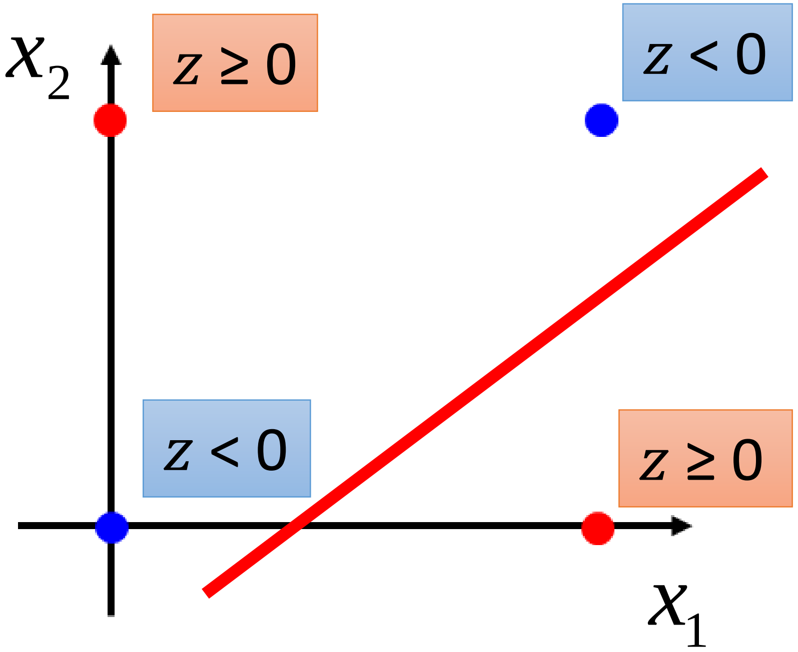
$$z = w_1 x_1 + w_2 x_2 + b$$



$$\begin{cases} Class1 & y \geq 0.5 \quad (z \geq 0) \\ Class2 & y < 0.5 \quad (z < 0) \end{cases}$$

Can we?

Input Feature		Label
x_1	x_2	
0	0	Class 2
0	1	Class 1
1	0	Class 1
1	1	Class 2

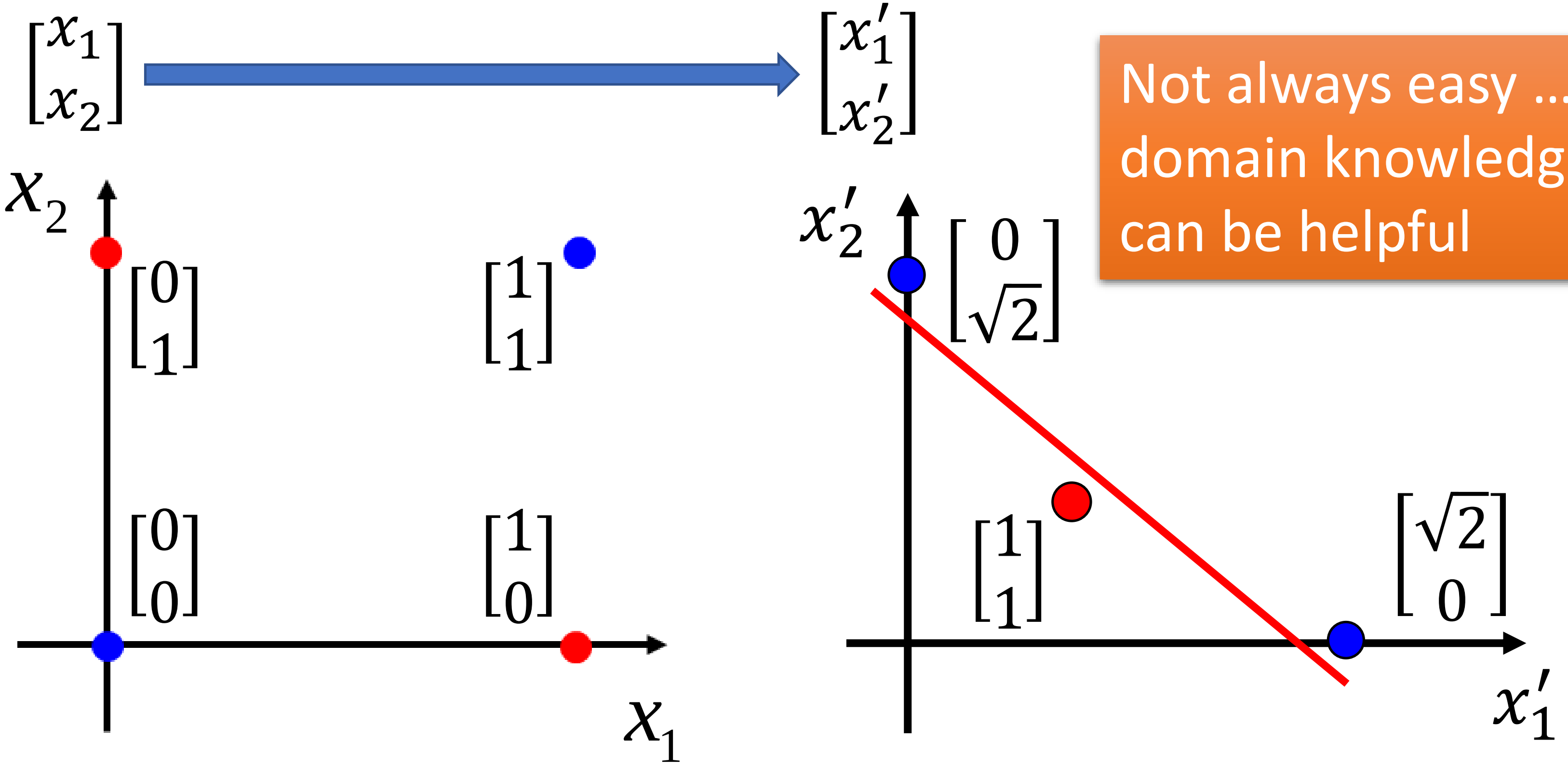


Limitation of Logistic Regression

- ***Feature Representation***

x'_1: distance to $\begin{bmatrix}0\\0\end{bmatrix}$

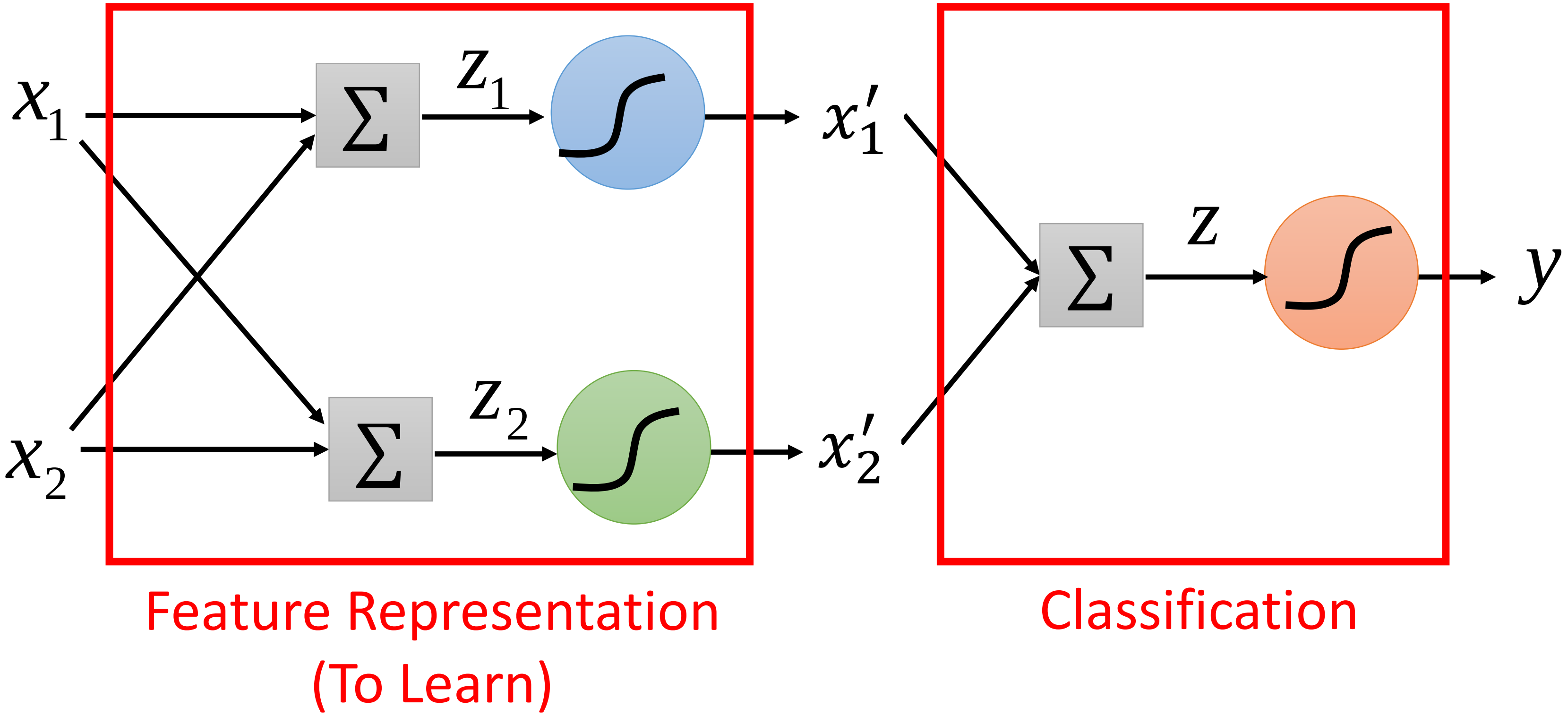
x'_2: distance to $\begin{bmatrix}1\\1\end{bmatrix}$



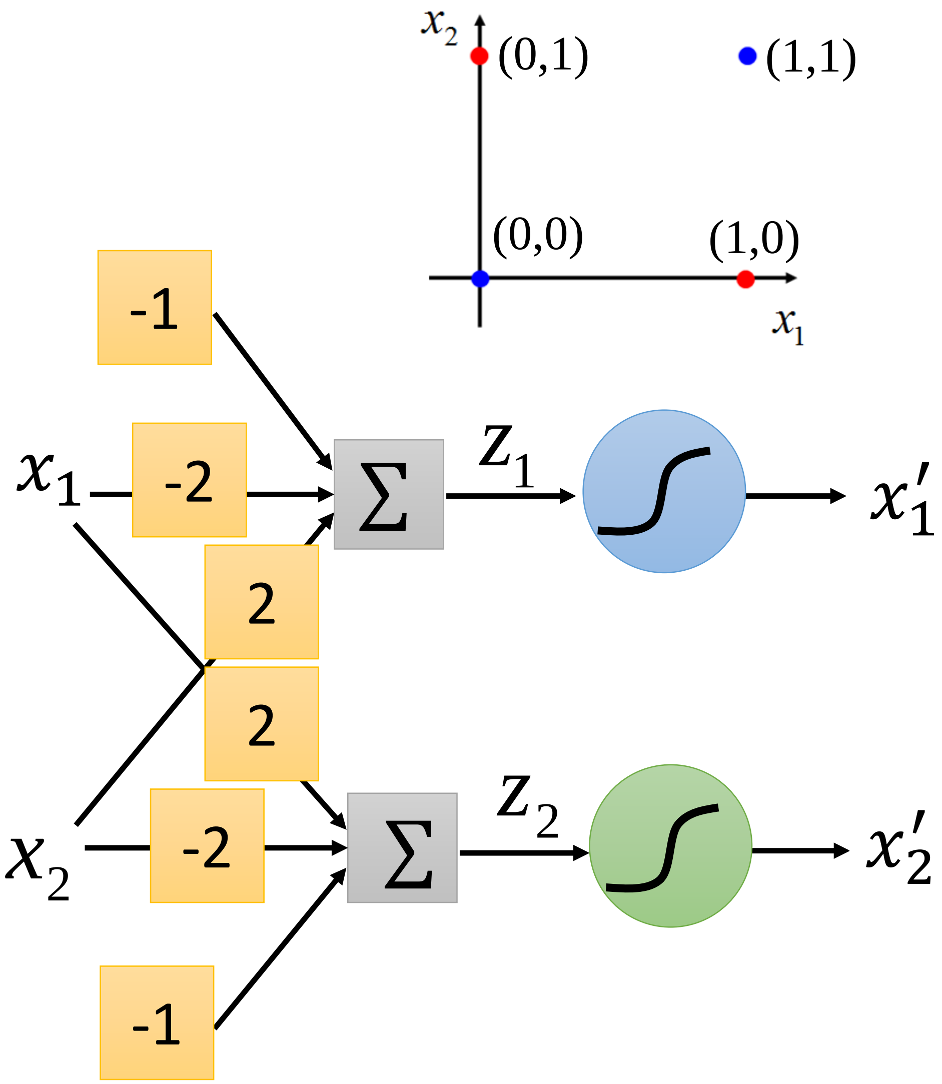
Not always easy ….. domain knowledge can be helpful

Limitation of Logistic Regression

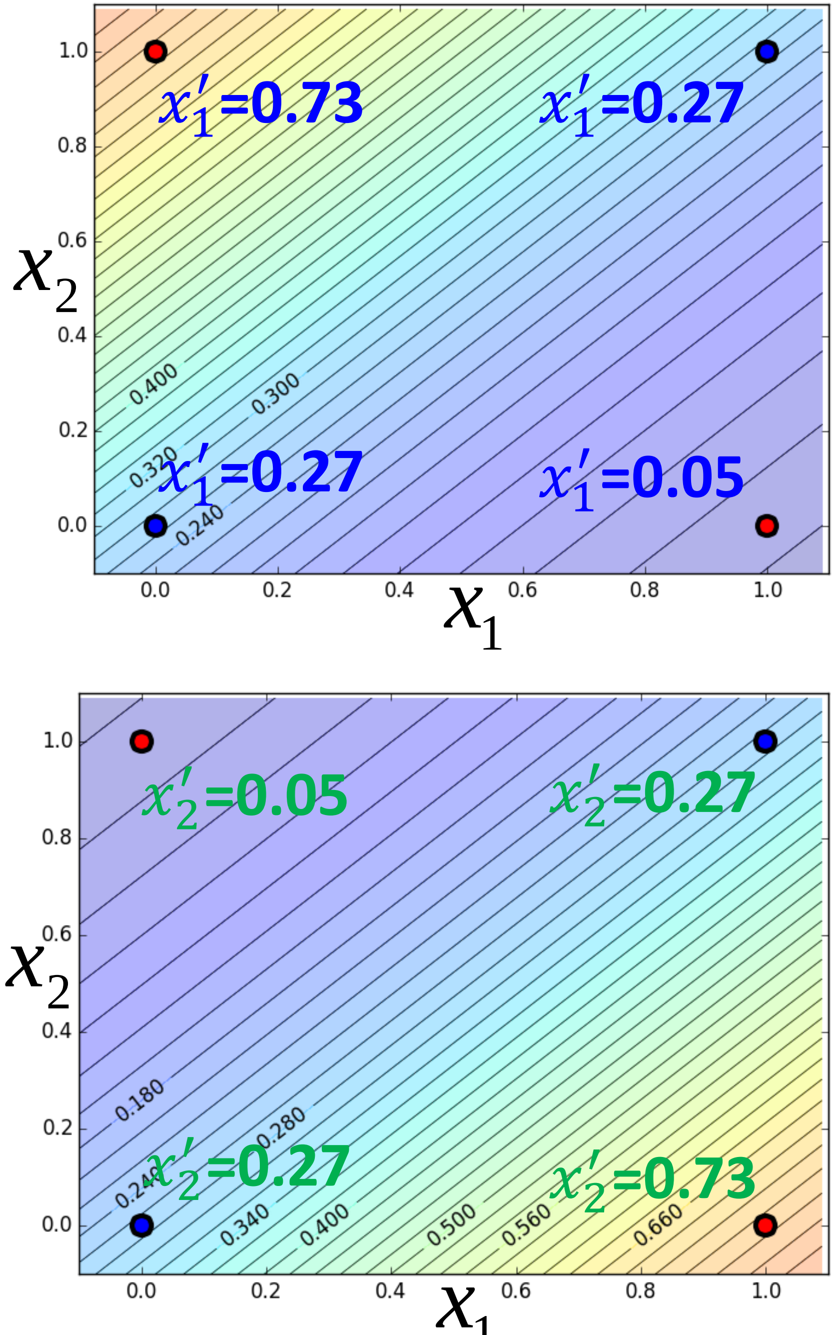
- Cascading logistic regression models



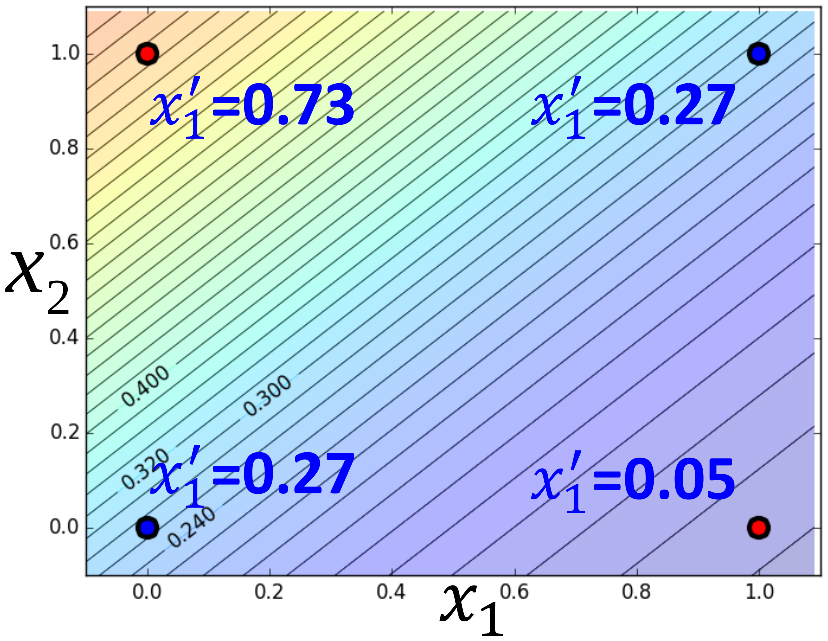
(ignore bias in this figure)

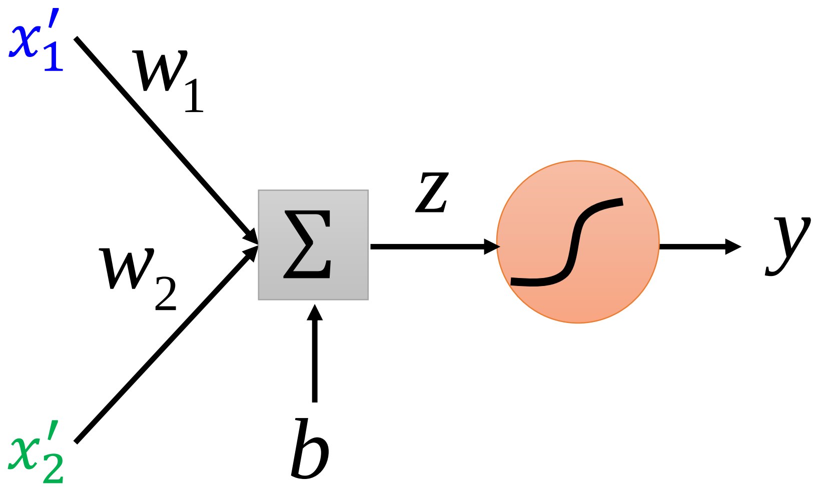


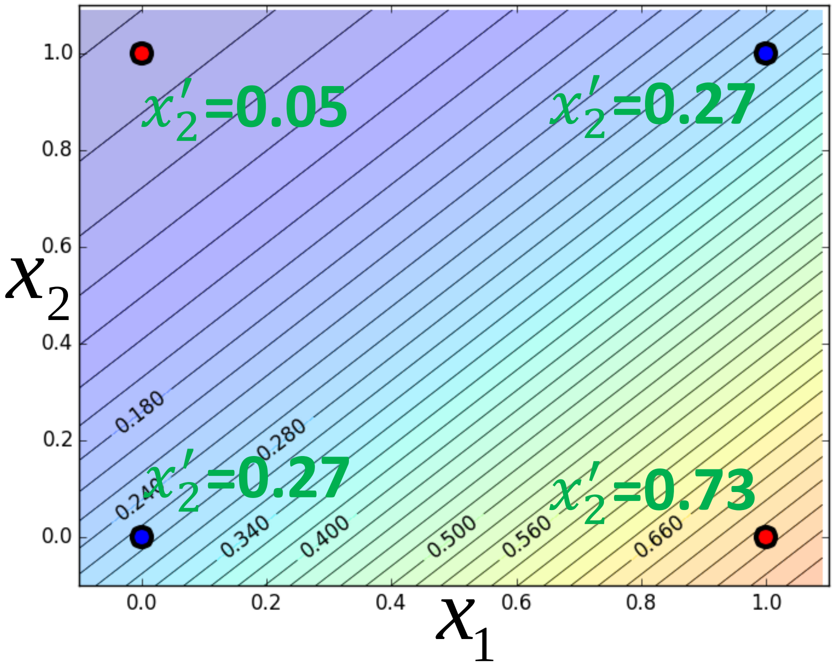
x_2
(0,1)
(1,1)
(0,0)
(1,0)
x_1
-1
x_1
-2
Σ
z_1
x'_1
2
2
x_2
-2
Σ
z_2
x'_2
-1

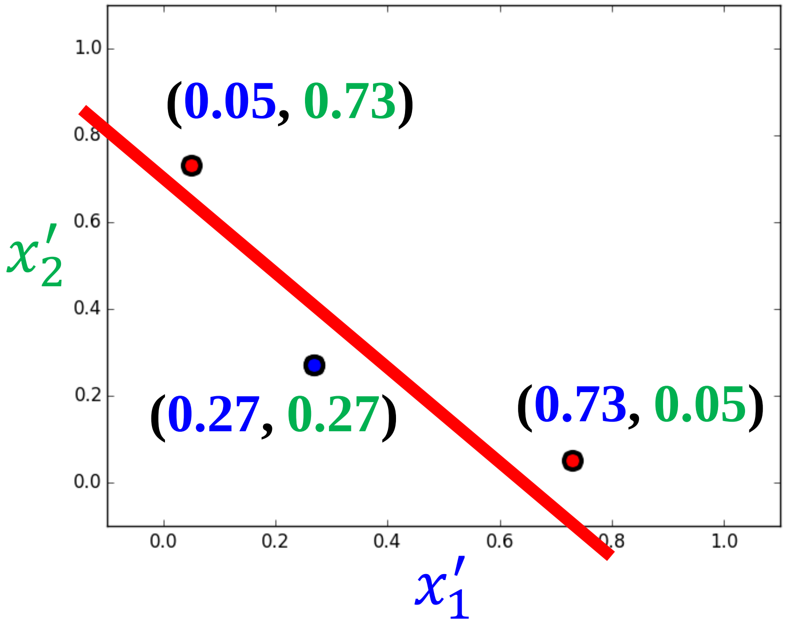


x'_1=0.73
x'_1=0.27
x'_1=0.27
x'_1=0.05
0.400
0.300
0.320
0.240
X_2
X_1
x'_2=0.05
x'_2=0.27
x'_2=0.27
x'_2=0.73
0.180
0.280
0.240
0.340
0.400
0.500
0.560
0.660


x_1'=0.73
x_1'=0.27
x_1'=0.27
x_1'=0.05
x_1
x_2

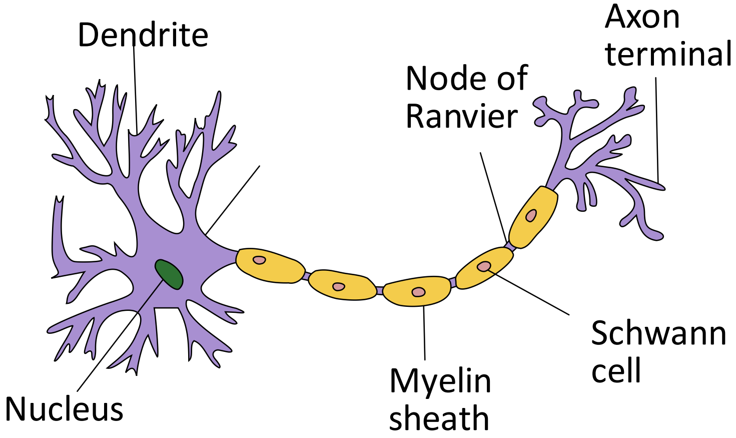

x_1'
x_2'
w_1
w_2
Σ
z
y
b

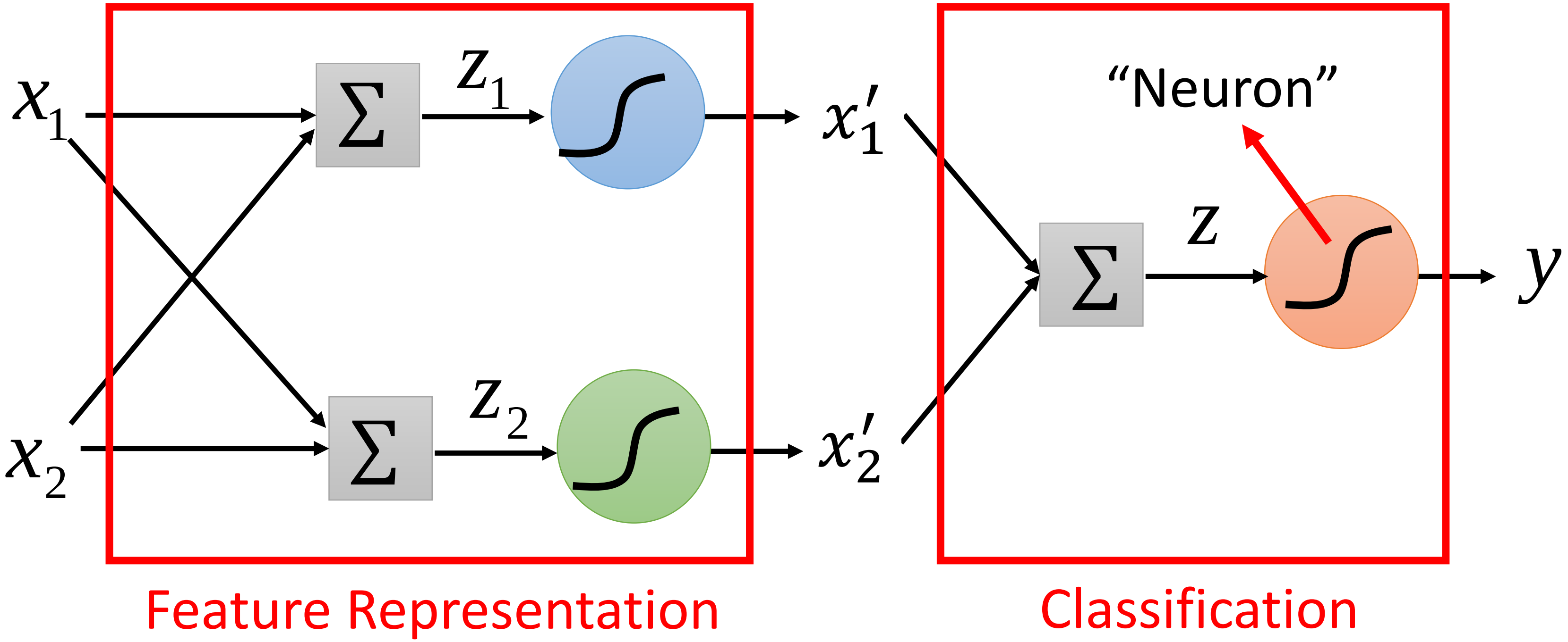

x_2'=0.05
x_2'=0.27
x_2'=0.27
x_2'=0.73
x_1
x_2


(0.05, 0.73)
(0.27, 0.27)
(0.73, 0.05)
x_1'
x_2'

Deep Learning!

All the parameters of the logistic regressions are jointly learned.





Neural Network