Machine learning: Neural networks

Bulat Ibragimov

bulat@di.ku.dk

Department of Computer Science
University of Copenhagen

UNIVERSITY OF COPENHAGEN

Learning Objectives

- One-neuron network
- Loss function
- Backpropagation
- Activation function
- Multi-neuron network

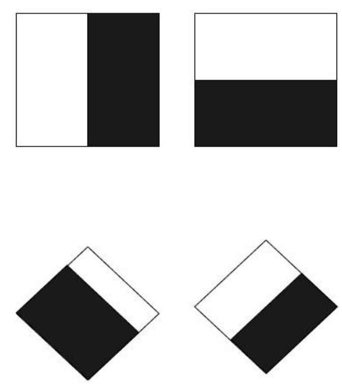
Machine learning (ML) in 1 line

1-line summary of ML: $y = \mathrm{sign}\big(f(z)\big)$

Data representation

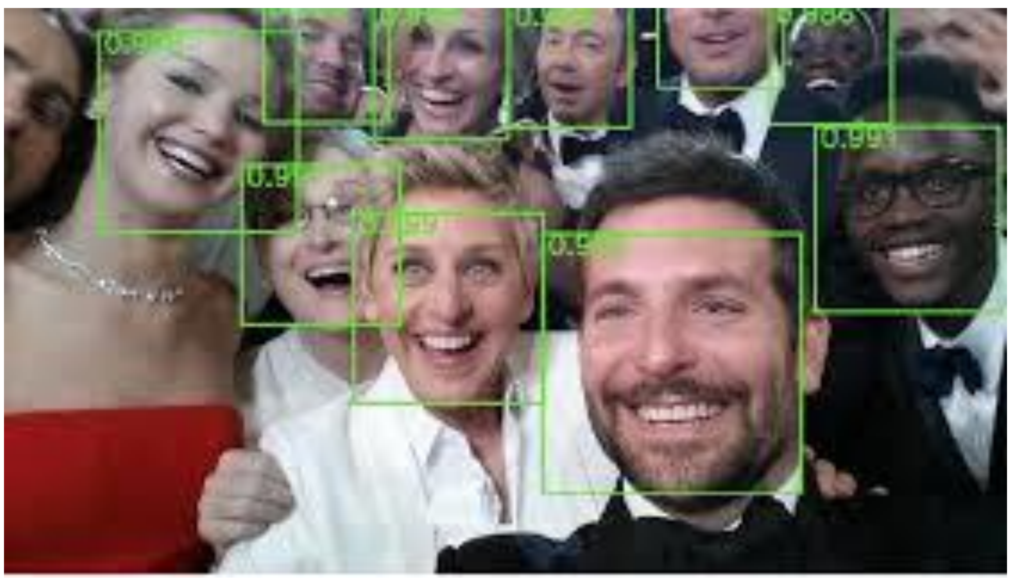
- Simple example of $f(x) = Wz + b$

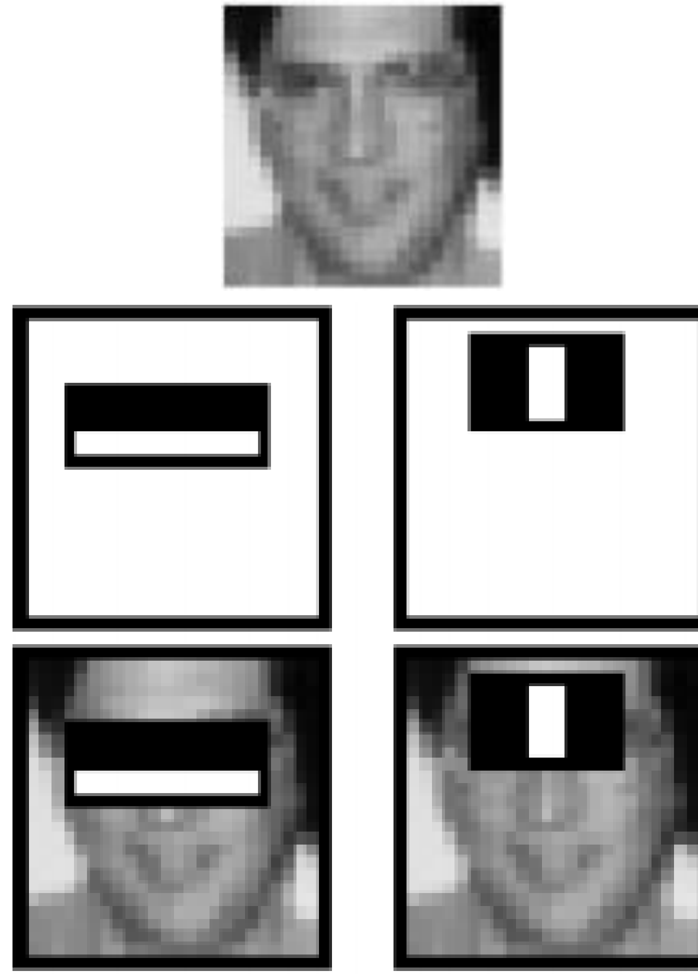
- $\cdots$ you just need to have the right z.



Machine learning in medical image analysis

- How useful hand-crafted features are?
- Very useful, but limited

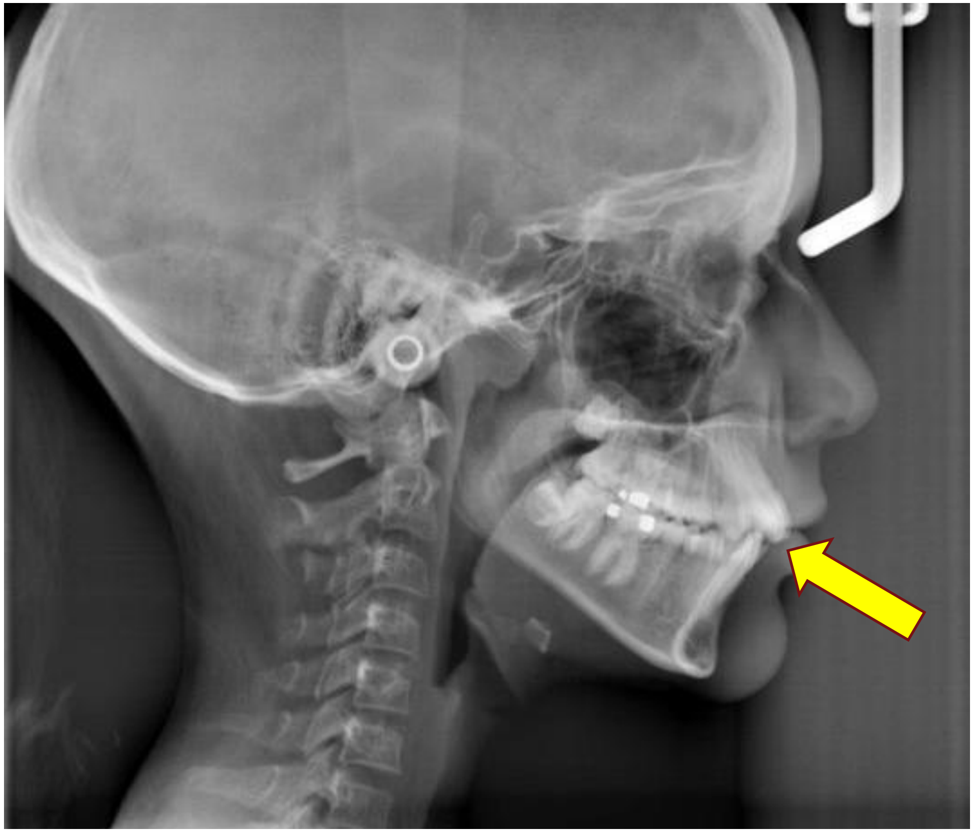




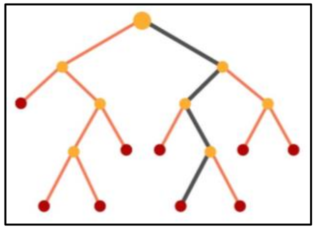
https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-IJCV-01.pdf

Machine learning in medical image analysis

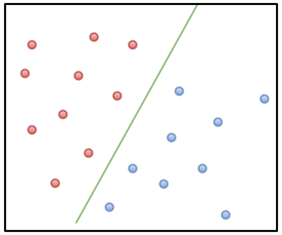
Which hand-crafted features will turn to be useful for a selected classifier?



Decision forests



Support vector machines



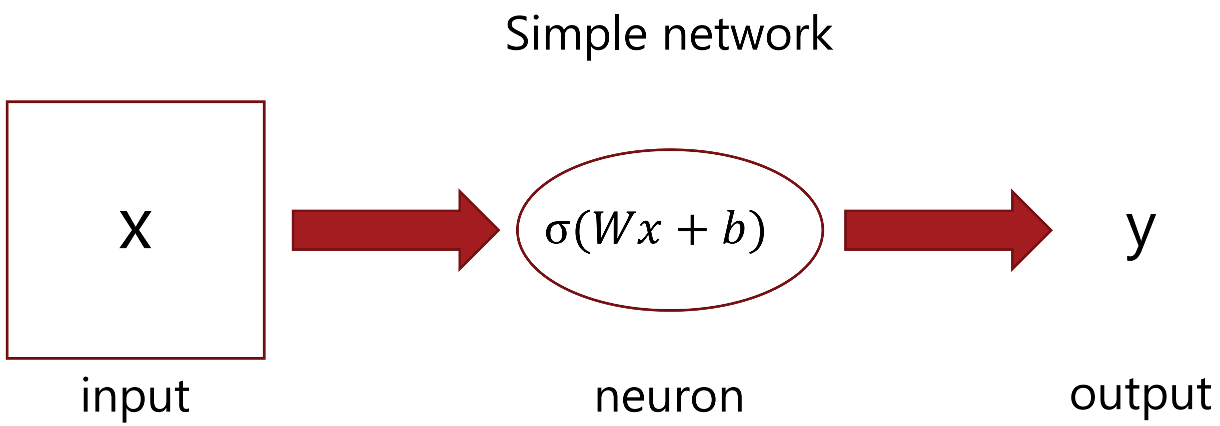
Machine learning in medical image analysis

Hand-crafted features work well when target objects are:

- Oriented and positioned in a predictable way


- Relatively similar to each other in appearance


Neural networks



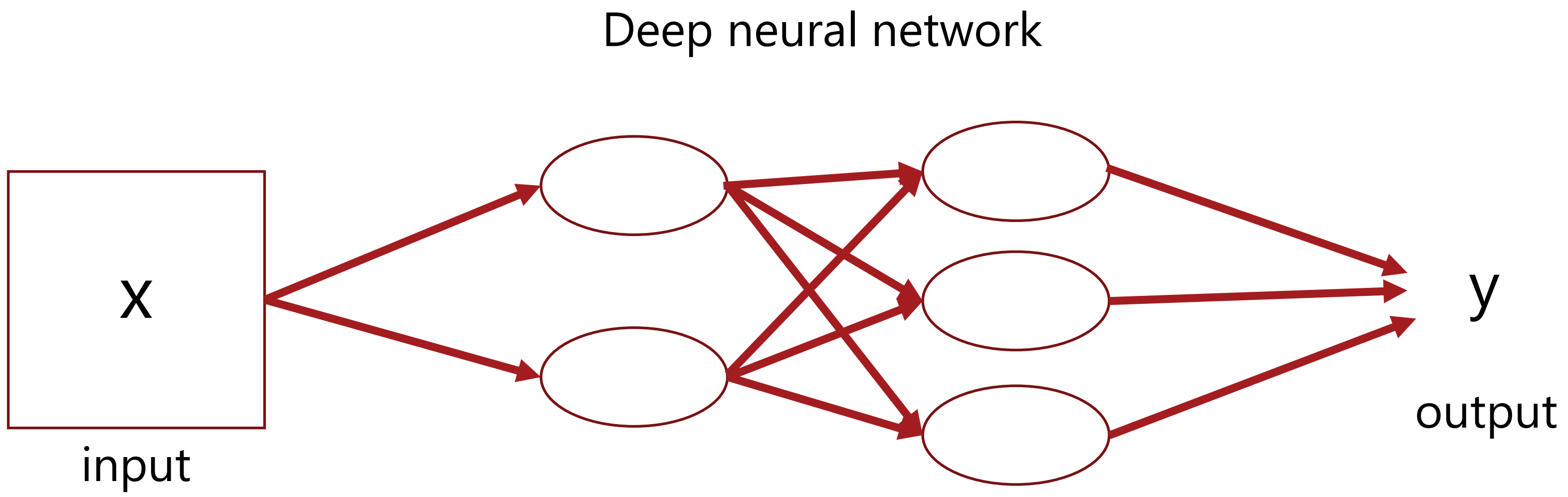
Neuron components:

- Feature -> $W^T x + b$
- Activation -> σ

Neural networks

Deep neural network

- Output of a neuron is a number
- Output of a neuron can be passed to next neurons as an input

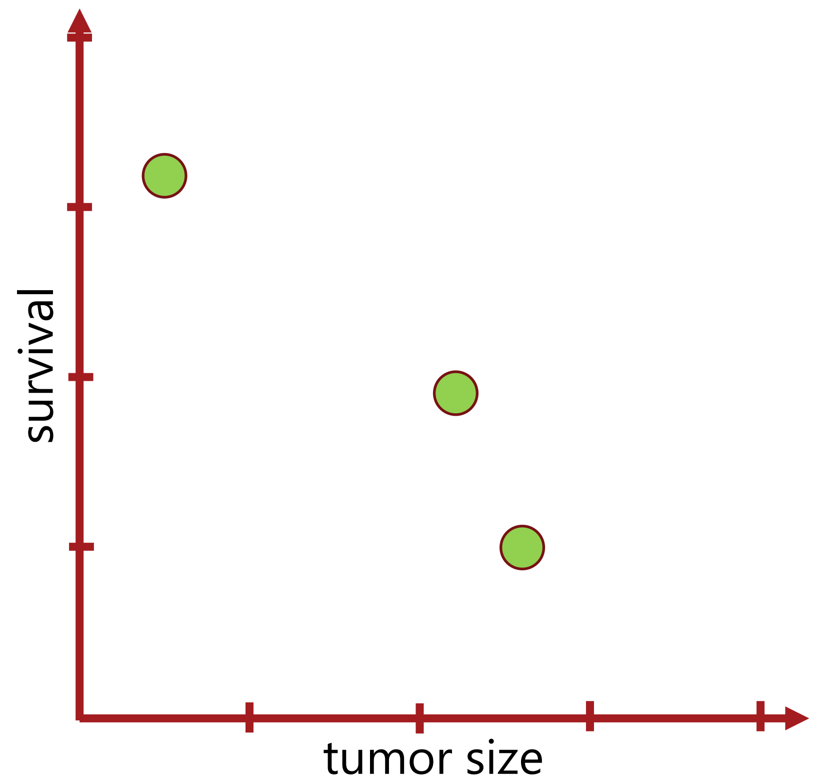


1-neuron network

Patient survival depends on cancer size:

- Bigger tumors are worse
- Can we model this dependency?

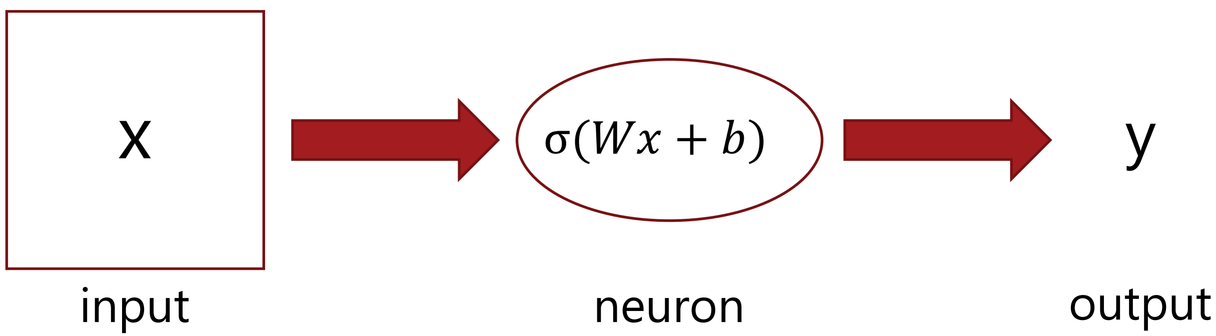
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0



1-neuron network

Network design:

- Input x is 1-dimensional (tumor size)
- Output y is 1-dimensional (survival time)
- Activation function σ is identity function



1-neuron network: initialization

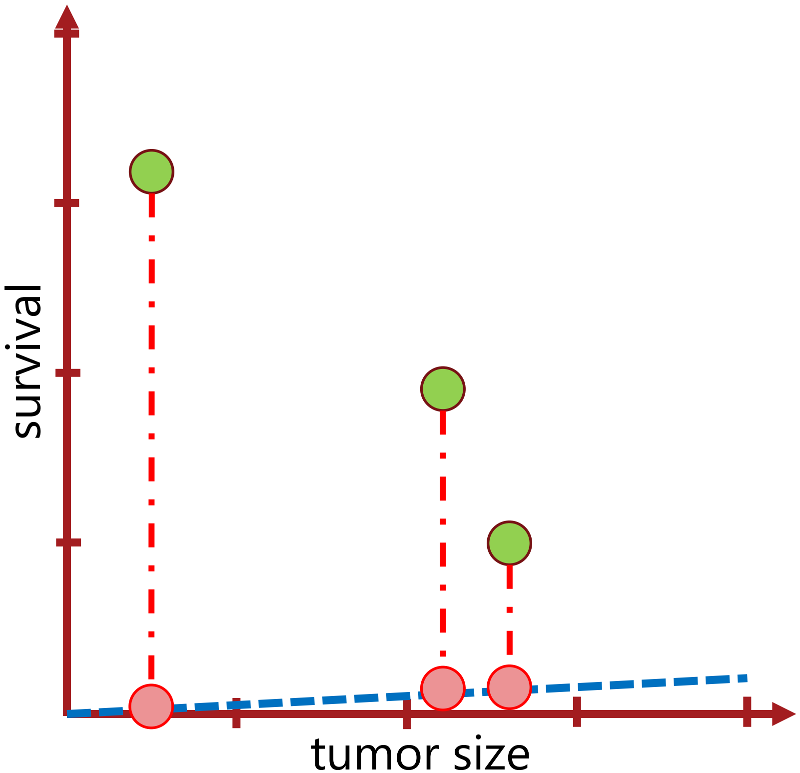
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

Let's initialize our network:

- $W = 0.01;\ b = 0.01$

The performance of the network is low:

- $y_1 = Wx_1 + b = 0.01 * 0.5 + 0.01 = 0.015$
- $y_2 = Wx_2 + b = 0.01 * 2.3 + 0.01 = 0.033$
- $y_3 = Wx_3 + b = 0.01 * 2.9 + 0.01 = 0.039$



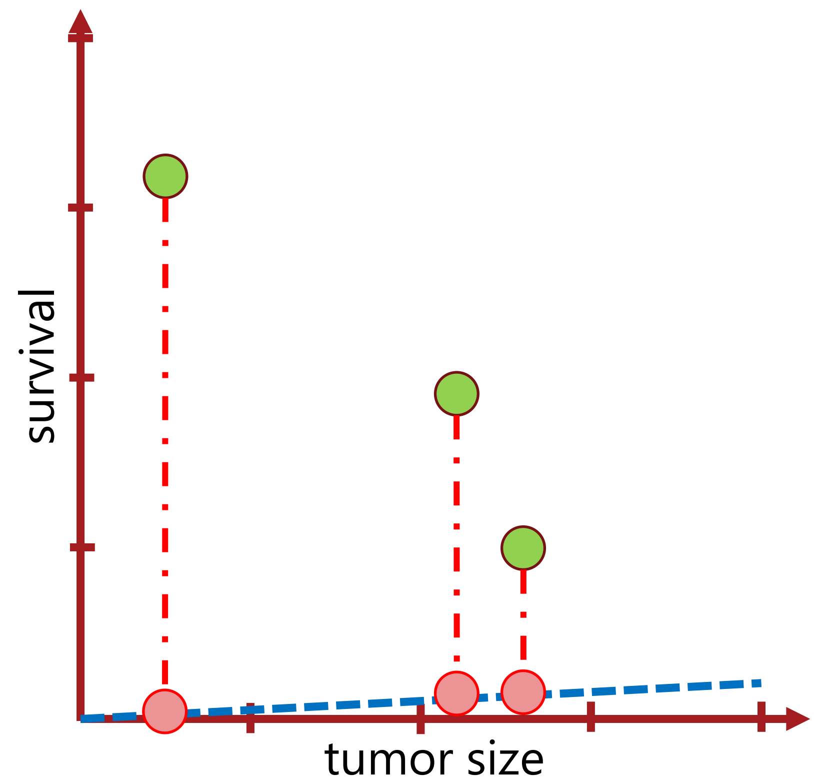
1-neuron network: loss function

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

The performance of the network is low, but how low?

Sum of absolute differences:

- $Loss = \sum_i |y_i' - y_i| = \sum_i |(Wx_i + b_i) - y_i|$
- $Loss = |0.015 - 3.2| + |0.033 - 1.9| + |0.039 - 1.0| = 6.013$



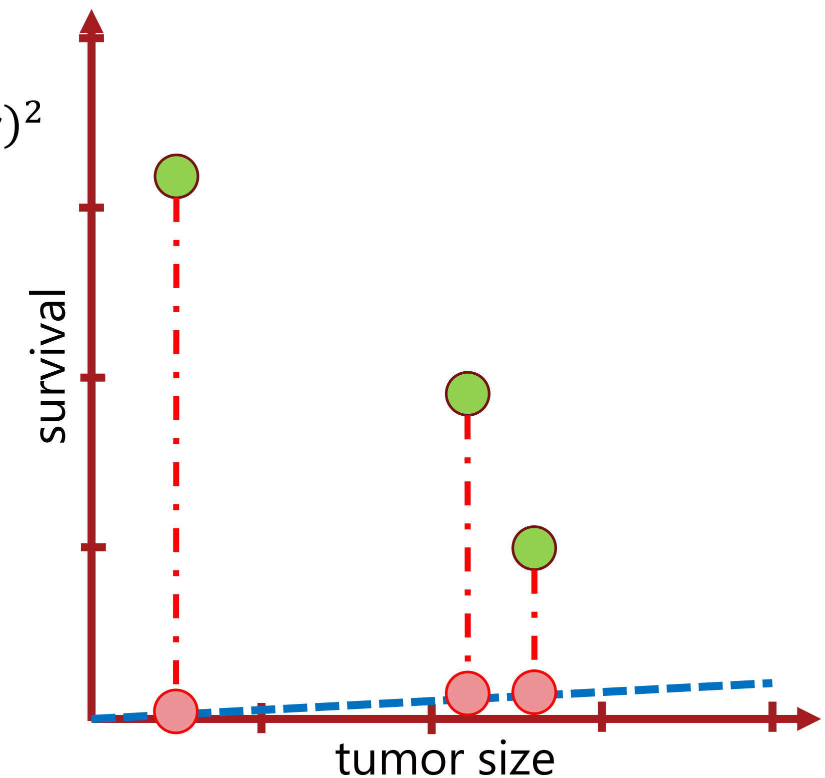
1-neuron network: loss function

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

The performance of the network is low, but how low?

Mean squared error:

- $Loss = \sum_i (y_i' - y_i)^2 = \sum_i ((Wx + b) - y)^2$
- $Loss =$
  $(0.015 - 3.2)^2 +$
  $(0.033 - 1.9)^2 +$
  $(0.039 - 1.0)^2 = 14.55$

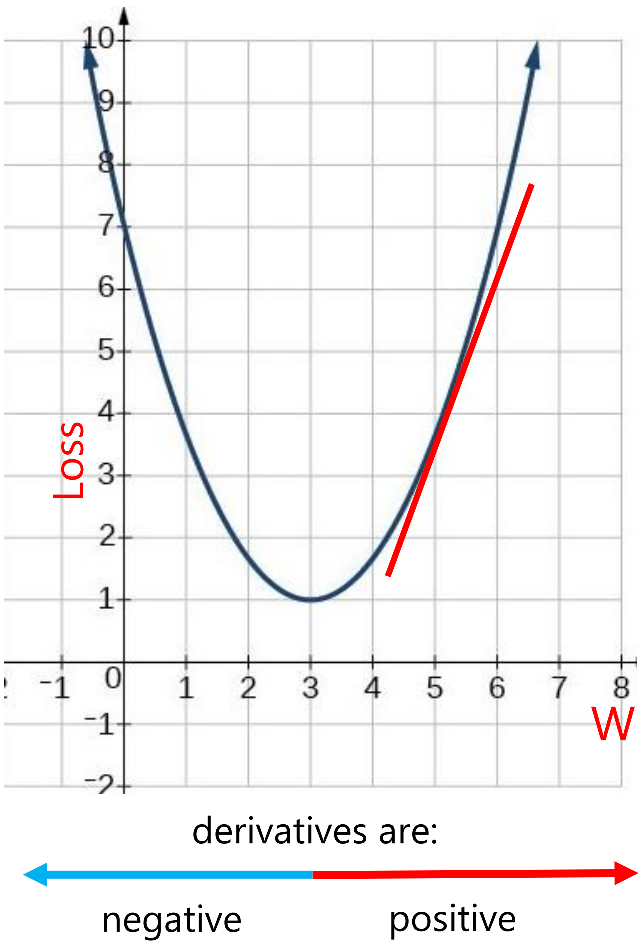


1-neuron network: derivatives

We want the loss to be as small as possible, i.e. find its minimum.

We use derivatives to find minima/maxima of a function:

- How fast function changes
- Will it increase or decrease



- Chain rule:
  - $\frac{\partial}{\partial x} f\big(g(x)\big) = \frac{\partial}{\partial g} f\big(g(x)\big) \cdot \frac{\partial}{\partial x} g(x)$

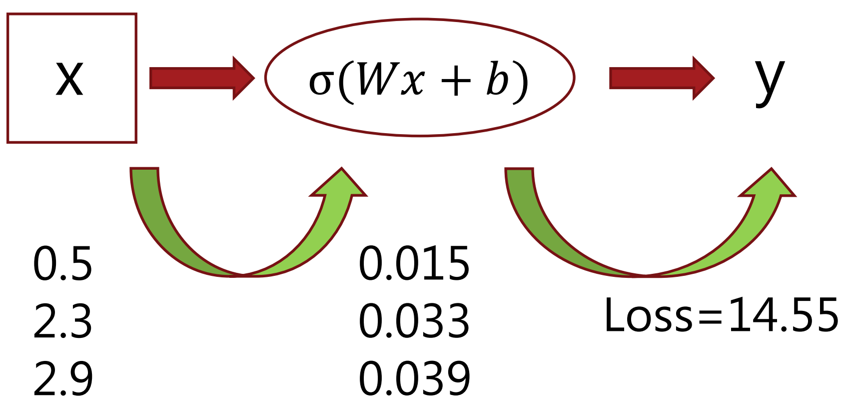
1-neuron network : backpropagation

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

Initialization:

- $W = 0.01, b = 0.01$

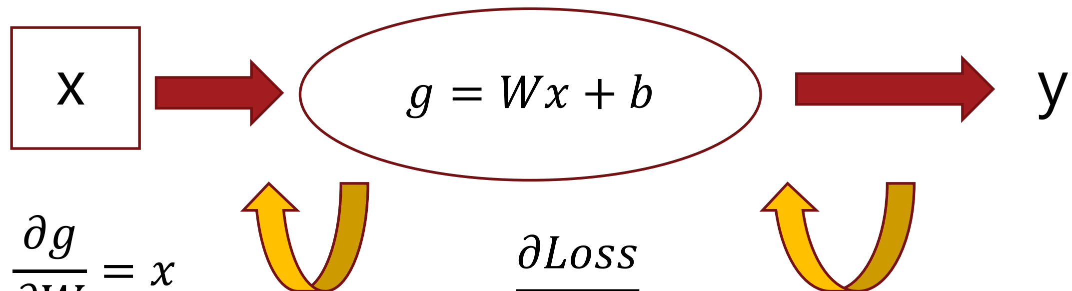
Forward pass:



1-neuron network: backpropagation

Compute the derivatives:

$$Loss = \sum_i (y' - y)^2$$



- How changes in $\sigma(Wx + b)$ affect the loss

$$\frac{\partial g}{\partial W} = x$$

$$\frac{\partial Loss}{\partial g}$$

14.55

- How changes in W and b affect g

$$\frac{\partial g}{\partial b} = 1$$

$$= 2\sum_i (g(x_i) - y_i)$$

- $\frac{\partial Loss}{\partial W}$ and $\frac{\partial Loss}{\partial b}$ will be computed using the chain rule:
  - $\frac{\partial Loss}{\partial W} = 2\sum_i \big((g(x_i) - y_i) \cdot x_i\big)$
  - $\frac{\partial Loss}{\partial b} = 2\sum_i \big((g(x_i) - y_i) \cdot 1\big)$

1-neuron network: optimization step

Update W and b according to the derivatives:

- $W \leftarrow W - \lambda \frac{\partial Loss}{\partial W}$; $b \leftarrow b - \lambda \frac{\partial Loss}{\partial b}$

Learning rate

The results of the optimization step:

- $W \leftarrow W - 0.1 \cdot 2 \sum_i \big((g(x_i) - y_i) \cdot x_i \big) = 0.01 - 0.1 \cdot 2 \cdot \big((0.015 - 3.2)0.5 + (0.033 - 1.9)2.3 + (0.039 - 1.0)2.9 \big) = 0.01 - 0.1 \cdot -8.67 = 0.88$

- $b \leftarrow b - 0.1 \cdot 2 \sum_i \big((g(x_i) - y_i) \big) = 0.61$

W	b
0.01	0.01

	Tumor Size	Survival	First solution
Case1	0.5	3.2	0.015
Case2	2.3	1.9	0.033
Case3	2.9	1.0	0.039

1-neuron network: optimization step

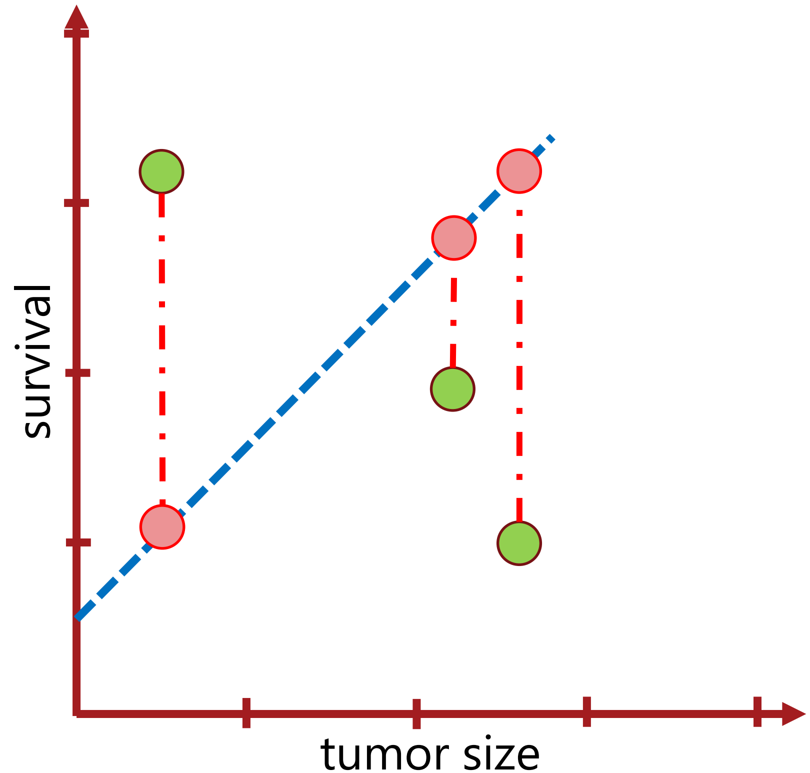
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

After parameter update:

- $W = 0.88;\ b = 0.61$

The performance of the network is better:

- $y_1 = Wx_1 + b = 0.70 * 0.5 + 0.61 = 1.05$
- $y_2 = Wx_2 + b = 0.70 * 2.3 + 0.61 = 2.63$
- $y_3 = Wx_3 + b = 0.70 * 2.9 + 0.61 = 3.16$



Loss improves $Loss\ = 9.8$

1-neuron network: optimization step

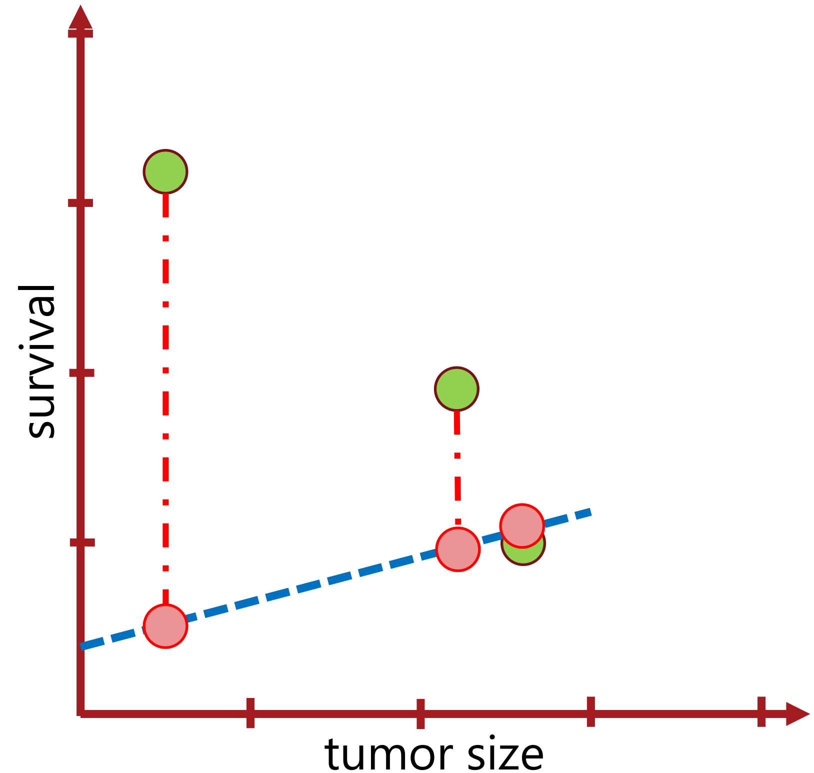
	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0

After next parameter update:

- $W = 0.19;\ b = 0.54$

The performance of the network is better:

- $y_1 = Wx_1 + b = 0.45 * 0.5 + 0.38 = 0.63$
- $y_2 = Wx_2 + b = 0.45 * 2.3 + 0.38 = 0.98$
- $y_3 = Wx_3 + b = 0.45 * 2.9 + 0.38 = 1.10$



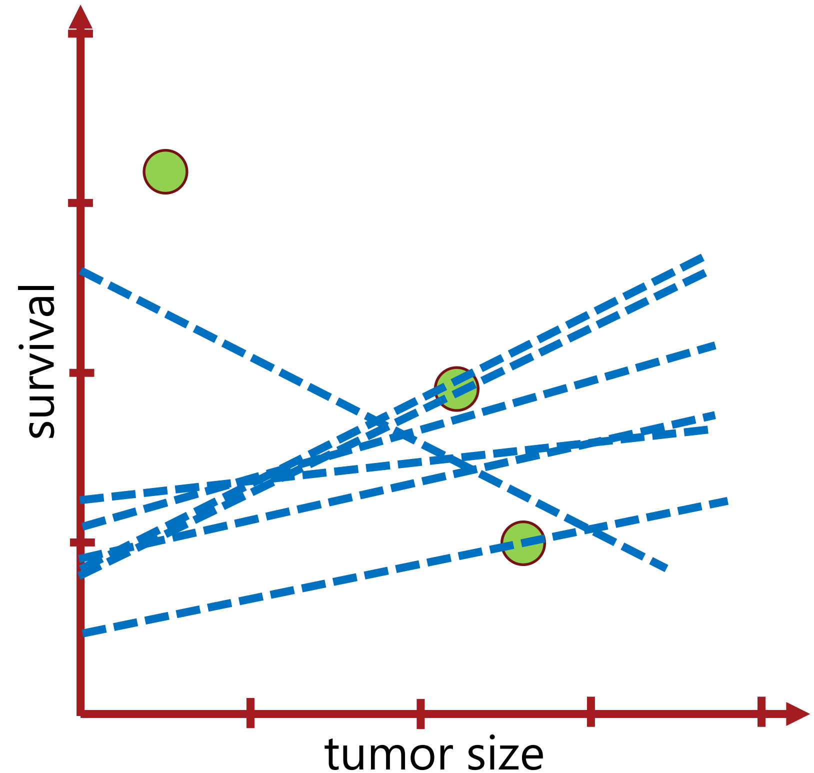
Loss improves $Loss\ = 7.46$

1-neuron network: optimization step

After next parameter update:

- $W = 0.50;\ b = 0.88, Loss = 6.07$
- $W = 0.19;\ b = 0.50, Loss = 7.75$
- $W = 0.53;\ b = 0.85, Loss = 6.29$
- $W = 0.19;\ b = 0.90, Loss = 5.37$
- $W = 0.29;\ b = 1.13, Loss = 4.67$
- $W = 0.12;\ b = 1.23, Loss = 4.14$
- …
- $W = 0.04;\ b = 1.51, Loss = 3.27$
- $W = -0.15;\ b = 1.98, Loss = 2.06$
- $W = -0.22;\ b = 2.18, Loss = 1.63$
- $W = -0.29;\ b = 2.35, Loss = 1.31$
- $W = -0.36;\ b = 2.50, Loss = 1.04$
- $W = -0.43;\ b = 2.63, Loss = 0.82$

	Tumor Size	Survival
Case1	0.5	3.2
Case2	2.3	1.9
Case3	2.9	1.0



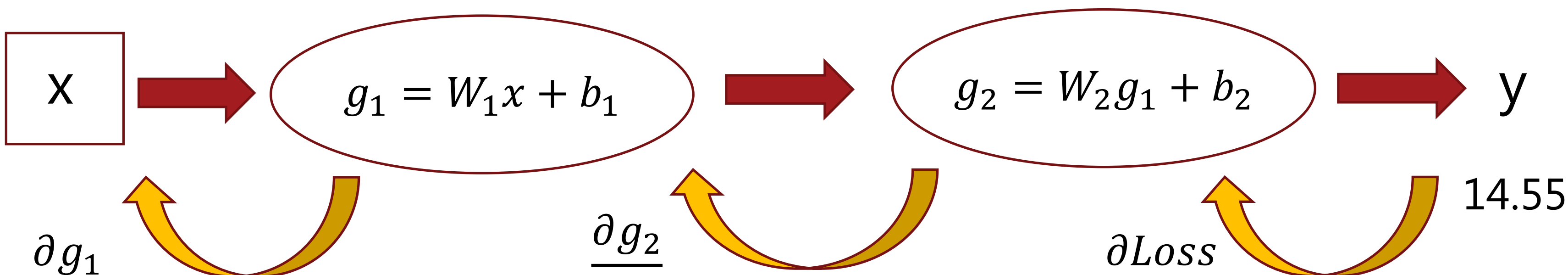
More neurons

The chain rule can be applied for longer chains of neurons:

$$Loss = \sum_i (g_2(g_1(x_i)) - y_i)^2$$

Let's initialize our network:

- $\boxed{W_1 = \quad ; b_1 = \quad}$ $\quad W_2 = 0.01; b_2 = 0.01$



$$\frac{\partial g_1}{\partial W_1} = x \qquad \frac{\partial g_2}{\partial g_1} = W_2 \qquad \frac{\partial Loss}{\partial g_2} = 2\sum_i (g_2(g_1(x_i)) - y_i)$$

$$\frac{\partial Loss}{\partial W_1} = 2\sum_i (g_2(g_1(x_i)) - y_i) W_2 x$$

More neurons: parameter update

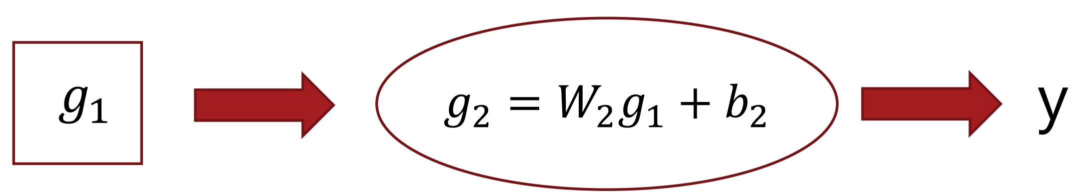
We need to update all network parameters W_1, b_1, W_2, b_2:

- First we update the parameters on the lowest level: W_1, b_1

$$\frac{\partial Loss}{\partial W_1}; \frac{\partial Loss}{\partial b_1}$$

- Then we move to the next level: W_2, b_2

$$\frac{\partial Loss}{\partial W_2}; \frac{\partial Loss}{\partial b_2}$$

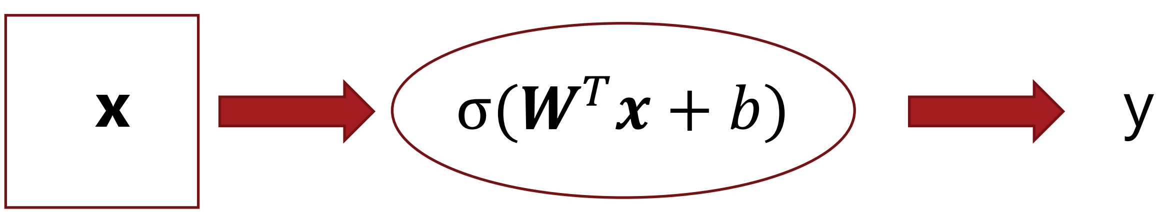


Multidimensional input

$\boldsymbol{W}$ will be in agreement with input $\boldsymbol{x}$ dimensions :

	Tumor Size	Temperature	Survival
Case1	0.5	37.6	3.2
Case2	2.3	39.1	1.9
Case3	2.9	36.2	1.0

- $g = \sigma(W_1(tumor_{size}) + W_2(temperature) + b)$



Activation function

Let's compute this:

$$x \rightarrow g_1 = W_1x + b_1 \rightarrow g_2 = W_2g_1 + b_2 \rightarrow y$$

$$y' = g_2\big(g_1(x)\big) = W_2(W_1x + b_1) + b_2 = W_2W_1x + (W_2b_1 + b_2)$$

Can we simplify the network?

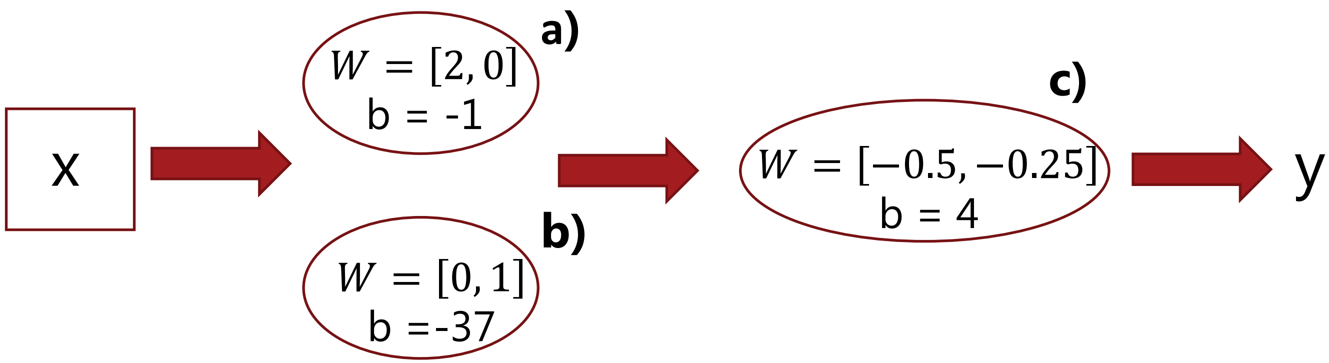
$$\Downarrow$$

$$x \rightarrow g = W_2W_1x + (W_2b_1 + b_2) \rightarrow y$$

The network is not deep at all

Activation function

On high level, what the following neurons try to capture?



Neuron a) gets excited about

Neuron b) gets excited about

Neuron c) gets predicts

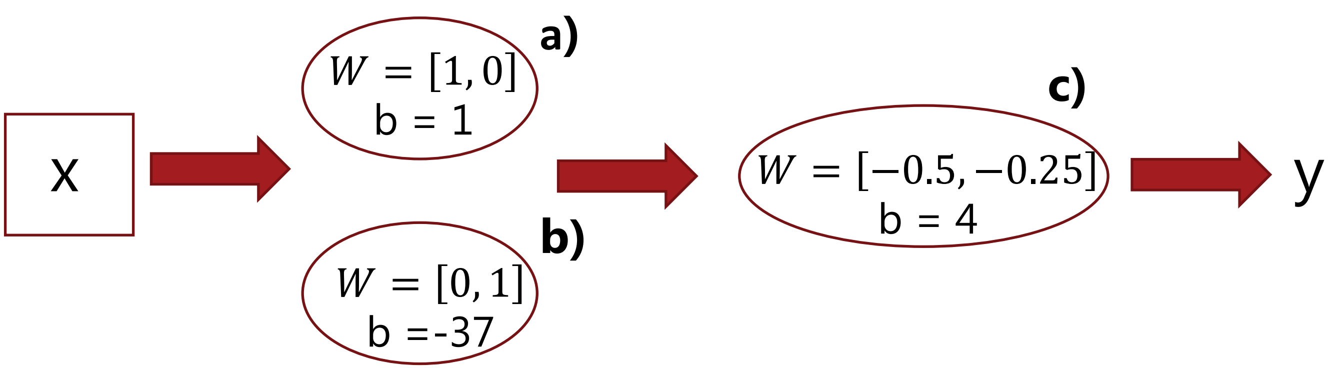
	Tumor Size	Temperature	Survival
Case1	0.5	37.6	3.2
Case2	2.3	39.1	1.9
Case3	2.9	36.2	1.0

Activation function

Let's add some more data into the database

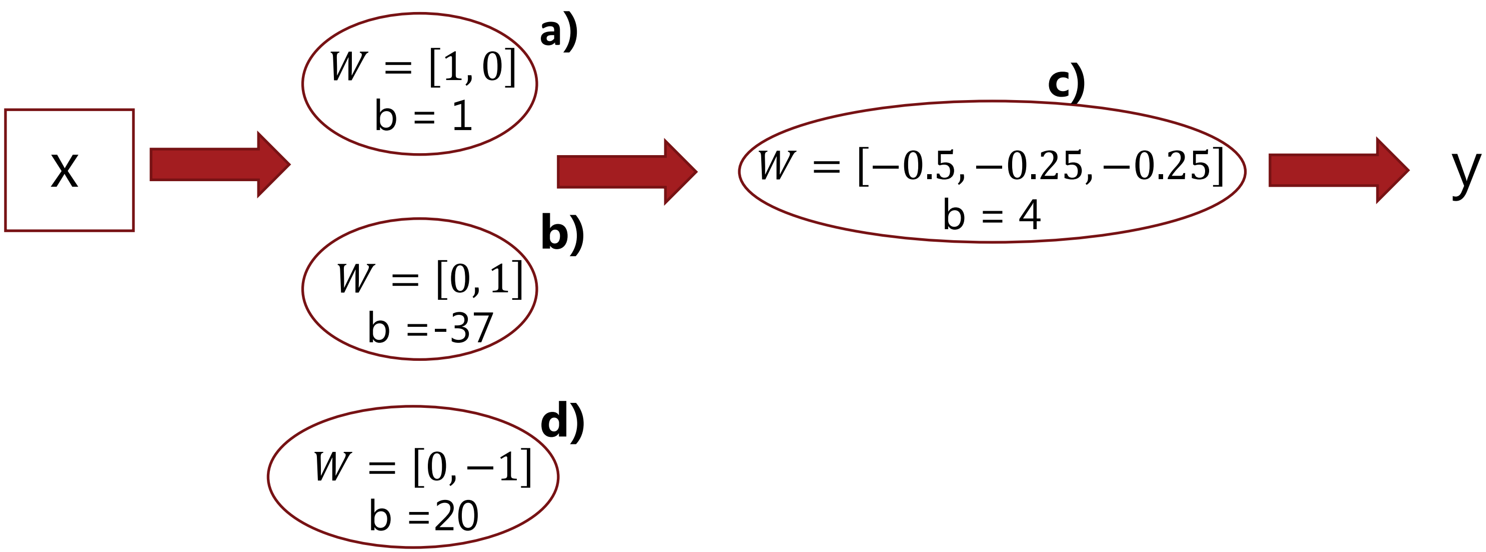
What will happen with the network behavior?

	Tumor Size	Temperature	Survival
Case1	0.5	37.6	3.2
Case2	2.3	39.1	1.9
Case3	2.9	36.2	1.0
Case4	2.8	22	0.0
Case5	2.1	23	0.0



Activation function

Maybe we can add a neuron to catch dead patients



Will this new neuron fix the network behavior on dead patients?

Activation function

Neuron

b) $W = [0, 1]$ b =-37

d) $W = [0, -1]$ b =20

	Tumor Size	Temperature	Survival
Case2	2.3	39.1	1.9
Case5	2.1	23	0.0

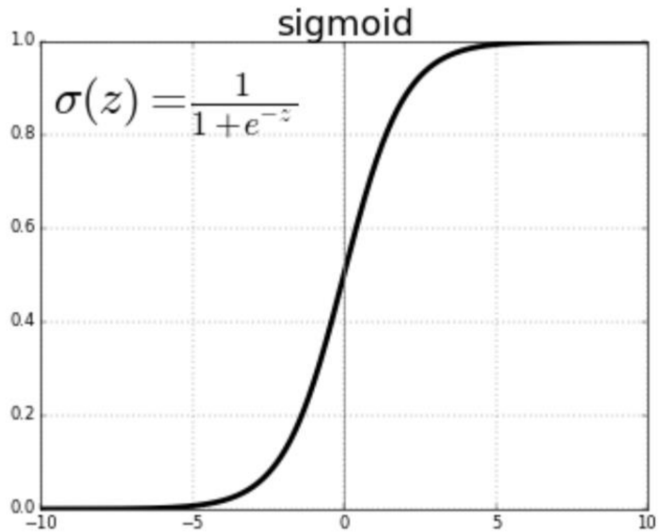
b)	d)
2.1	-19.1
-14	3

The idea of neuron b) is to capture fever and not to care about low temperature

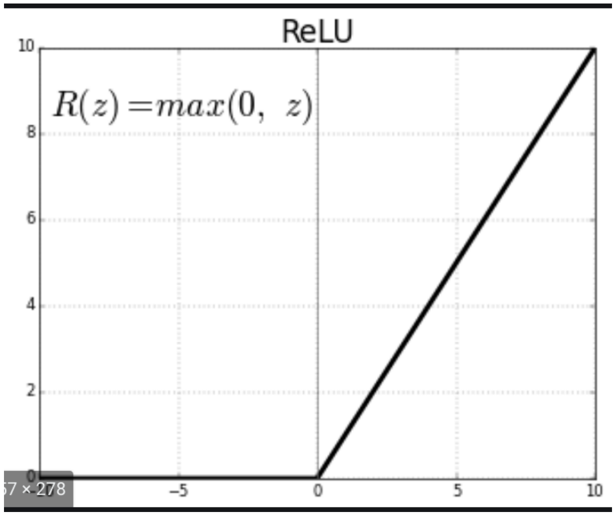
The idea of neuron d) is to capture dead body temperature and not to care about fever

Activation function

Sigmoid activation:



ReLU:



Regularization

Let's augment our loss functions:

$$Loss = \sum_{i} (y' - y)^2 + \mu \sum_{j} |W_j|$$

$$Loss = \sum_{i} (y' - y)^2 + \mu \sum_{j} (W_j)^2$$

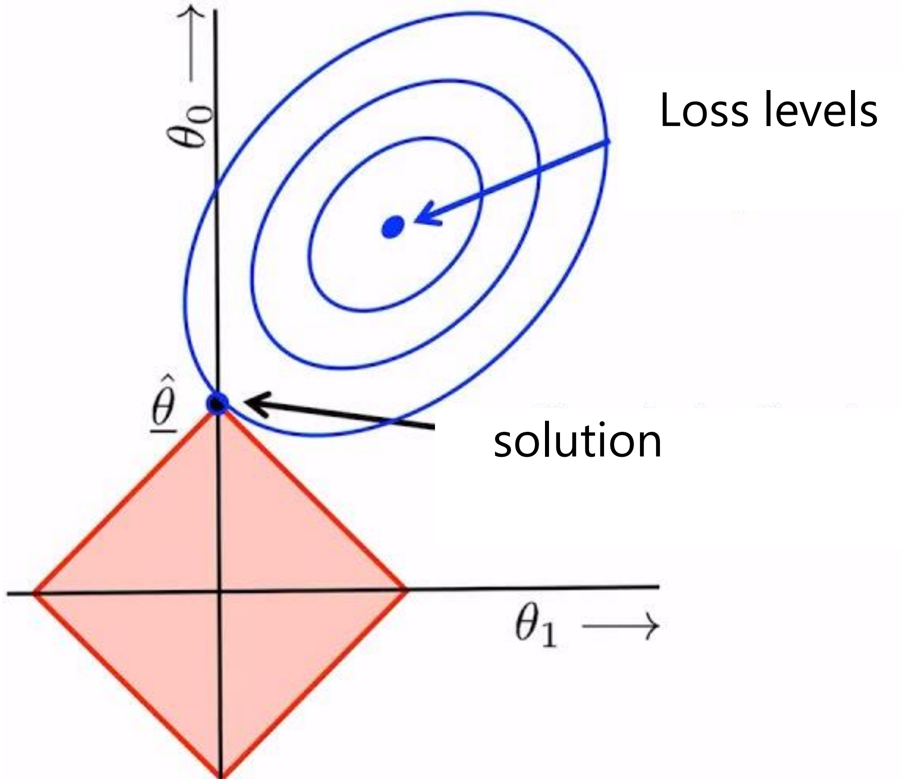
What will these additional terms do?

Regularization

L1 regularization tries to minimize the number of features used in the analysis

More features you use – the more likely it is to find some dependencies.

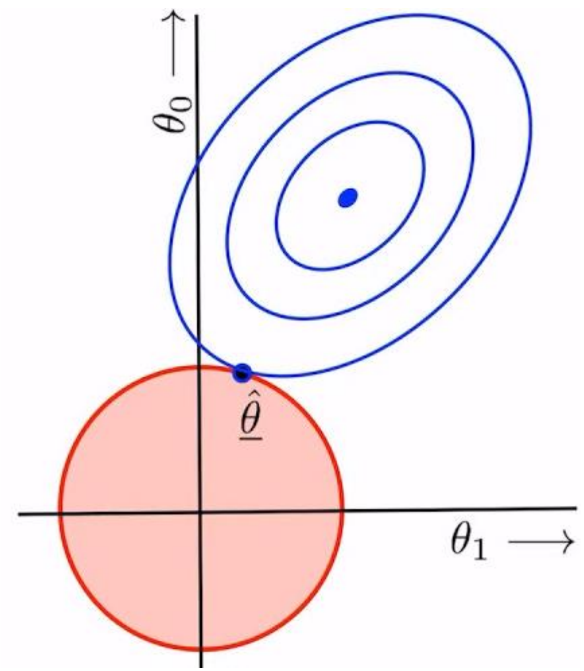
One person with a rare name X is carpenter, dose this mean that X can predict the profession?



Regularization

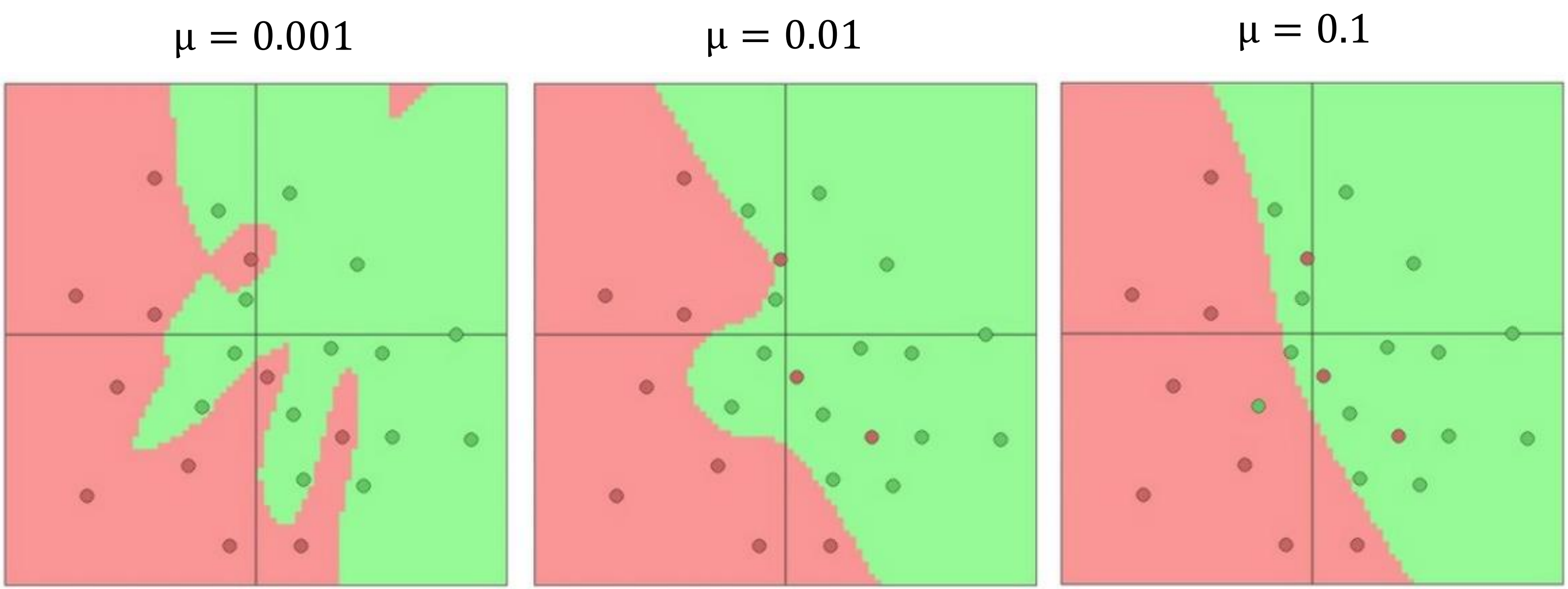
L2 regularization tries to use all features equally

Do not rely on one feature too much, you can get outliers.



Brief CNNs: L2 regularization

- Regularization makes your network more robust

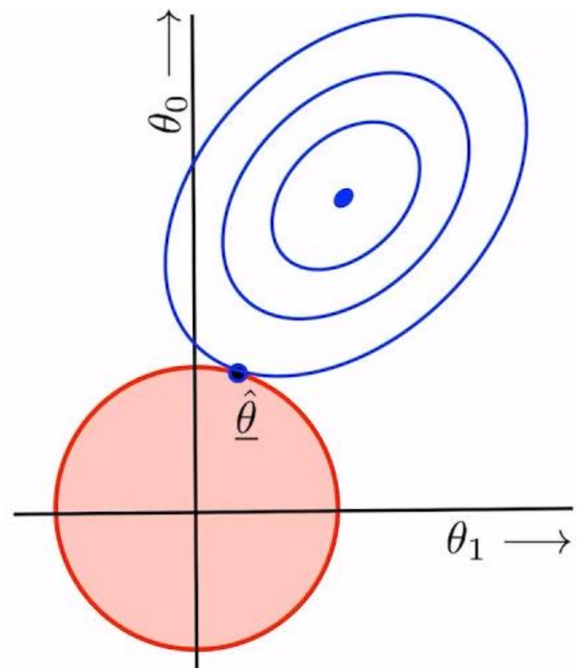


http://cs231n.github.io/neural-networks-1/

Regression/classification

L2 regularization tries to use all features equally

Do not rely on one feature too much, you can get outliers.



Questions?