Basics of neural network

Outline

- Basic operations of deep model
- Basics of optimization
- Optimization for deep models

Machine Learning

$x \longrightarrow \boxed{F(X;\ W)} \longrightarrow y$

y { Class label (Classification) / Vector (Estimation) }

Object recognition →

{dog, cat, horse, flower, …}

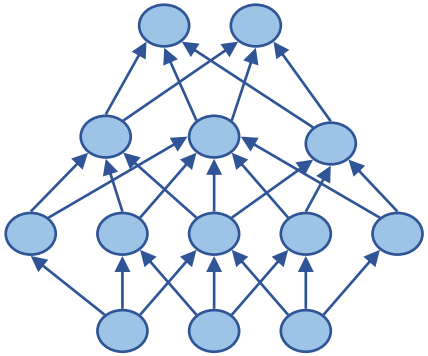
Low-resolution image

Super resolution →

High-resolution image

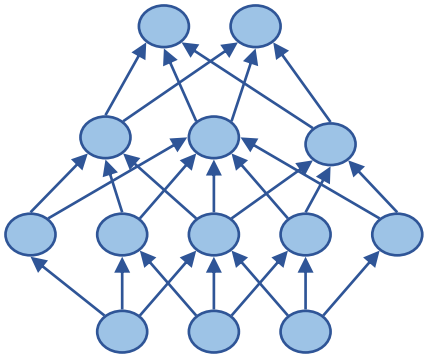
Neural network

Multilayer Perception (MLP)



x input

Neural network

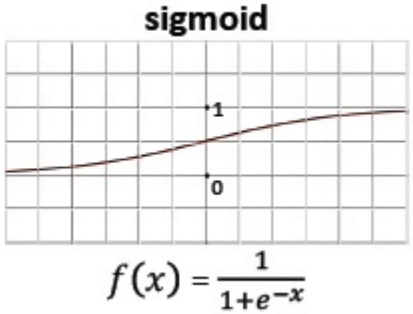


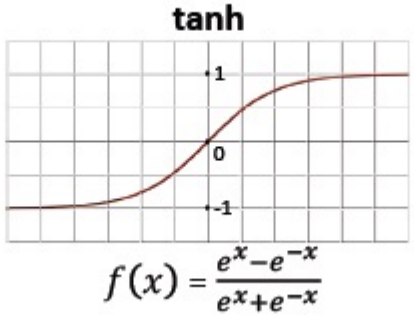
$\mathbf{h_1}=f(\mathbf{W_1 x})$

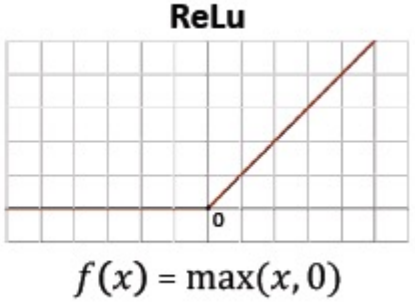
$\mathbf{x}$

f: Non-linear/activation function

sigmoid: The larges response is 1, the smallest response is 0.
tanh: The larges response is 1, the smallest response is -1
Relu (rectified linear unit): Thresholding, when smaller than 0, set to 0.

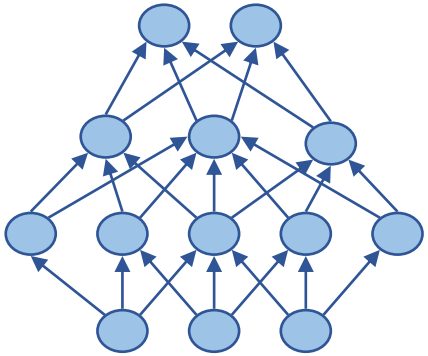






Neural network

Multilayer Perception (MLP)



$\mathbf{h}_2 = f(\mathbf{W}_2\mathbf{h}_1)$

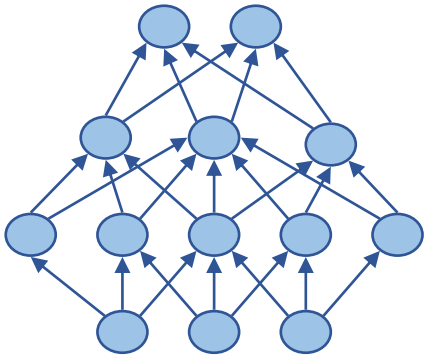
$\mathbf{h}_1 = f(\mathbf{W}_1\mathbf{x})$

$\mathbf{x}$

f: Non-linear function

Neural network

Multilayer Perception (MLP)



$\mathbf{y}=f(\mathbf{W}_3\mathbf{h}_2)$

$\mathbf{h}_2=f(\mathbf{W}_2\mathbf{h}_1)$

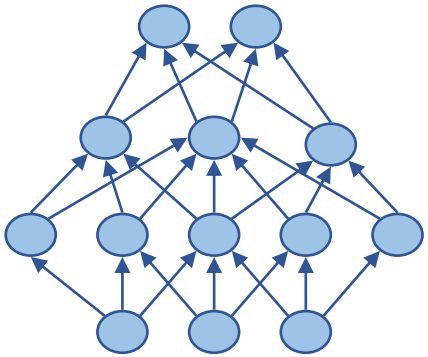
$\mathbf{h}_1=f(\mathbf{W}_1\mathbf{x})$

$\mathbf{x}$

f: Non-linear function

Neural network

Multilayer Perception (MLP)



$\mathbf{y}=f(\mathbf{W}_3\mathbf{h}_2)$

$\mathbf{h}_2=f(\mathbf{W}_2\mathbf{h}_1)$

$\mathbf{h}_1=f(\mathbf{W}_1\mathbf{x})$

$\mathbf{x}$

apple

pear

Neural network

Back propagation

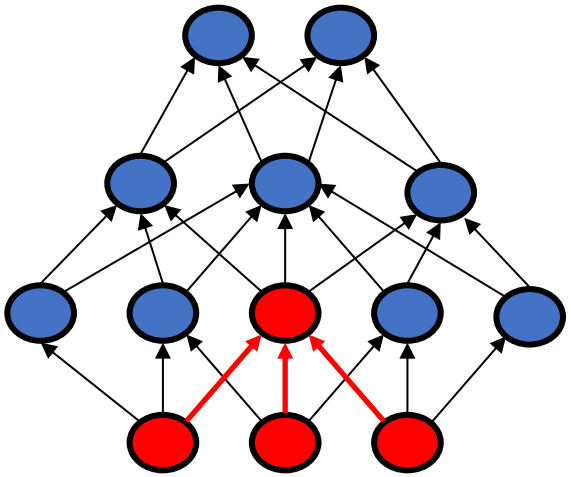
Nature

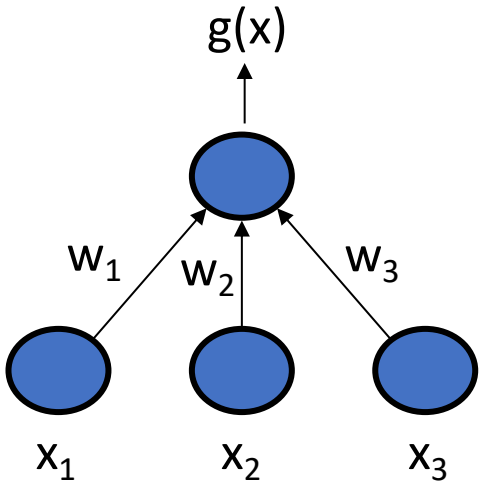
Geoffrey Hinton

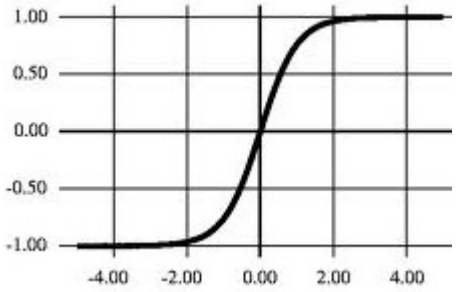
1940s

1986

Lecun Yann joined Geoffery Hinton's group in 1987

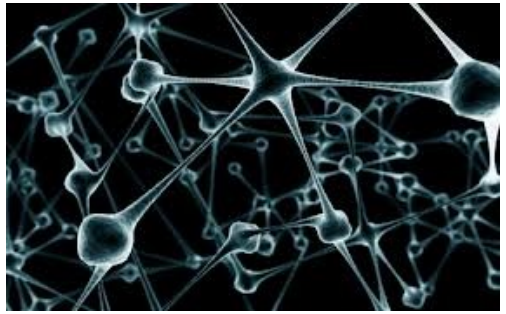






f(*net*)

$$g(\mathbf{x}) = f\left(\sum_{i=1}^{d} x_i w_i + w_0\right) = f(\mathbf{w}^t \mathbf{x})$$



Neural network

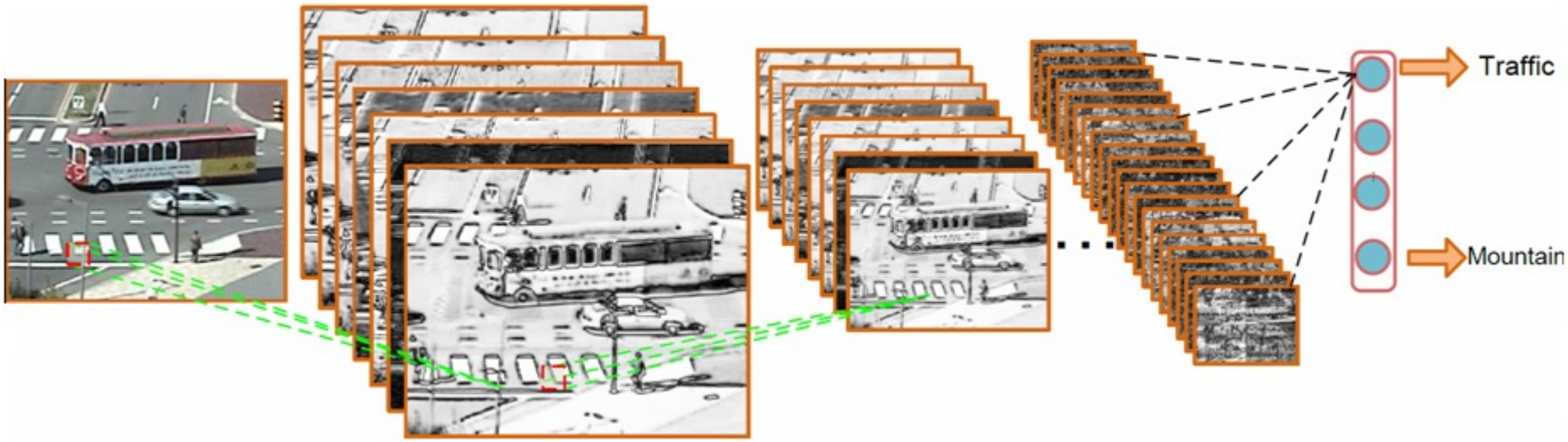
Back propagation

Convolutional neural network

1940s

1986

1998



Neural network

Back propagation

Convolutional neural network

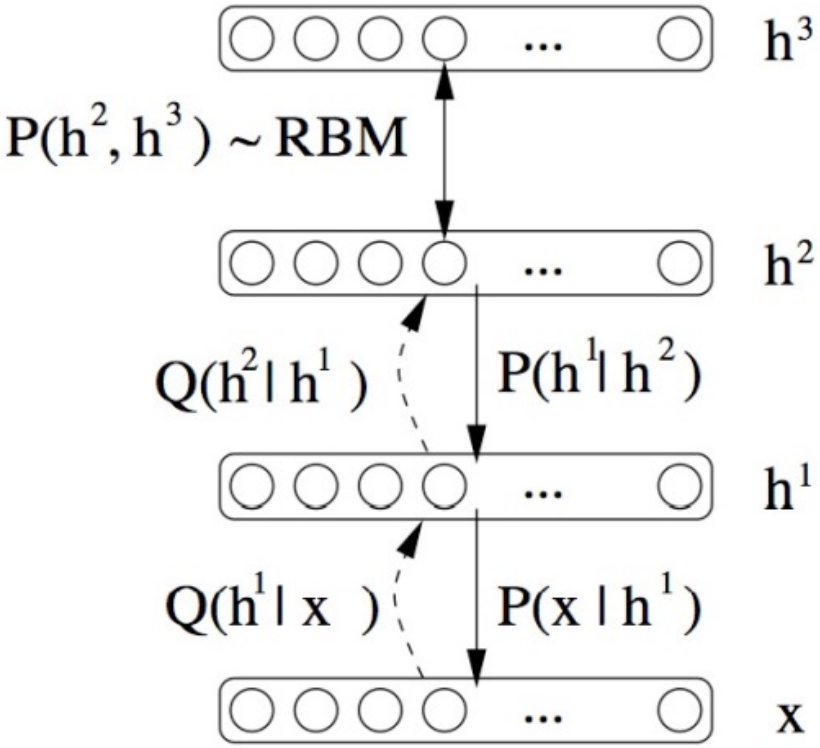
Deep belief net

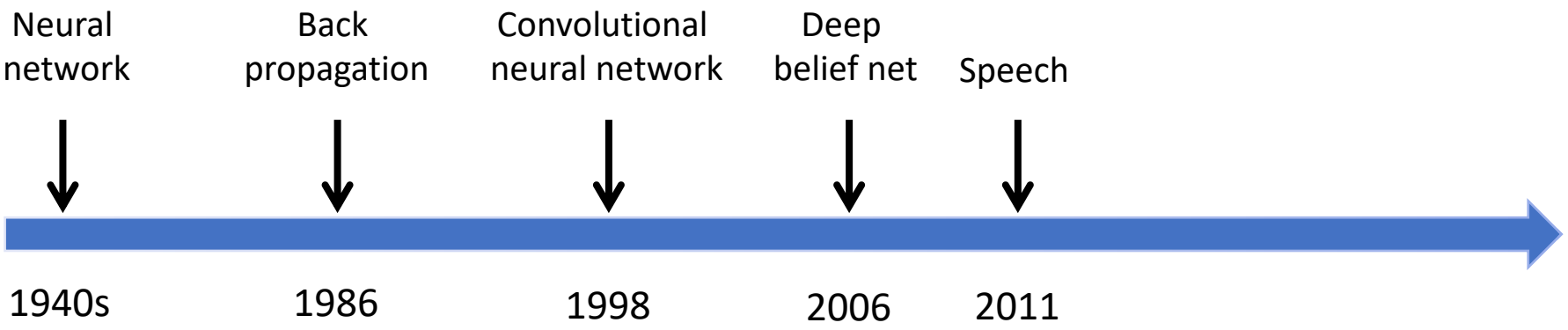
1940s

1986

1998

2006





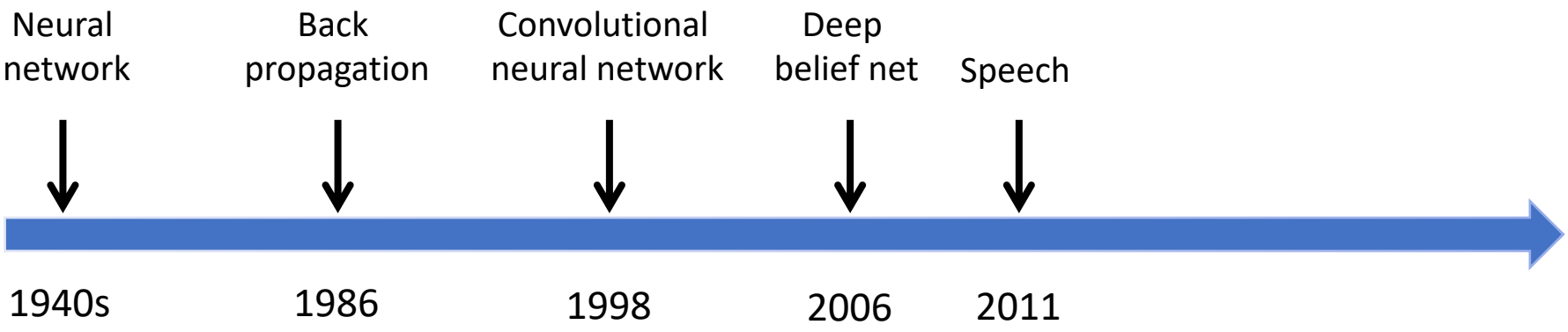
deep learning results

task	hours of training data	DNN-HMM	GMM-HMM with same data
Switchboard (test set 1)	309	18.5	27.4
Switchboard (test set 2)	309	16.1	23.6
English Broadcast News	50	17.5	18.8
Bing Voice Search (Sentence error rates)	24	30.4	36.2
Google Voice Input	5,870	12.3	
Youtube	1,400	47.6	52.3

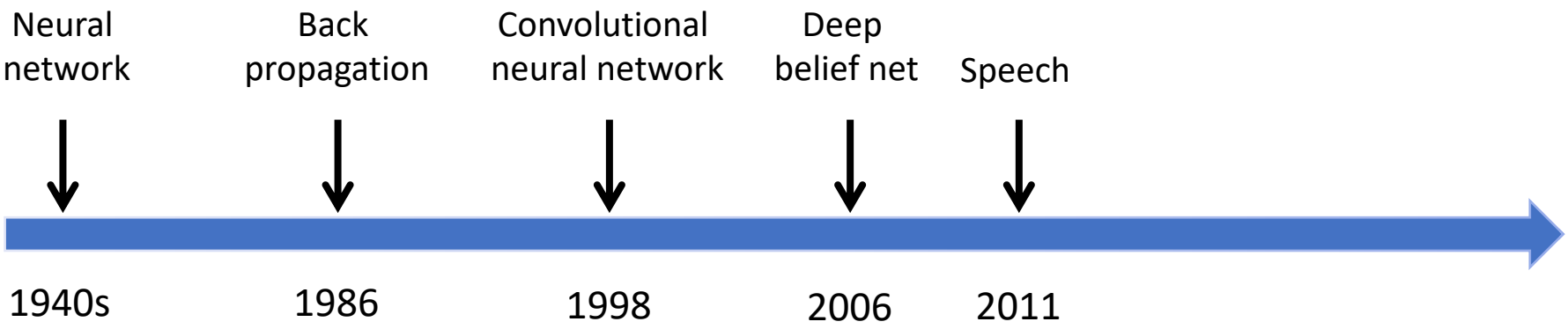
Deep Networks Advance State of Art in Speech

Deep Learning leads to breakthrough in speech recognition at MSR.

Microsoft



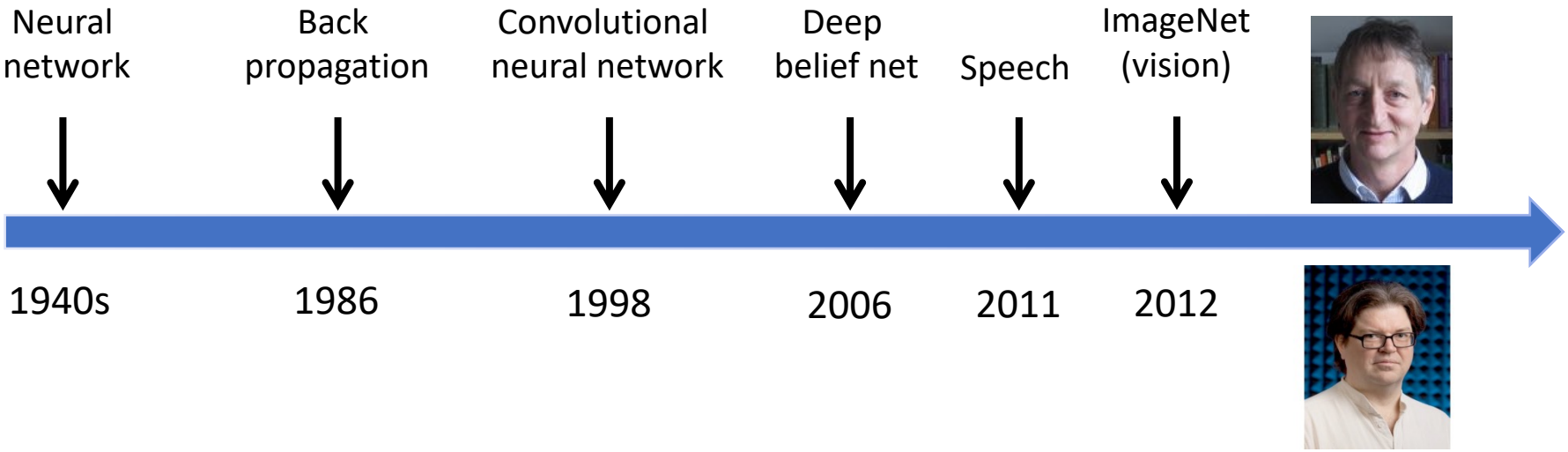
Not well accepted by the vision community ☹



LeCun's open letter in CVPR 2012

So, I'm giving up on submitting to computer vision conferences altogether. CV reviewers are just too likely to be clueless or hostile towards our brand of methods. Submitting our papers is just a waste of everyone's time (and incredibly demoralizing to my lab members)

I might come back in a few years, if at least two things change:
- Enough people in CV become interested in feature learning that the probability of getting a non-clueless and non-hostile reviewer is more than 50% (hopefully [Computer Vision Researcher]'s tutorial on the topic at CVPR will have some positive effect).
- CV conference proceedings become open access.



Rank	Name	Error rate	Description
1	**U. Toronto**	0.15315	Deep learning
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

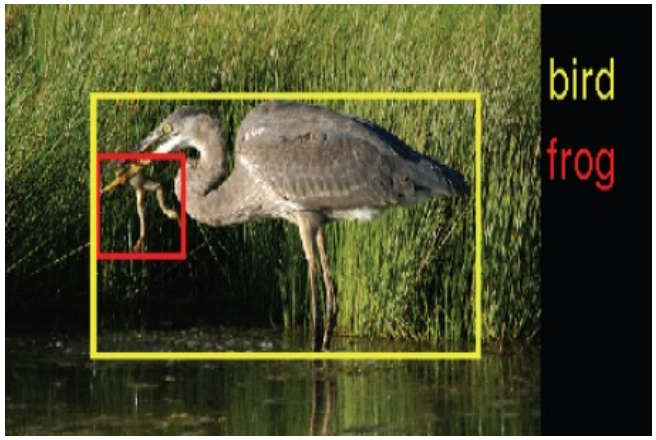
Current best result < 0.03

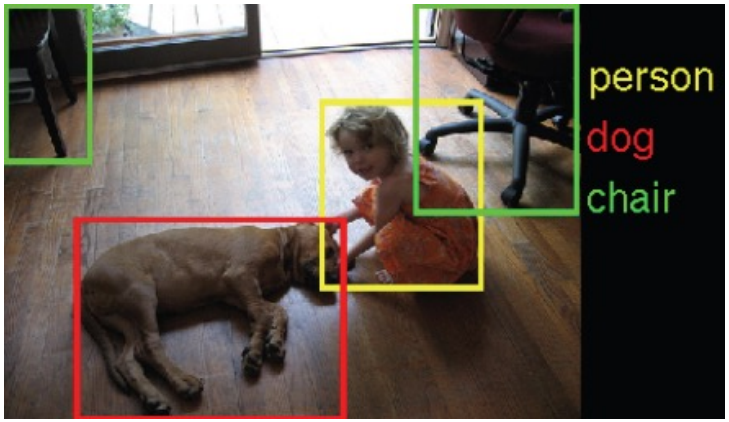
A. Krizhevsky, L. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, 2012.

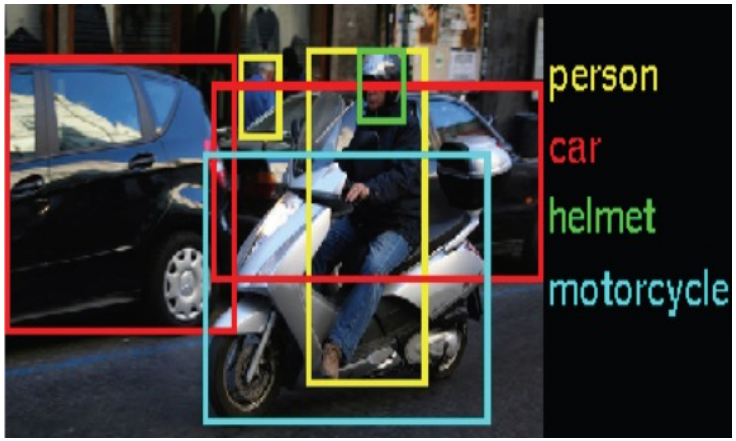
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

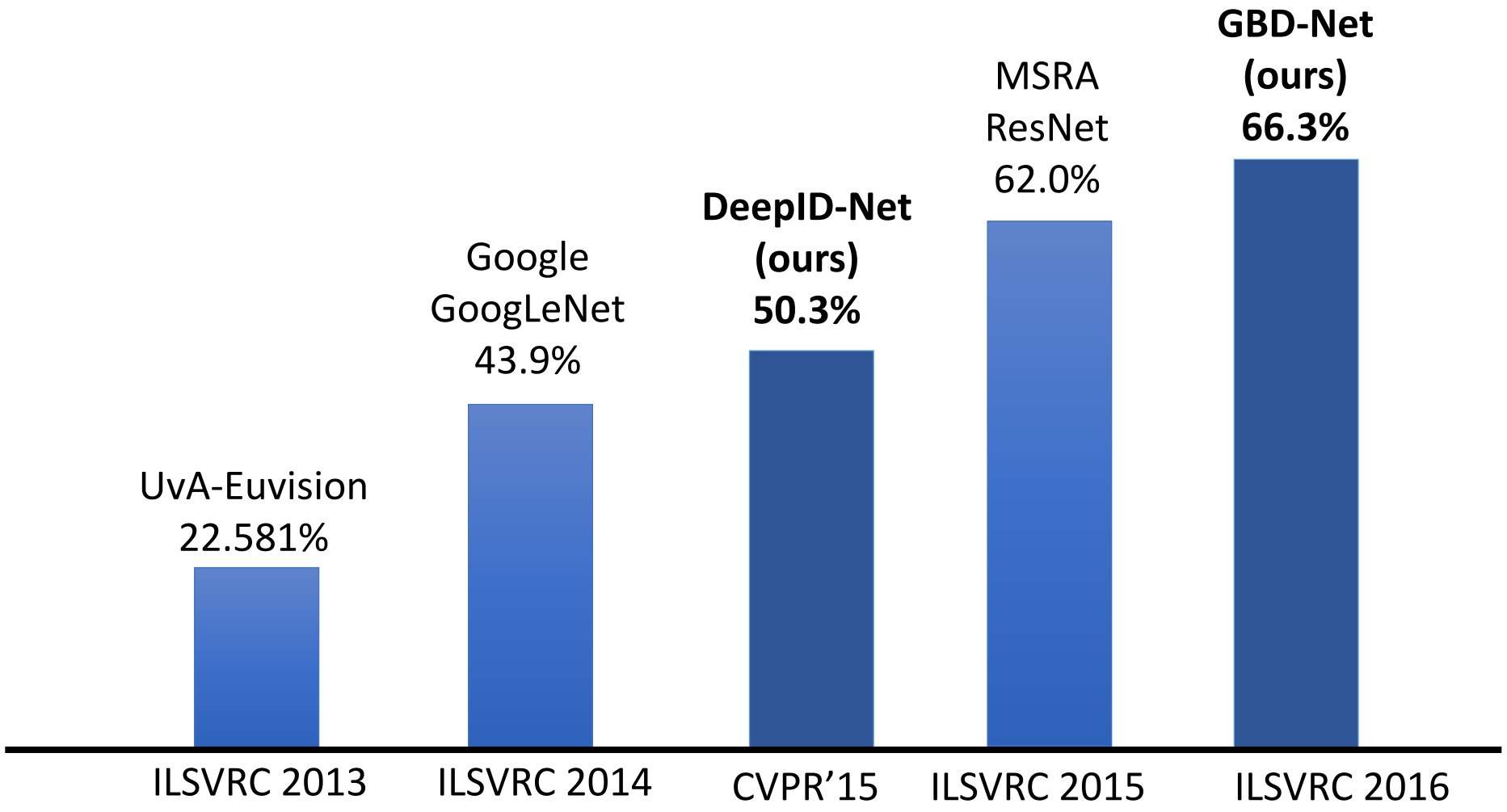
ImageNet Object Detection Task

- 200 object classes
- 60,000 test images







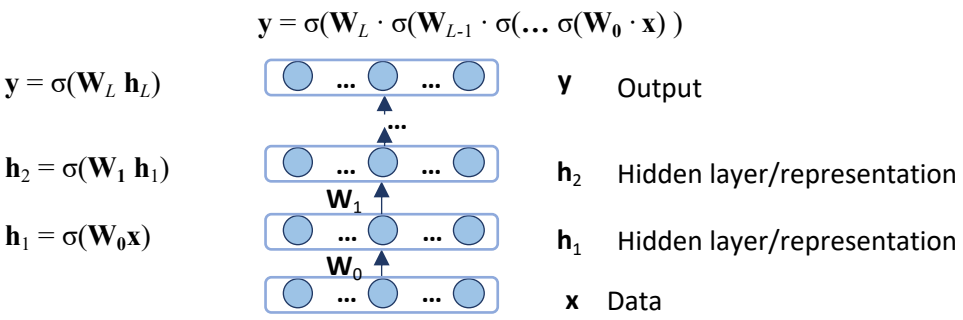


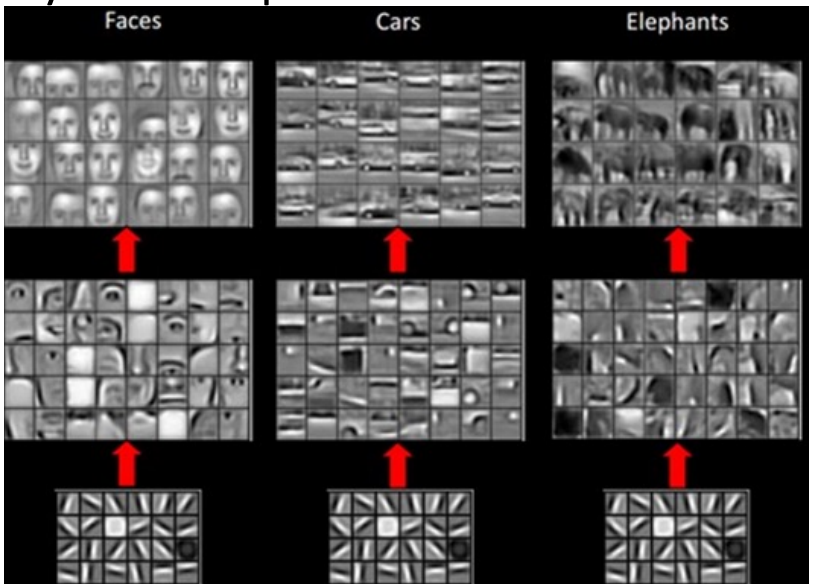
W. Ouyang and X. Wang, et al. "DeepID-Net: Deformable Deep Convolutional Neural Networks for Object Detection," CVPR 2015/TPAMI17

X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang, "Gated Bi-directional CNN for Object Detection," ECCV 2016/ /TPAMI17

Deep learning

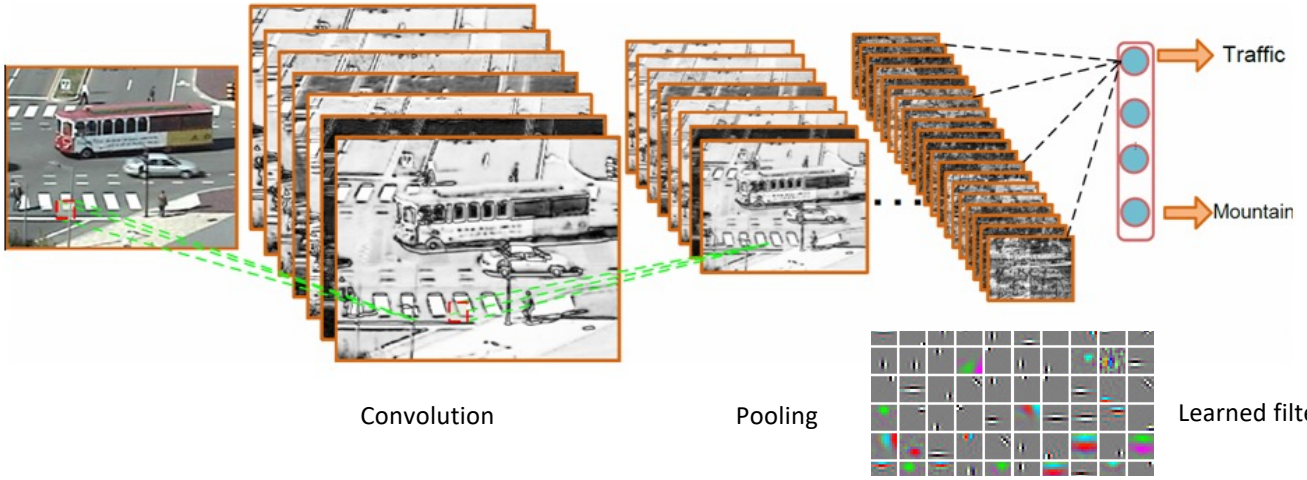
- Learning of multi-layer non-linear transformations
- Multiple layers have multiple levels of representations with multiple non-linear transformations.
- Higher level representations are derived from lower level representations to form a hierarchical representation.
- Multiple levels of representations correspond to different levels of abstraction; the levels form a hierarchy of concepts.





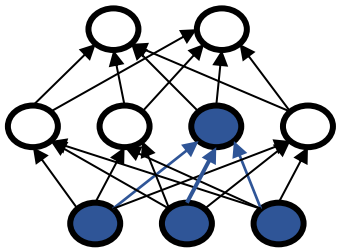
Classical Deep Models

- Convolutional Neural Networks (CNN)
  - First proposed by Fukushima in 1980
  - Improved by LeCun, Bottou, Bengio and Haffner in 1998



Comparison of fully connected network and convolutional neural network

- Fully connected network (FC)
  - Each neuron at the current layer is connected with all the neurons in the previous layer



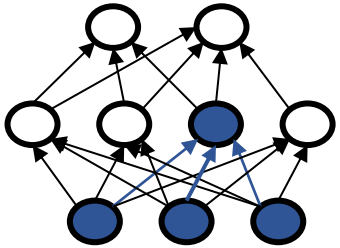
$\mathbf{y}=f(\mathbf{W}_2\mathbf{h}_1)$

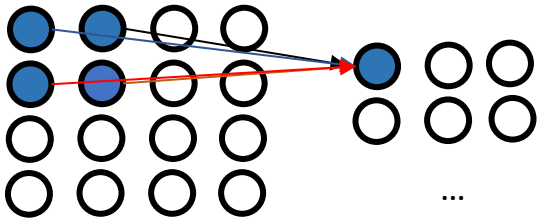
$\mathbf{h}_1=f(\mathbf{W}_1\mathbf{x})$

$\mathbf{x}$

Comparison of fully connected network and convolutional neural network

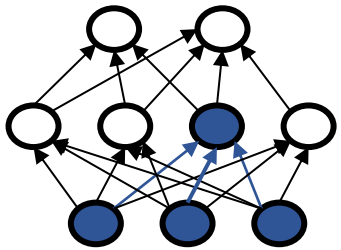
- Fully connected network (FC)
  - Every output unit interacts with every input unit
- Convolutional neural network (CNN)
  - Every output unit interacts with a small set of input units, depending on location
  - Parameters are shared

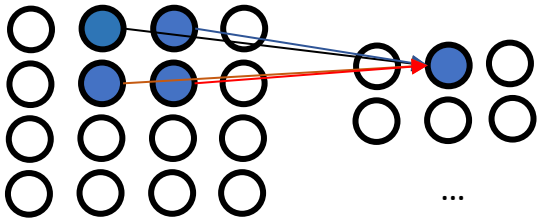




Comparison of fully connected network and convolutional neural network

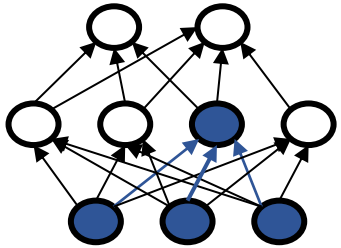
- Fully connected network (FC)
  - Every output unit interacts with every input unit
- Convolutional neural network (CNN)
  - Every output unit interacts with a small set of input units, depending on location
  - Parameters are shared

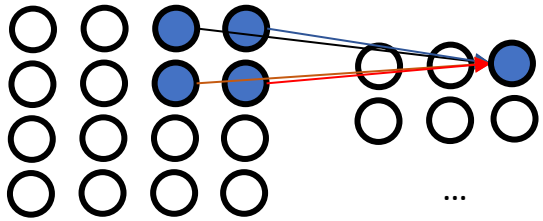




Comparison of fully connected network and convolutional neural network

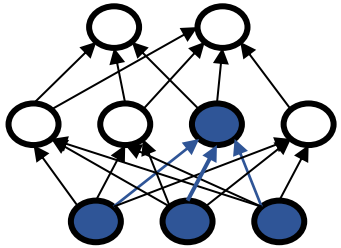
- Fully connected network (FC)
  - Every output unit interacts with every input unit
- Convolutional neural network (CNN)
  - Every output unit interacts with a small set of input units, depending on location
  - Parameters are shared

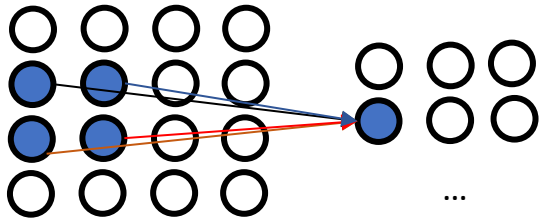




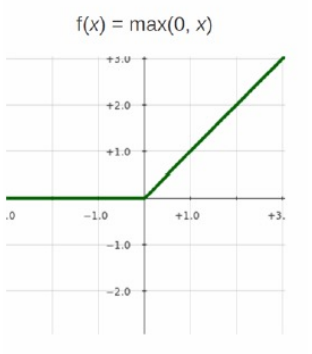
Comparison of fully connected network and convolutional neural network

- Fully connected network (FC)
  - Every output unit interacts with every input unit
- Convolutional neural network (CNN)
  - Every output unit interacts with a small set of input units, depending on location
  - Parameters are shared

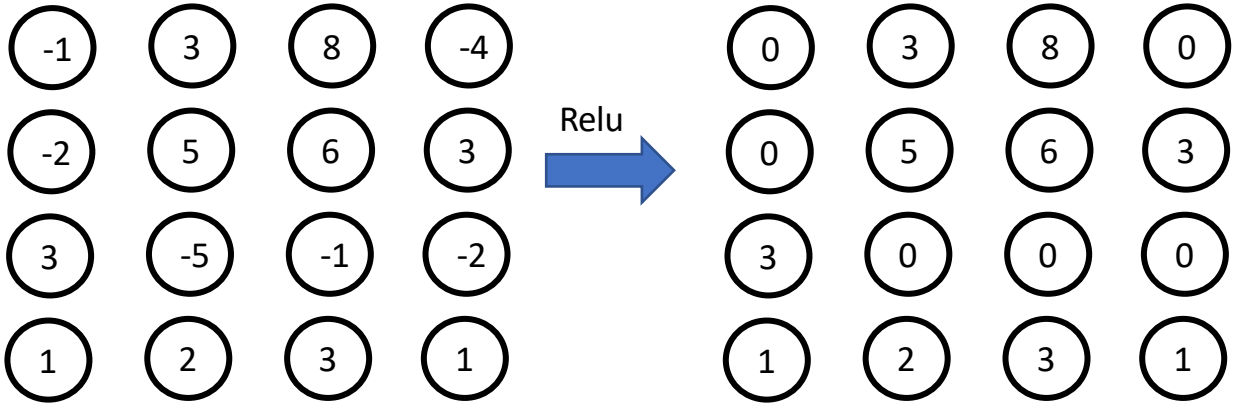




Non-linear function Relu

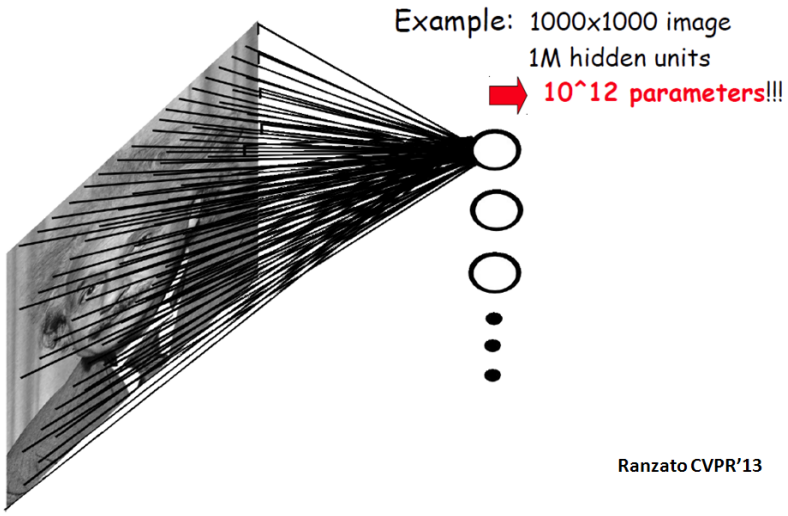


- Rectified linear unit (Relu)
- f(x)= max(x, 0);



Problems of fully connected neural networks

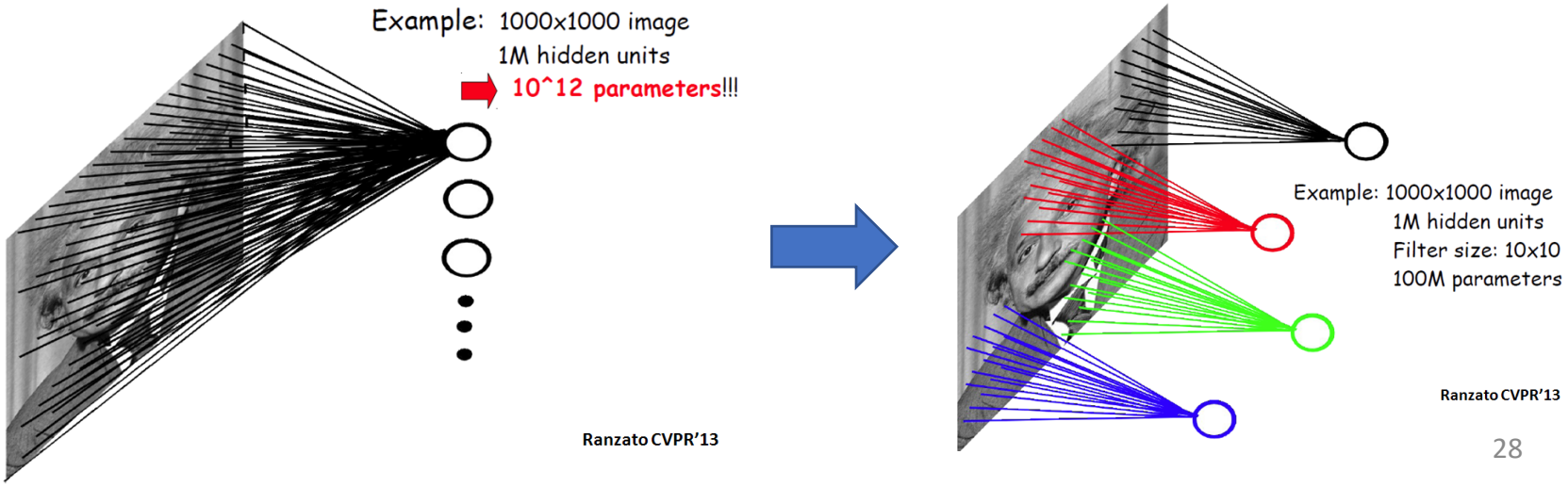
- Every output unit interacts with every input unit
- The number of weights grows largely with the size of the input image
- Pixels in distance are less correlated



Ranzato CVPR'13

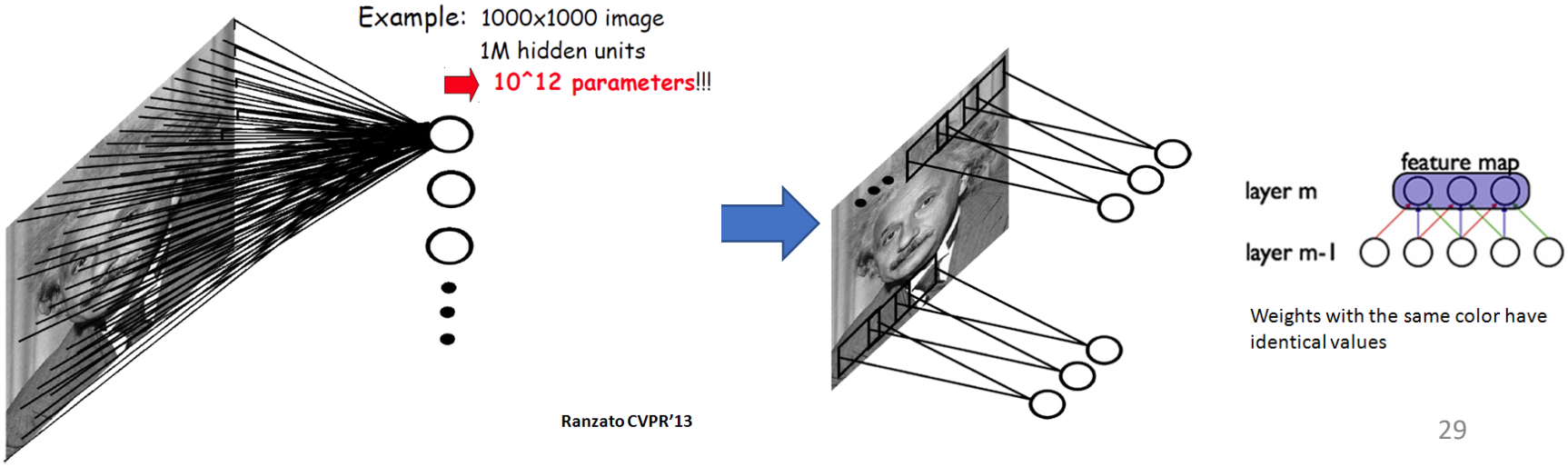
Locally connected neural networks

- Sparse connectivity: a hidden unit is only connected to a local patch (weights connected to the patch are called filter or kernel)
- It is inspired by biological systems, where a cell is sensitive to a small sub-region of the input space, called a receptive field. Many cells are tiled to cover the entire visual field.
- The design of such sparse connectivity is based on domain knowledge.



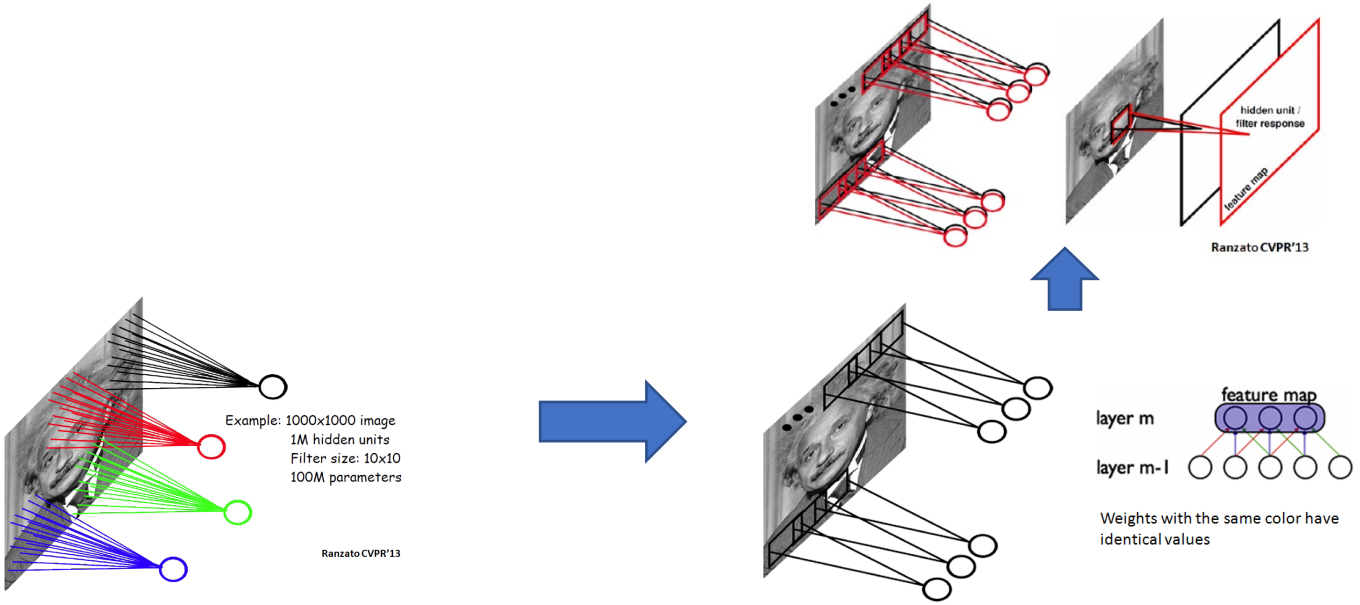
Locally connected neural networks with shared weights

- Translation invariance: capture statistics in local patches and they are independent of location
  - Similar edges may appear at different locations
- Hidden nodes at different locations share the same weights. It greatly reduces the number of parameters to learn
- In some applications (especially images with regular structures), we may only locally share weights or not share weights at top layers



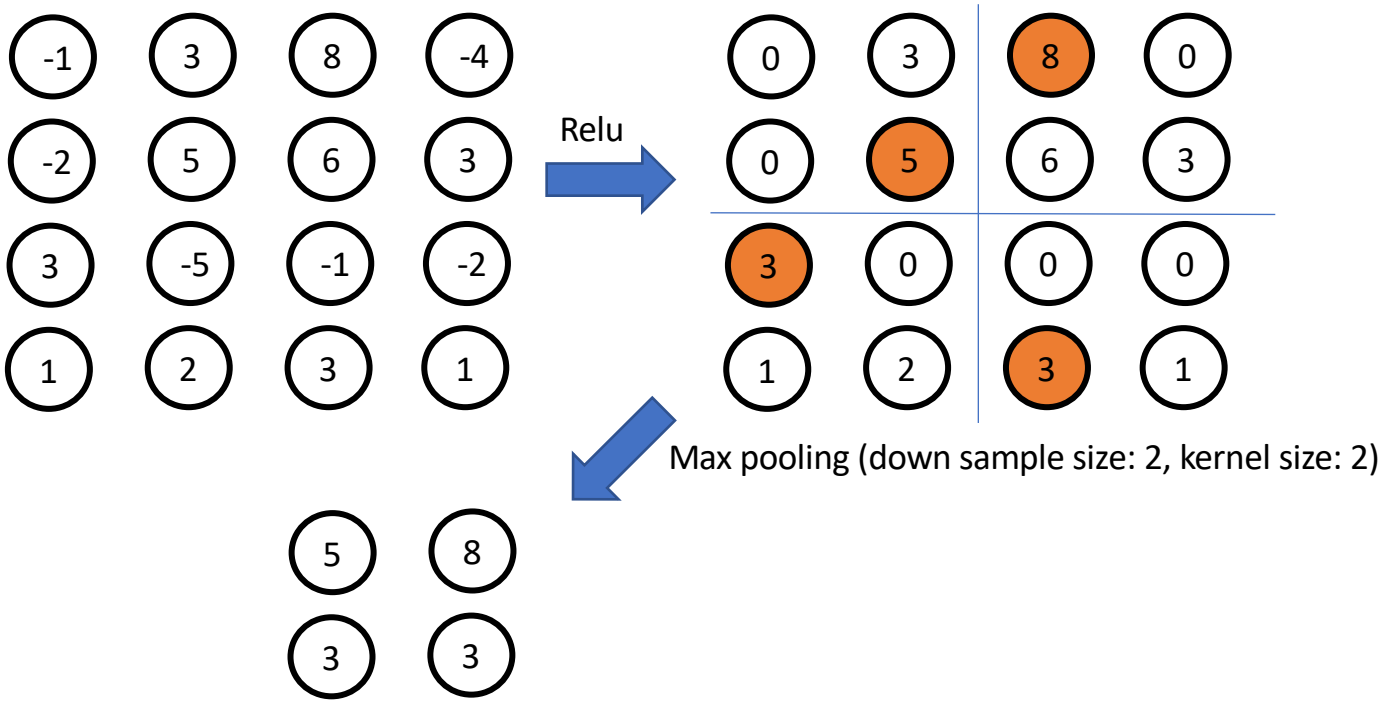
Multiple filters

- Multiple filters generate multiple feature maps
- Detect the spatial distributions of multiple visual patterns



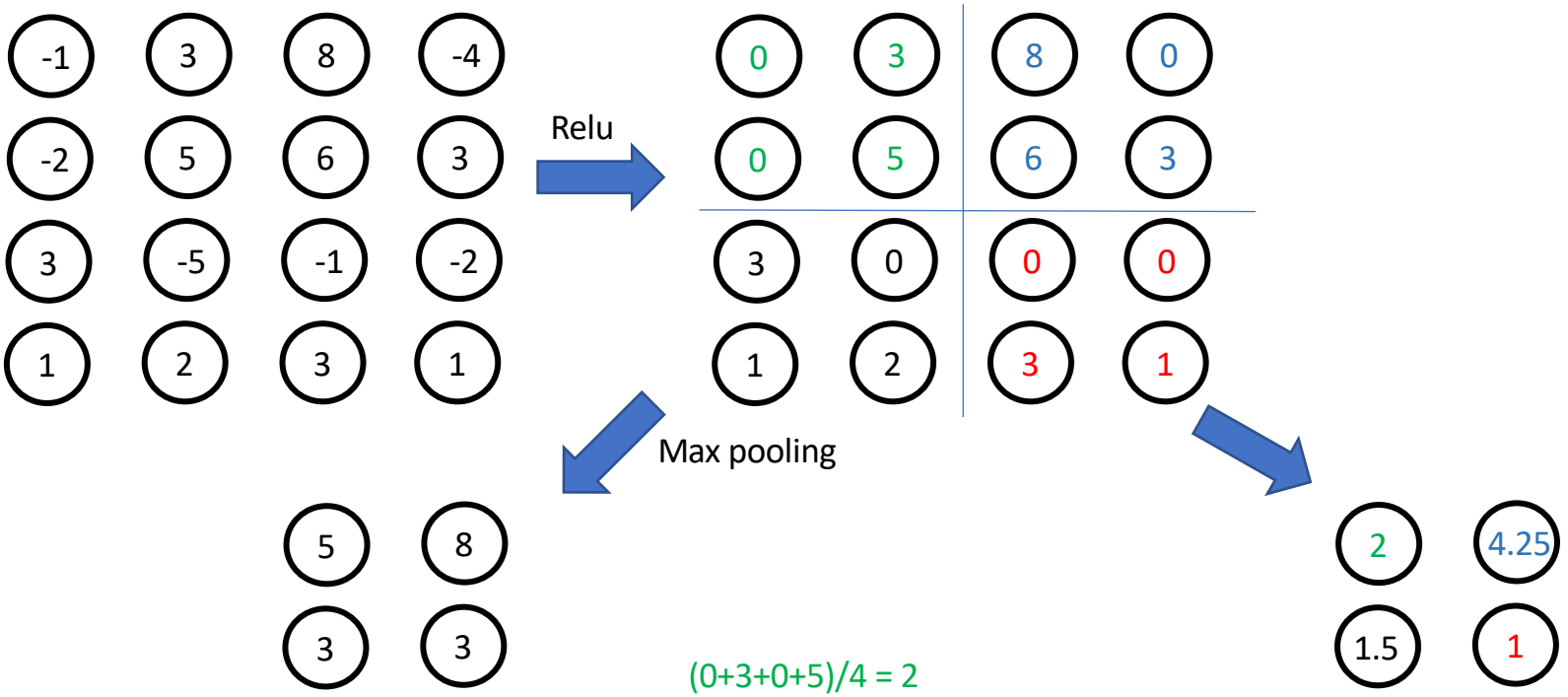
Pooling

- Max pooling



Pooling

- Max pooling and average pooling



Convolutional Neural Networks (CNN)

- Visualization results
  - https://www.youtube.com/watch?v=AgkfIQ4IGaM
  - https://www.youtube.com/watch?v=ixF5WNpTzCA

Outline

- Basic operations of deep model
- Basics of optimization
- Optimization for deep models

Optimization basics

- General form

$$\begin{aligned} &\min\ f_0(x) \\ &\text{s.t. } f_i(x) \leq b_i, \quad i = 1, \ldots, m \end{aligned}$$

$x = [\ x_1, \ldots, x_n\ ]^T \in \mathbf{R}^n$ optimization variables

$f_0 : \mathbf{R}^n \to \mathbf{R}$ objective function

$f_i : \mathbf{R}^n \to \mathbf{R},\ i = 1, \ldots, m$ constraint functions

Our aim is to find an **optimal solution** $x^\star$ that minimizes f_0 whilst satisfying all the constraints.

Example:

$$\begin{array}{ll} \text{maximize} & \mathbf{c^T x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \end{array}$$

Goals of Messi

Training data

Testing data

Year	2013	2014	2015	2016	2017	2018
Goals (club)	39	50	48	51	50	?

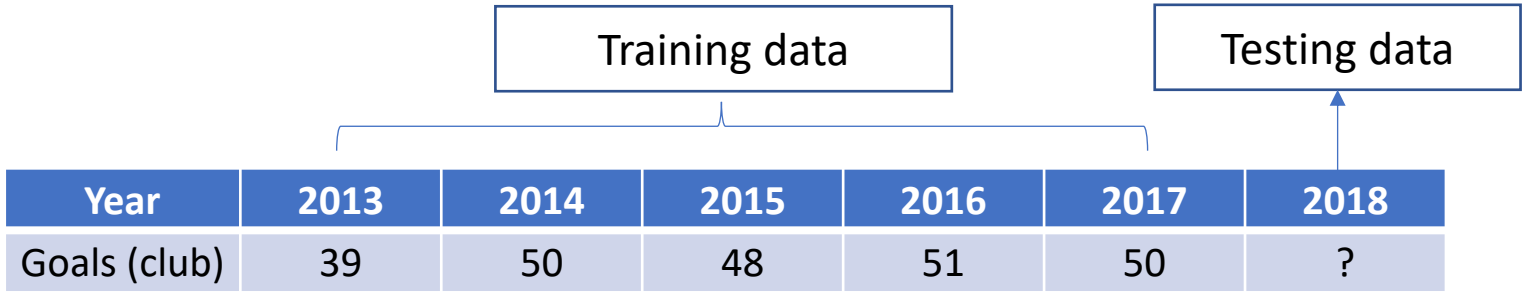
- Minimize $\mathbf{W}, b$ $\sum_i (y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b)^2$ ← Objective/loss function

$F(X; \boldsymbol{W}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b$ ← Prediction function

Picture from http://www.goal.com/en-gb/news/how-messi-became-the-worlds-deadliest-goalscorer/f1llyv71yqs91nzapr8yvo5td

Goals from http://messivsronaldo.net/calendar-year-stats/

Terms in machine learning



- Minimize $\mathbf{W}, b$ $\sum_i (y_i - \mathbf{w}^T \mathbf{x}_i - b)^2$ ← Objective/loss function

- **Training data:** The data used for training the model, i.e. learning the parameters.
- $(x_1, y_1)=(2013, 39)$, $(x_2, y_2)=(2014, 50)$ …
- $\mathbf{W}, b$: parameters to be learned from training data
- **Testing data:** The data used for evaluating the model. How effective is the learned model?
- Minimize: $(39-w*2013-b)^2+(50-w*2014-b)^2+(48-w*2015-b)^2+(51-w*2016-b)^2+(50-w*2017-b)^2$

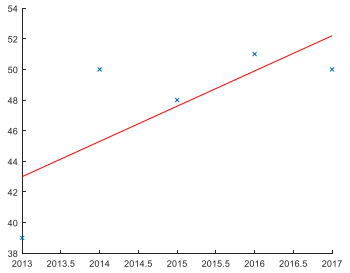
Linear Regression

Year	2013	2014	2015	2016	2017	2018
Goals (club)	39	50	48	51	50	?

- Minimize $\sum_i (y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b)^2$
  $\mathbf{w}$

```
x=[2013 2014 2015 2016 2017];
y=[39 50 48 51 50];
mdl = LinearModel.fit(x,y);
b = mdl.Coefficients.Estimate(1);
a = mdl.Coefficients.Estimate(2);
```

```
y2 = a*x+b;
hold on;
plot(x,y,'x');
plot(x,y2,'r');
hold off;
a*2018+b
```

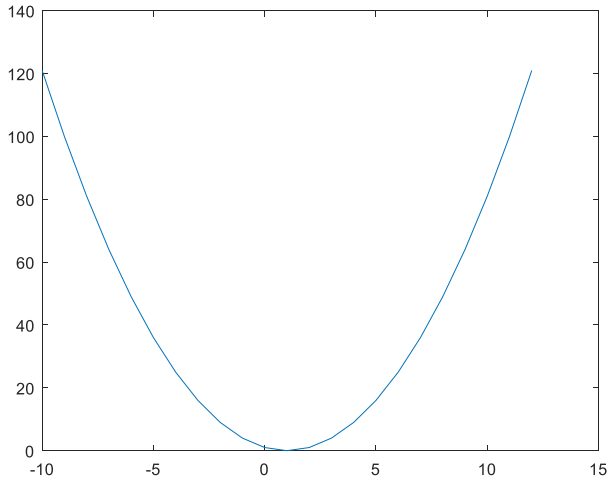



```
>> a*2018+b

ans =

   54.5000
```

Linear Regression

- Minimize $\sum_i (y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b)^2$
  w

- Minimize **$(x-1)^2$**



Gradient descent for linear regression

- Gradient descent
  - Follow the inverse direction of gradient for minimizing the objective function
- $x_t = x_{t-1} - \alpha J(x_t - 1)'$

Learning rate

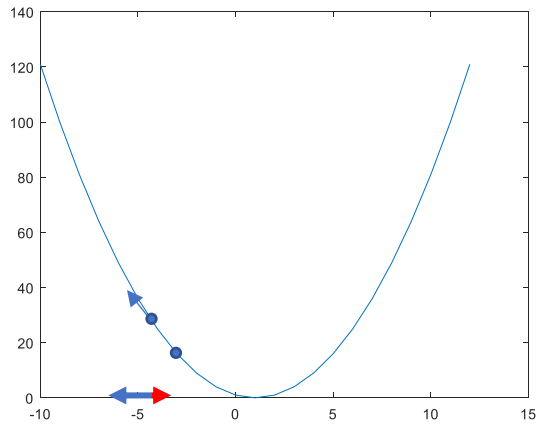
$\alpha = 0.1$

$x_0 = -4$

$x_1 = x_0 - \alpha f(x_0)' = -4 - \alpha * (-10) = -3$

$f(-4)' = 2*(-4-1) = -10$

- Minimize $(x-1)^2$
- $J(x) = (x-1)^2$
- $J(x)' = 2(x-1)$



Gradient descent for linear regression

- Gradient descent
  - Follow the inverse direction of gradient for minimizing the objective function
- $x_t = x_{t-1} - \alpha J(x_t - 1)'$
- Minimize $(x-1)^2$
- $J(x) = (x-1)^2$
- $J(x)' = 2(x-1)$

$\alpha = 0.1$

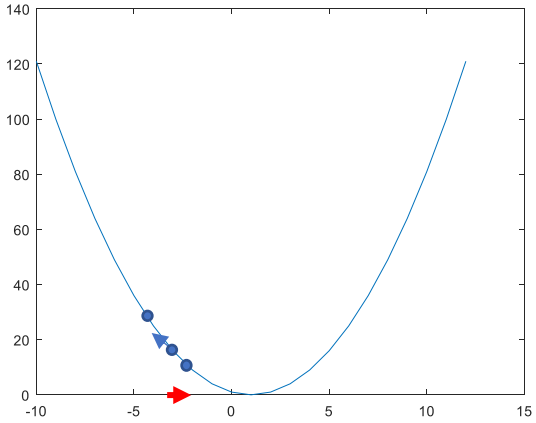
$x_0 = -4$

$x_1 = x_0 - \alpha f(x_0)' = -4 - \alpha*(-10) = -3$

$x_2 = x_1 - \alpha f(x_1)' = -3 - \alpha*(-10) = -2.2$

$J(-4)' = 2*(-4-1) = -10$

$J(-3)' = 2*(-3-1) = -8$



Gradient descent for linear regression

- Gradient descent
  - Follow the inverse direction of gradient for minimizing the objective function
- $x_t = x_{t-1} - \alpha J(x_t - 1)'$
- Minimize $(x-1)^2$
- $J(x) = (x-1)^2$
- $J(x)' = 2(x-1)$

$\alpha = 0.1$

$x_0 = -4$

$x_1 = x_0 - \alpha f(x_0)' = -4 - \alpha*(-10) = -3$

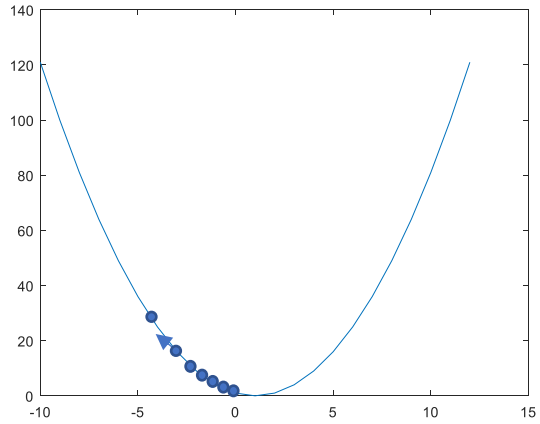
$x_2 = x_1 - \alpha f(x_1)' = -3 - \alpha*(-10) = -2.2$

…

…

$J(-4)' = 2*(-4-1) = -10$

$J(-3)' = 2*(-3-1) = -8$



Loss function

$0.5\lambda||\mathbf{w}||^2$: Regularization term, the magnitude of $\mathbf{w}$ should not be too large.

- Prediction p_i : $p_i = \mathbf{w}^T\mathbf{x}_i - b$

  y_i can be any continuous value

- $J(\mathbf{w}) = \sum_i (y_i - p_i)^2 + 0.5\lambda||\mathbf{w}||^2$

  - Least square loss for linear regression, pixel value?

- $J(\mathbf{w}) = \sum_i \max(0, 1 - y_i * p_i) + 0.5\lambda||\mathbf{w}||^2$

  $y_i = -1$ *or* 1

  -Soft-margin loss for linear support vector machine, dog or not?

- $J(\mathbf{w}) = -\sum_i \sum_c y_{i,c} \log p_{i,c} + 0.5\lambda||\mathbf{w}||^2$

  $y_{i,c} = 1$ *or* 0 $\quad\quad$ $\sum_c y_{i,c} = 1$

  - Cross entropy loss for multi-class classification, dog, cat, or flower?

$y_{i,1} = 1$

$[1, 0, 0]$

$y_{i,2} = 1$

$[0, 1, 0]$

$y_{i,3} = 1$

$[0, 0, 1]$

$[y_{i,1}, y_{i,2}, y_{i,3}]$

$0.5\lambda||\mathbf{w}||^2$: Regularization term, the magnitude of $\mathbf{w}$ should not be too large.

Gradients for loss functions

$p_i = \mathbf{w}^T\mathbf{x}_i - b$

- Least square: $J(\mathbf{w}) = \sum_i (p_i - y_i)^2 + 0.5\lambda||\mathbf{w}||^2$ y_i can be any continuous value

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_i 2(p_i - y_i)\frac{\partial p_i}{\partial \mathbf{w}} + \lambda\mathbf{w}$$

- Soft-margin loss: $J(\mathbf{w}) = \sum_i \max(0, 1 - y_i * p_i) + 0.5\lambda||\mathbf{w}||^2$ $y_i = -1$ *or* 1

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_i (-y_i)u(1 - y_i * p_i)\frac{\partial p_i}{\partial \mathbf{w}} + \lambda\mathbf{w}$$

$$u(x) = 1, if\ x > 0,$$
$$u(x) = 0, if\ x \leq 0.$$

- Cross entropy loss: $J(\mathbf{w}) = -\sum_i \sum_c y_{i,c} \log p_{i,c} + 0.5\lambda||\mathbf{w}||^2$ ith sample, cth class

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -\sum_i \sum_c y_{i,c}\frac{1}{p_{i,c}}\frac{\partial p_{i,c}}{\partial \mathbf{w}} + \lambda\mathbf{w}$$

$y_{i,c} = 1$ *or* 0 $\sum_c y_{i,c} = 1$

$y_{i,1} = 1$

$y_{i,2} = 1$

$y_{i,3} = 1$

$[y_{i,1}, y_{i,2}, y_{i,3}]$

$[1, 0, 0]$ $[0, 1, 0]$ $[0, 0, 1]$

$$J(\mathbf{w}) = \sum_i \max(0, 1 - y_i * p_i) + 0.5\lambda ||\mathbf{w}||^2$$

$$\max(0, 1 - y_i * p_i) = 1 - y_i * p_i \quad \textbf{if } 1 - y_i * p_i > 0$$
$$\text{or } = 0 \quad \textbf{if } 1 - y_i * p_i \leq 0$$

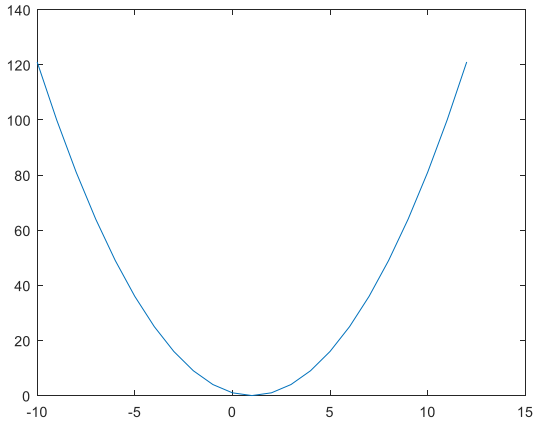
$$\frac{\partial J(p)}{\partial \boldsymbol{p}} = -y_i \quad \textbf{if } 1 - y_i * p_i > 0$$
$$\text{or } = 0 \quad \textbf{if } 1 - y_i * p_i \leq 0$$

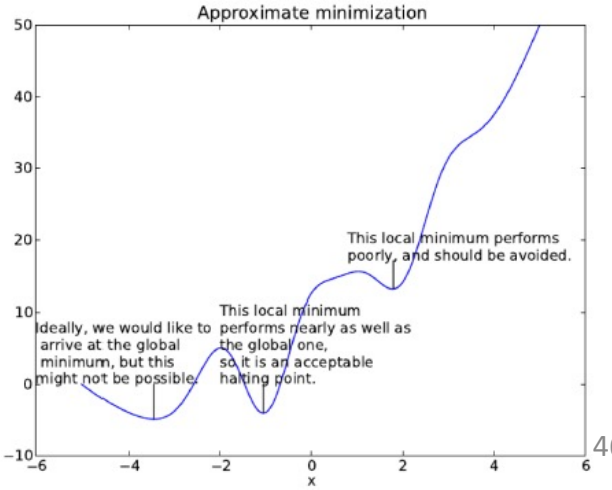
$$u(x) = 1, if\ x > 0,$$
$$u(x) = 0, if\ x \leq 0.$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_i (-y_i) u(1 - y_i * p_i) \frac{\partial p_i}{\partial \mathbf{w}} + \lambda \mathbf{w}$$

Local minimum and saddle points

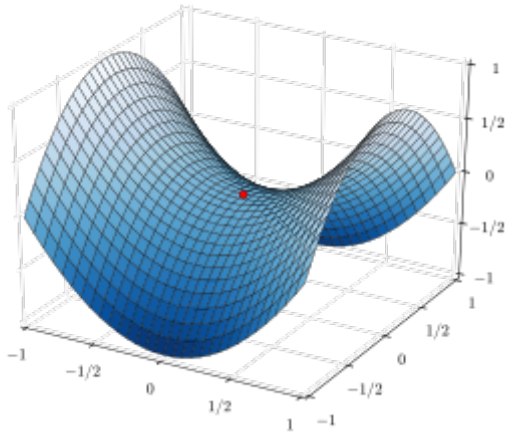
- Minima
  - Smallest value of the function, either within a given range (the **local** minima) or on the entire domain of a function (the **global** minima)
- Local minima = global minima => Convex
- Local minima ≠ global minima => Non-Convex





Local minimum and saddle points

- Minima
- Saddle point:
  - a point on the surface of the graph of a function where the slopes (derivatives) in orthogonal directions are both zero, but which is not a **local** minima/maxima of the function.



Outline

- Basic operations of deep model
- Basics of optimization
- Optimization for deep models

Back-propagation

- A chaining rule for obtaining gradient
- $p=f_1(h)$
- $J=f_2(p)$

$\frac{\partial J}{\partial h}$?

$$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h} = f_2(p)'f_1(h)'$$

$h=f_1(h), p = f_2(p)$

$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h}$

Back-propagation

- A chaining rule for obtaining gradient
- $p=f_1(h)=2h$
- $J=f_2(p)=(p\text{-}1)^2$

$\frac{\partial J}{\partial h}$?

$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h} = f_2(p)'f_1(h)'=2(p\text{-}1)\cdot 2=2(2h\text{-}1)\cdot 2=4\cdot(2h\text{-}1)$

$J = f_2(\mathrm{f}_1(\mathrm{h})) = (2h-1)^2$

$\frac{\partial J}{\partial h} = 4\cdot(2h-1)$

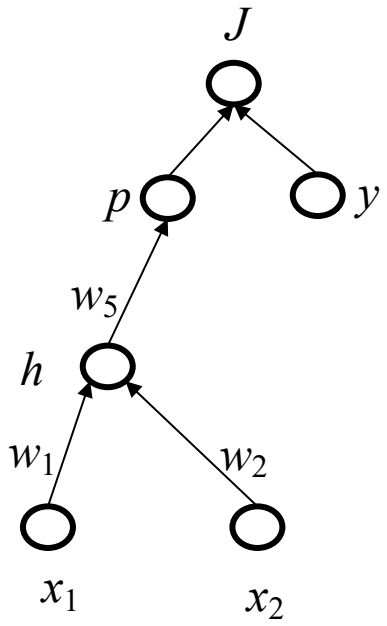
$h=f_1(h), p = f_2(p)$

$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h}$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i=\mathrm{f}_1(\mathbf{x}_i; w_1, w_2)=\max(w_1x_{1,i}+w_2x_{2,i}, 0)$
- $p_i=\mathrm{f}_2(h_i; \mathrm{w}_5)=w_5h_i$

- Loss:
- $\boldsymbol{J(\mathbf{p}, \mathbf{w})=\sum_i(p_i-y_i)^2}$



$h=f_1(h), p = f_2(p)$

$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h}$

Back-propagation

- A chaining rule for obtaining gradient

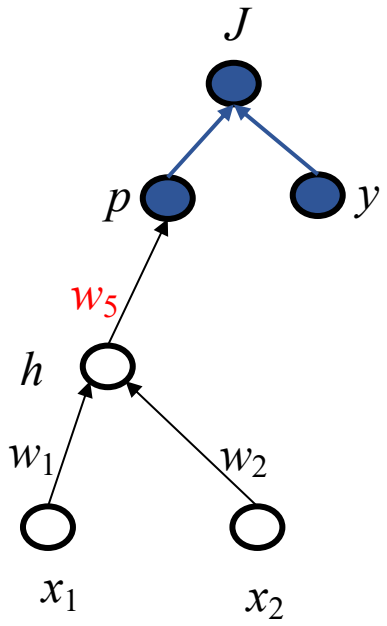
$h_i=\mathrm{f}_1(\mathbf{x}_i; w_1, w_2)=\max(w_1 x_{1,i}+w_2 x_{2,i}, 0)$

$p_i=\mathrm{f}_2(h_i; \mathrm{w}_5)=w_5 h_i$

- Loss:

$J(\mathbf{p}) = \sum_i (p_i - y_i)^2$

$\frac{\partial J}{\partial w_5} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial w_5}$

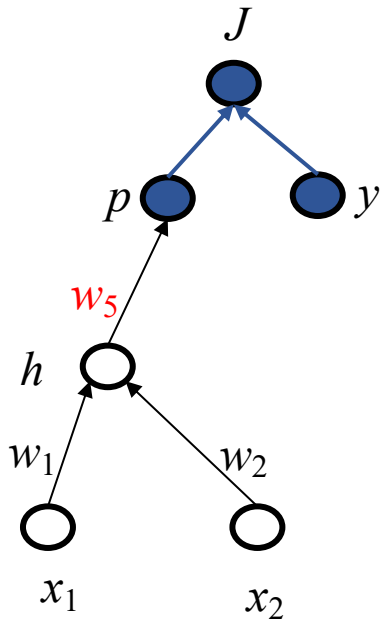


$$h=f_1(h), p = f_2(p)$$
$$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h}$$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i=\mathrm{f}_1(\mathbf{x}_i; w_1, w_2)=\max(w_1x_{1,i}+w_2x_{2,i}, 0)$
- $p_i=\mathrm{f}_2(h_i; \mathrm{w}_5)=w_5h_i$

- Loss:
- $J(\mathbf{p}, \mathbf{w})=\sum_i(p_i-y_i)^2$



$$\frac{\partial J}{\partial w_5} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial w_5}$$
$$= \sum_i 2(p_i-y_i)\frac{\partial p_i}{\partial w_5}$$

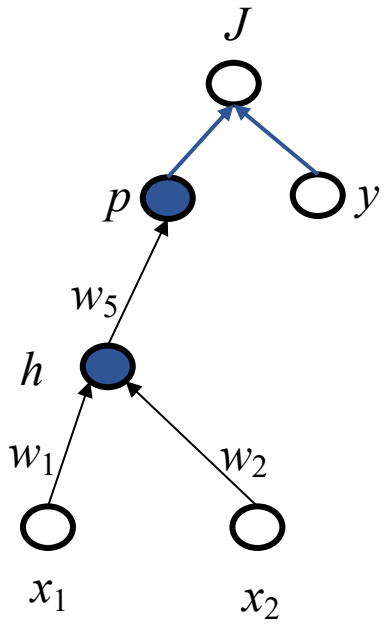
$h=f_1(h), p = f_2(p)$

$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h}$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i=\mathrm{f}_1(\mathbf{x}_i; w_1, w_2)=\max(w_1x_{1,i}+w_2x_{2,i}, 0)$
- $p_i=\mathrm{f}_2(h_i; \mathrm{w}_5)=w_5h_i$

- Loss:
- $J(\mathbf{p}, \mathbf{w})=\sum_i(p_i-y_i)^2$



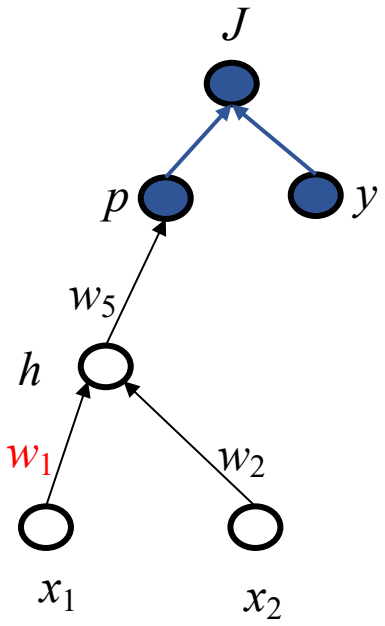
$$\Delta p_i = \frac{\partial J}{\partial p_i} = 2(p_i - y_i)$$

$$\frac{\partial J}{\partial w_5} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial w_5}$$
$$= \sum_i 2(p_i - y_i)\frac{\partial p_i}{\partial w_5}$$
$$= \sum_i 2(p_i - y_i)h_i$$

$$h=f_1(h), p = f_2(p)$$
$$\frac{\partial J}{\partial h} = \frac{\partial J}{\partial p}\frac{\partial p}{\partial h}$$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i=\mathrm{f}_1(\mathbf{x}_i; w_1, w_2)=\max(w_1 x_{1,i}+w_2 x_{2,i}, 0)$
- $p_i=\mathrm{f}_2(h_i; \mathrm{w}_5)=w_5 h_i$



- Loss:
- $J(\mathbf{p}, \mathbf{w})=\sum_i(p_i-y_i)^2$

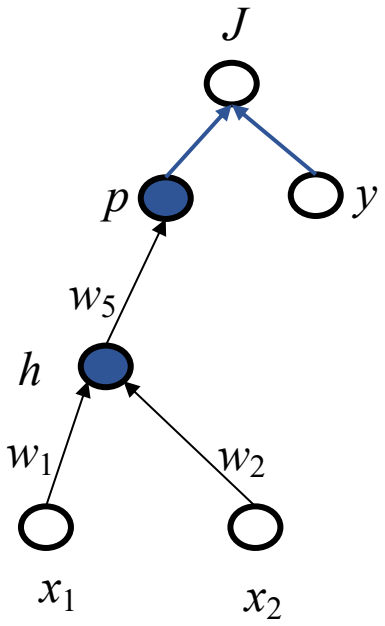
$$\frac{\partial J}{\partial w_1} = \sum_i \boxed{\frac{\partial J}{\partial p_i}} \frac{\partial p_i}{\partial w_1}$$
$$= \sum_i \boxed{2(p_i-y_i)} \frac{\partial p_i}{\partial w_1}$$
$$= \sum_i \boxed{\Delta p_i} \frac{\partial p_i}{\partial w_1}$$

$$\Delta p_i = \frac{\partial J}{\partial p_i} = 2(p_i-y_i)$$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i = \mathrm{f}_1(\mathbf{x}_i; w_1, w_2) = \max(w_1 x_{1,i} + w_2 x_{2,i}, 0)$
- $p_i = \mathrm{f}_2(h_i; \mathrm{w}_5) = w_5 h_i$

- Loss:
- $J(\mathbf{p}, \mathbf{w}) = \sum_i (p_i - y_i)^2$



$$\frac{\partial J}{\partial w_1} = \sum_i \Delta p_i \frac{\partial p_i}{\partial w_1} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial h_i}\frac{\partial h_i}{\partial w_1} = \sum_i \Delta h_i \frac{\partial h_i}{\partial w_1}$$

$$\Delta p_i = \frac{\partial J}{\partial p_i} = 2(p_i - y_i)$$

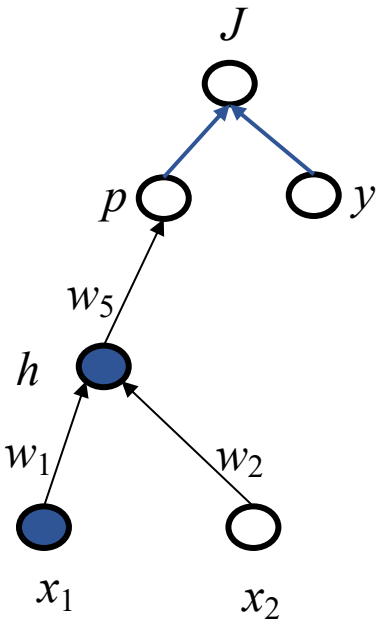
$$\Delta h_i = \frac{\partial J}{\partial h_i} = \Delta p_i \frac{\partial p_i}{\partial h_i} = 2(p_i - y_i) w_5$$

$u(x) = 1, if\ x > 0,$
$u(x) = 0, if\ x \leq 0.$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i = \mathrm{f}_1(\mathbf{x}_i; w_1, w_2) = \max(w_1 x_{1,i} + w_2 x_{2,i}, 0)$
- $p_i = \mathrm{f}_2(h_i; \mathrm{w}_5) = w_5 h_i$

- Loss:
- $J(\mathbf{p}, \mathbf{w}) = \sum_i (p_i - y_i)^2$



$$\frac{\partial J}{\partial w_1} = \sum_i \Delta p_i \frac{\partial p_i}{\partial w_1} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial h_i}\frac{\partial h_i}{\partial w_1} = \sum_i \Delta h_i \frac{\partial h_i}{\partial w_1} = \sum_i \Delta h_i\ u(w_1 x_{1,i} + w_2 x_{2,i}) x_{1,i}$$

$$\Delta p_i = \frac{\partial J}{\partial p_i} = 2(p_i - y_i)$$

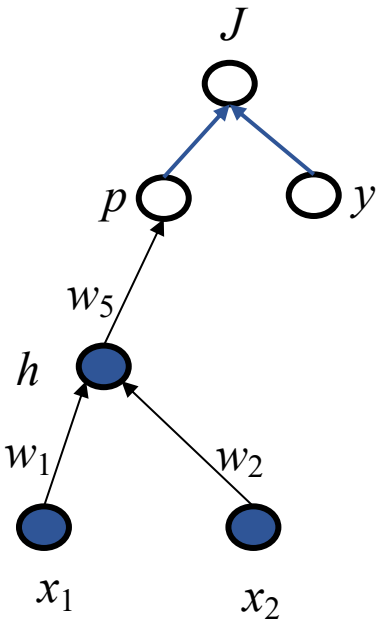
$$\Delta h_i = \frac{\partial J}{\partial h_i} = \Delta p_i \frac{\partial p_i}{\partial h_i} = 2(p_i - y_i) w_5$$

$$\Delta x_{1,i} = \frac{\partial J}{\partial x_{1,i}} = \Delta h_i \frac{\partial h_i}{\partial x_{1,i}} = 2(p_i - y_i) w_5 u(w_1 x_{1,i} + w_2 x_{2,i}) w_1$$

$u(x) = 1, if\ x > 0,$
$u(x) = 0, if\ x \leq 0.$

Back-propagation

- A chaining rule for obtaining gradient
- $h_i = \mathrm{f}_1(\mathbf{x}_i; w_1, w_2) = \max(w_1 x_{1,i} + w_2 x_{2,i}, 0)$
- $p_i = \mathrm{f}_2(h_i; \mathrm{w}_5) = w_5 h_i$



- Loss:
- $J(\mathbf{p}, \mathbf{w}) = \sum_i (p_i - y_i)^2$

$$\frac{\partial J}{\partial w_1} = \sum_i \Delta p_i \frac{\partial p_i}{\partial w_1} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial h_i}\frac{\partial h_i}{\partial w_1} = \sum_i \Delta h_i \frac{\partial h_i}{\partial w_1} = \sum_i 2(p_i - y_i) w_5 u(w_1 x_{1,i} + w_2 x_{2,i}) x_{1,i}$$

$$\Delta h_i = 2(p_i - y_i) w_5$$

$$\frac{\partial J}{\partial w_2} = \sum_i \Delta p_i \frac{\partial p_i}{\partial w_2} = \sum_i \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial h_i}\frac{\partial h_i}{\partial w_2} = \sum_i \Delta h_i \frac{\partial h_i}{\partial w_2} = \sum_i 2(p_i - y_i) w_5 u(w_1 x_{1,i} + w_2 x_{2,i}) x_{2,i}$$

$$\frac{\partial h_i}{\partial w_1} = \frac{\partial \max(w_1 x_{1,i} + w_2 x_{2,i}, 0)}{\partial w_1}$$

$$\max(w_1 x_{1,i} + w_2 x_{2,i}, 0) = w_1 x_{1,i} + w_2 x_{2,i} \text{ if } w_1 x_{1,i} + w_2 x_{2,i} > 0$$
$$\text{or } = 0 \text{ otherwise}$$

$$\frac{\partial \max(w_1 x_{1,i} + w_2 x_{2,i}, 0)}{\partial w_1} = w_1 x_{1,I} \quad \text{if } w_1 x_{1,i} + w_2 x_{2,i} > 0$$
$$\text{or } = 0 \text{ otherwise}$$

$$\frac{\partial \max(w_1 x_{1,i} + w_2 x_{2,i}, 0)}{\partial w_1} = u(w_1 x_{1,i} + w_2 x_{2,i}) x_{1,i}$$

$u(x) = 1, if\ x > 0,$
$u(x) = 0, if\ x \leq 0.$

Back-propagation

- A chaining rule for obtaining gradient
- Gradients to non-linear function?

$Sigmoid(x)' = Sigmoid(x)(1 - Sigmoid(x))$
$Tanh(x)' = 1 - 2Tanh^2(x)$
$Relu(x)' = u(x)$