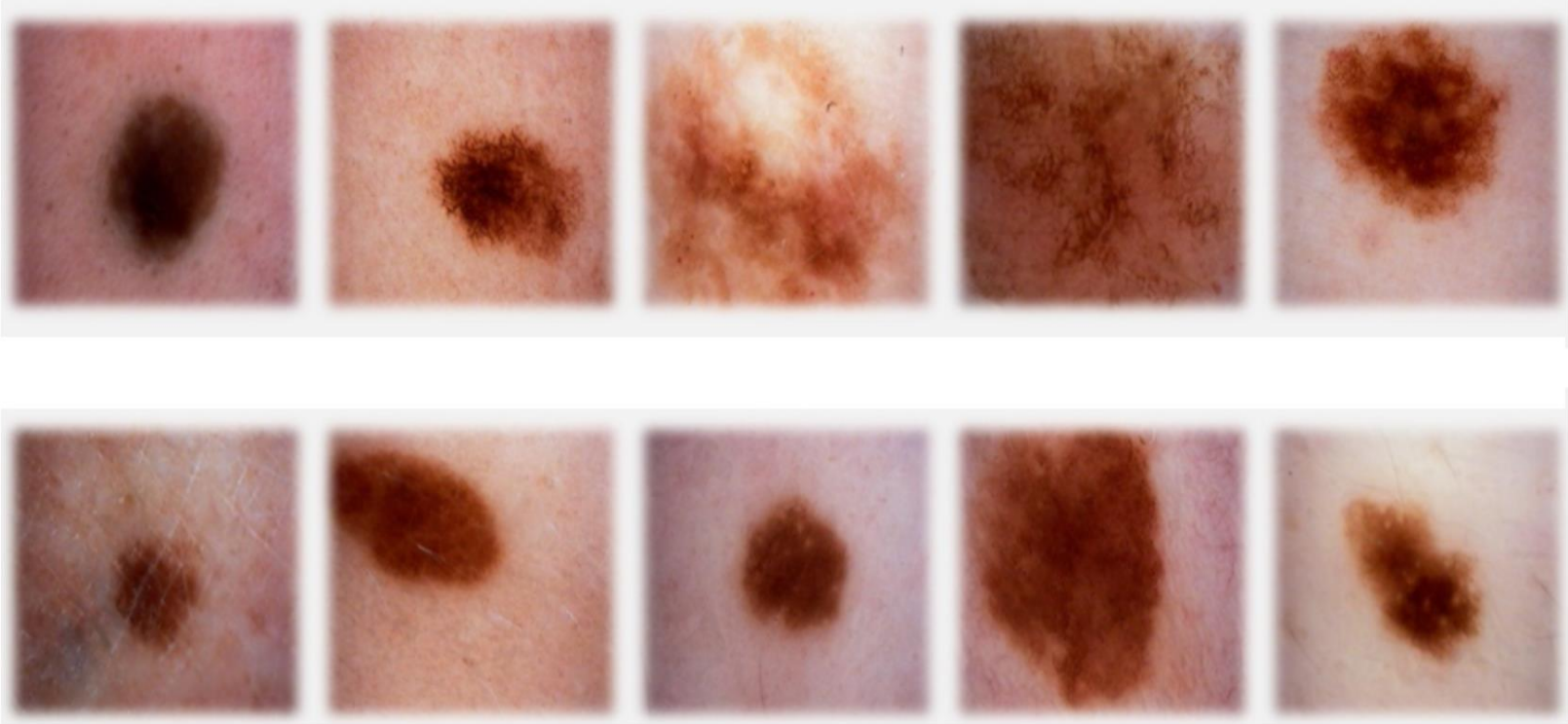
Classification: labelled data

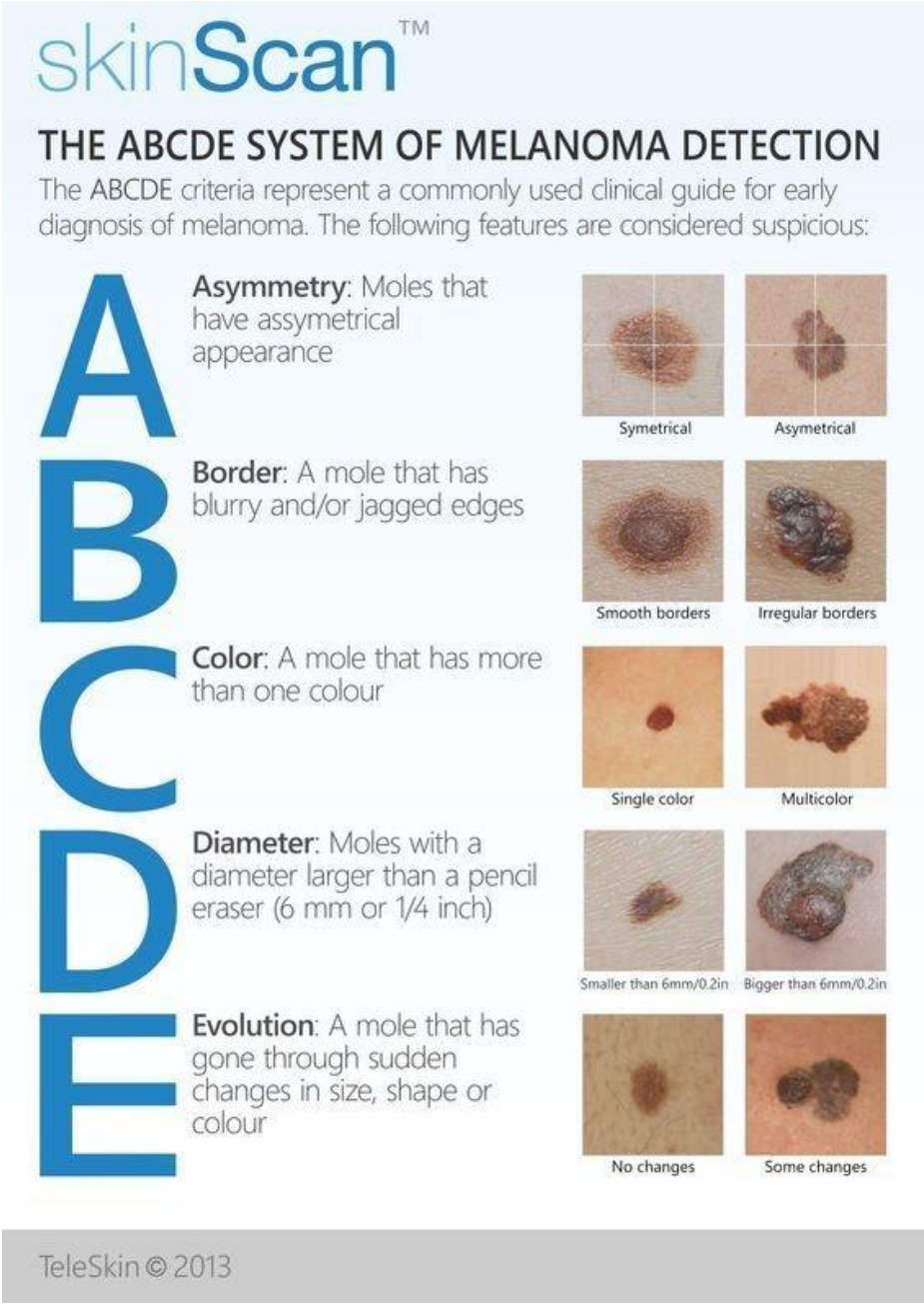


sign of cancer

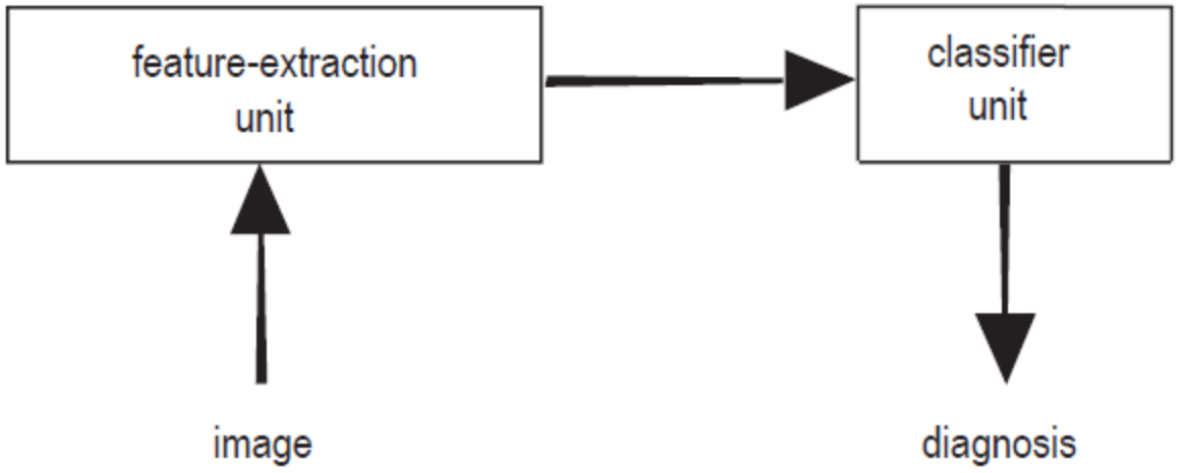
top row malignant

bottom row benign

Classification: features



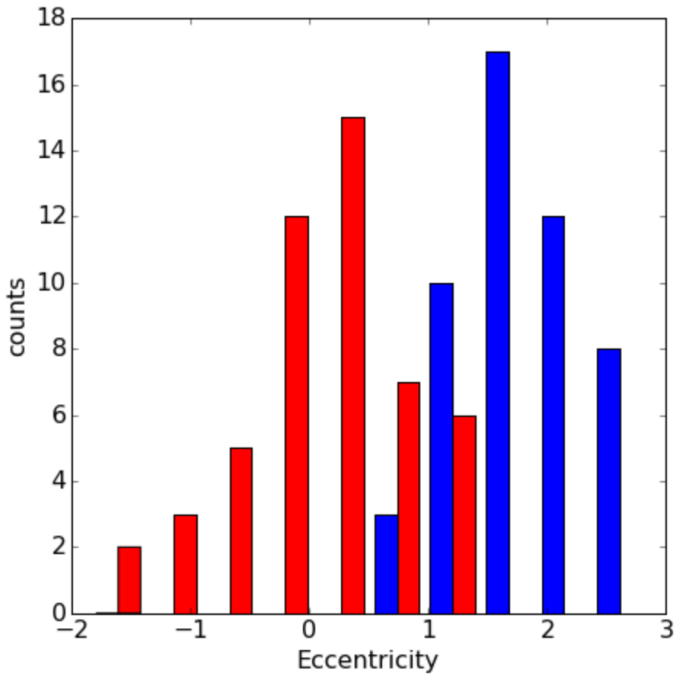
Classification: model induction

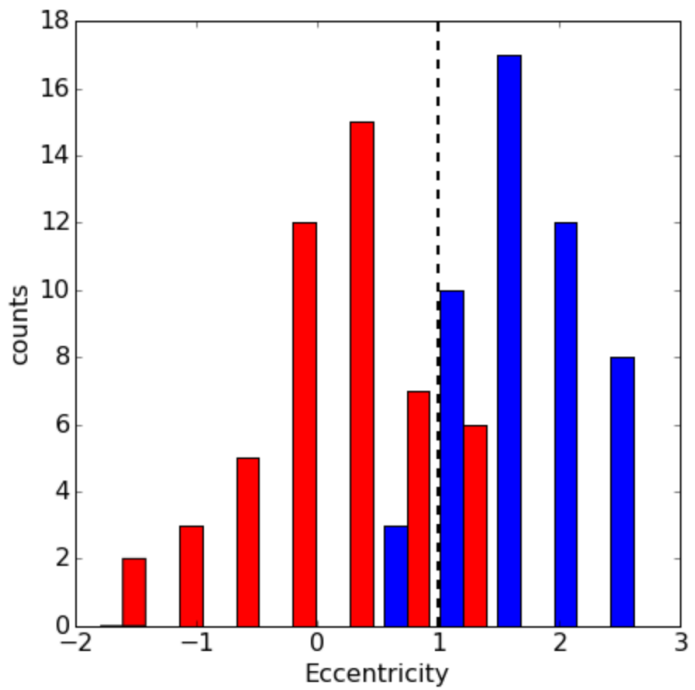


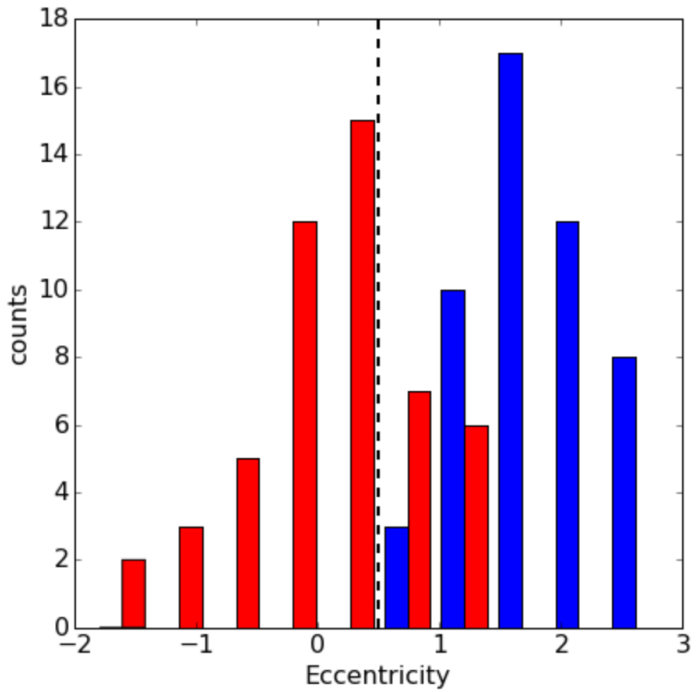
feature extraction: features (a.k.a. properties or attributes)

data set, sample (a.k.a. example, instance or data point), label (a.k.a. target)

Classification: feature values







eccentricity of lesion (how nearly circular the lesion is)

threshold ***t***

consequence of the predictions

Classification: errors

malignant positive class, benign negative

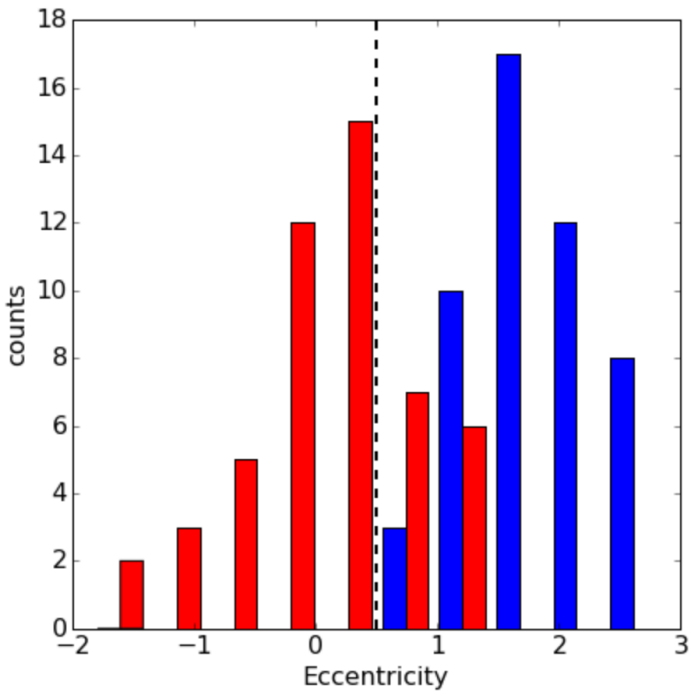
consequence of the predictions

count the number of malignant images with eccentricity value $\geq t$: **true positive** predictions (TP)

count the number of malignant images with eccentricity value $< t$: **false negative** predictions (FN)

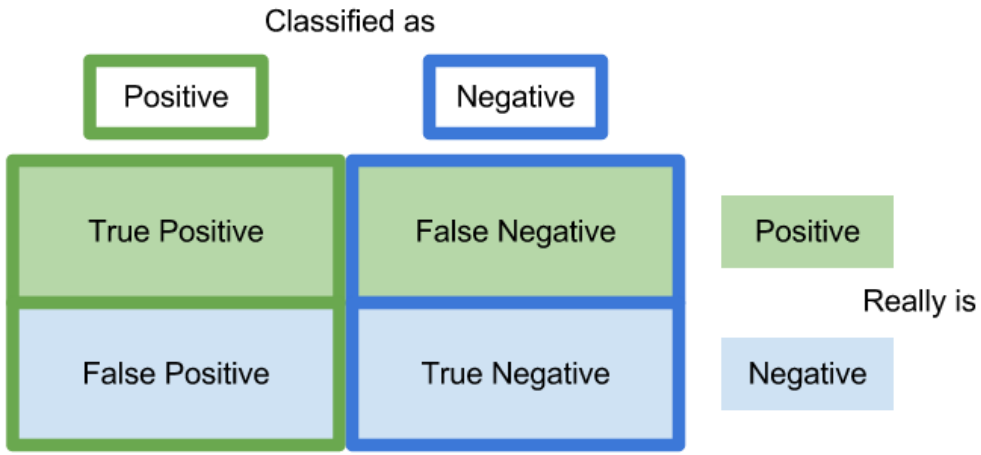
count the number of benign images with eccentricity value $\geq t$: **false positive** predictions (FP)

count the number of benign images with eccentricity value $< t$: **true negative** predictions (TN)



	Classified as: Positive	Classified as: Negative
Really is: Positive	True Positive	False Negative
Really is: Negative	False Positive	True Negative

Classification: model evaluation metrics



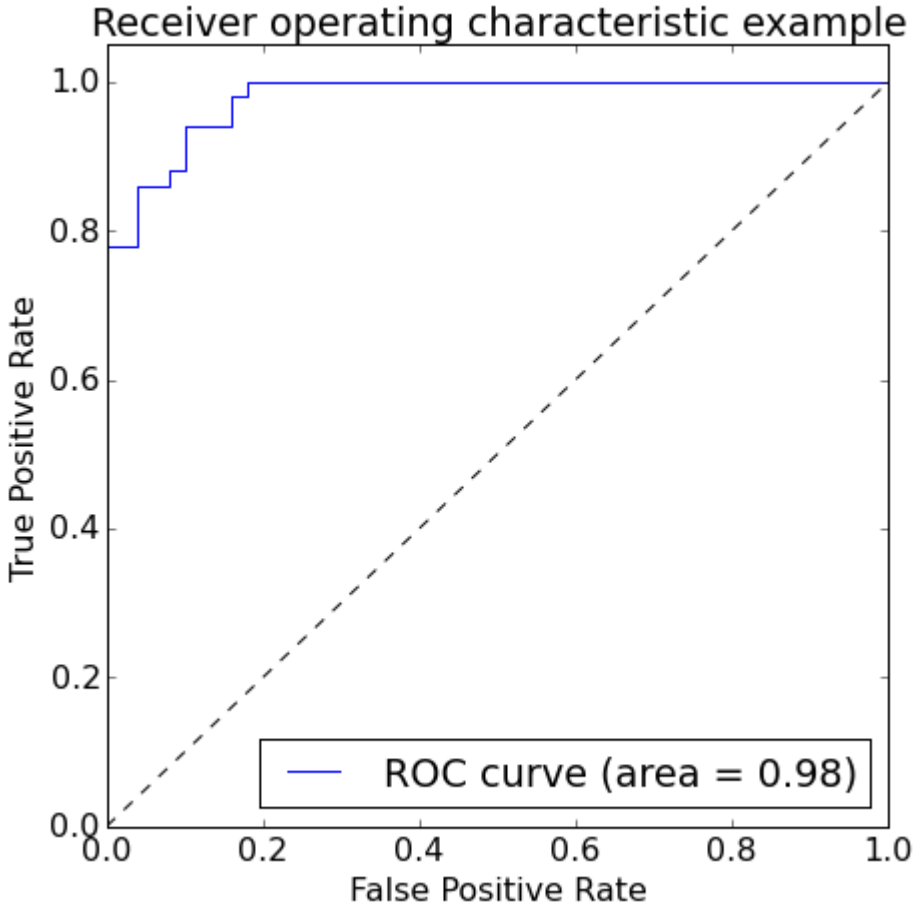
$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{TPR} = \frac{TP}{TP + FN}$$

$$\text{FPR} = \frac{FP}{FP + TN}$$

Classification: ROC & AUC

$$\mathrm{TPR} = \frac{TP}{TP + FN}$$



$$\mathrm{FPR} = \frac{FP}{FP + TN}$$

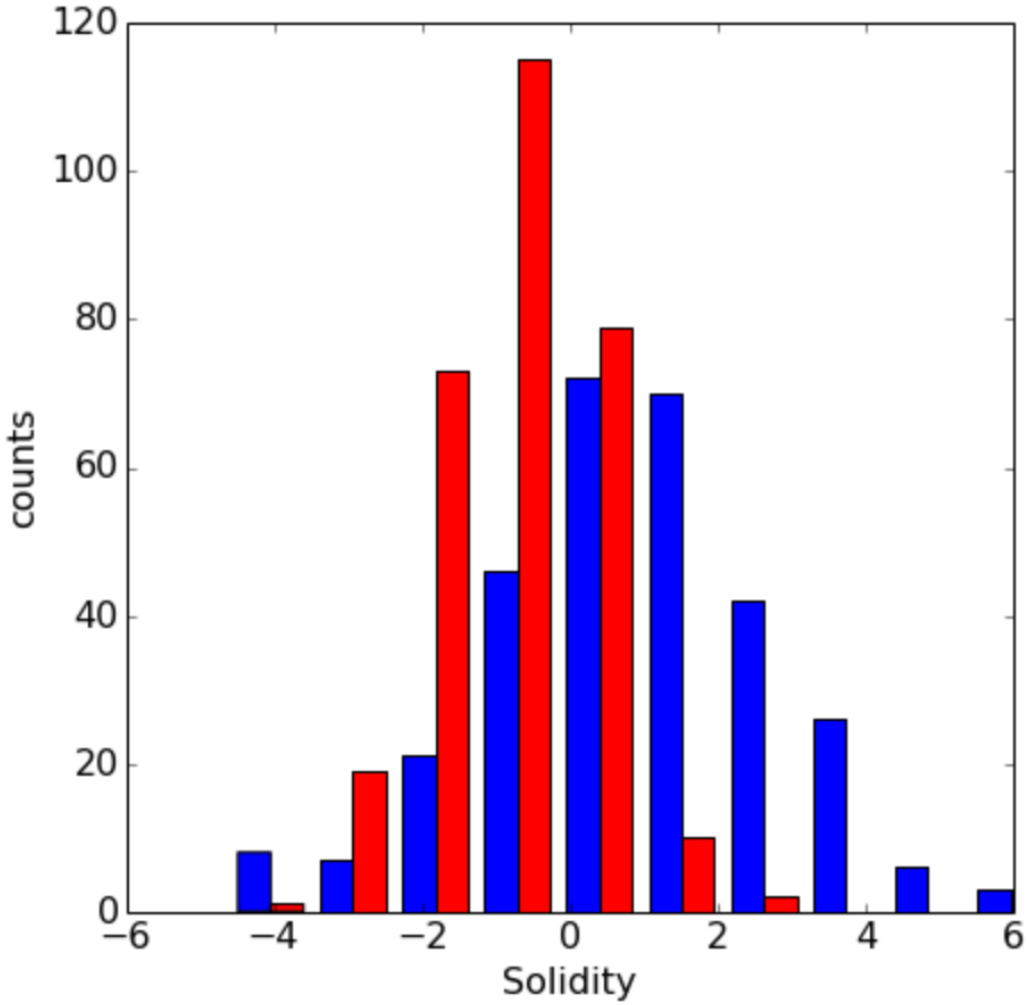
model that classifies all images as malignant:
TPR=1 and FPR=1

model that classifies all images a benign:
TPR=0 and FPR=0

vary threshold *t*

AUC

Classification: feature vector

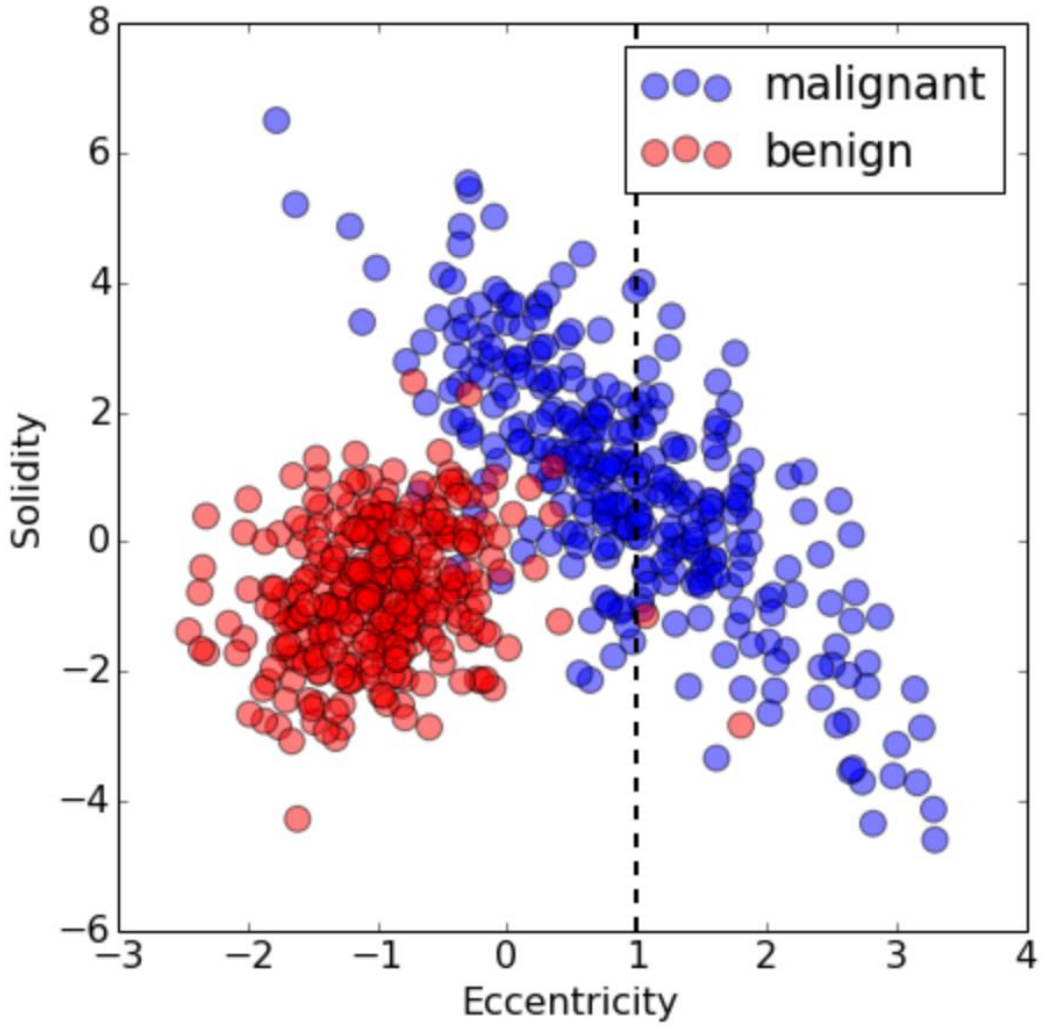


add another feature?

feature vector X

Euclidean vector space

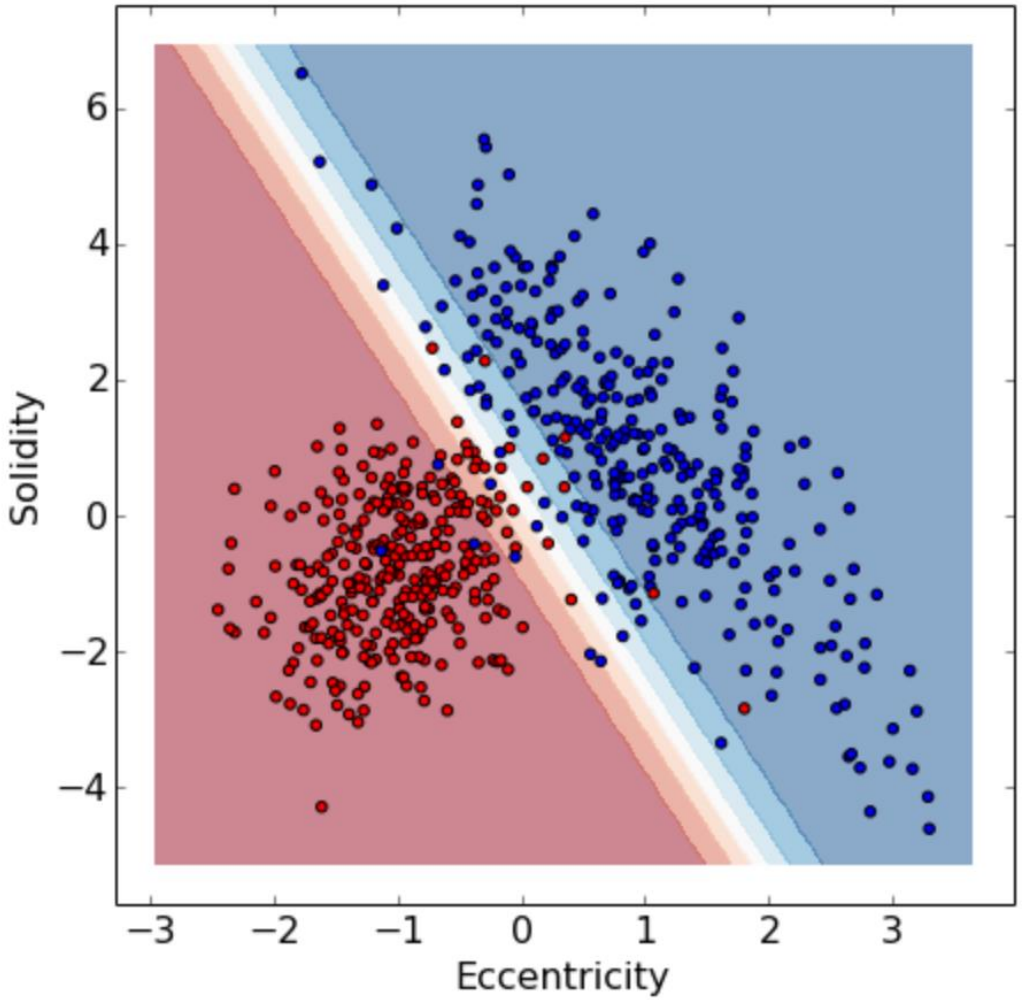
Classification: feature space



feature vector X

Euclidean vector space

Classification: decision boundary



linear decision boundary

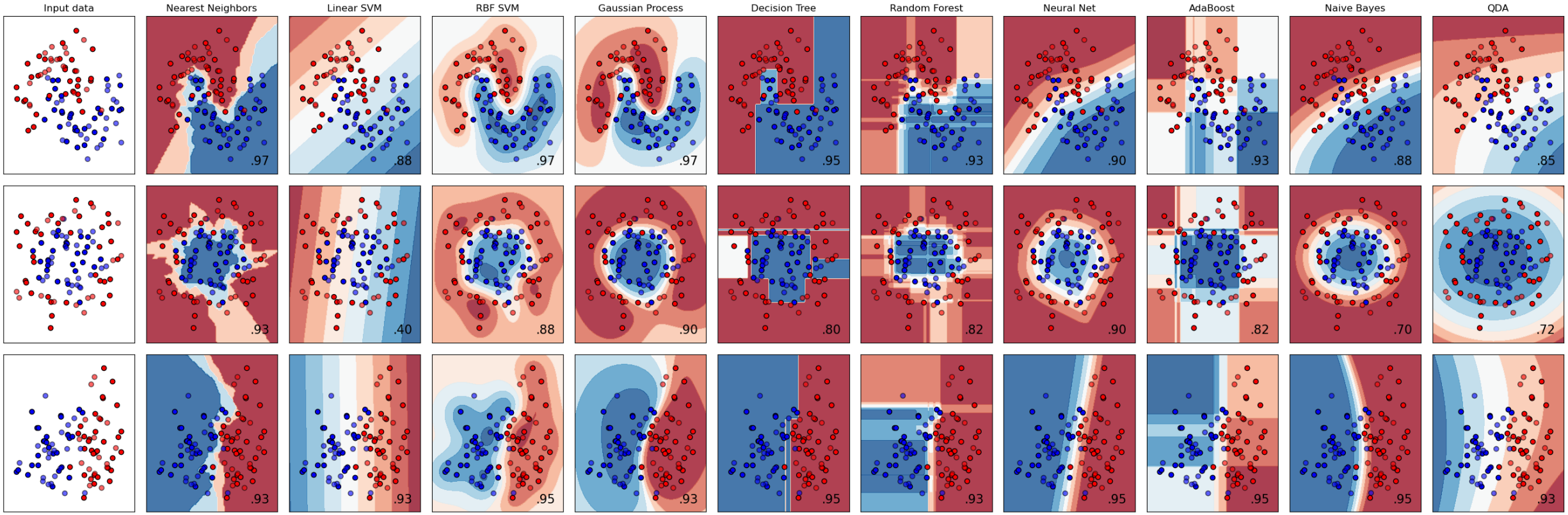
blue region malignant, red region benign

yet more features

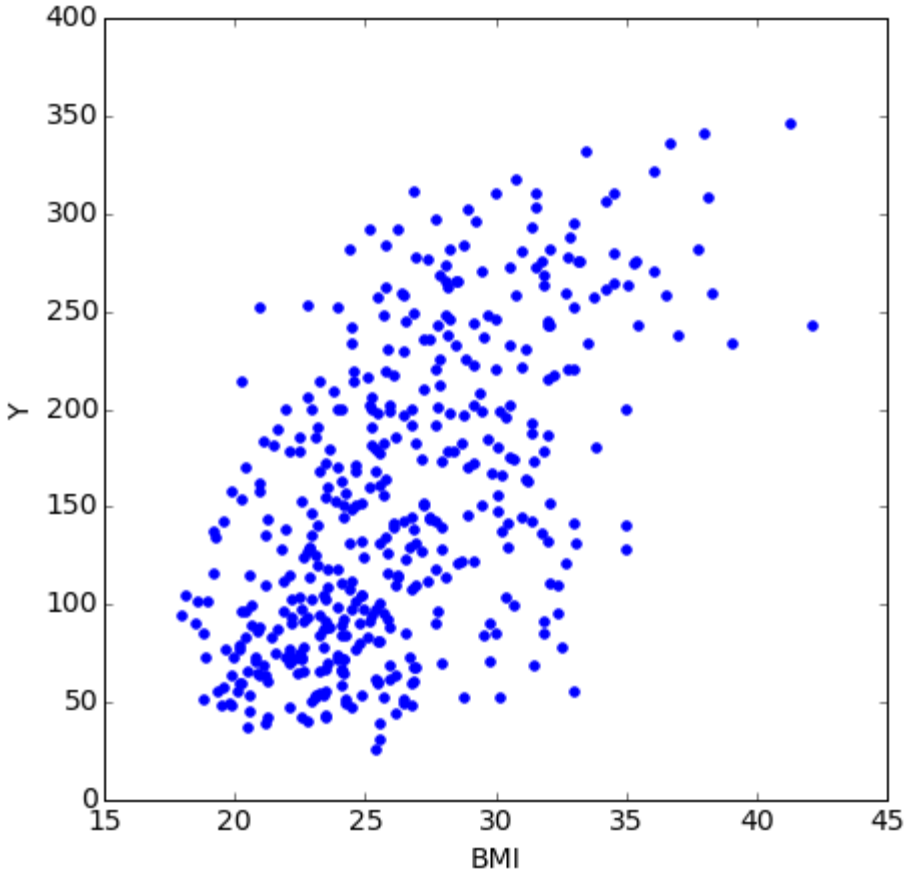
can’t look at the decision boundary

more complex

Classification: model representations



Linear regression



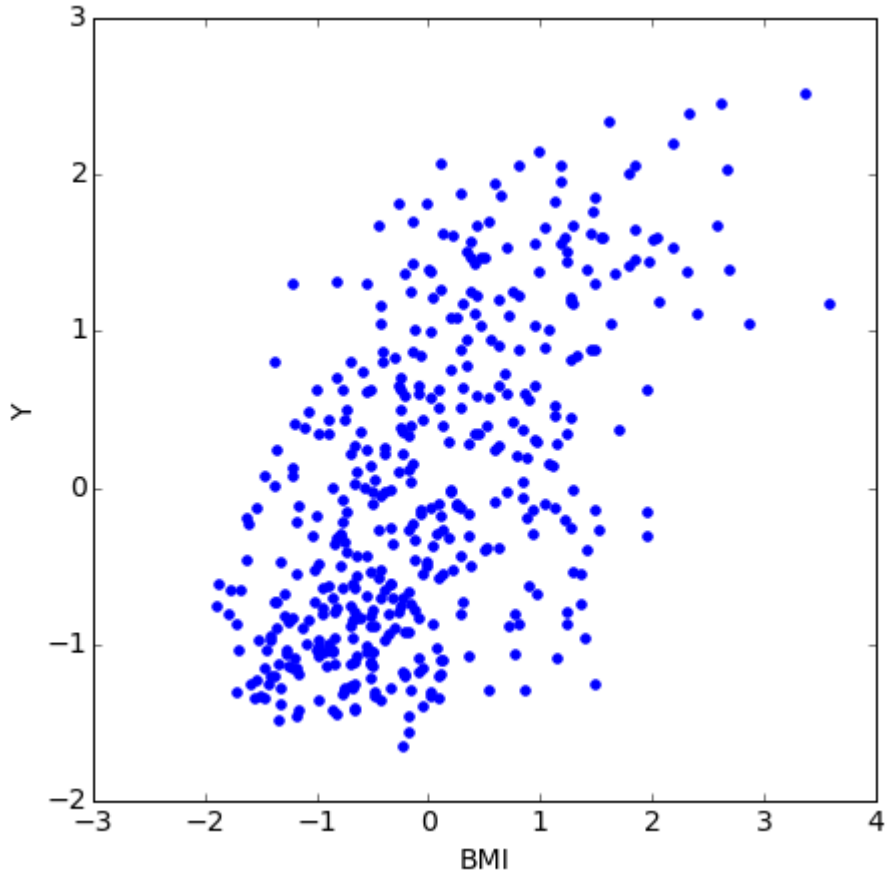
Can BMI explain Y?

Can BMI predict Y?

How does Y vary with BMI?

Regression is a general term for **modeling the relationship** between a **label** (a.k.a. dependent variable or target) and one or more **features** (a.k.a. the attributes, the independent or explanatory variable(s)).

Linear regression



We need to make

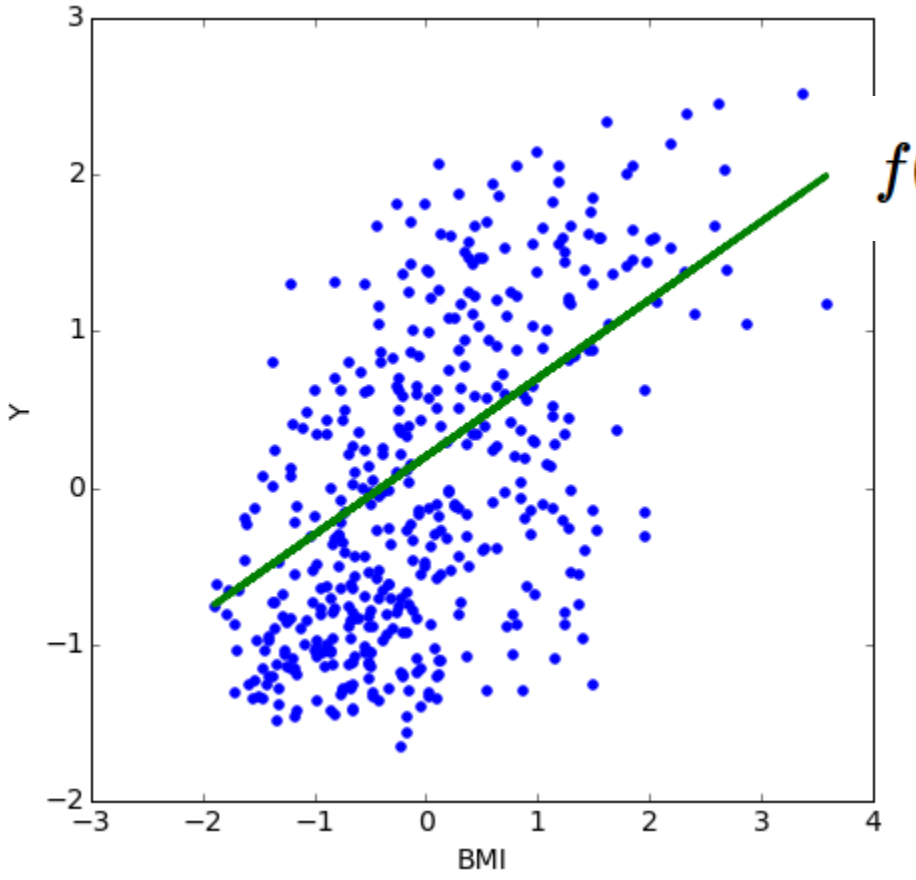
assumptions *linear relationship*

about the

model *linear model*

that generated the data.

Linear regression



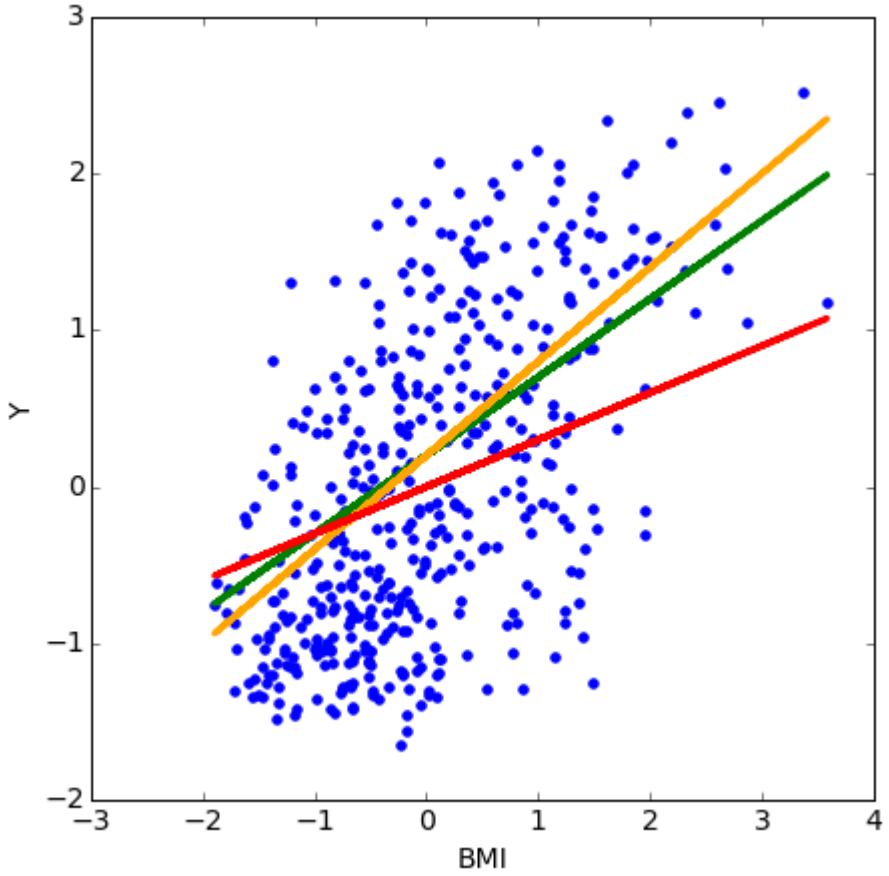
$$f(x) = 0.5 \cdot x + 0.2 = ax + b$$

a and *b* are model parameters

a is the slope to the line (direction)

b is the intercept or bias (position)

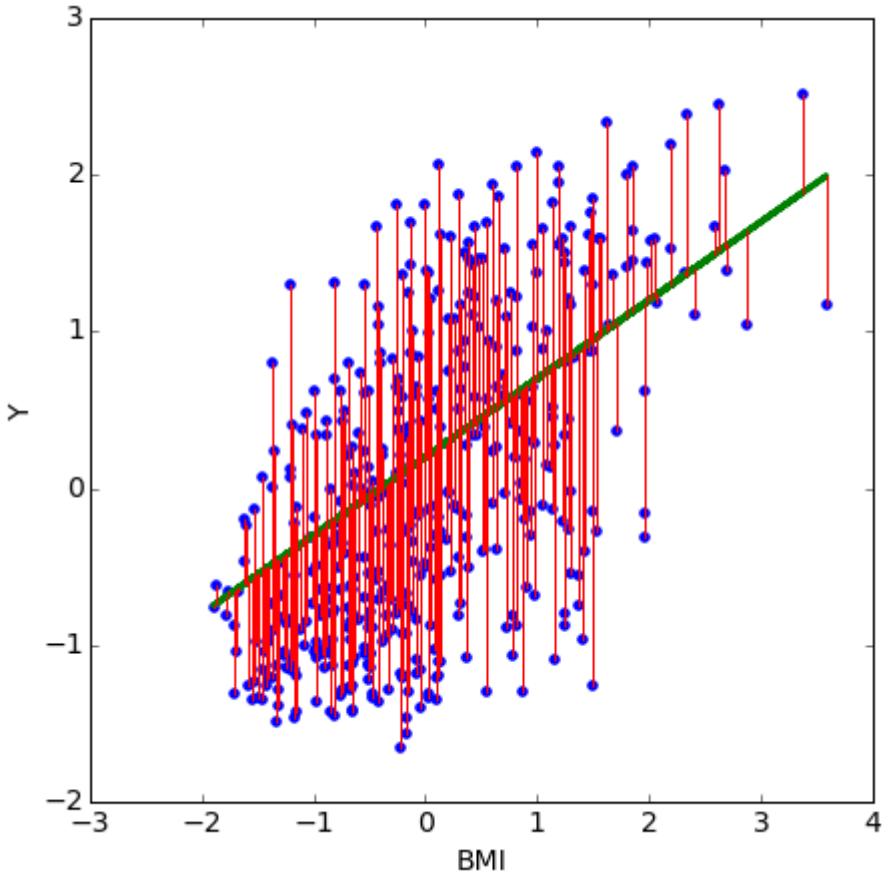
Data fitting

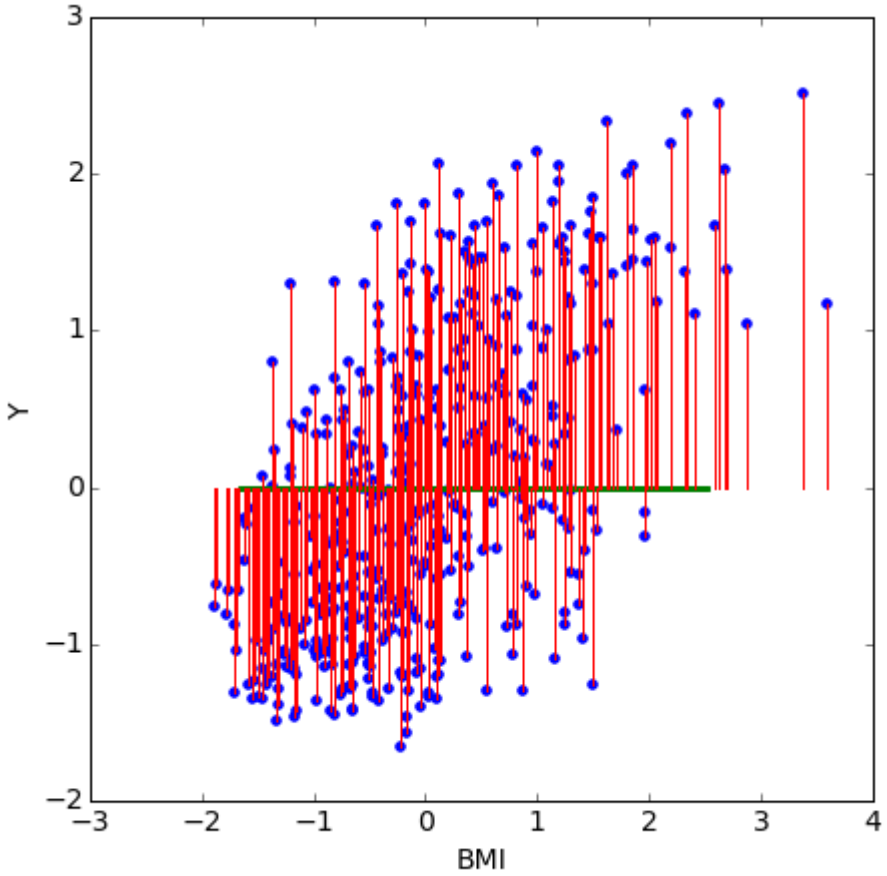


What values for *a* and *b* fit the data best?

How can we evaluate them?

Data fitting





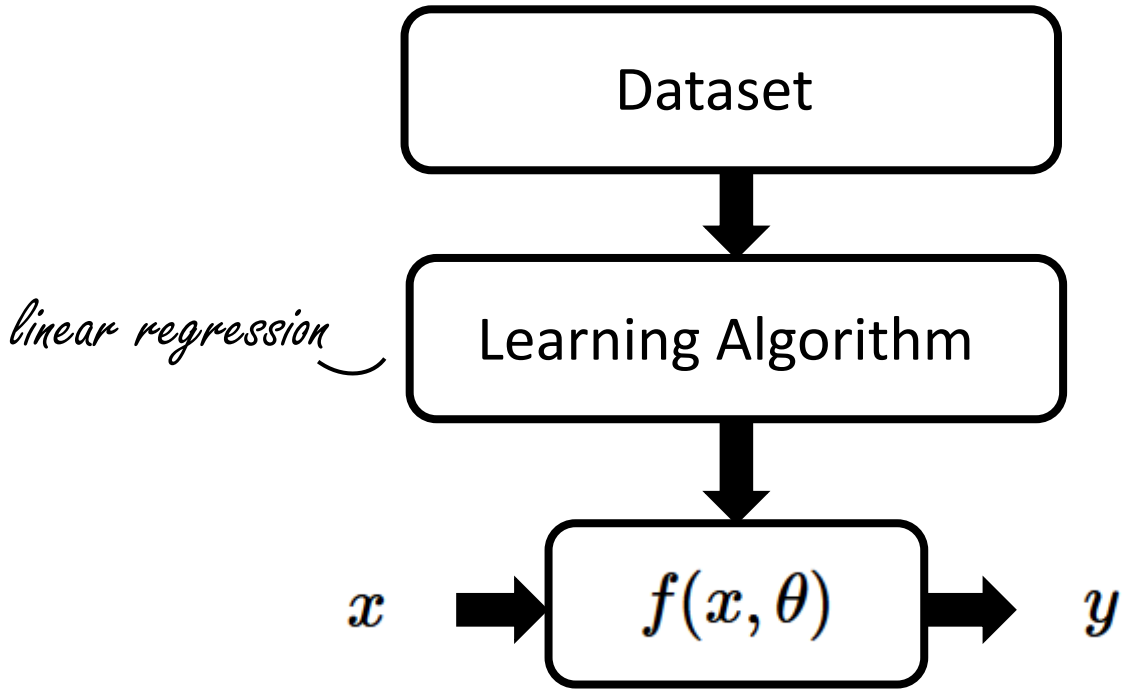
$$R^2 = 1 - \frac{SS_{error}}{SS_{total}}$$

$$R^2 = 0.296$$

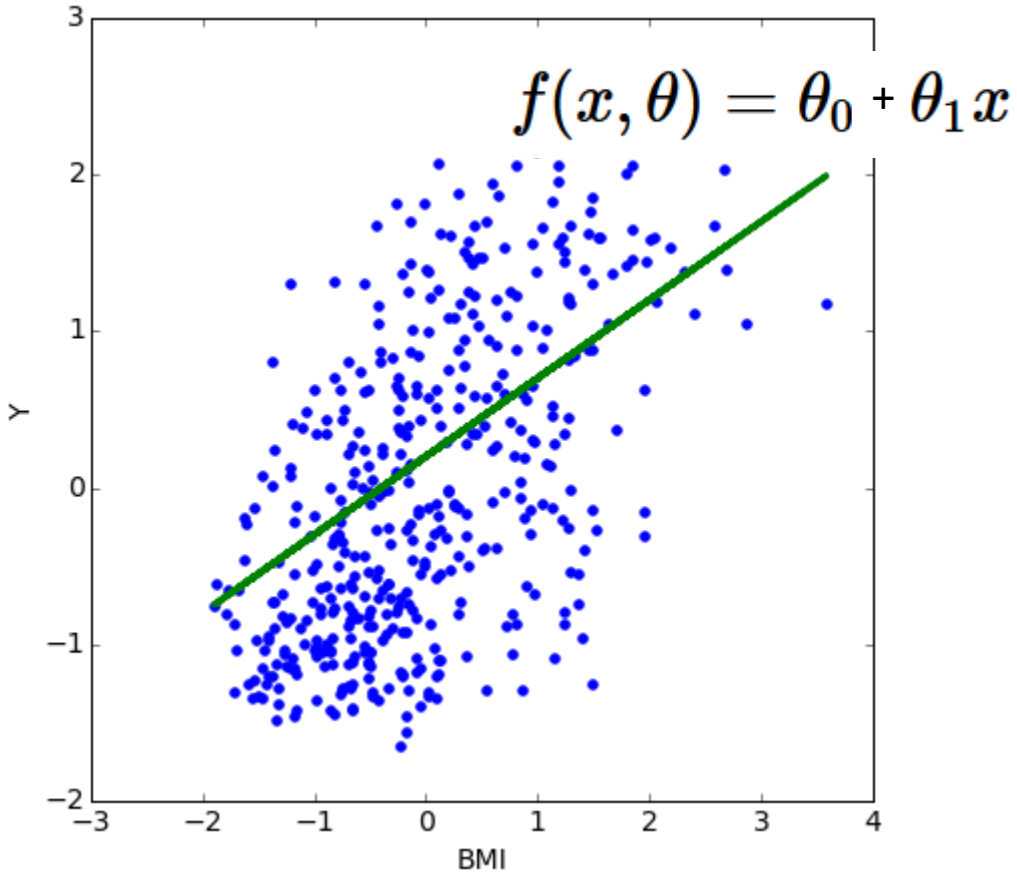
$$SS_{error} = \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2$$

$$SS_{total} = \sum_{i=1}^{n} (y^{(i)} - \bar{y})^2$$

Linear regression



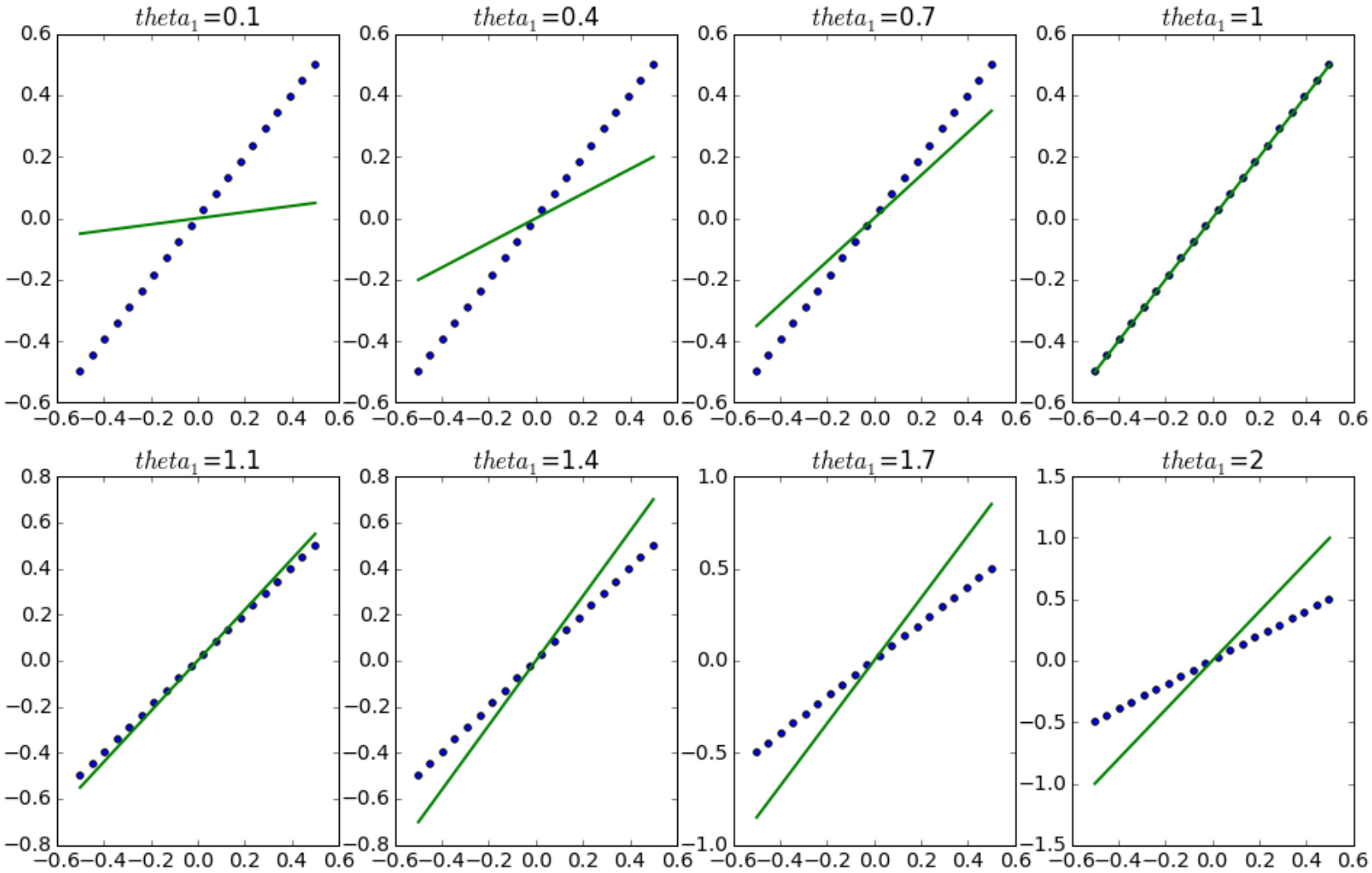
$$f(x,\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m = \theta' x$$



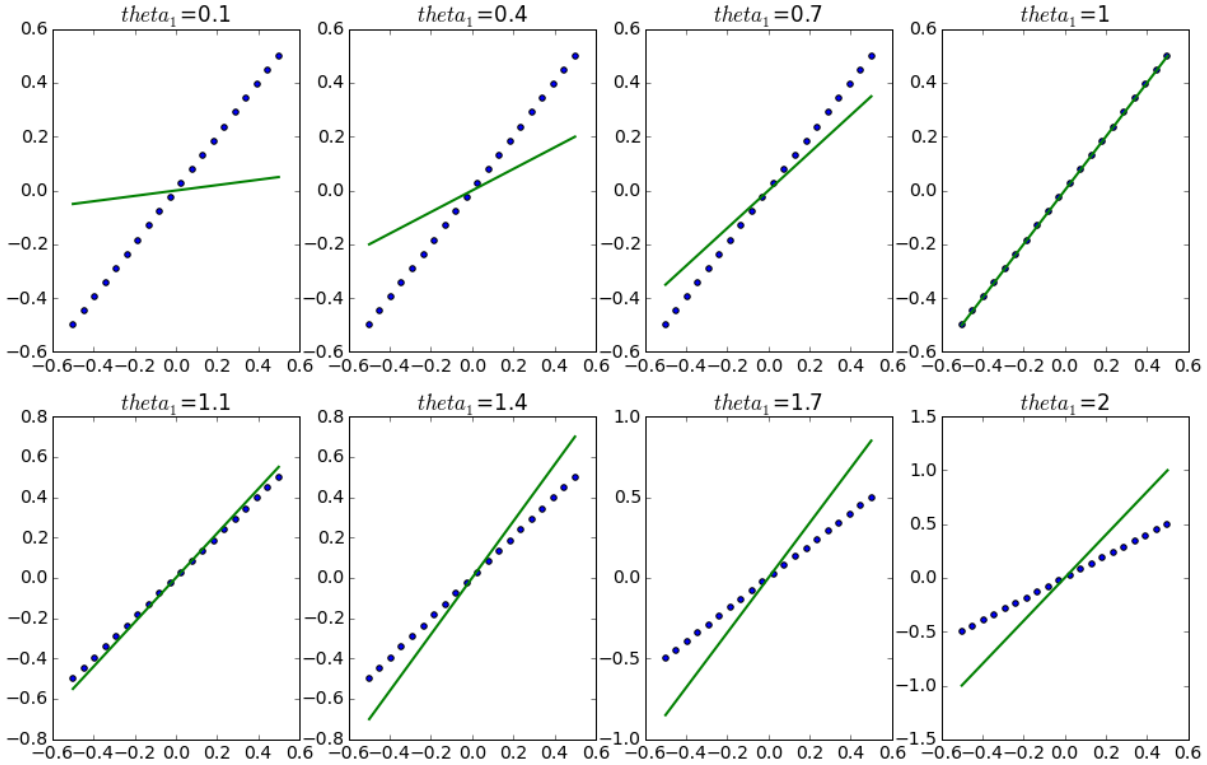
Cost function

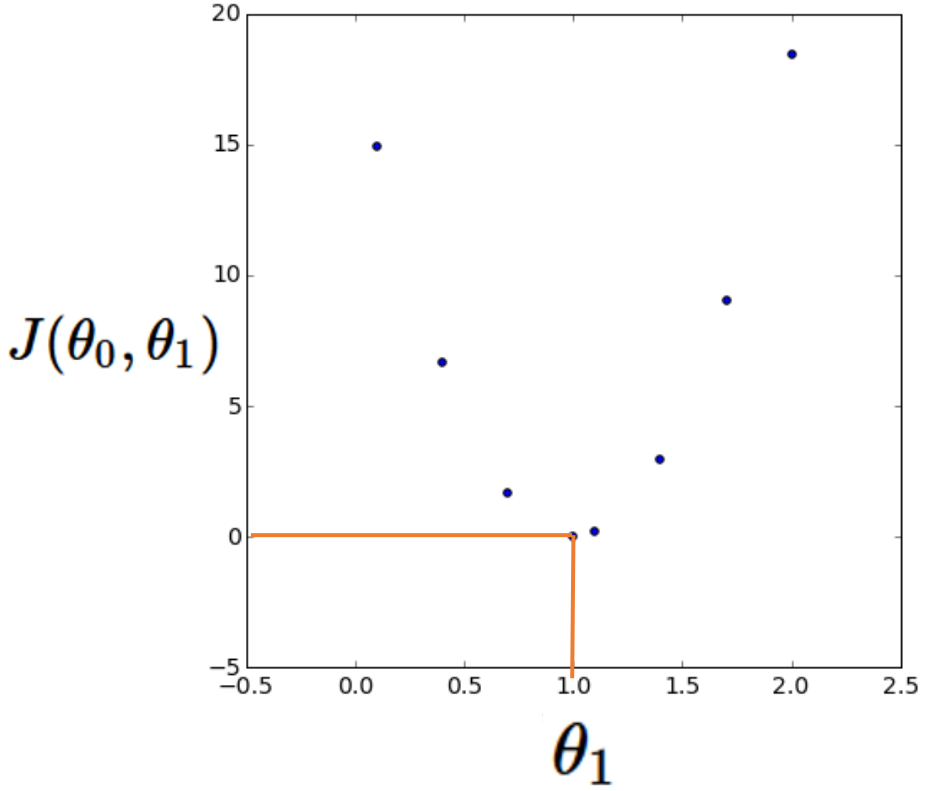
$$f(x) = x$$

$$(\theta_1 = 1, \theta_0 = 0)$$



Cost function





$$J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} (f(x^{(i)}, \theta) - y^{(i)})^2$$

Linear regression

Fit a linear model

$$f(x, \theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m = \theta' x$$

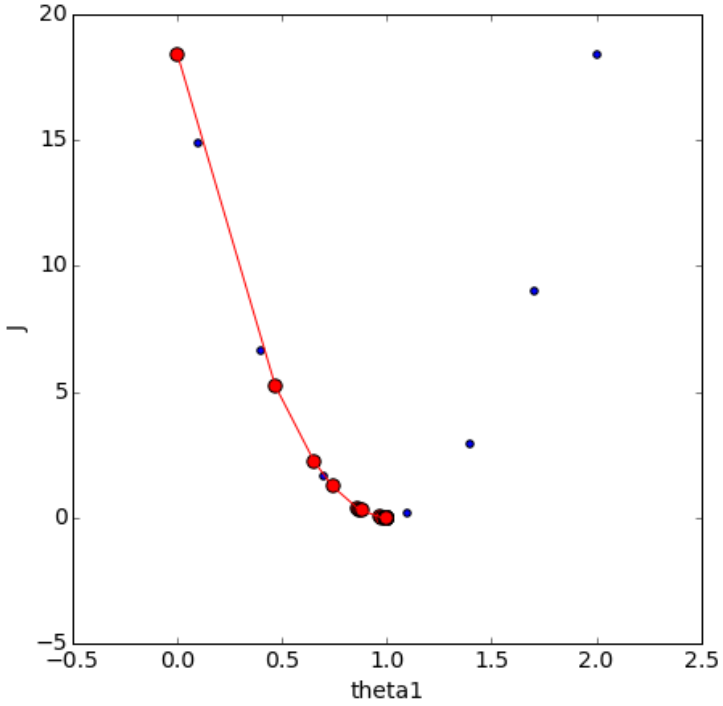
to the data set such that the cost function

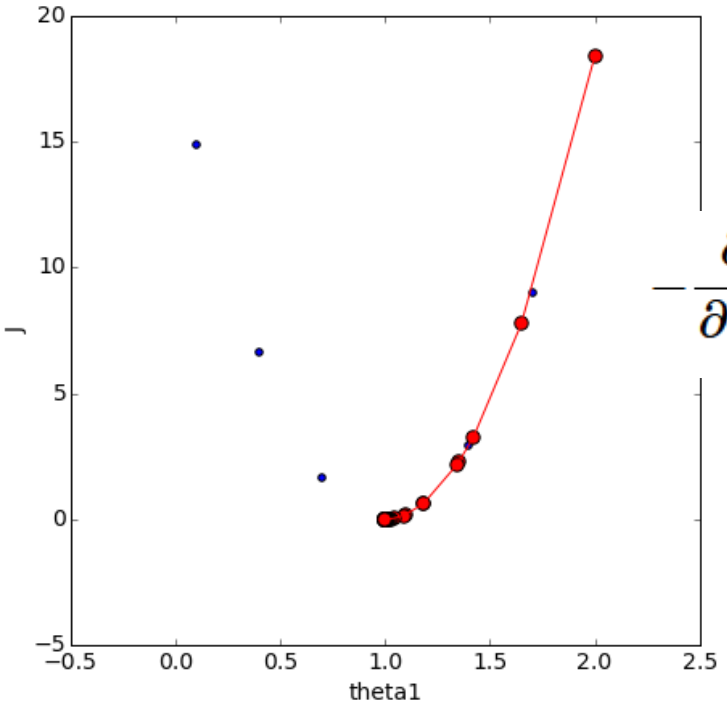
$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (f(x^{(i)}, \theta) - y^{(i)})^2$$

is minimal.

Gradient descent

1. start with some random initialization of $\boldsymbol{\theta}$, i.e. a randomly chosen model
2. increment or decrement the value(s) of $\boldsymbol{\theta}$ slightly such that $J(\boldsymbol{\theta})$ is reduced
3. repeat step 2. until we observe convergence





$$-\frac{\partial}{\partial \theta_j} J(\theta)$$

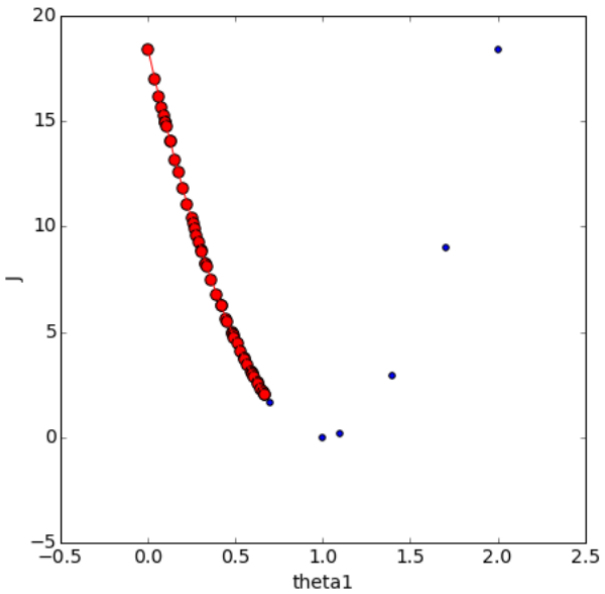
Gradient descent

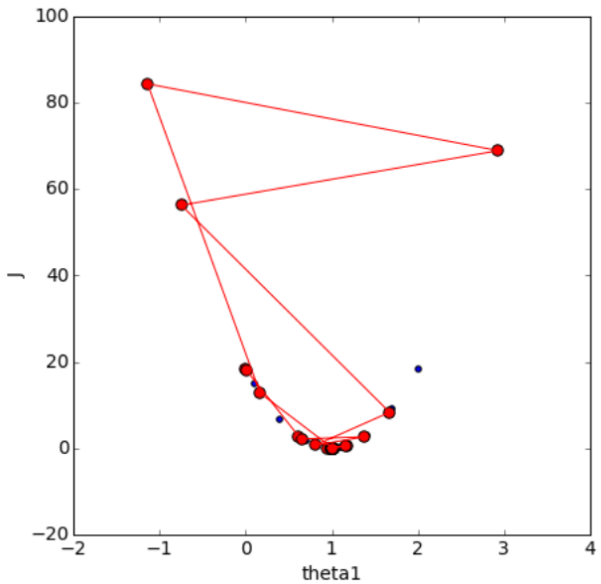
$$\theta_0 := \theta_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} (f(x^{(i)}, \theta) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} (f(x^{(i)}, \theta) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{n} \sum_{i=1}^{n} (f(x^{(i)}, \theta) - y^{(i)}) x_2^{(i)}$$

$$\ldots$$





Linear regression

model: $$f(x,\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m$$

cost function: $$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(f(x^{(i)},\theta) - y^{(i)})^2$$

Goal: $$\underset{J(\theta)}{\text{minimize}} \quad J(\theta)$$

Learning:
1. start with some θ
2. change θ to reduce $J(\theta)$
3. repeat 2. until convergence

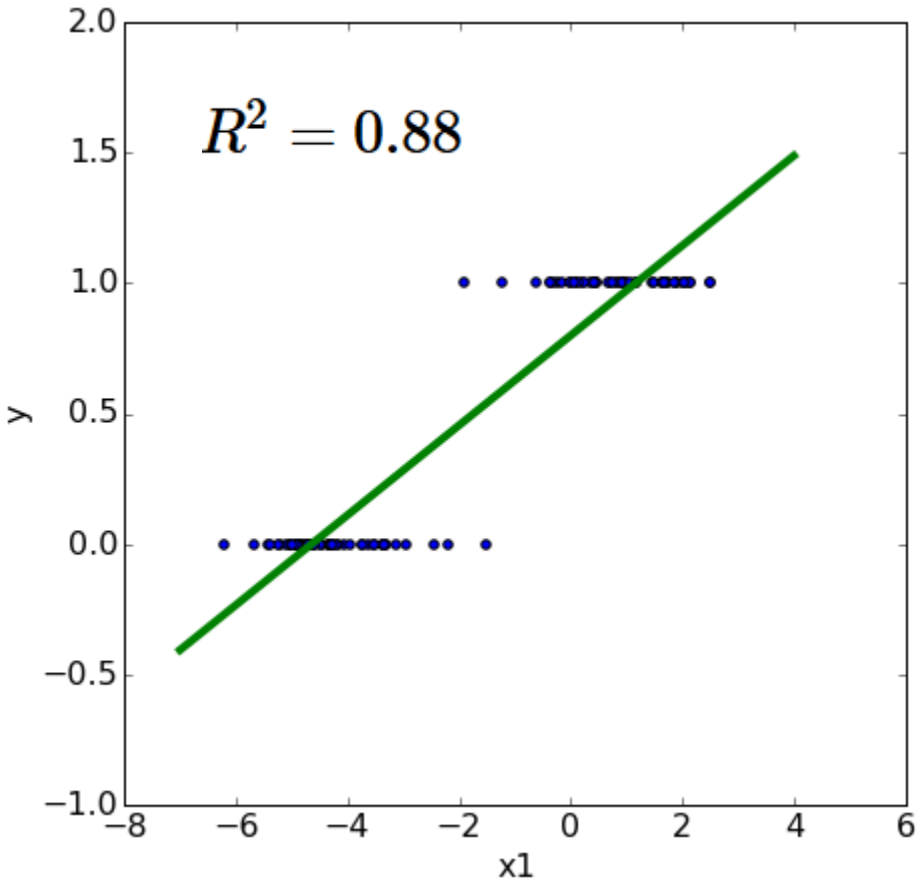
$$\theta_0 := \theta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}(f(x^{(i)},\theta) - y^{(i)})x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha\frac{1}{n}\sum_{i=1}^{n}(f(x^{(i)},\theta) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha\frac{1}{n}\sum_{i=1}^{n}(f(x^{(i)},\theta) - y^{(i)})x_2^{(i)}$$

...

Logistic regression



We need to make

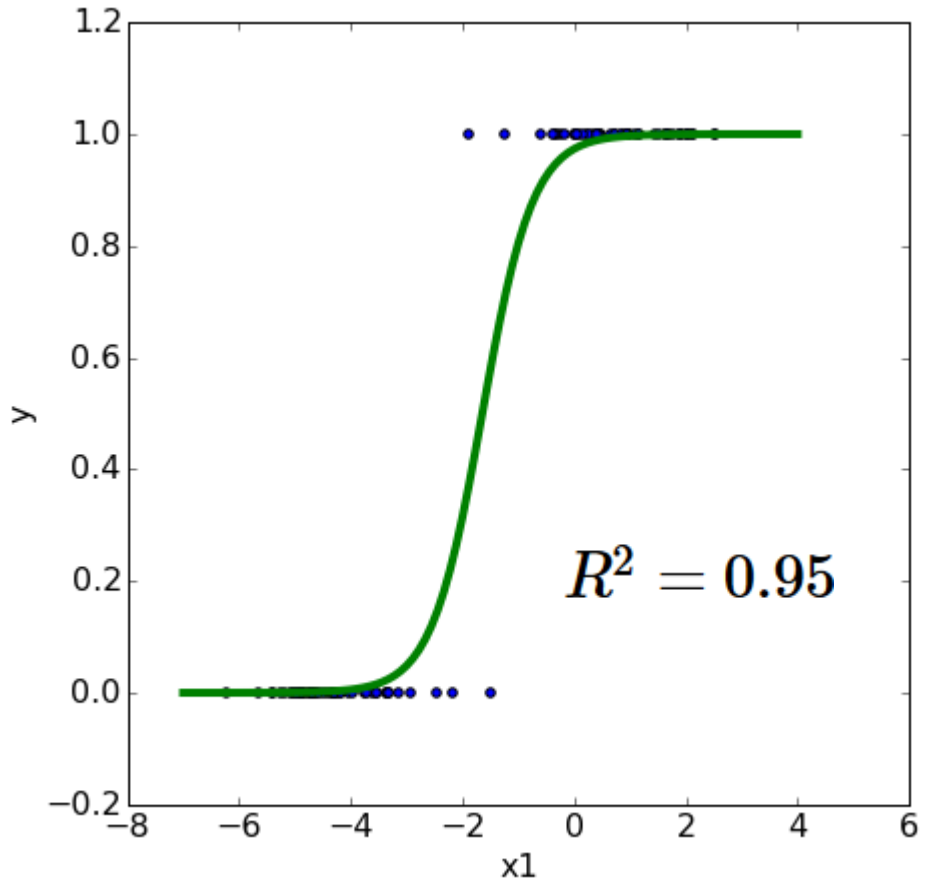
assumptions *linear relationship*

about the

model *linear model*

that generated the data.

Logistic regression



We need to make

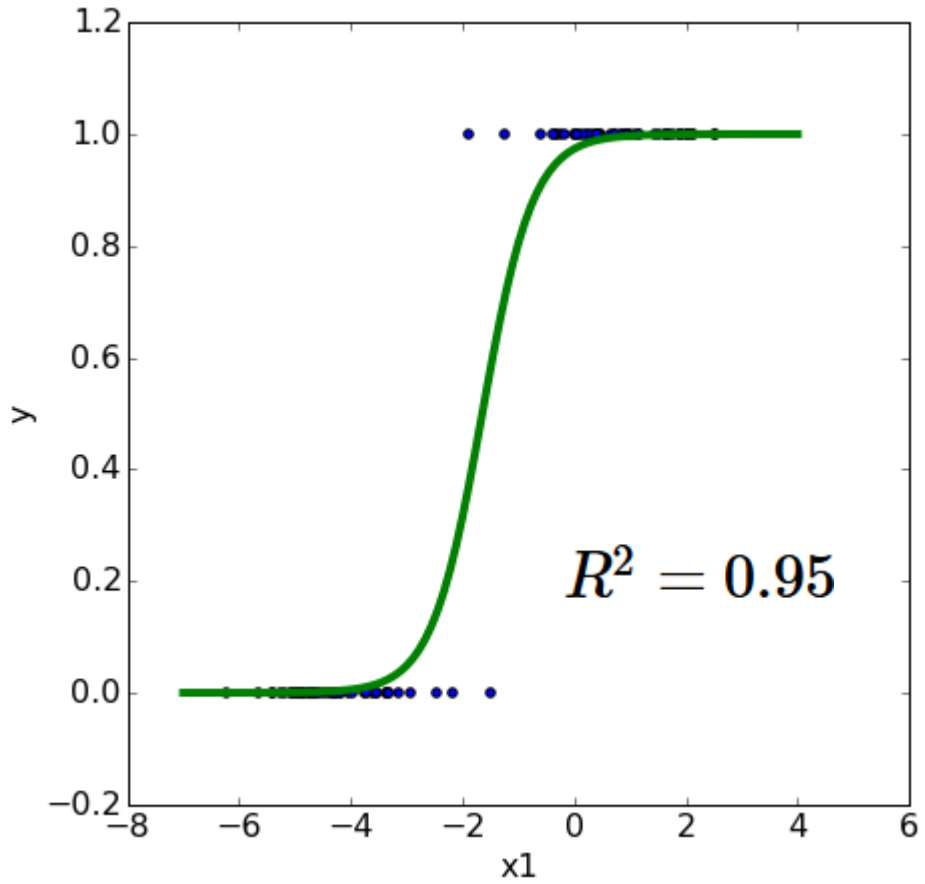
assumptions *linearly separable*

about the

model *logistic model*

that generated the data.

Logistic regression



$$f(x, \theta) = g(\theta_0 + \theta_1 x_1)$$

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

Logistic regression

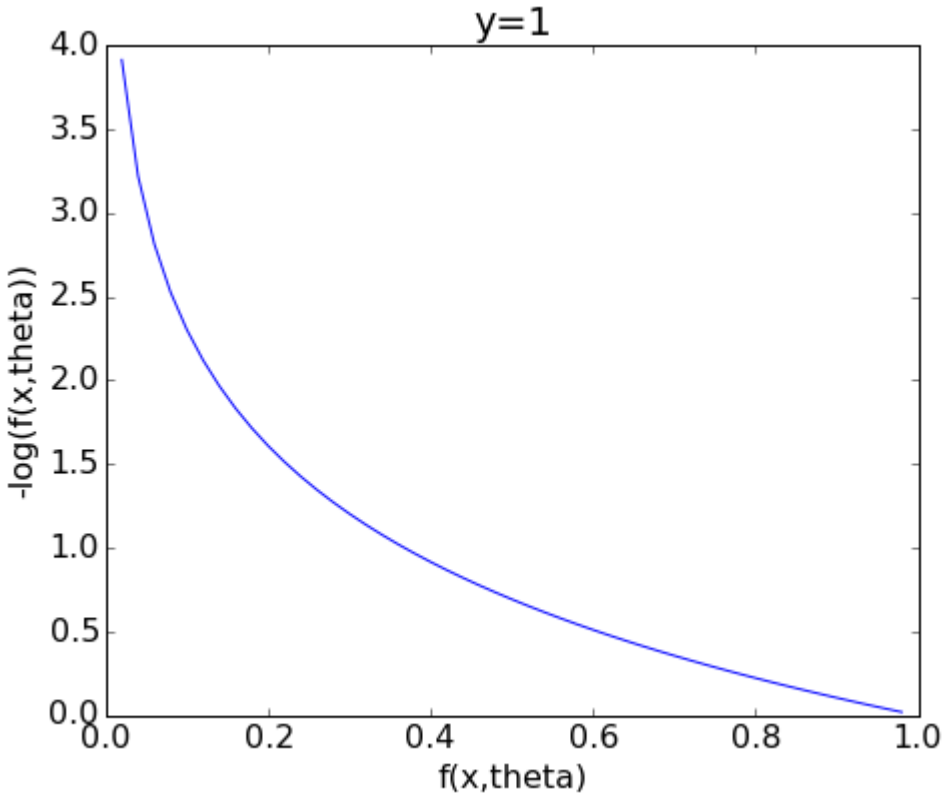
$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)}, \theta)) + (1 - y^{(i)}) log(1 - f(x^{(i)}, \theta))]$$

We know that $y^{(i)}$ is either 0 or 1. If $y^{(i)} = 1$ then the cost function $J(\theta)$ is incremented by $-log(f(x^{(i)}, \theta))$.

Similarly, if $y^{(i)} = 0$ then the cost function $J(\theta)$ is incremented by $-log(1 - f(x^{(i)}, \theta))$.

Cost function

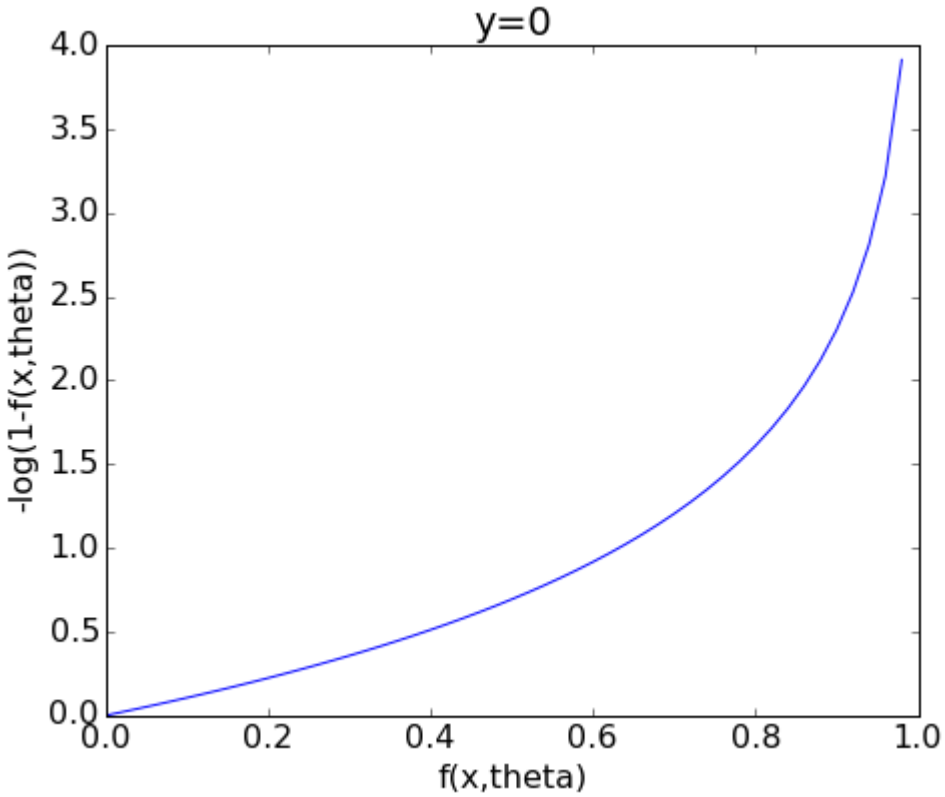
We know that $y^{(i)}$ is either 0 or 1. If $y^{(i)} = 1$ then the cost function $J(\theta)$ is incremented by $-log(f(x^{(i)}, \theta))$.



Cost function

Similarly, if $y^{(i)} = 0$ then the cost function $J(\theta)$ is incremented by

$$-log(1 - f(x^{(i)}, \theta)).$$



Logistic regression

Fit a logistic model

$$f(x,\theta) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m) = g(\theta' x)$$

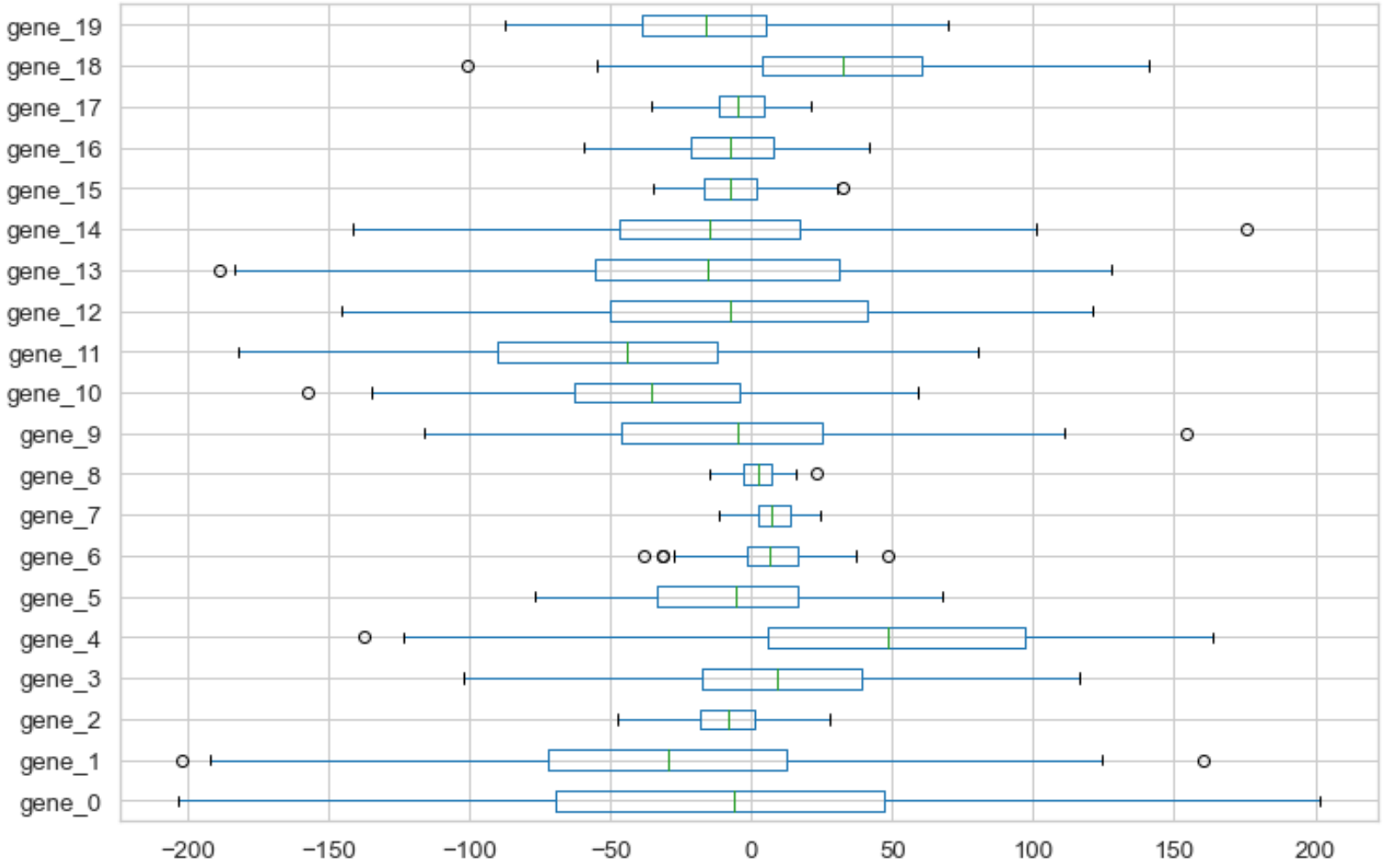
to the data set such that the cost function

$$J(\theta) = -[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} log(f(x^{(i)},\theta)) + (1 - y^{(i)}) log(1 - f(x^{(i)},\theta))]$$

is minimal using gradient descent

$$\theta_j := \theta_j - \alpha \frac{1}{n}\sum_{i=1}^{n}(f(x^{(i)},\theta) - y^{(i)})x_j^{(i)}$$

Classification: feature normalization



Classification: feature normalization

same scale for features

e.g. Eccentricity [0,100], Solidity [-5,7]

weights all features equally in their representation

standardization

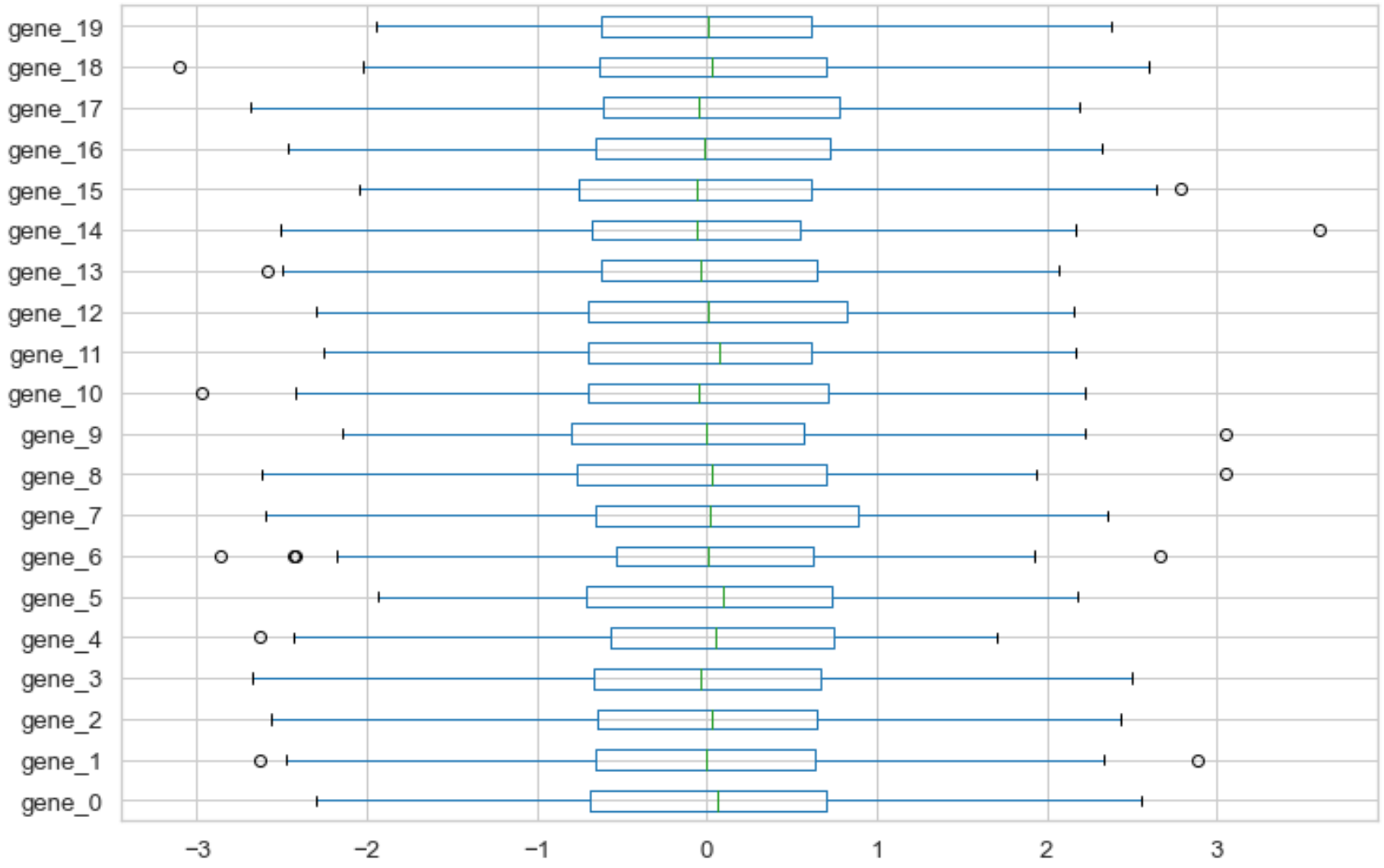
$$\mu = 0 \quad \sigma = 1$$

$$x_{norm} = \frac{x - \mu}{\sigma}$$

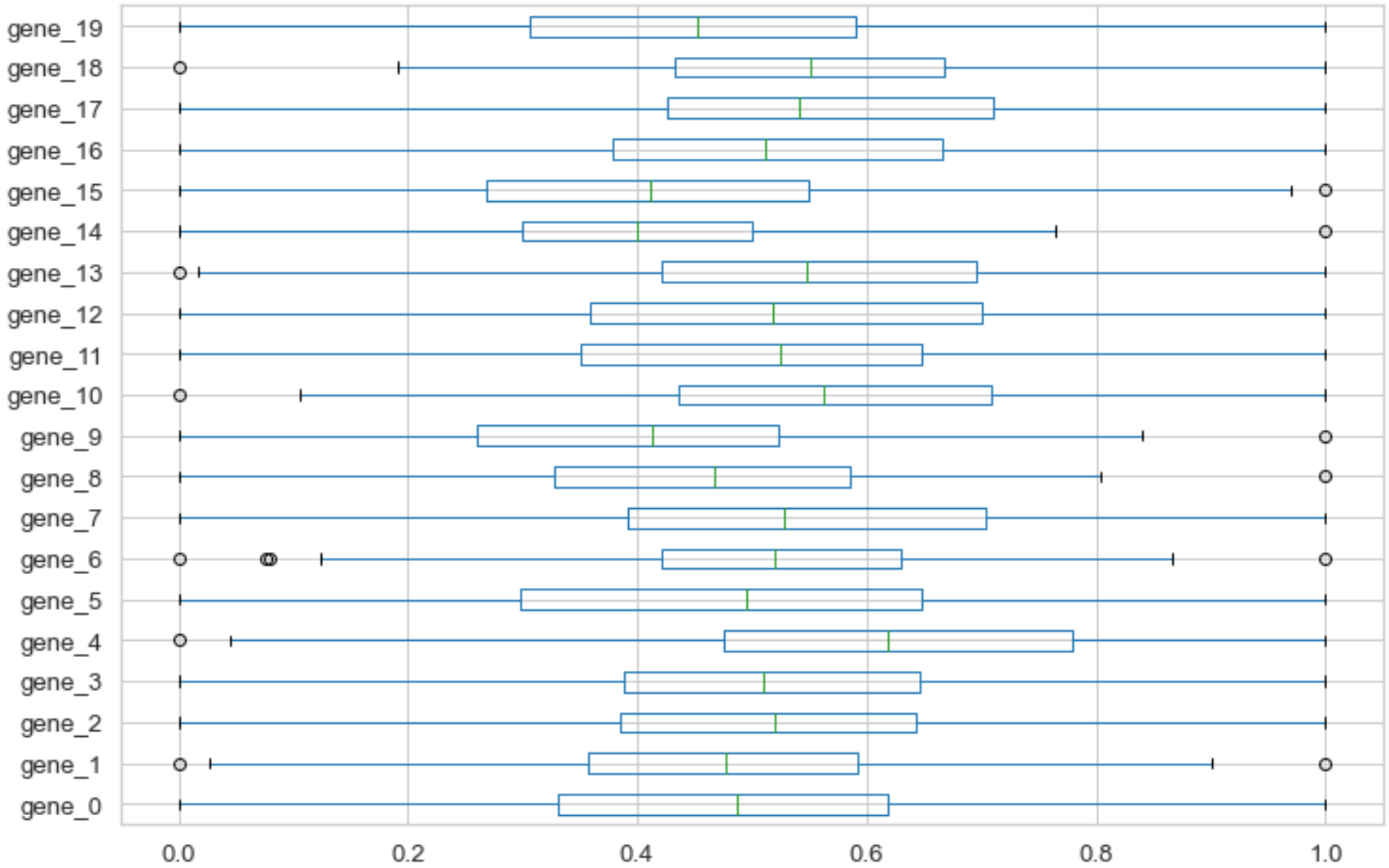
$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

min-max scaling: scale the features to a fixed range

Classification: feature standardization

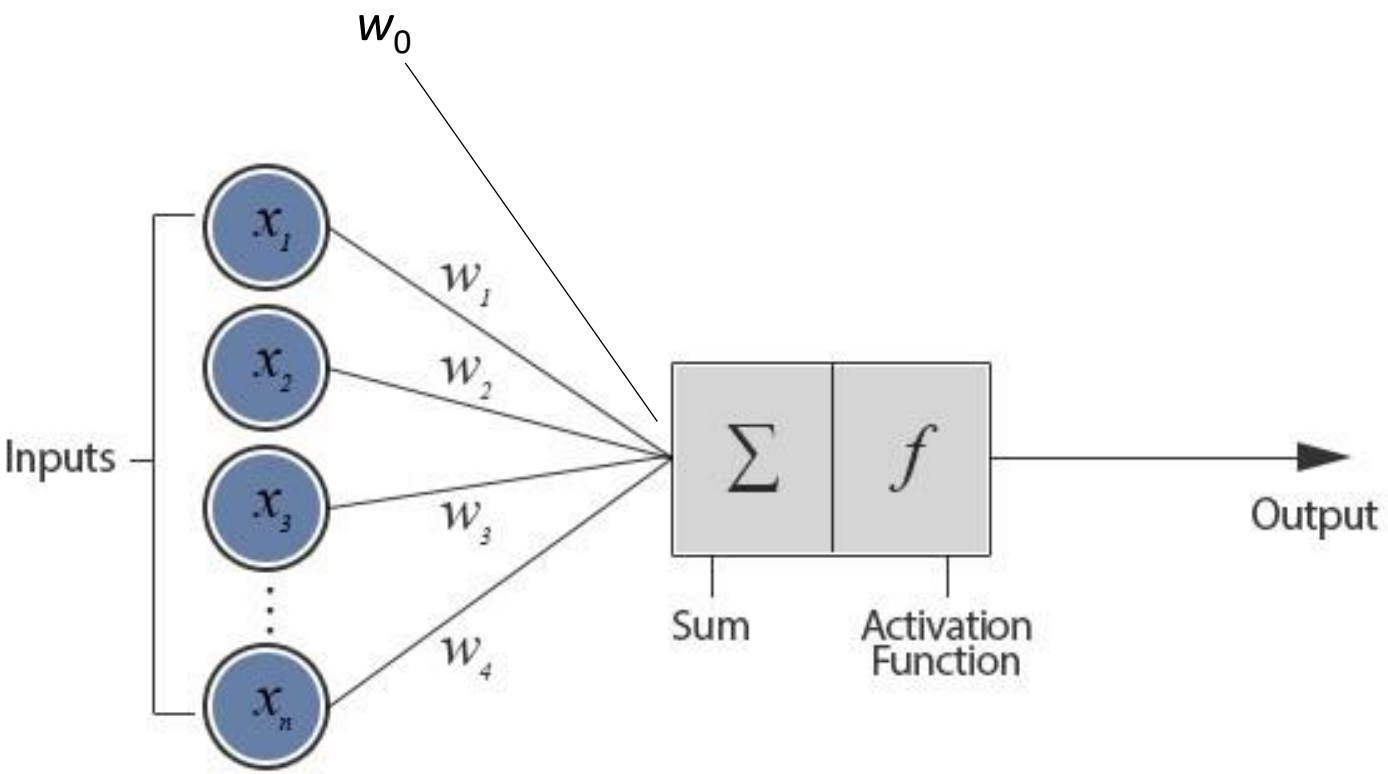


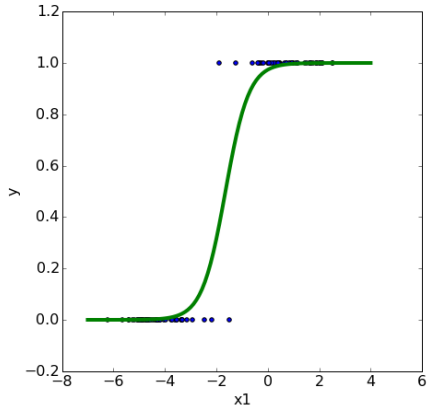
Classification: feature scaling



Neural networks

$$f(x,\theta) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m) = g(\theta' x)$$





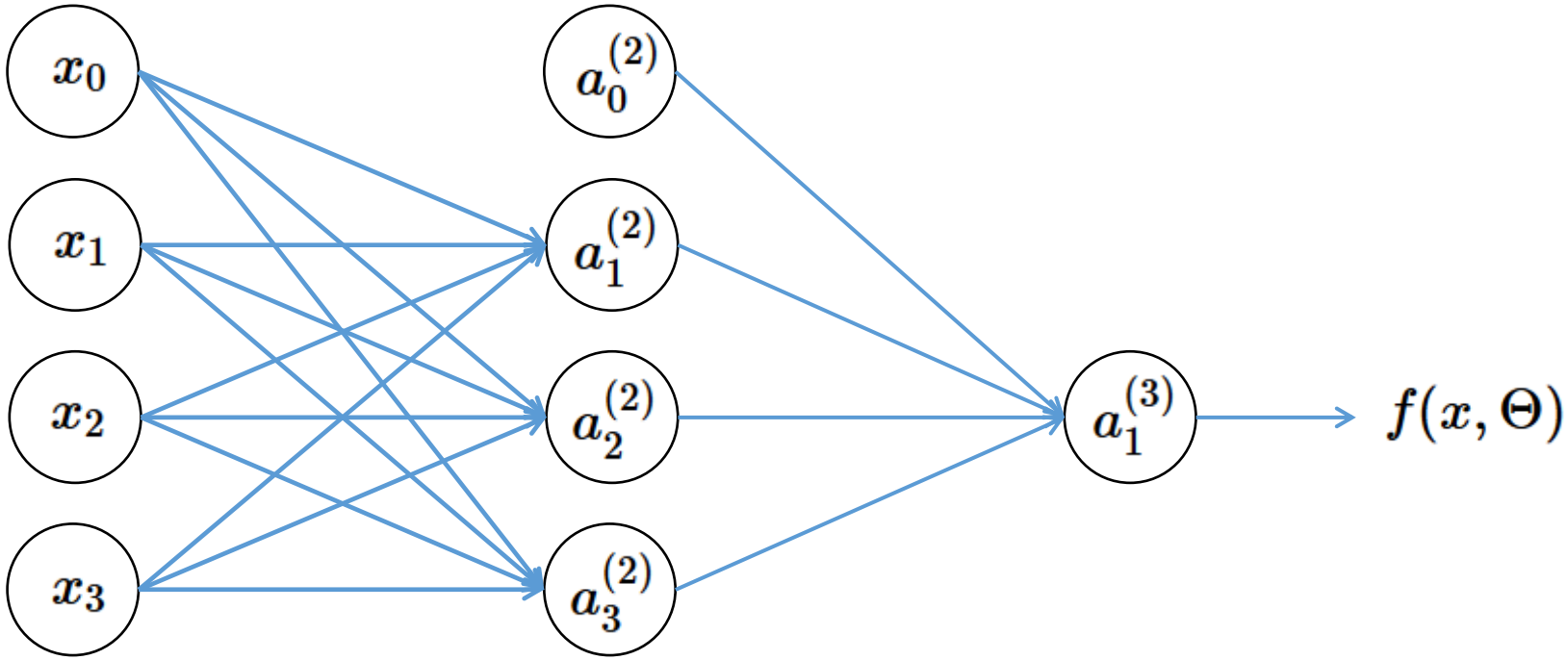
Neural networks

$$f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



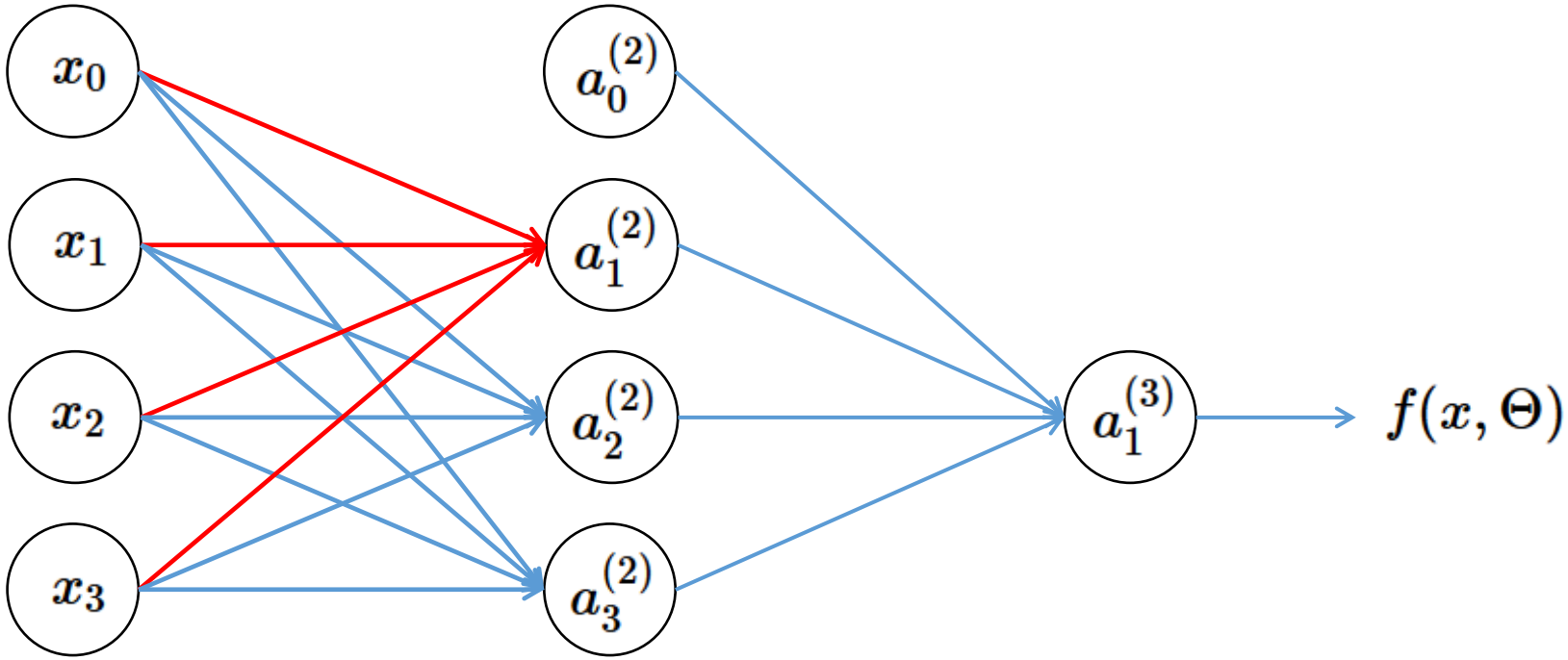
Neural networks

$$f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$\boxed{a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)}$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



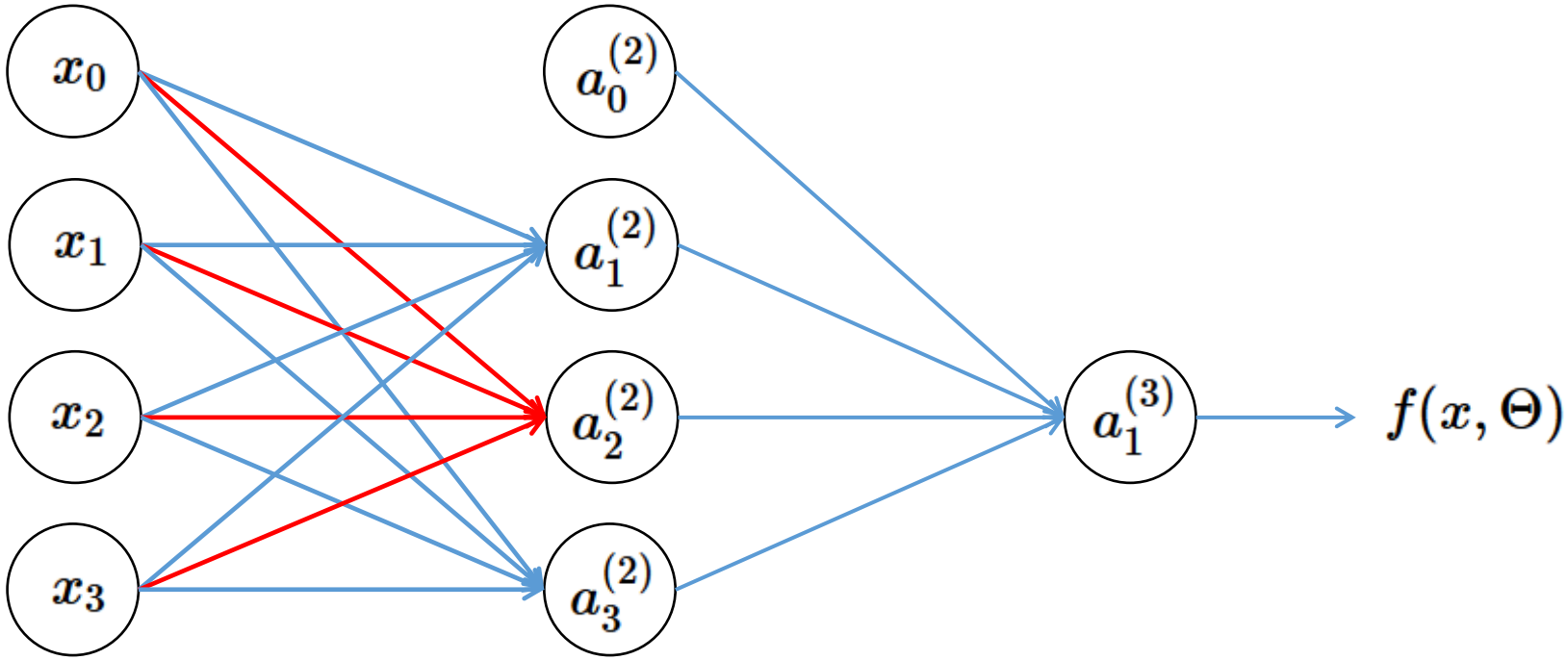
Neural networks

$$f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$\boxed{a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)}$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



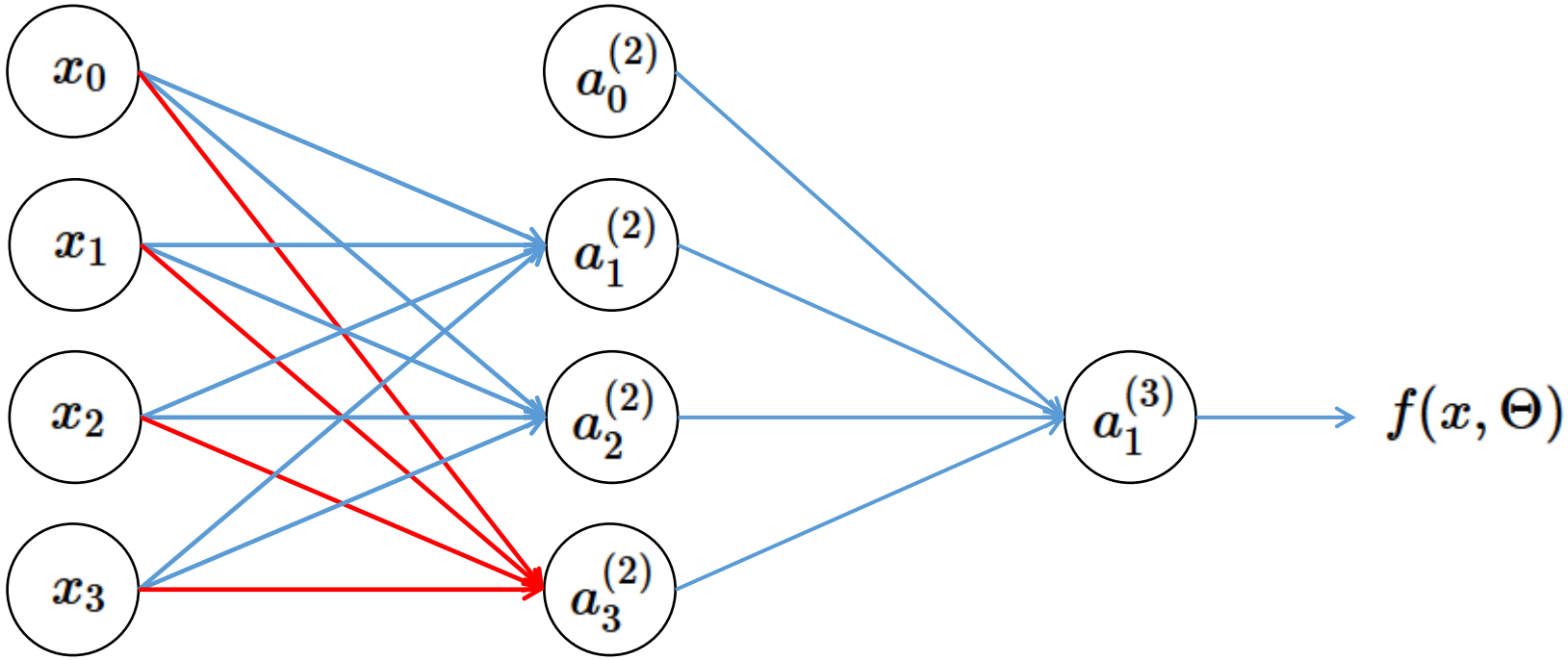
Neural networks

$$f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$\boxed{a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)}$$



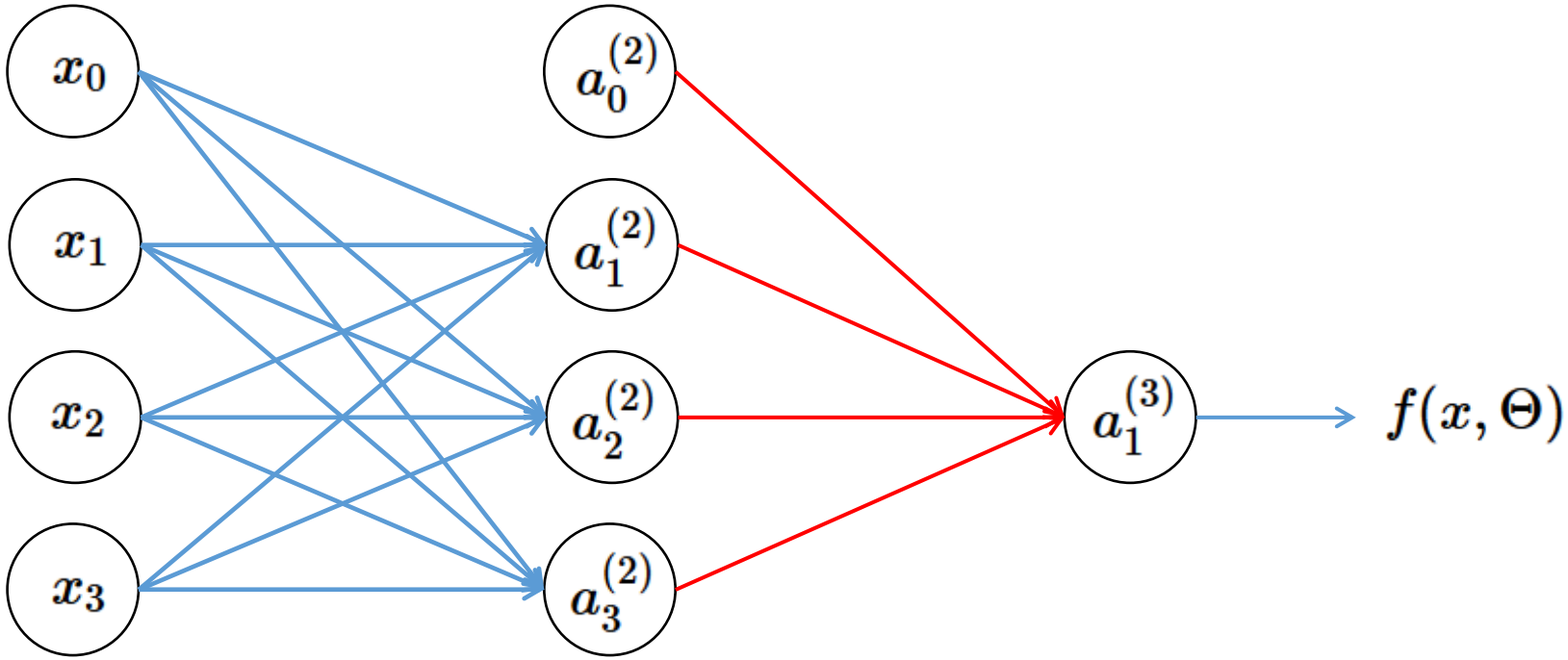
Neural networks

$$\boxed{f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})}$$

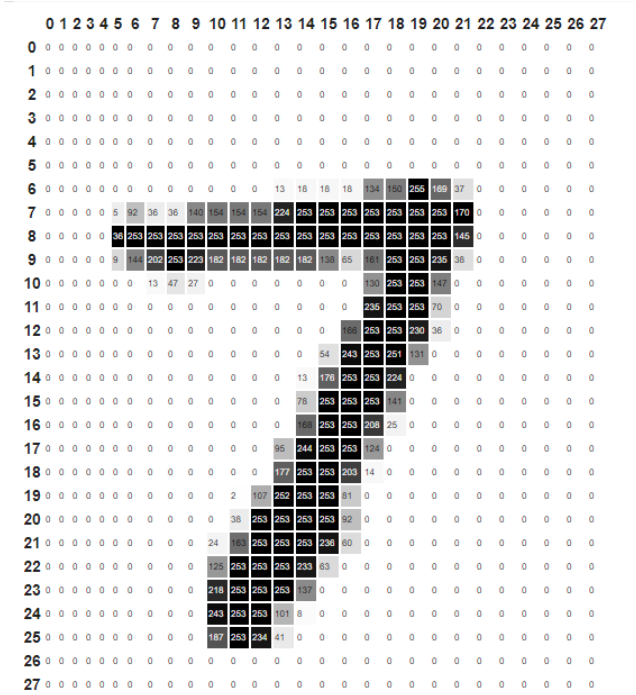
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

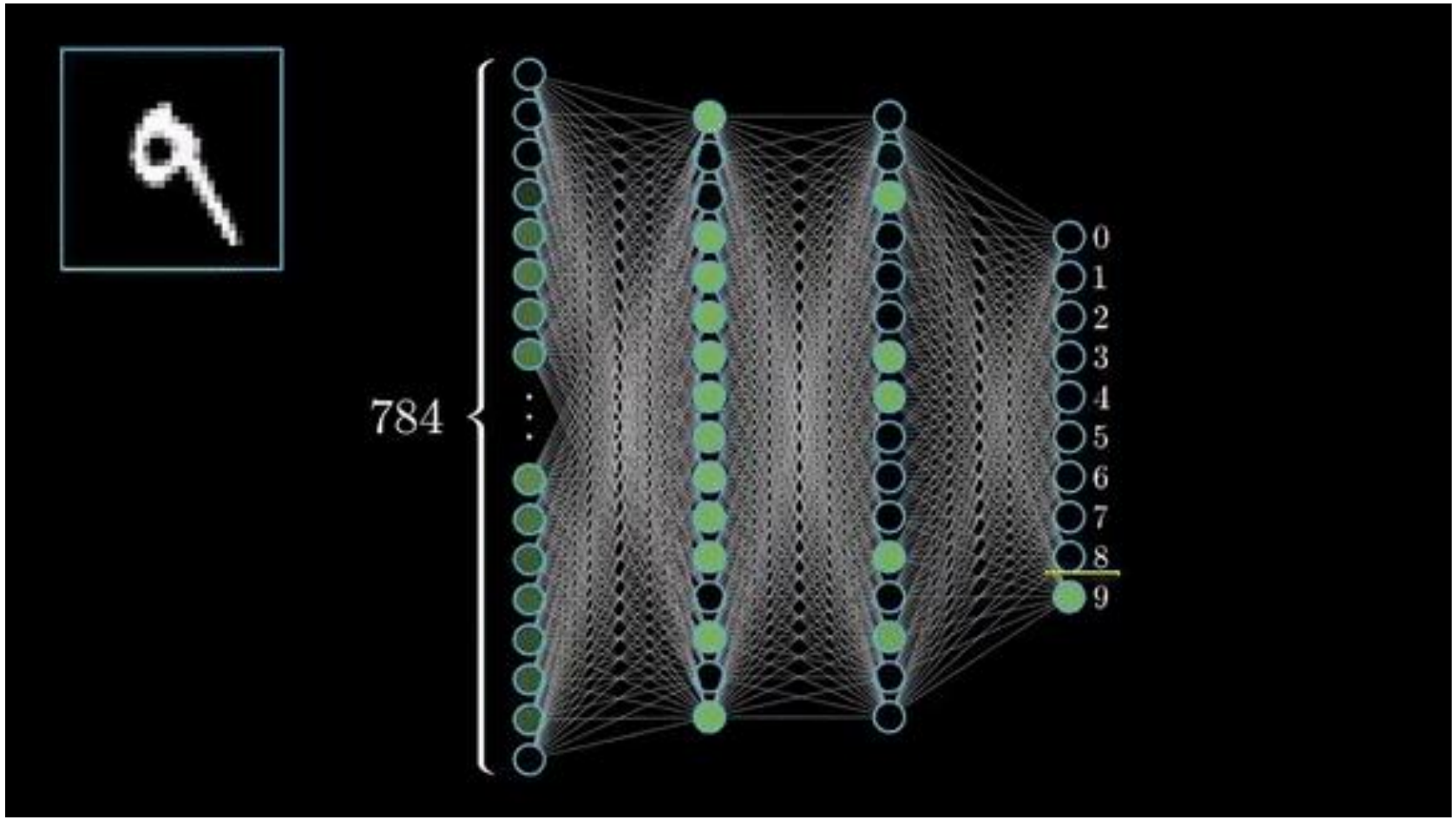
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



Neural networks





Neural networks

Cost function

Logistic regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

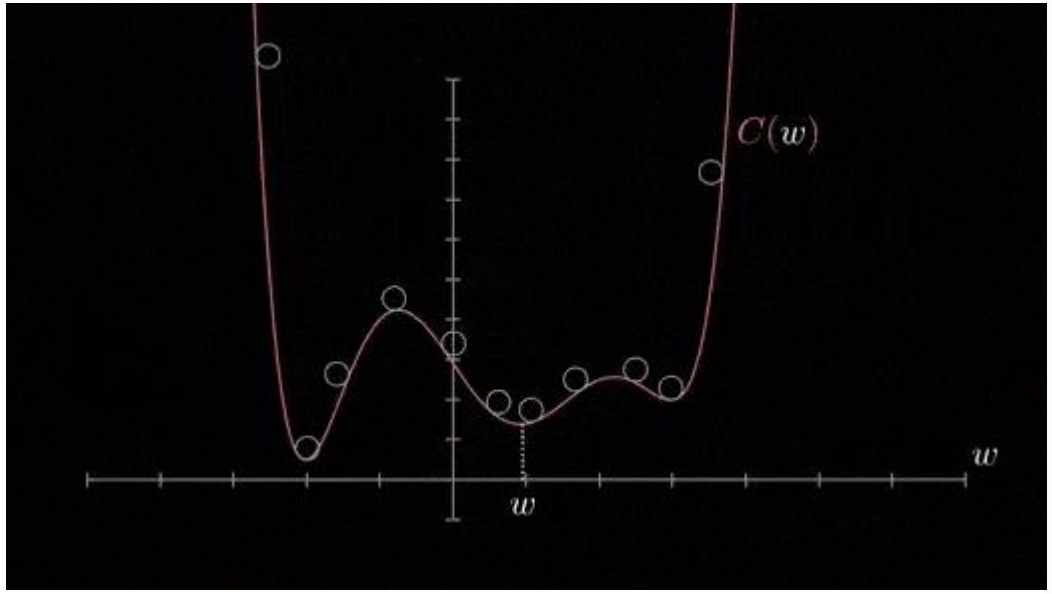
Neural network:

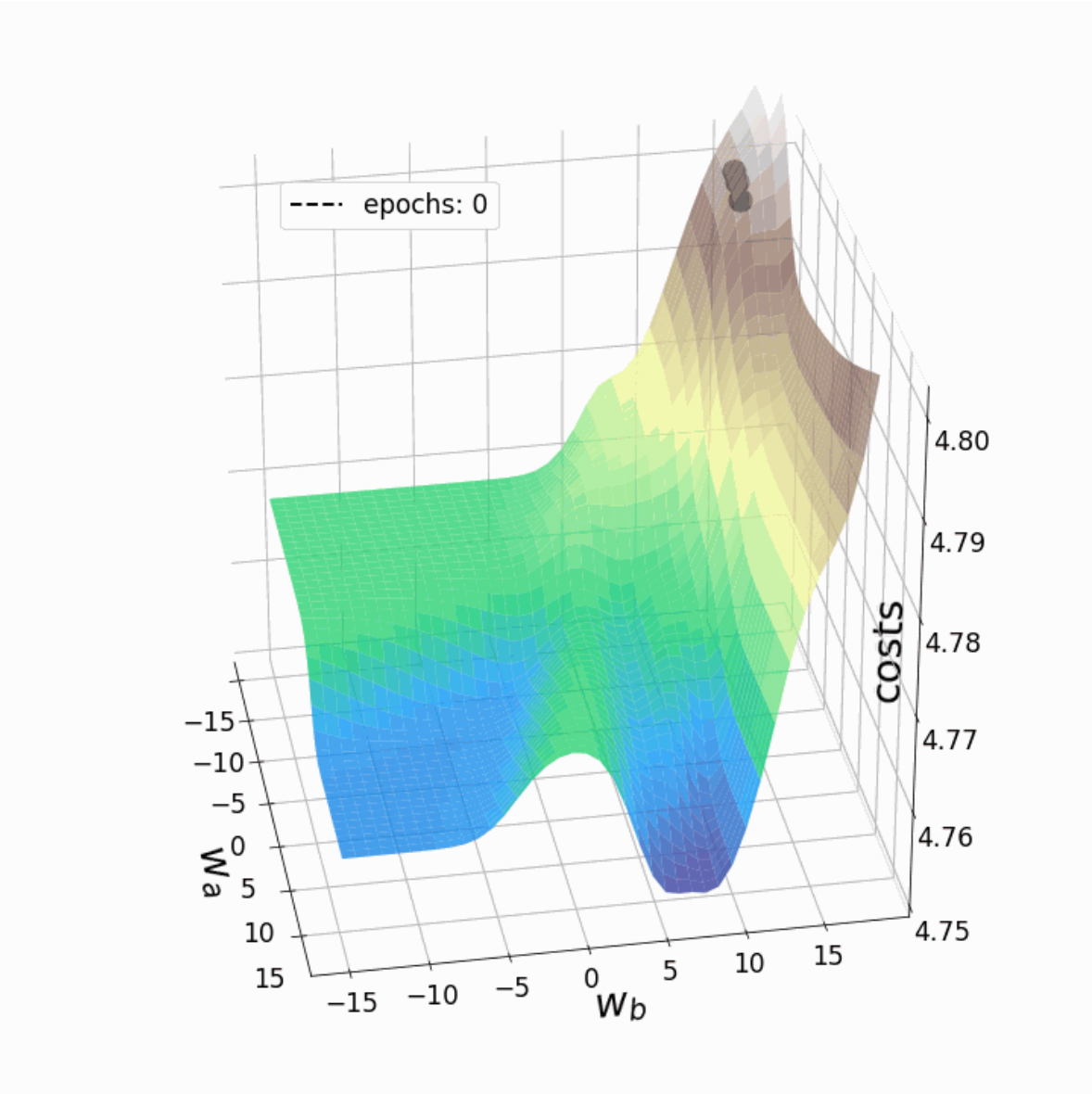
$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

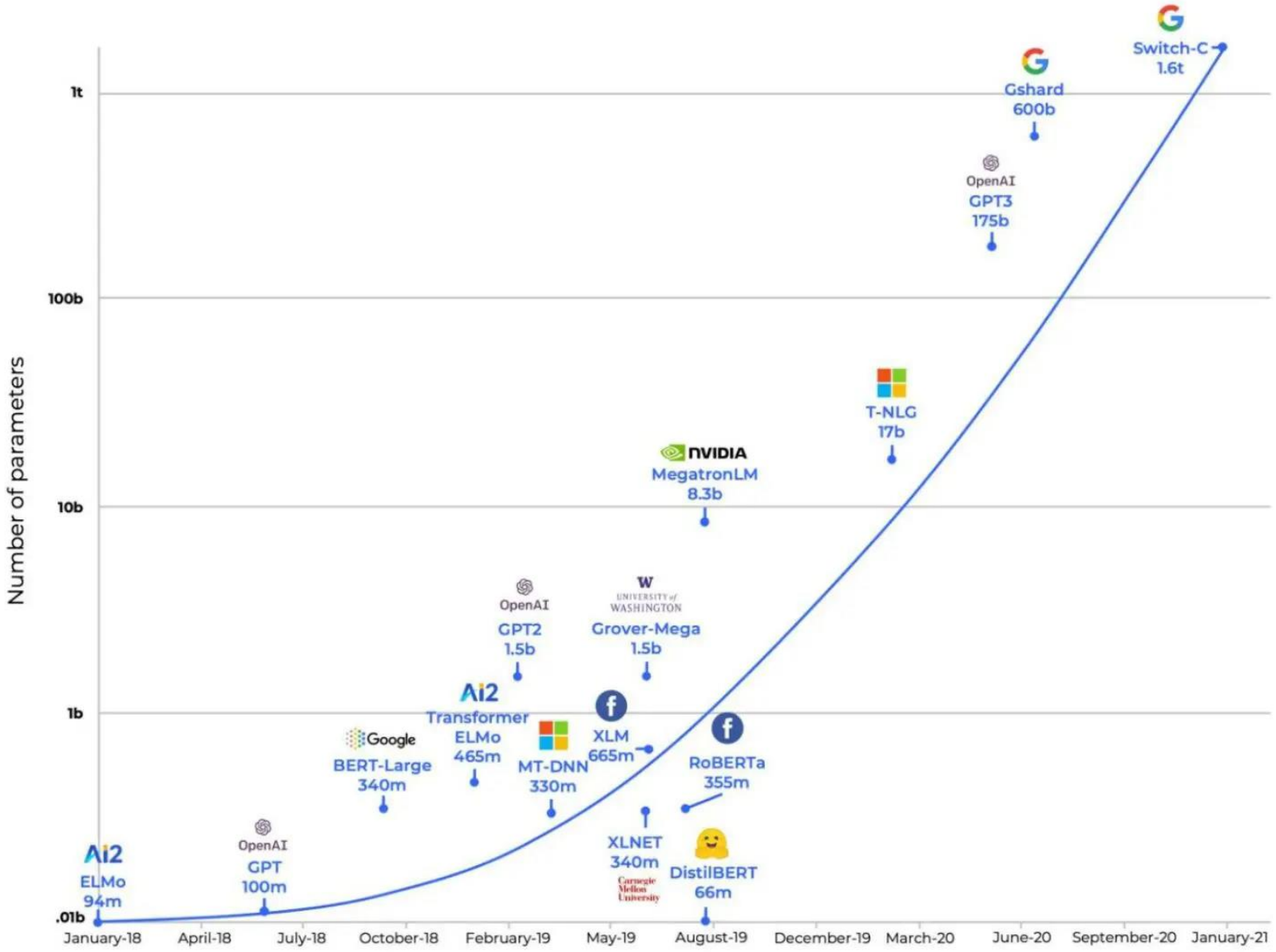
$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

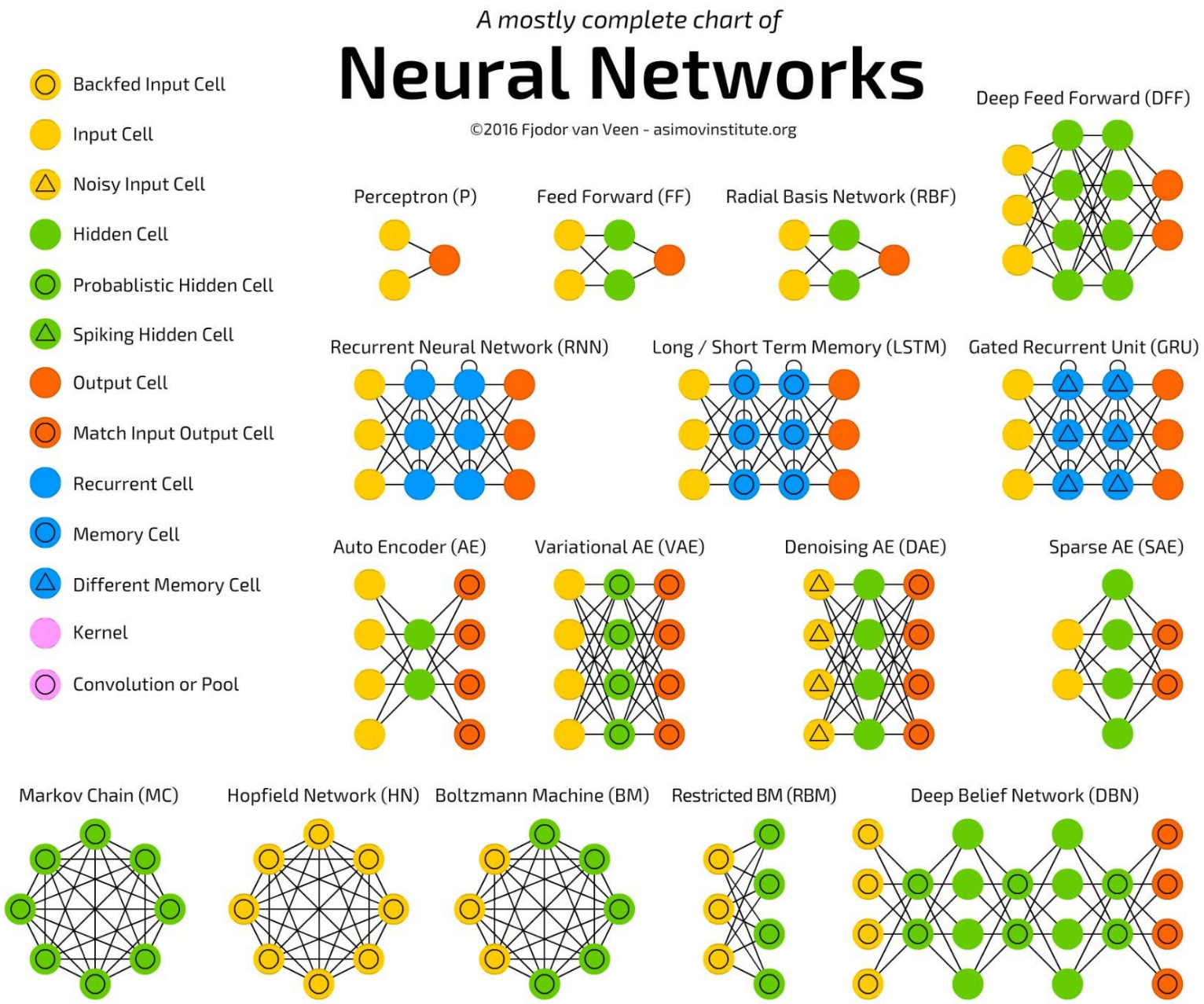
Gradient descent (back-propagation)



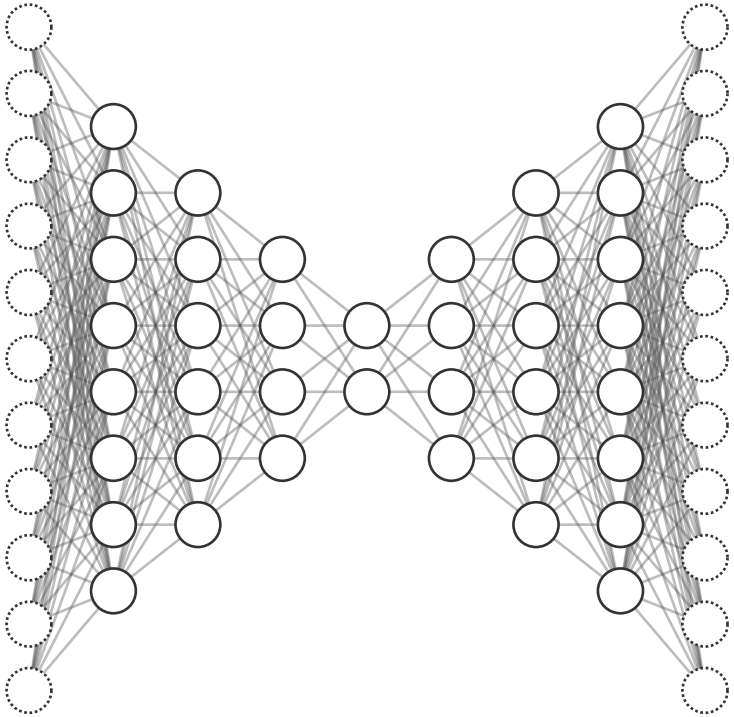




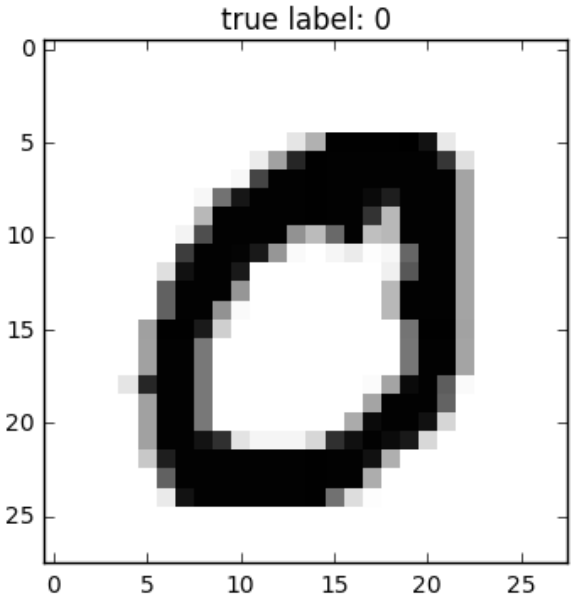
Source: https://deci.ai

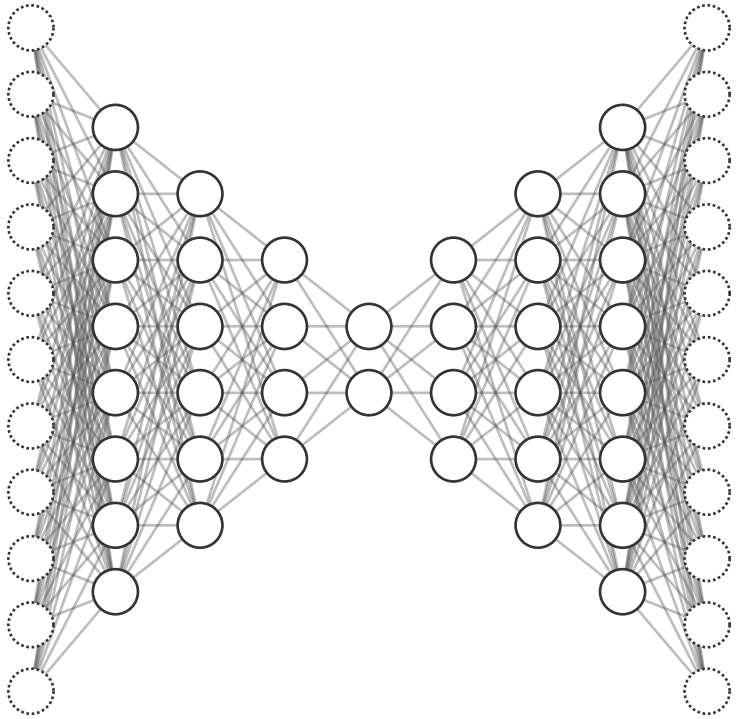

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)

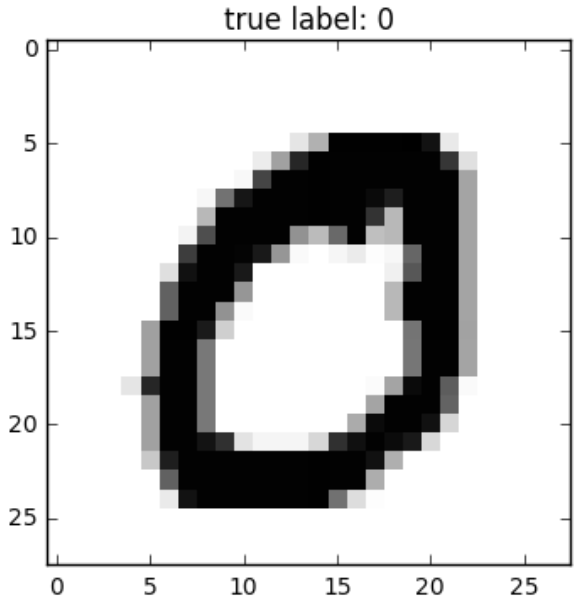
Auto encoder



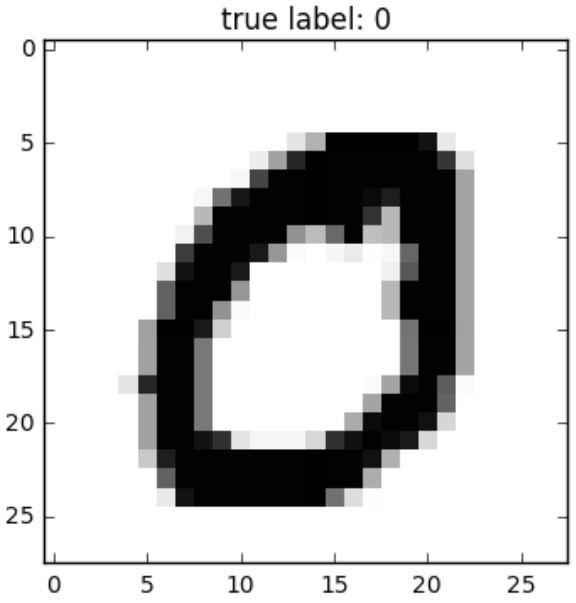
Auto encoder

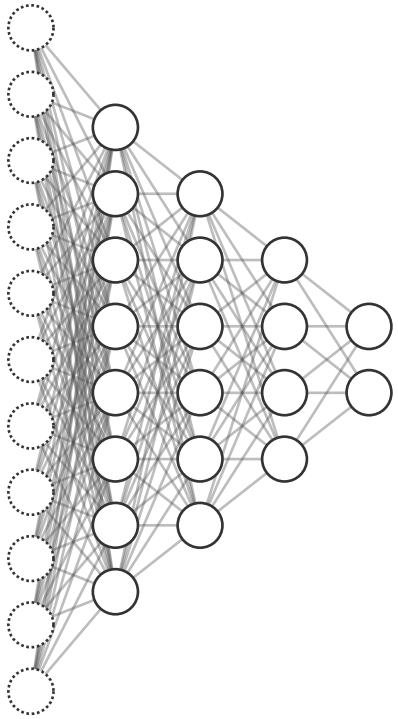


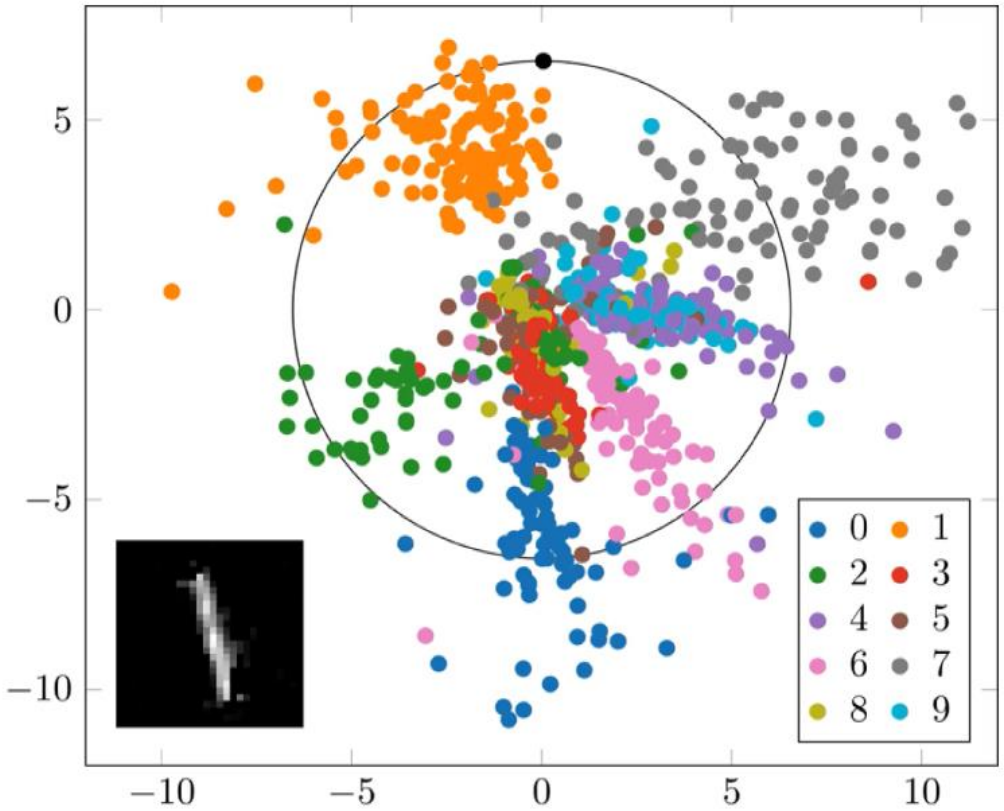




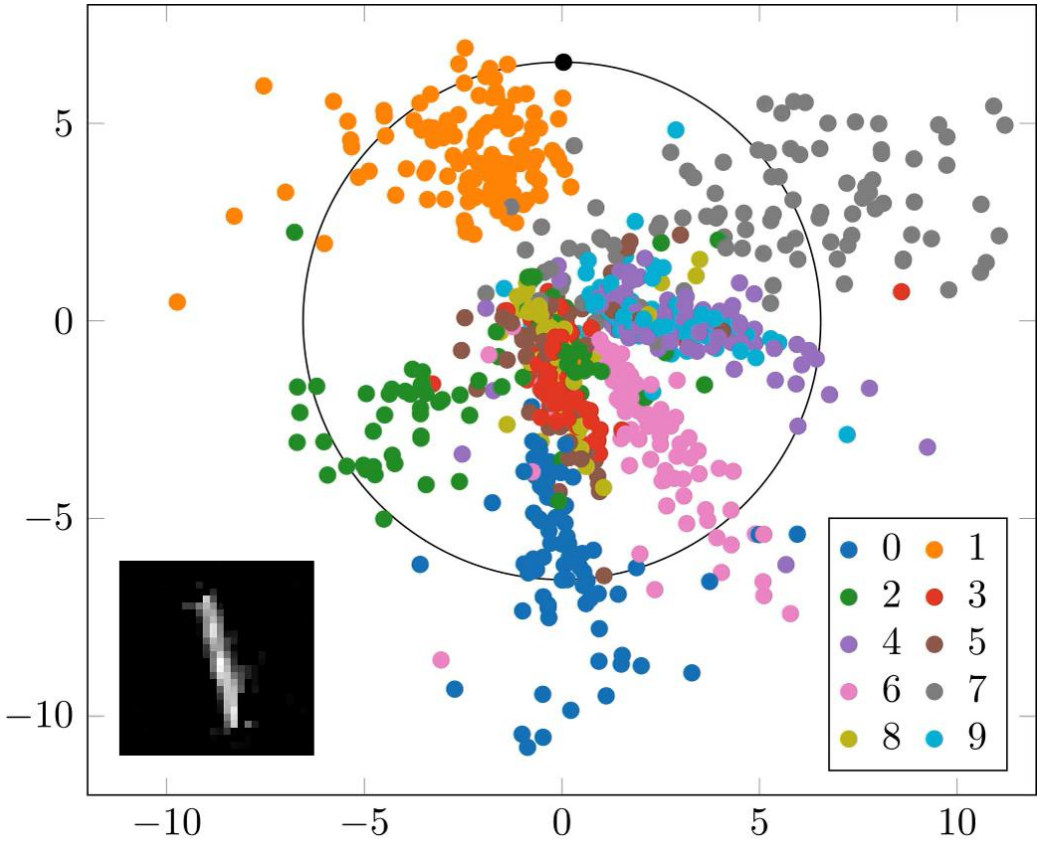
Auto encoder

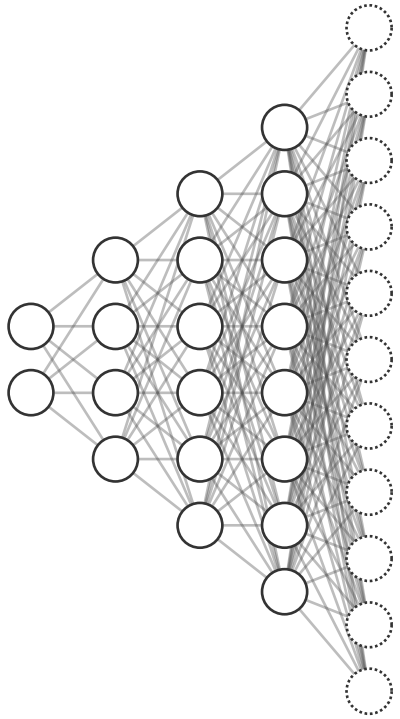


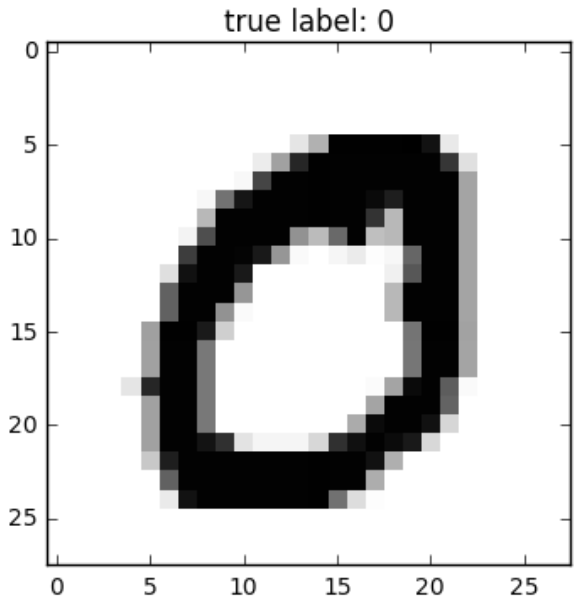




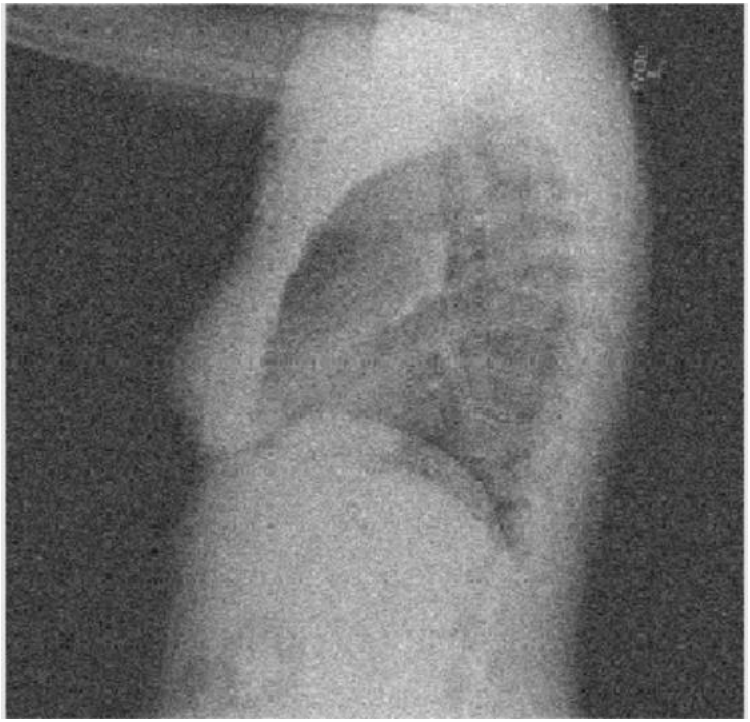
Auto encoder

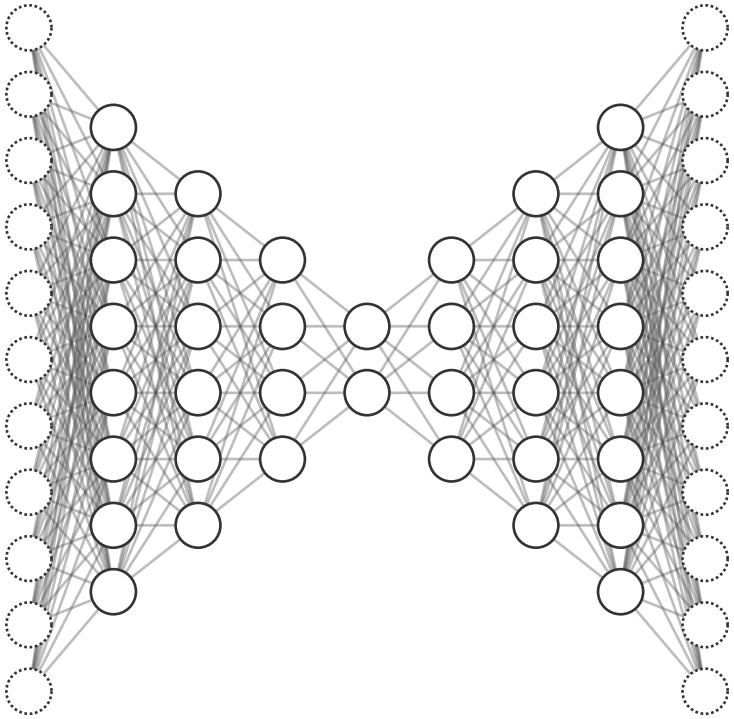


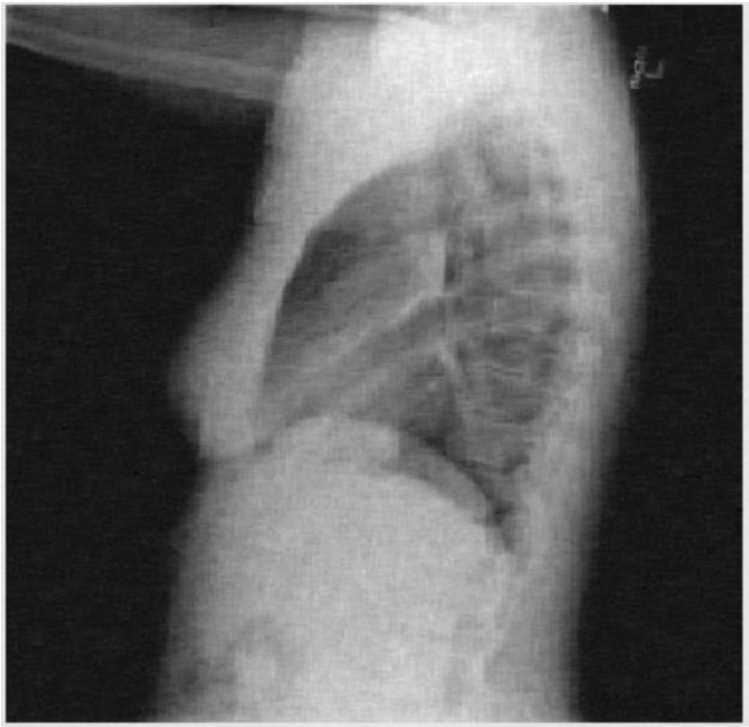




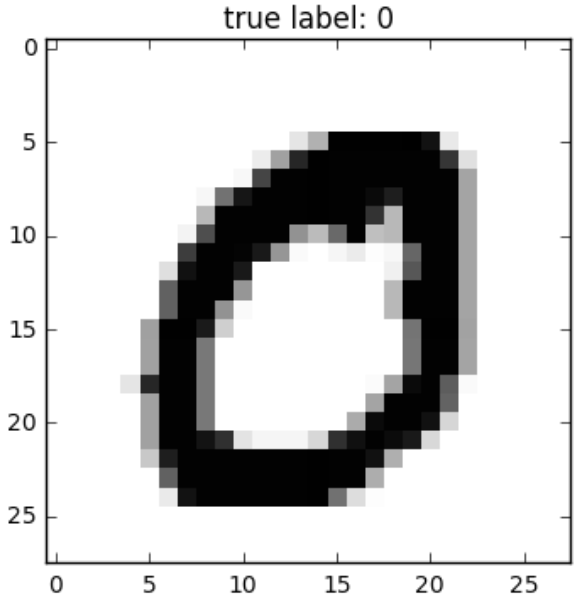
Denoising auto encoder







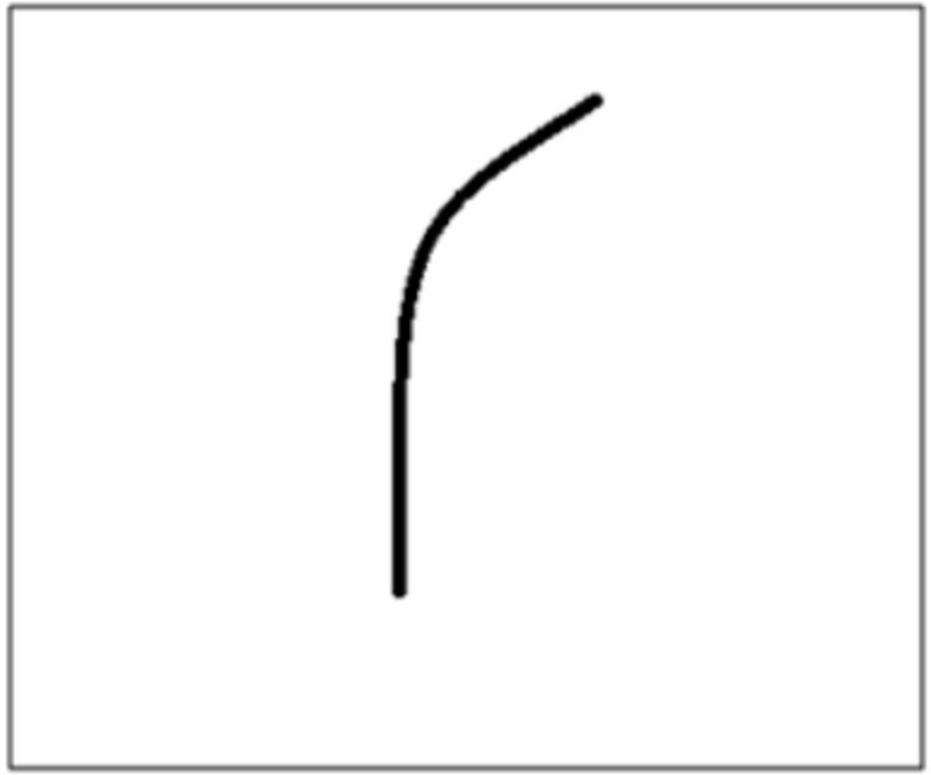
Convolutional neural network (CNN)



Convolutional neural network (CNN)

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

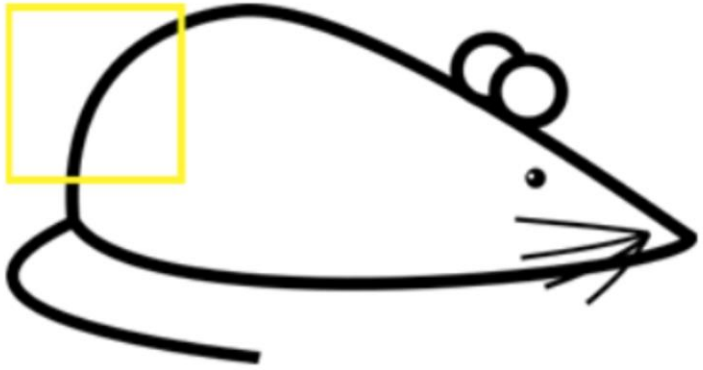


Visualization of a curve detector filter

Source: https://setosa.io/ev/image-kernels/

Convolutional neural network (CNN)

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Source: https://setosa.io/ev/image-kernels/

Convolutional neural network (CNN)

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

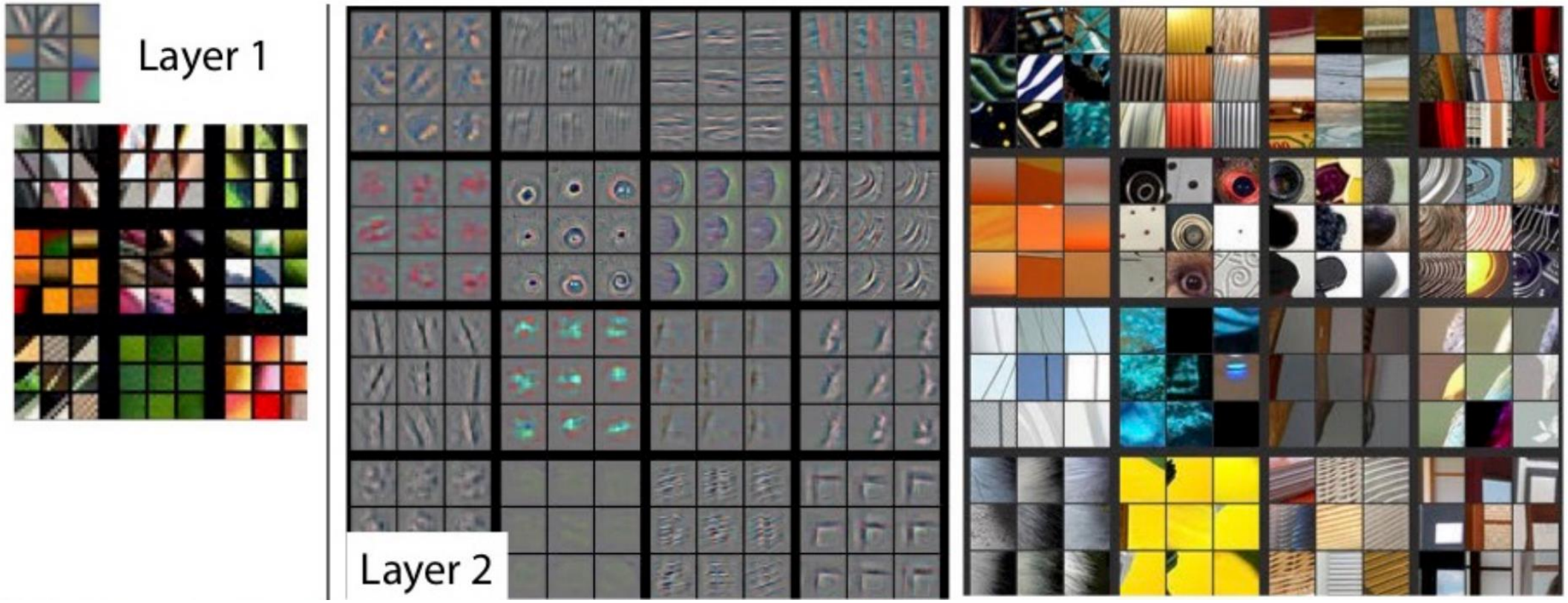
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

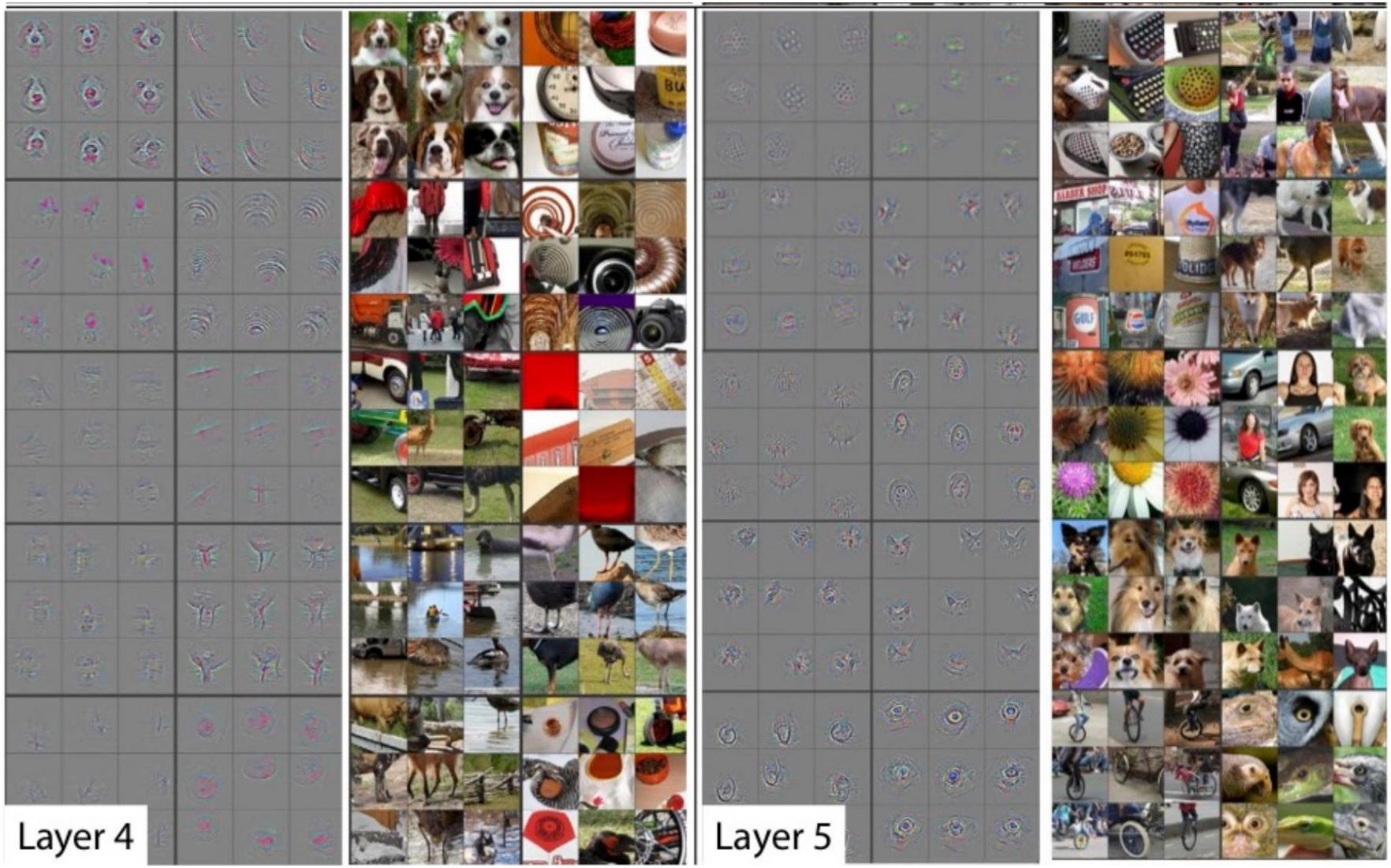
Source: https://setosa.io/ev/image-kernels/

Convolutional neural network (CNN)



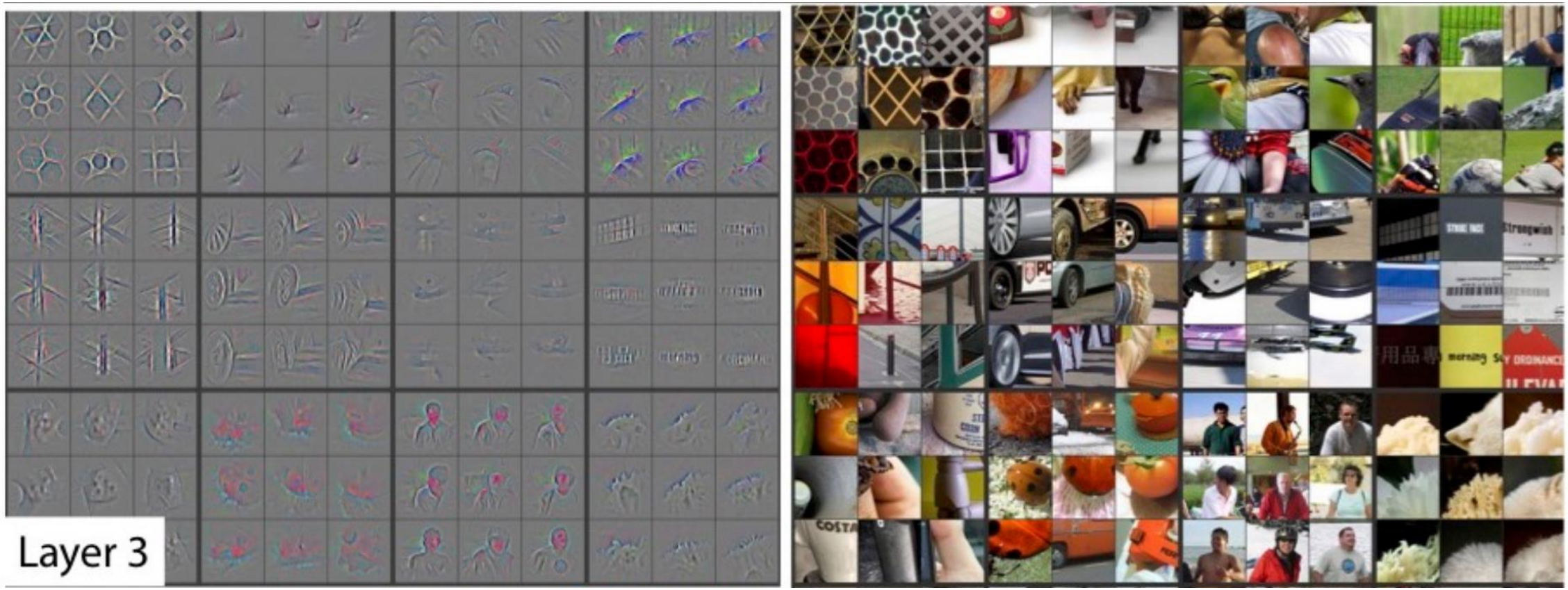
Source: https://towardsdatascience.com

Convolutional neural network (CNN)



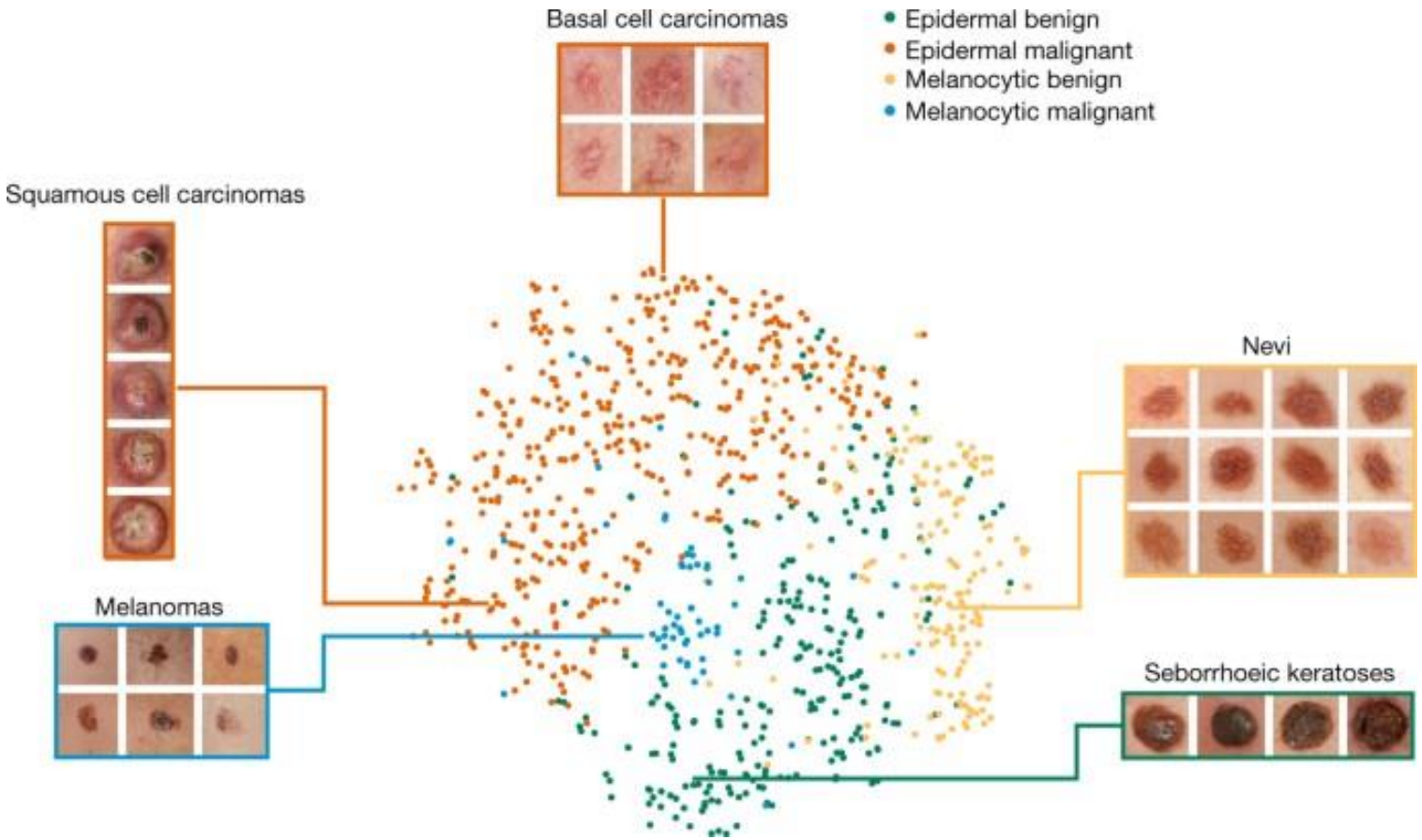
Source: https://towardsdatascience.com

Convolutional neural network (CNN)



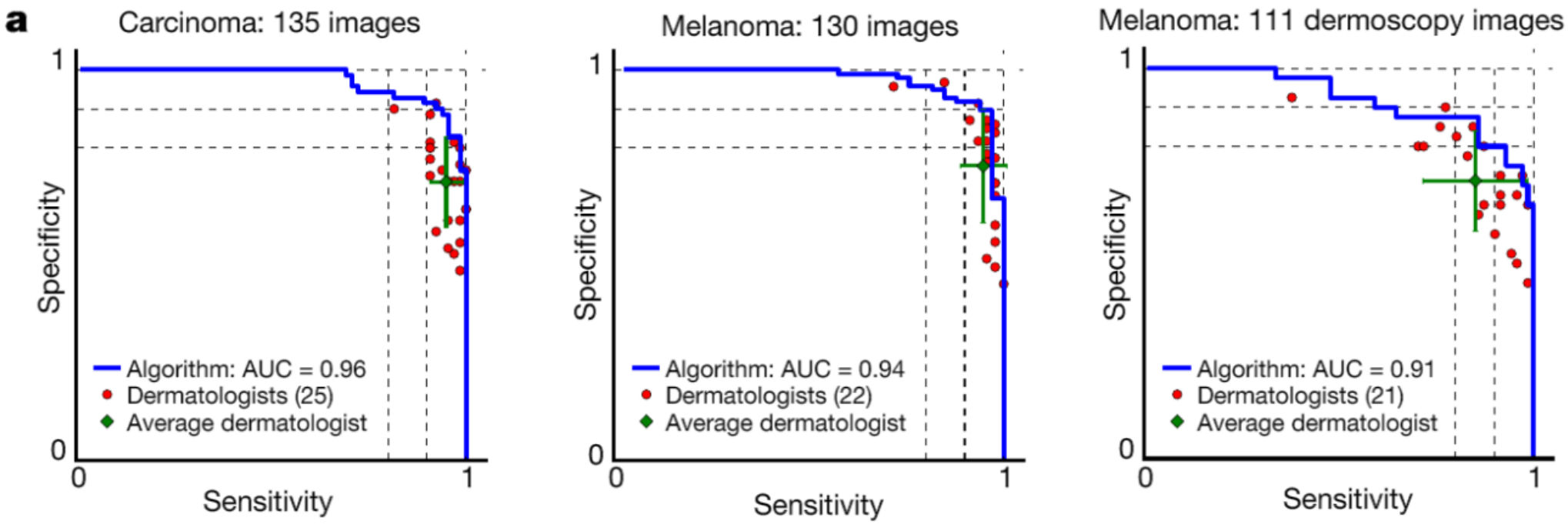
Source: https://towardsdatascience.com

Convolutional neural network (CNN)



Thrun *et al*., Dermatologist-level classification of skin cancer with deep neural networks, Nature (2017)

Convolutional neural network (CNN)



keratinocyte carcinomas: 65
benign seborrheic keratoses: 75

malignant melanomas: 33
benign nevi: 97

malignant melanomas: 71
benign nevi: 40

Thrun *et al.*, Dermatologist-level classification of skin cancer with deep neural networks, Nature (2017)