Artificial Neural Networks

Esra Suel

CASA0006: Data Science for Spatial Systems

Slides adapted from Ender Konukoglu at ETH Zurich & Huanfa Chen at UCL

Outline

- Basic concepts and some math
- Simple models: linear regression and logistic regression
- Perceptron model and multilayer extension: Artificial Neural Networks (ANNs)

We will focus on the supervised learning

There is a **task**, e.g. house price prediction, segmentation, detection, recognition, …

Examples have both **features** (predictors, images) and **labels** (prices, classes) related to the task

At inference time you only have the features

Example task: segmentation

- Appearance variation across classes
- Ignore variance within instances of the same class

Training examples are extremely important!

Examples to learn from

Test image Prediction

Basic concepts and some math

Basic notation

$$\mathbf{x} = \{x_1, x_2, \ldots, x_d\}$$

Features

- Observed information
- Numerical or categorical

- # bedrooms, location, …

- Hand-crafted explicit features
- Multispectral images

$$\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$$

Labels

- Unobserved information at prediction
- Numerical (regression) or categorical (classification)

- House prices

- Semantic segmentation labels – pixel-wise
- Object categories – image-wise

Example: Regression

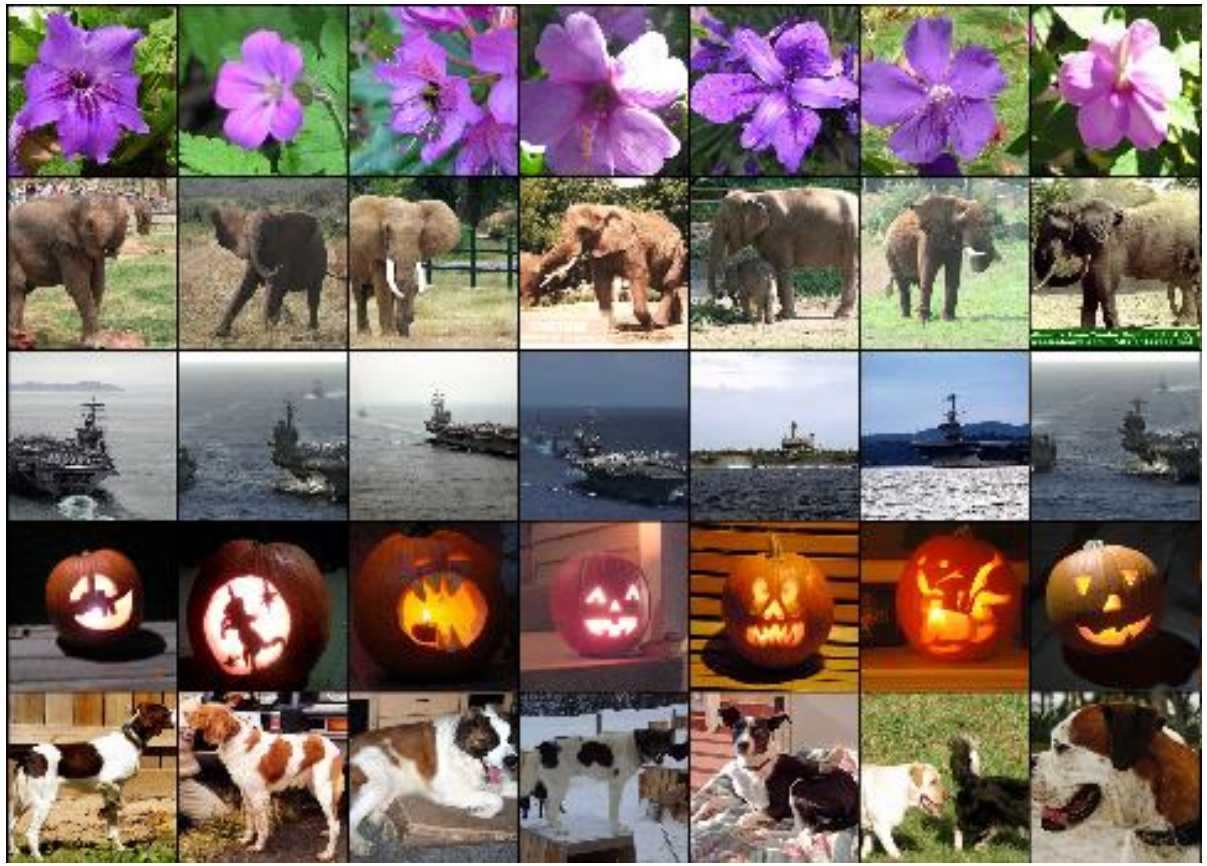
Features:

- # bedrooms
- Size
- Location
- Postcode
- …

Labels: house price

size of house (square feet)	# of bedrooms	price (1000$)
523	1	115
645	1	0.001
708	unknown	210
1034	3	unknown
unknown	4	355
2545	unknown	440

Example: Classification



Flower

Elephant

Ship

Carved Pumpkin

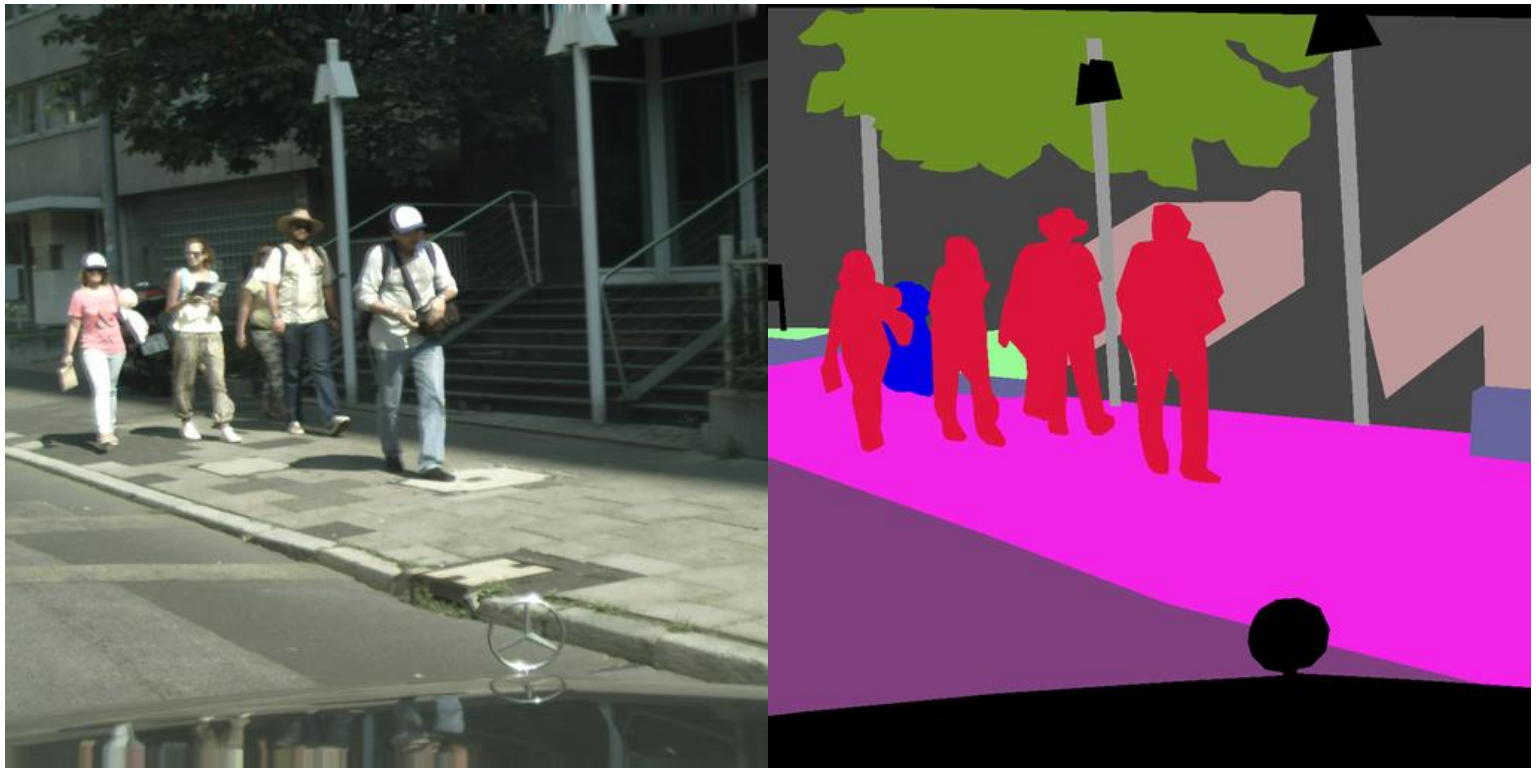
Dog

Features: images, intensity values at each pixel

Labels: object names

[Figure from Krizhevsky, Sutskever and Hinton 2012 – Predictions on ImageNet with CNNs]

Example: Segmentation



Features: Images
intensities at all pixels

Labels: Segmentation maps
object-categories at all pixels

Example: Sentiment prediction

- Sentiment Analysis assigns a sentiment score to a piece of text

I love data science, and our teachers are awesome	+4 (Strongly positive)
Beer is disgusting, why do people even like it?	-1 (Weakly negative)
It's so great that my train is late every single day	+1 (Weakly positive)

- Input: a piece of text
- Output: a piece of text with a sentiment score assigned each line

Basic concepts - mapping

$$\mathbf{x} = [x_1, x_2, \ldots, x_d] \qquad \mathbf{y} = [y_1, y_2, \ldots, y_m]$$

- Learn a mapping between features and labels
- We will focus on parametric mappings with parameters: θ

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Basic concepts - learning

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

- Determine the ``best'' parameters using the examples
- Labeled examples used for learning are called **“Training set”**
- The examples form a paired dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$
- “Best” parameters are the ones that minimize a cost defined between predictions and “ground-truth” labels

Basic concepts – best parameters

- Cost between ground truth label and prediction

$$\mathcal{L}(y, f(x;\theta))$$

- I can compute this cost for every training pair and sum

$$L(\theta) = \mathcal{L}(y_1, f(x_1;\theta)) + \mathcal{L}(y_2, f(x_2;\theta)) + \mathcal{L}(y_3, f(x_3;\theta)) + \dots$$

- I determine the parameter that achieves the minimum loss

$$\theta^* = \arg_\theta \min L(\theta)$$

- Another notation for this operation is

$$\theta^* = \arg_\theta \min \sum_{n=1}^{N} \mathcal{L}\left(\mathbf{y}_n, f(\mathbf{x}_n;\theta)\right)$$

Basic concepts – cost function

$$\mathcal{L}(y, f(x;\theta)) \qquad \theta^* = \arg_\theta \min \sum_{n=1}^{N} \mathcal{L}\left(\mathbf{y}_n, f(\mathbf{x}_n;\theta)\right)$$

- We model the definition of “best” using the cost function
- Task dependent and ideally defined for your final goal
- Regression: e.g., squared difference

$$\mathcal{L}(y_n, f(x_n;\theta) = (y_n - f(x_n;\theta))^2$$

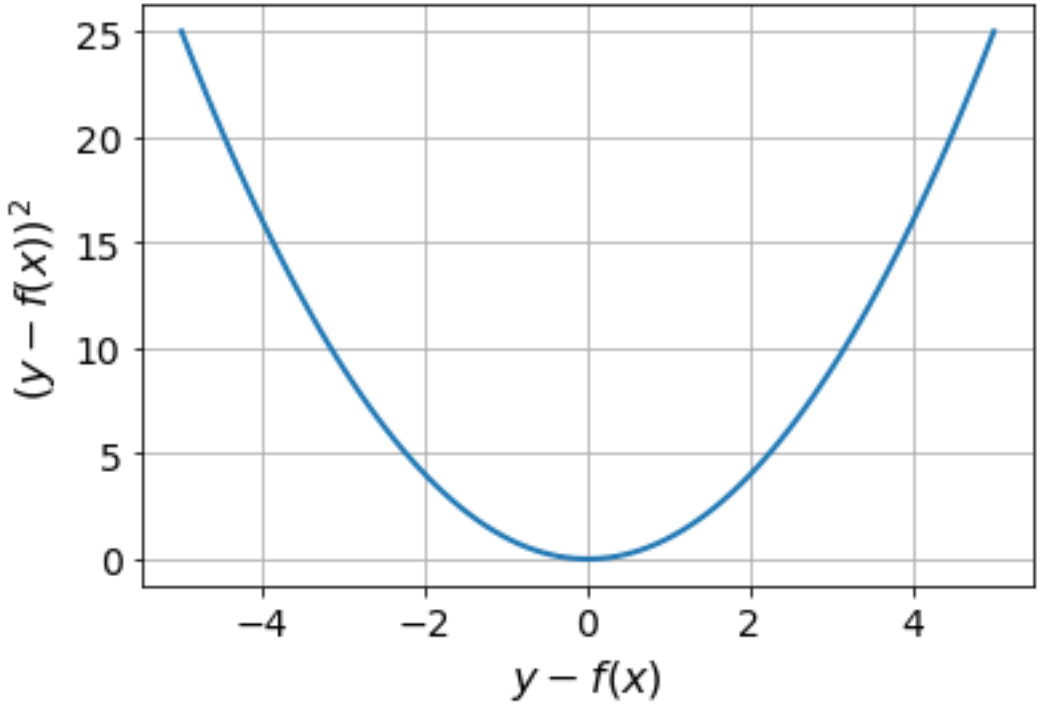
- You can also use different loss functions, e.g.

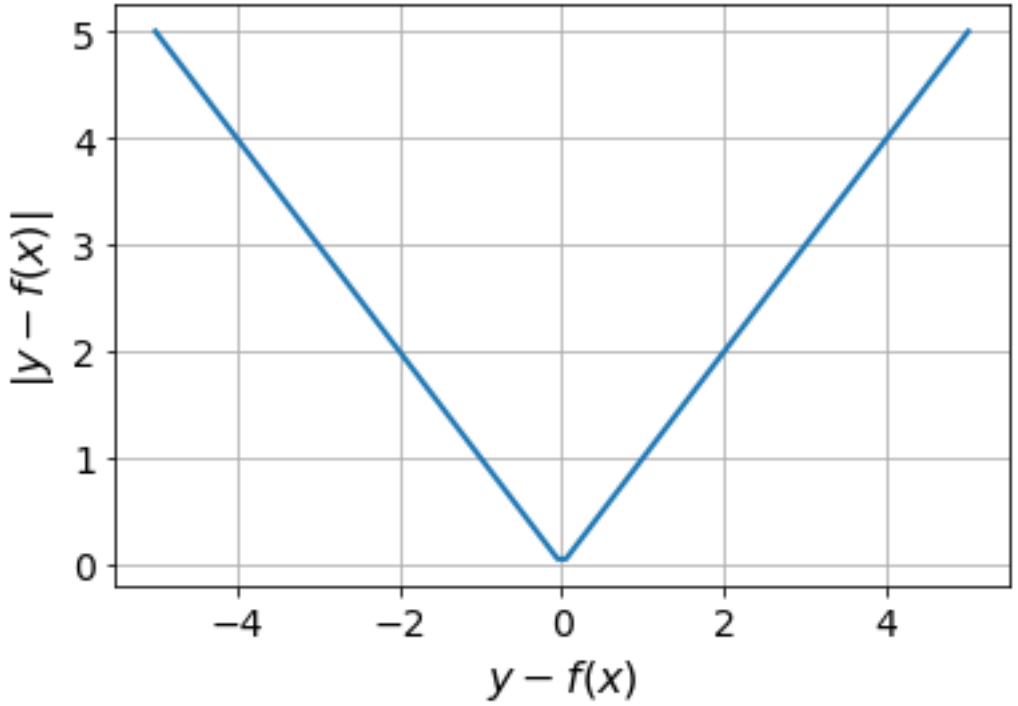
$$\mathcal{L}(y_n, f(x_n;\theta) = |y_n - f(x_n;\theta)|$$

Basic concepts – regression costs

$$\mathcal{L}(y_n, f(x_n;\theta) = (y_n - f(x_n;\theta))^2$$

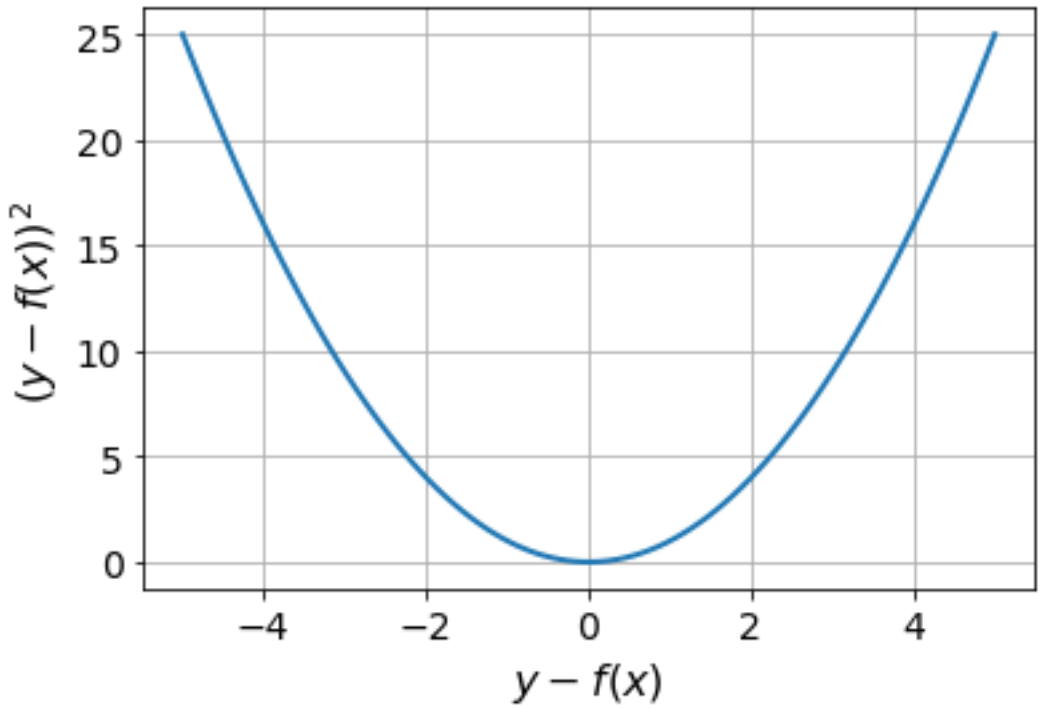
$$\mathcal{L}(y_n, f(x_n;\theta) = |y_n - f(x_n;\theta)|$$





Basic concepts – regression costs

$$\mathcal{L}(y_n, f(x_n;\theta) = (y_n - f(x_n;\theta))^2$$



Ground truth label y_n	Predicted value $f_1(x_n;\theta)$	Cost $L(y_n, f(x_n;\theta)$
3	2.7	0.3^2=0.09
3	1.7	1.3^2=1.69
-3	-1.7	1.3^2=1.69
3	4.3	1.3^2=1.69
4	1.7	2.3^2=5.29

Total cost is the sum of all individual costs

Basic concepts – cost function

$$\mathcal{L}(y, f(x;\theta)) \qquad \theta^* = \arg_\theta \min \sum_{n=1}^{N} \mathcal{L}\left(\mathbf{y}_n, f(\mathbf{x}_n;\theta)\right)$$

Cost function determines the definition of “best”

Task dependent and ideally defined for your final goal

Classification: e.g., cross-entropy for classification

$$y_n \in \mathcal{C}, \mathcal{C}\text{: set of possible classes}$$

$$f(\mathbf{x};\theta) = \left[f_{c_1}(\mathbf{x};\theta), f_{c_2}(\mathbf{x};\theta), \ldots\right], \sum_{k\in\mathcal{C}} f_k(\mathbf{x};\theta) = 1$$

where predictions are considered as class probabilities

Basic concepts – cost function

$$\mathcal{L}(y, f(x;\theta)) \qquad \theta^* = \arg_\theta \min \sum_{n=1}^{N} \mathcal{L}\left(\mathbf{y}_n, f(\mathbf{x}_n;\theta)\right)$$

Cost function determines the definition of “best”

Task dependent and ideally defined for your final goal

Classification: e.g., cross-entropy for classification

$$y_n \in \mathcal{C}, \mathcal{C}\text{: set of possible classes}$$

$$f(\mathbf{x};\theta) = \left[f_{c_1}(\mathbf{x};\theta), f_{c_2}(\mathbf{x};\theta), \ldots\right], \sum_{k\in\mathcal{C}} f_k(\mathbf{x};\theta) = 1$$

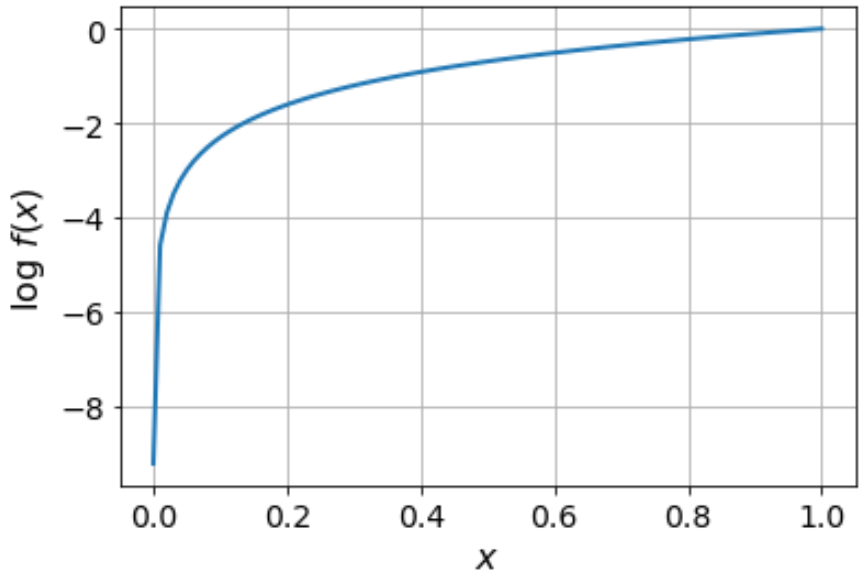
$$\mathcal{L}(y_n, f(\mathbf{x}_n;\theta)) = -\sum_{k\in\mathcal{C}} \log(f_k(\mathbf{x}_n;\theta))\mathbf{1}(y_n = k)$$

Basic concepts – cross entropy

$$\mathcal{L}(y_n, f(\mathbf{x}_n; \theta)) = -\sum_{k \in \mathcal{C}} \log(f_k(\mathbf{x}_n; \theta)) \mathbf{1}(y_n = k)$$

Binary example

$$\mathcal{L}(y_n, f(x_n; \theta)) = -\log(f_0(x_n; \theta)) \mathbf{1}(y_n = 0) - \log(f_1(x_n; \theta) \mathbf{1}(y_n = 1)$$

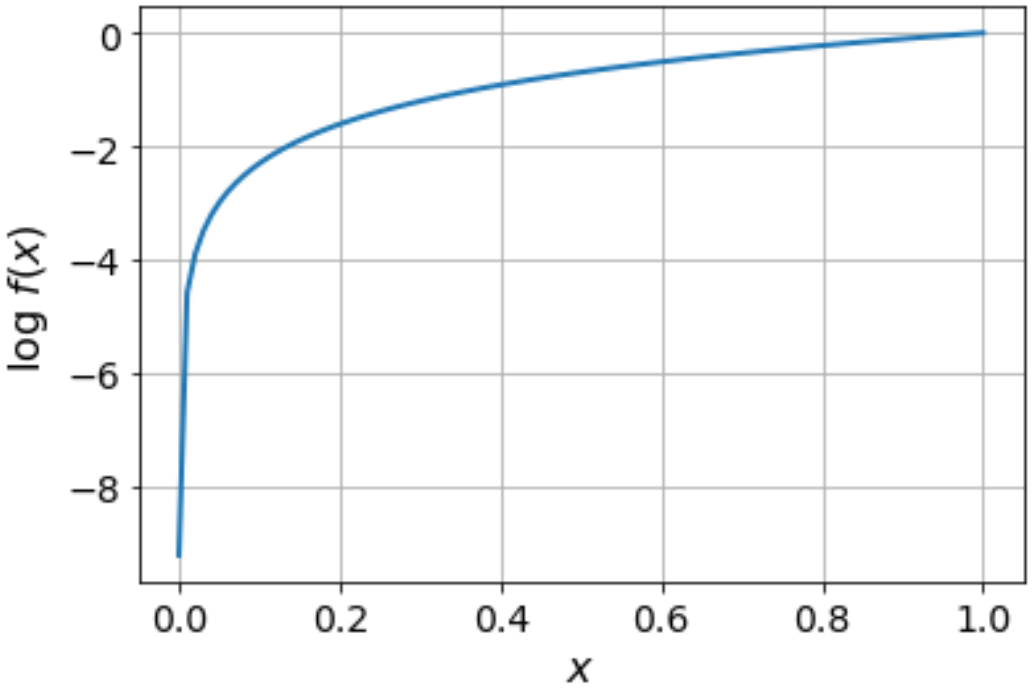


$$f_0(x_n; \theta)) + f_1(x_n; \theta) = 1$$

Basic concepts – cross entropy

$$\mathcal{L}(y_n, f(x_n;\theta)) = -\log(f_0(x_n;\theta))\mathbf{1}(y_n = 0) - \log(f_1(x_n;\theta)\mathbf{1}(y_n = 1)$$

$$f_0(x_n;\theta)) + f_1(x_n;\theta) = 1$$



Ground truth label y_n	Predicted probability $f_1(x_n;\theta)$	Cost $L(y_n, f(x_n;\theta)$
1	0.7	-log(0.7)
1	0.3	-log(0.3)
0	0.3	-log(0.7)
1	1	-log(1.0)
0	0.8	-log(0.2)

Total cost is the sum of all individual costs

Basic concepts - prediction

The model and the best parameters are determined

Prediction for a new sample also depends on your task

For regression:

$$\hat{\mathbf{y}} = f(\mathbf{x}; \theta^*)$$

Basic concepts - prediction

The model and the best parameters are determined

Prediction for a new sample also depends on your task

For classification:

$$\hat{\mathbf{y}} = \arg_k \max f_k(\mathbf{x}; \theta^*)$$

Choose the class with maximum probability

Basic concepts – prediction error

Prediction will have errors

For regression, the most used: Mean Squared Error (MSE), Mean Absolute Error (MAE)

$$MSE = \frac{1}{T}\sum_{t}^{T} \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|_2^2, \; MAE = \frac{1}{T}\sum_{t}^{T} |\hat{\mathbf{y}}_t - \mathbf{y}_t|$$

For classification, the most used: Classification error (Cerr)

$$Cerr = \frac{1}{T}\sum_{t}^{T} \mathbf{1}(\hat{\mathbf{y}}_t \neq \mathbf{y}_t)$$

Basic concepts - Three sets:
Training, validation and test

1. Determining the best model parameters
2. Determining the hyper-parameters
3. Estimating generalization accuracy

A. Training set

B. Validation set

C. Test set

Simple models

Simplest ML model: Linear Regression

Linear model is the main building block

Assumes a linear relationship between features and labels

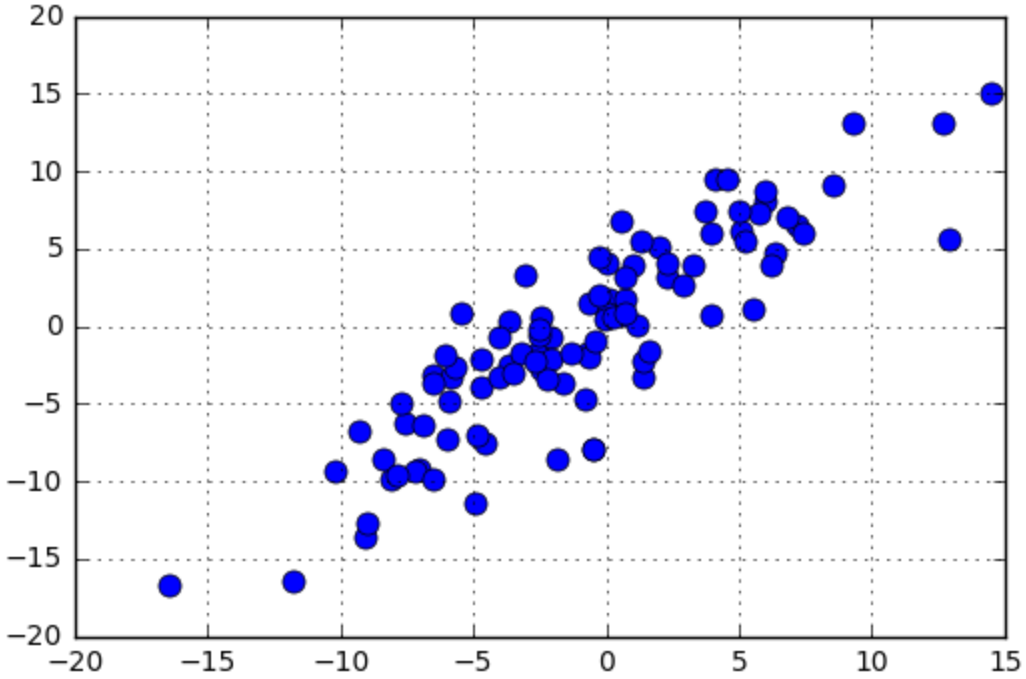
For simplicity, let us assume one-dimensional label: $\mathbf{y} = y$

And one-dimensional feature: $\mathbf{x} = x$

$$y = ax + b$$

Parameters of the linear model: $\theta = \{a, b\}$

Linear regression model



Dataset

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$$

Model

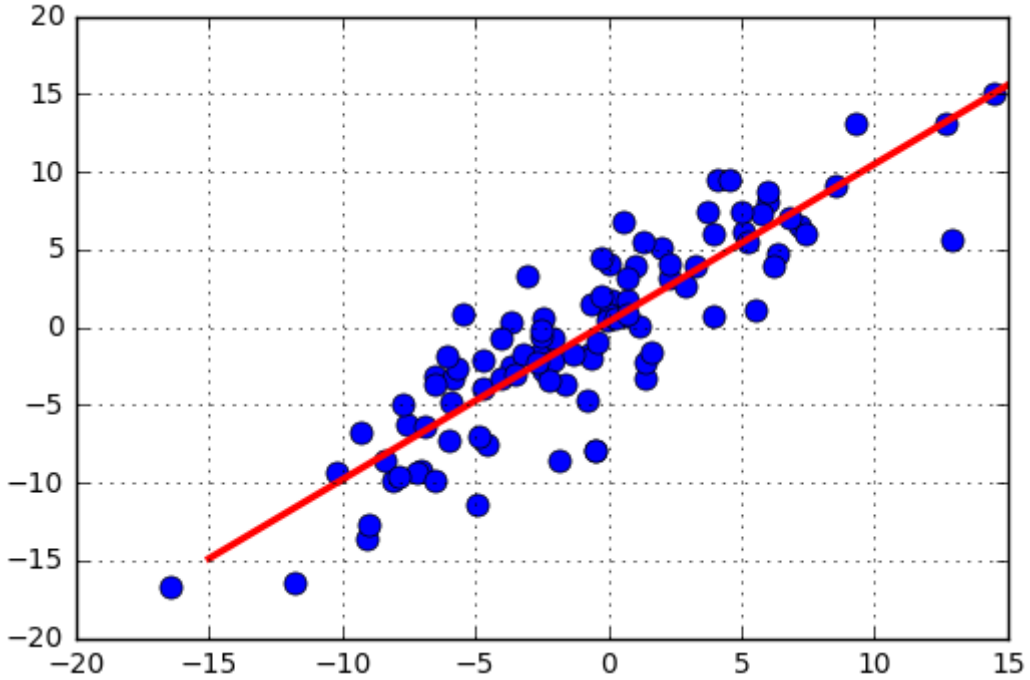
$$y = f(x;\theta) = f(x;a,b) = ax + b$$

Learning

$$a^*, b^* = \arg_{a,b} \min \sum_{n=1}^{N} \|y_n - ax_n - b\|_2^2$$

a^*, b^* : Optimal model parameters for this dataset

Linear regression model



Dataset

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$$

Model

$$y = f(x;\theta) = f(x;a,b) = ax + b$$

Learning

$$a^*, b^* = \arg_{a,b} \min \sum_{n=1}^{N} \|y_n - ax_n - b\|_2^2$$

a^*, b^* : Optimal model parameters for this dataset

Generalization: multiple features

One feature was easy $y = ax + b$

Generalizing to multiple features is also easy

For simplicity, let us assume one dimensional label: $\mathbf{y} = y$

$$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d = \theta^T \mathbf{x}$$

Parameters of the linear model: θ

Linear model is for continuous labels

Linear model for Classification: Logistic Regression Models

Linear model is the main building block

Assumes a linear relationship between features and labels

For simplicity, let us assume one dimensional label: $\mathbf{y} = y$

$$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d = \theta^T \mathbf{x}$$

Parameters of the linear model: θ

Linear regression model was for continuous labels

Logistic regression is the extension to binary labels

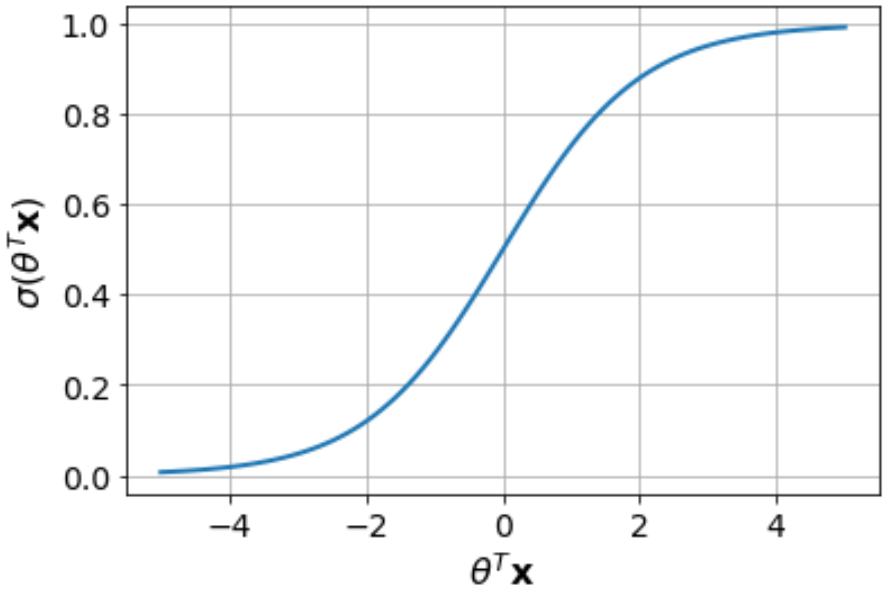
Extracting probabilities

$$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d = \theta^T \mathbf{x}$$

Continuous values that can go from -∞ to ∞

For binary classification we want probabilities

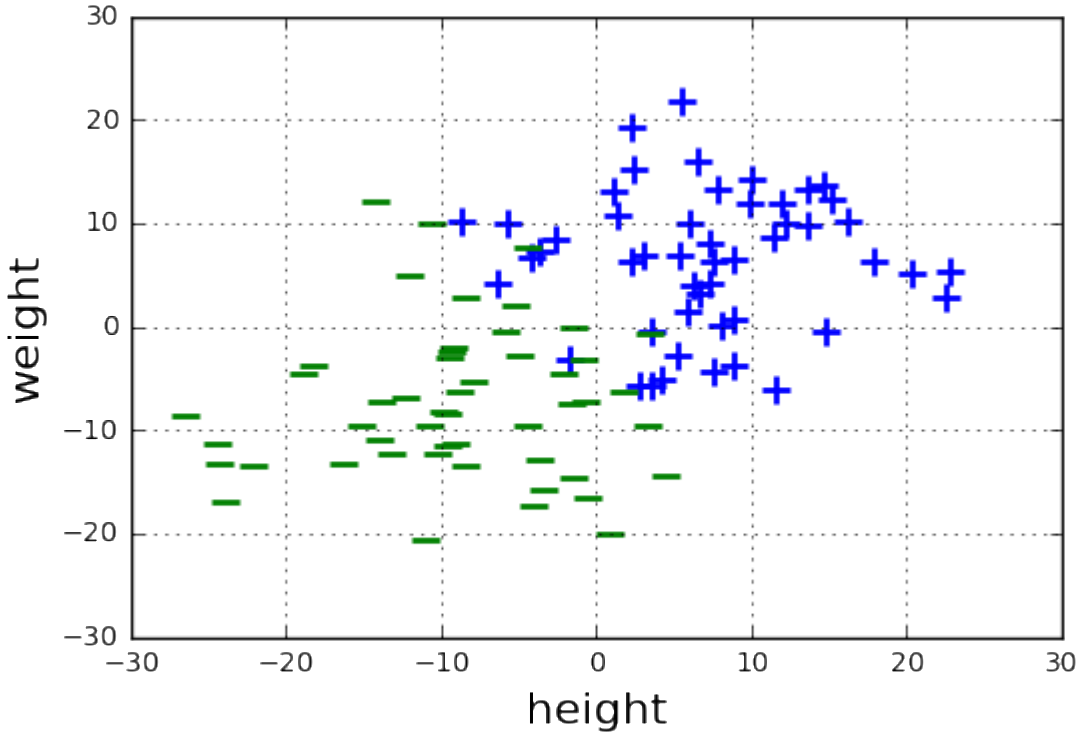
$$0 \leq f(\mathbf{x}; \theta) \leq 1$$



- Sigmoid function maps it to a probability

$$f(\mathbf{x}; \theta) = \sigma(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Logistic regression model



Dataset

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$$

Model

$$p(y = 1) = f(\mathbf{x}; \theta) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Learning

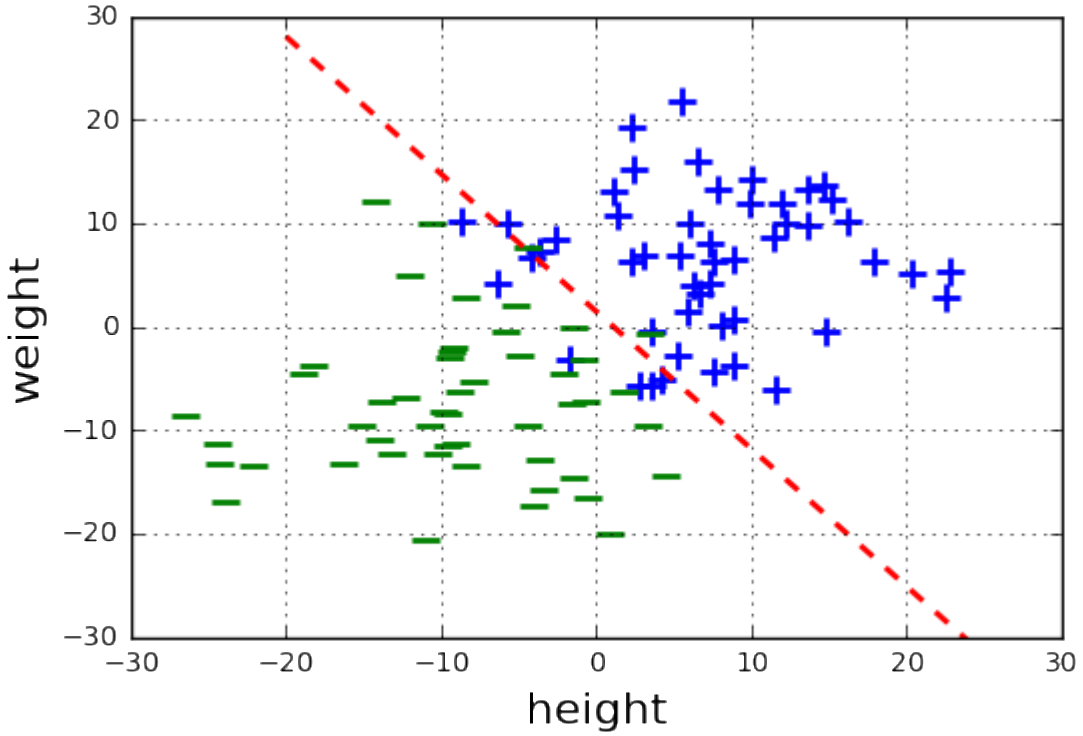
$$\theta^* = \arg_\theta \min \sum_{n=1}^{N} - \log f(\mathbf{x}_n; \theta) y_n$$

$$- \log(1 - f(\mathbf{x}; \theta))(1 - y_n)$$

Logistic regression model

Logistic regression is the building block of the perceptron model

Line denotes the points where p(y=1)=0.5



Dataset

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$$

Model

$$p(y = 1) = f(\mathbf{x}; \theta) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Learning

$$\theta^* = \arg_\theta \min \sum_{n=1}^{N} - \log f(\mathbf{x}_n; \theta) y_n$$

$$- \log(1 - f(\mathbf{x}; \theta))(1 - y_n)$$

Optimization - Learning

Whether classification or regression

$$\theta^* = \arg_\theta \min \mathcal{L}(\theta) \qquad \theta = [\theta_1, \theta_2, \ldots]$$

The main idea is to start from an initial estimate: θ^0

"Wiggle each parameter a bit" to find the direction that **minimizes the loss the most:** $\mathbf{v}^0$

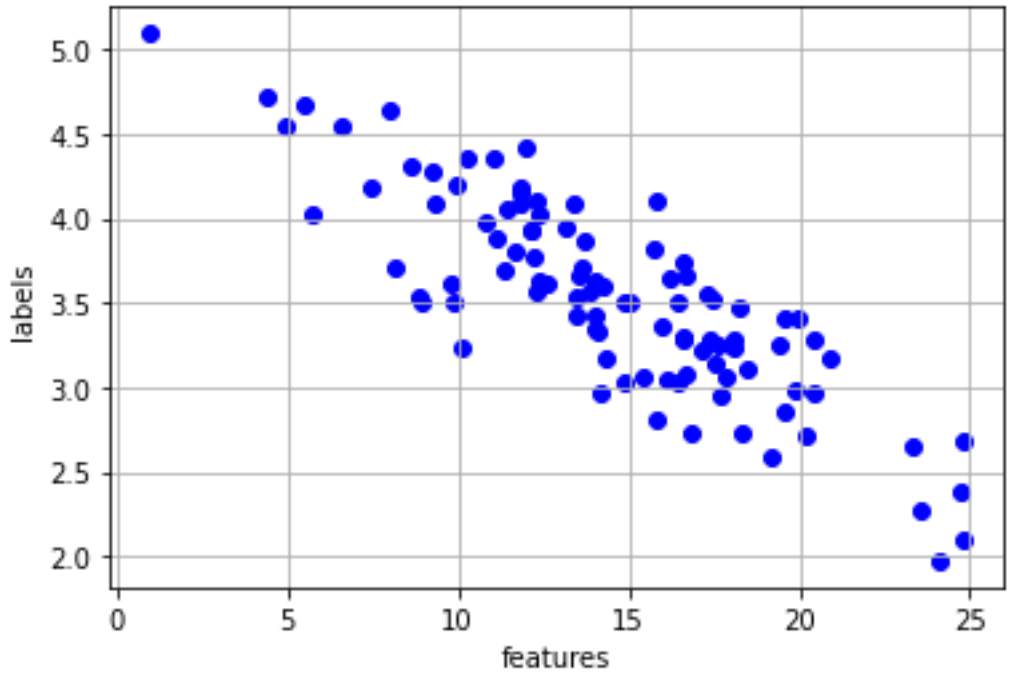
Update the parameters: $\theta^1 = \theta^0 + \mathbf{v}^0$

Continue doing this: $\theta^{t+1} = \theta^t + \mathbf{v}^t$

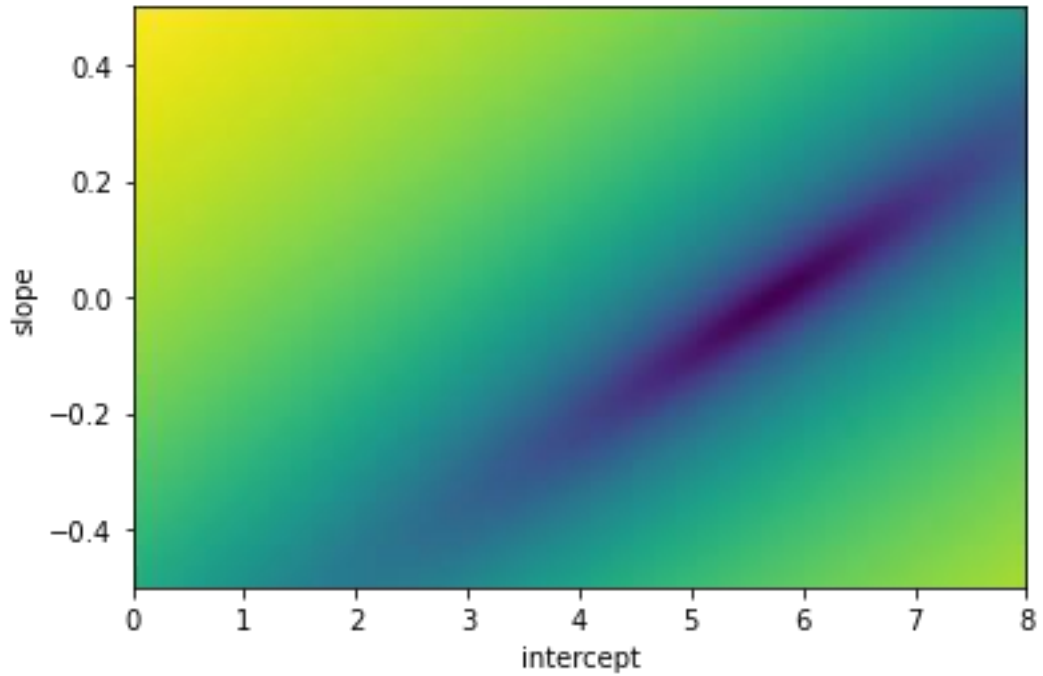
Stop when you can no longer minimize the loss

Let’s see it in action

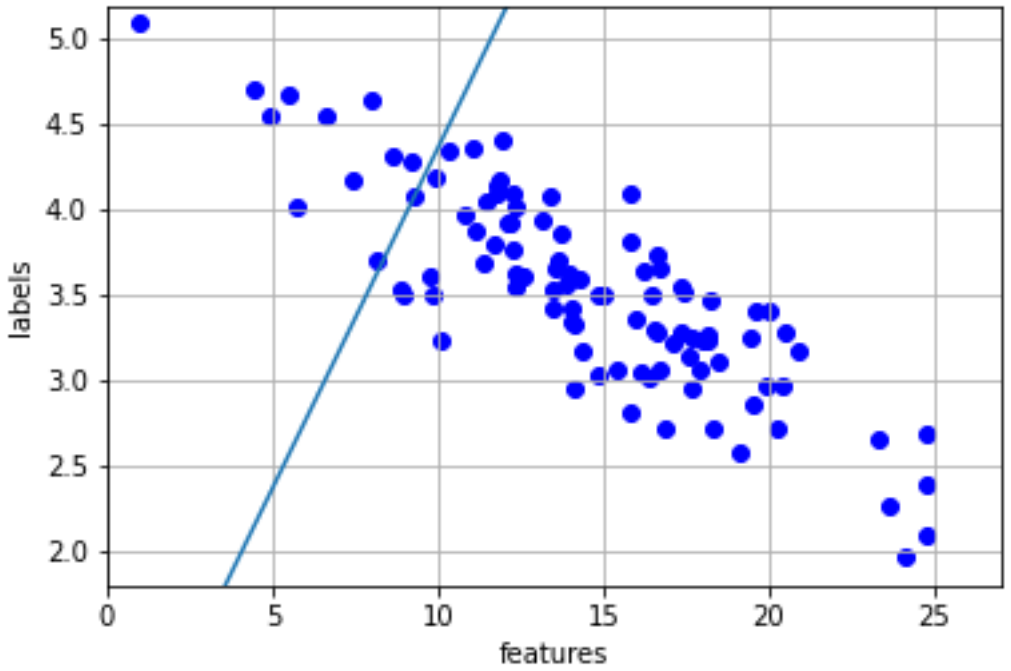
Samples

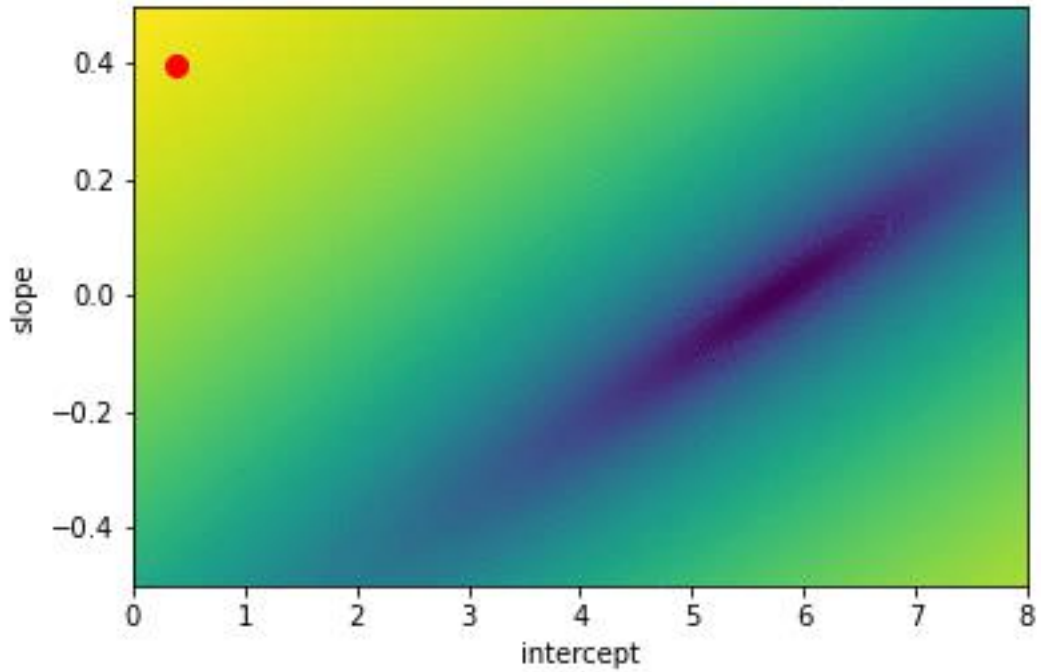


Loss function



$$\theta^{t+1} = \theta^t + \alpha \nabla \mathcal{L}(\theta) = \begin{bmatrix} \theta_0^t \\ \theta_1^t \\ \vdots \\ \theta_d^t \end{bmatrix} + \alpha \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \theta_1} \\ \frac{\partial \mathcal{L}}{\partial \theta_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \theta_d} \end{bmatrix}$$

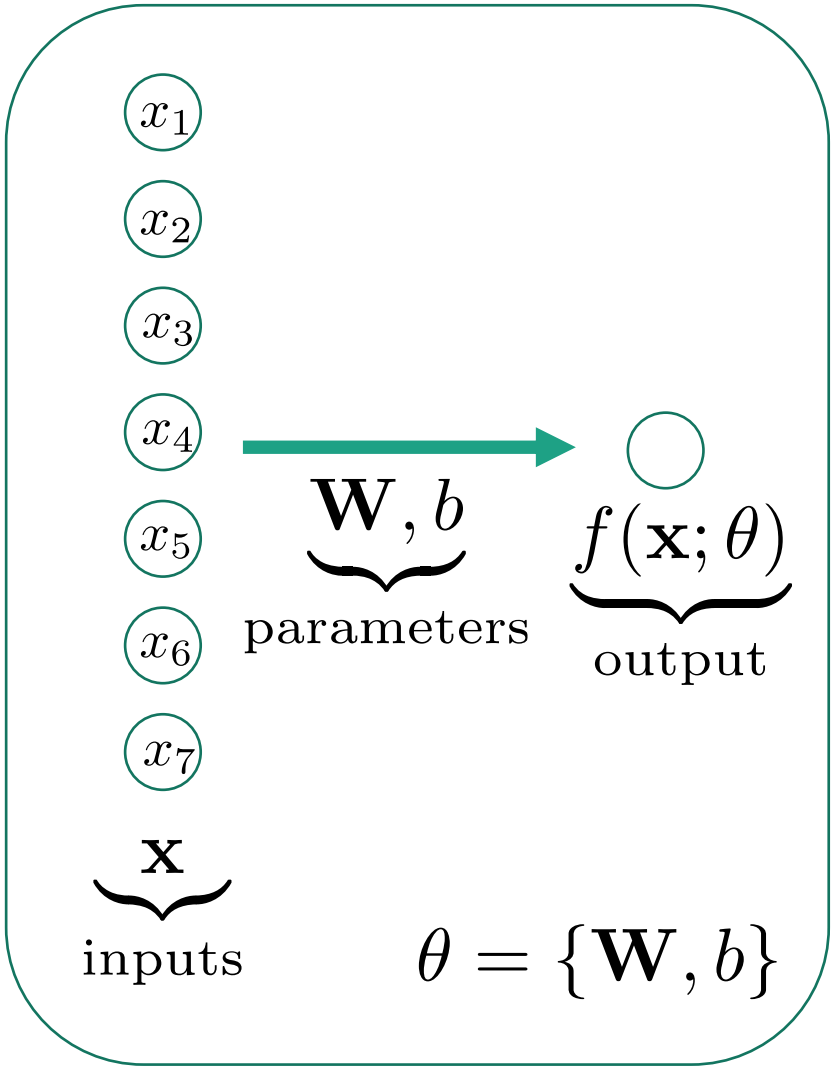




Perceptron model and multilayer extension

Artificial Neural Networks (ANNs)

Basic perceptron model



Model of binary classification

$$p(y=1) = f(\mathbf{x};\theta) = \sigma(\mathbf{W}\mathbf{x} + b)$$

Formed of two different parts

1. Activation

$$\underbrace{a}_{\text{activation}} = \underbrace{\mathbf{W}}_{\text{weights}}\mathbf{x} + \underbrace{b}_{\text{bias}}$$

2. Nonlinearity $f(\mathbf{x};\theta) = \sigma(a)$

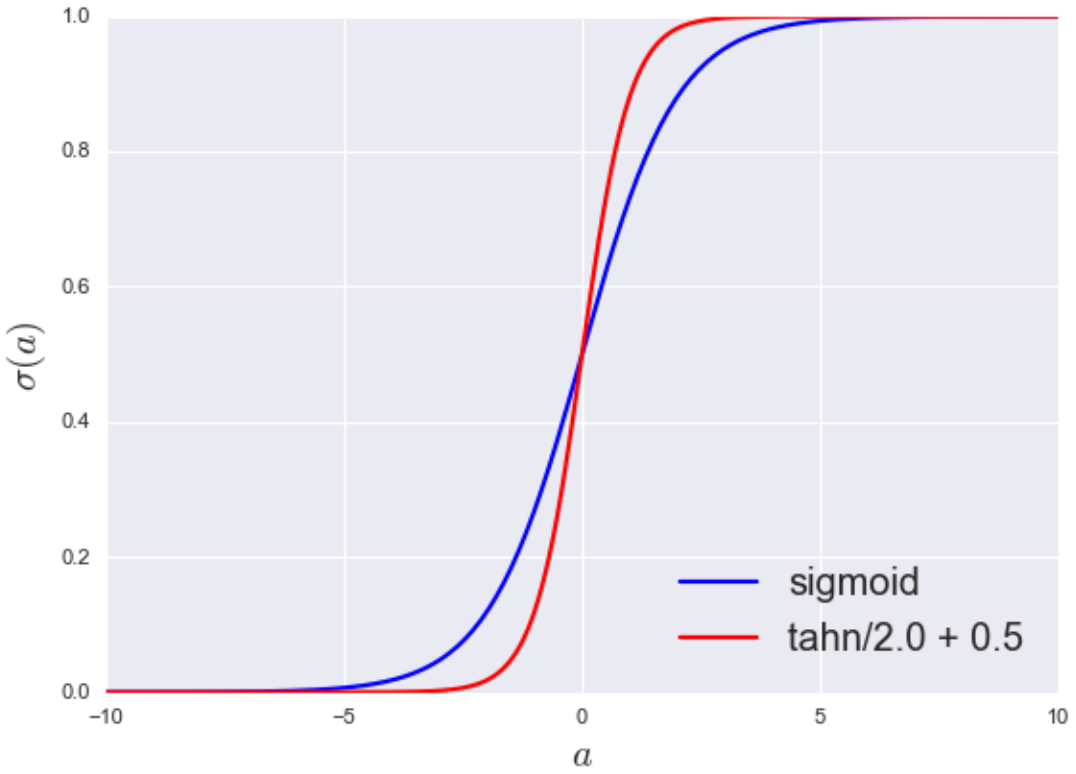
Activation

$$\underbrace{a}_{\text{activation}} = \underbrace{\mathbf{W}}_{\text{weights}}\,\mathbf{x} + \underbrace{b}_{\text{bias}} \longrightarrow a = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

- Linear transformation of the features
- d + 1 number of parameters $\mathbf{W} \in \mathbb{R}^{1 \times d}$ $b \in \mathbb{R}$

Non-linearity

Maps real line to probabilities



$$\sigma : \mathbb{R} \mapsto (0, 1)$$

Sigmoid function

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

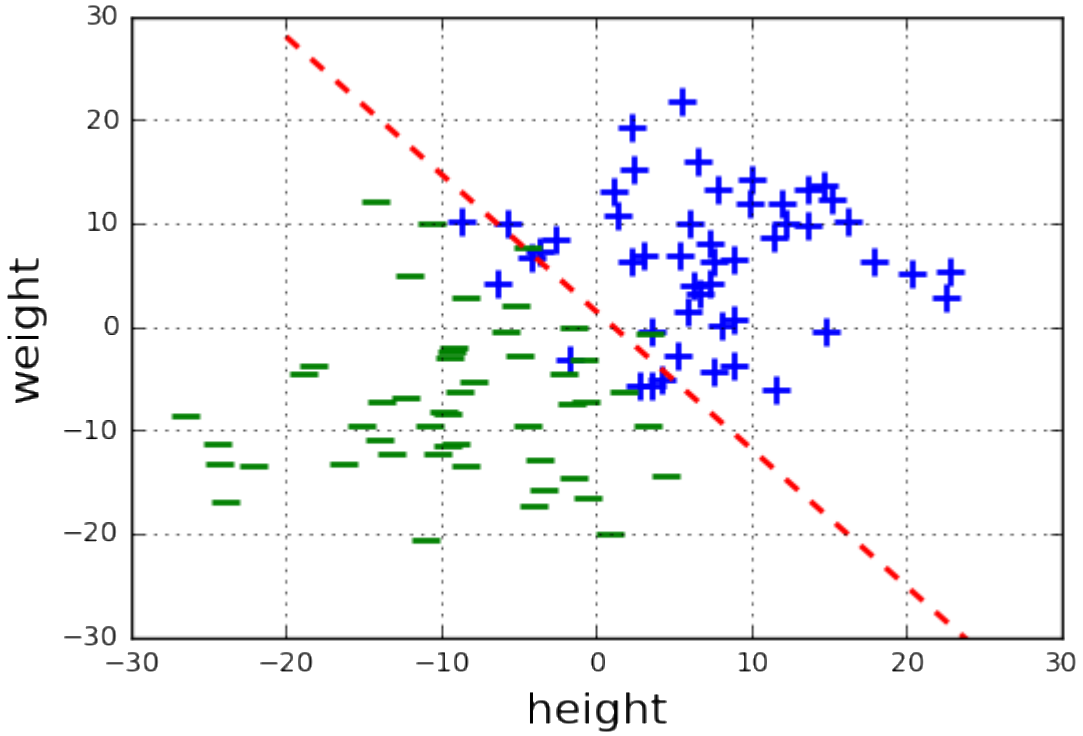
Tangent Hyperbolic

$$\sigma(a) = \tanh(a)/2.0 + 0.5$$

Remember - Logistic regression model

Logistic regression is the building block of the perceptron model

Line denotes the points where p(y=1)=0.5



Dataset

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$$

Model

$$p(y = 1) = f(\mathbf{x}; \theta) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

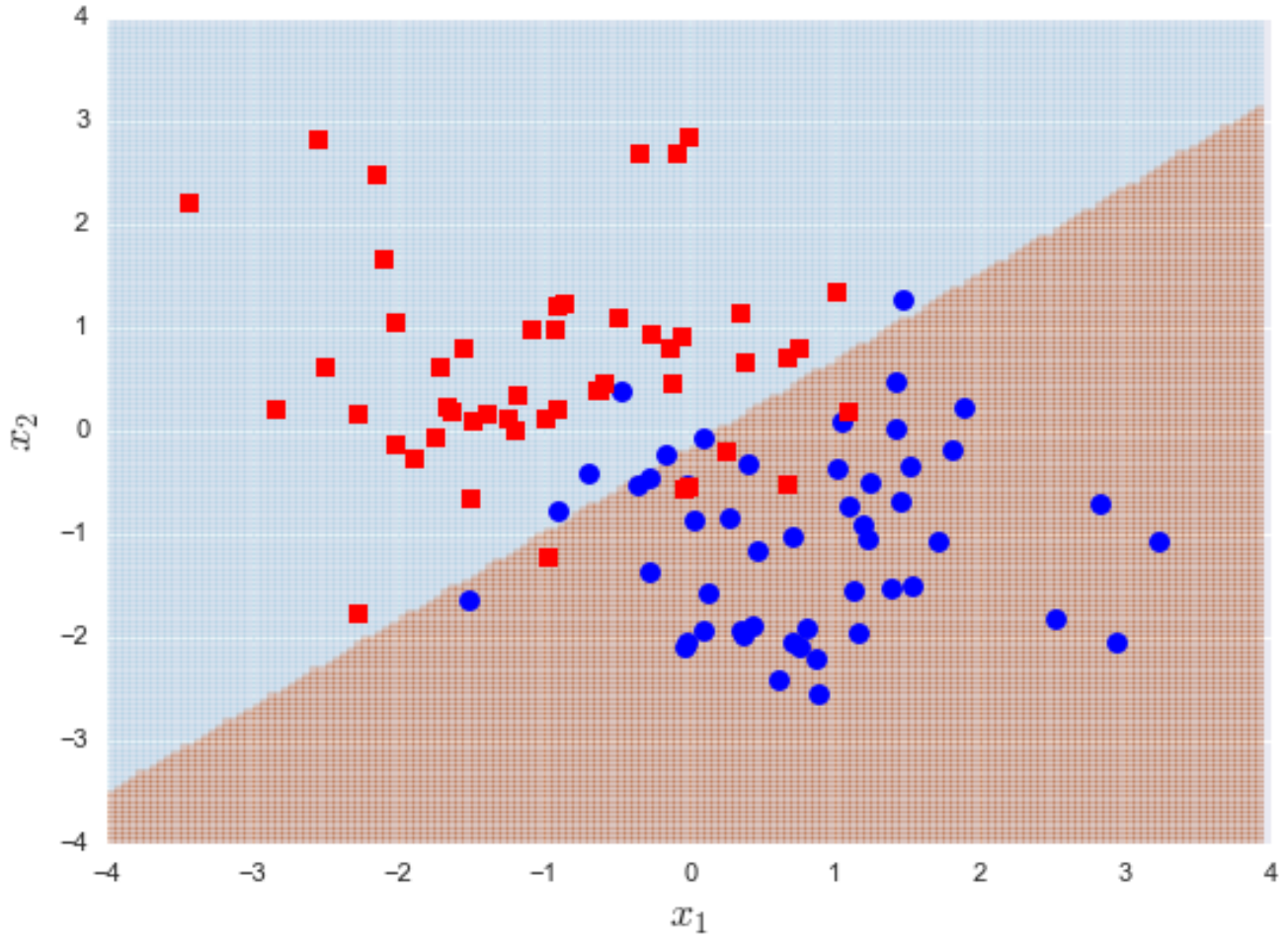
Learning

$$\theta^* = \arg_\theta \min \sum_{n=1}^{N} - \log f(\mathbf{x}_n; \theta) y_n$$

$$- \log(1 - f(\mathbf{x}; \theta))(1 - y_n)$$

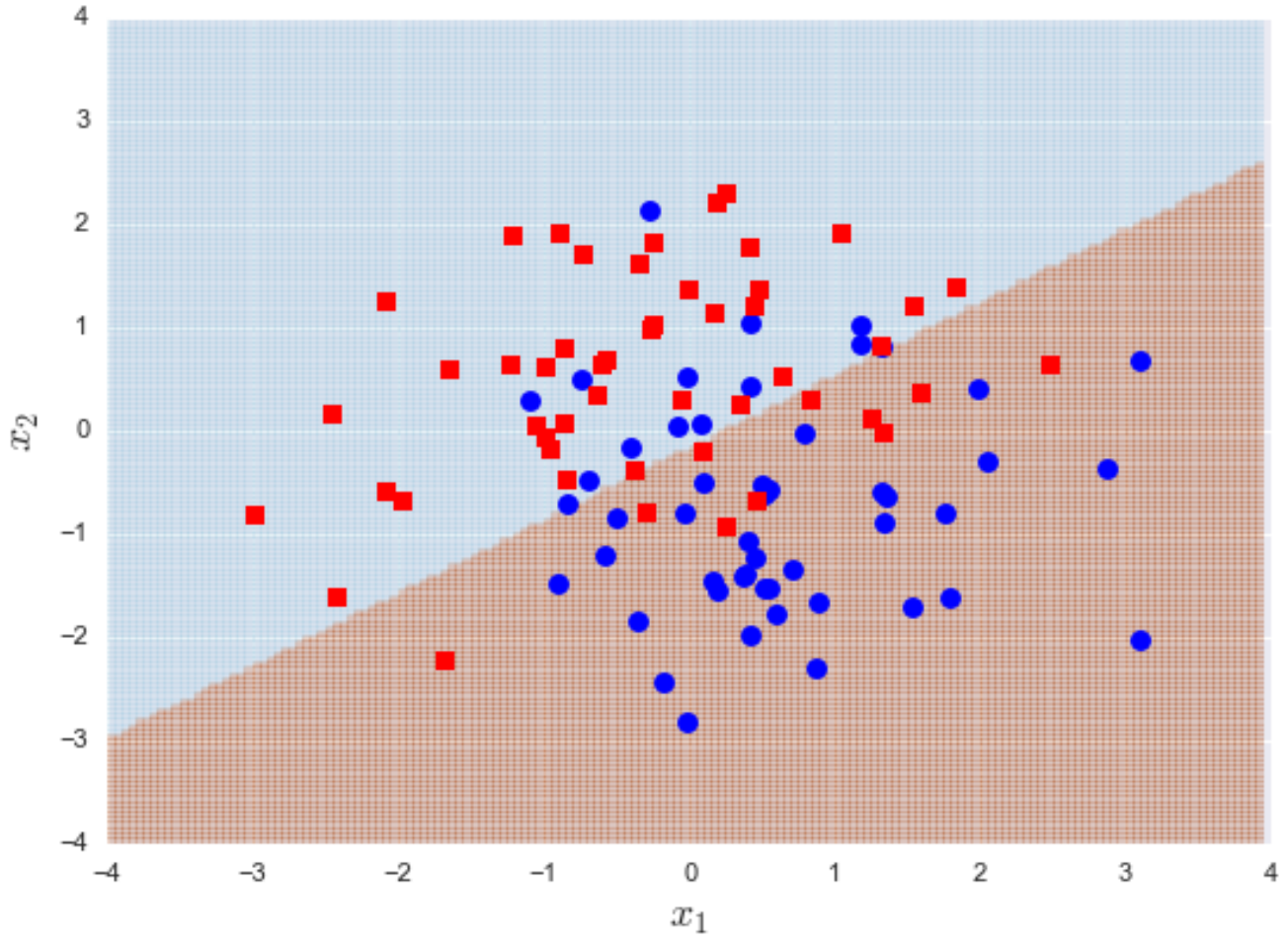
Decision boundary

$$f(\mathbf{x};\theta) = 0.5$$



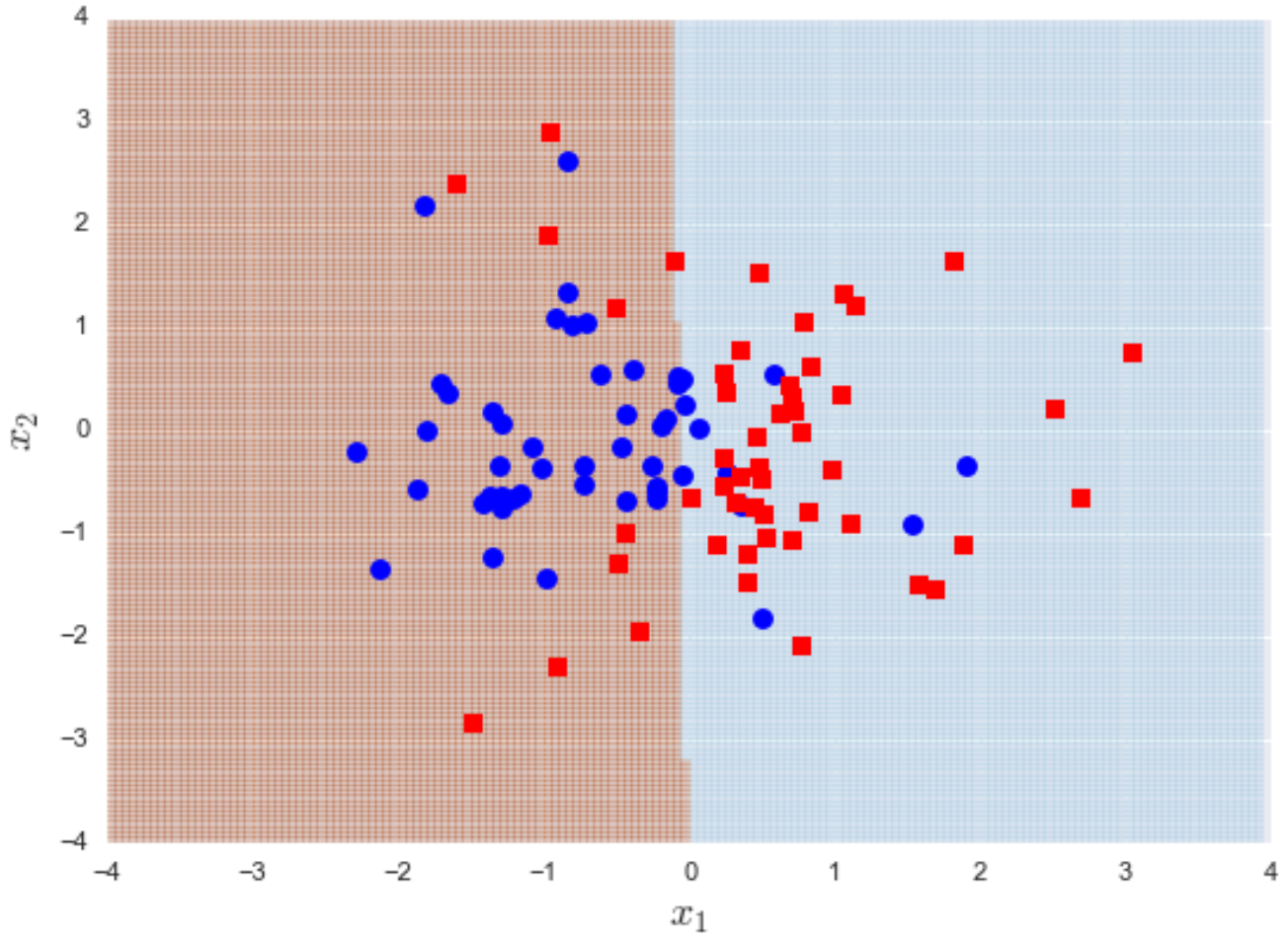
Decision boundary II

$$f(\mathbf{x};\theta) = 0.5$$



Decision boundary III

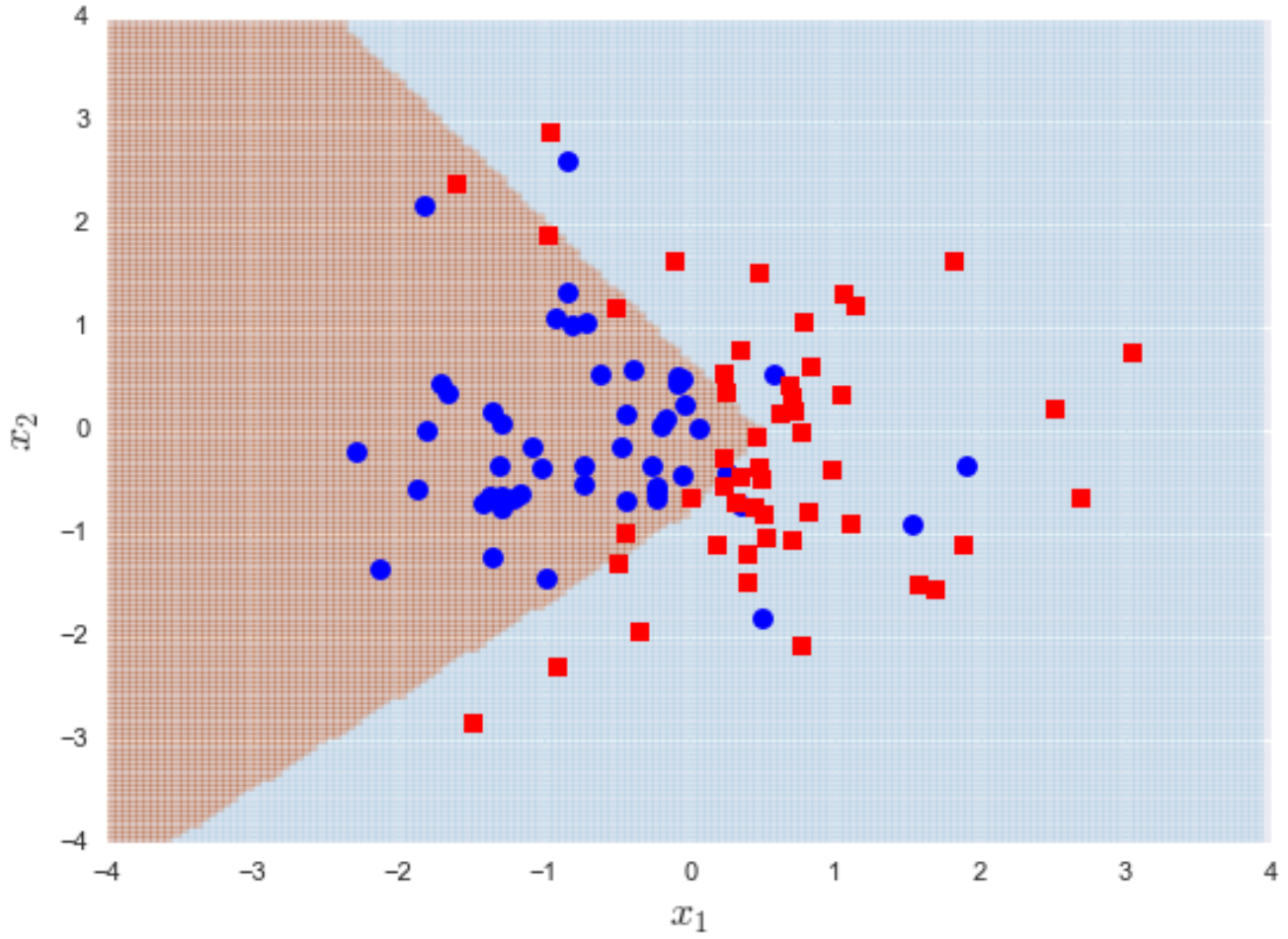
$$f(\mathbf{x};\theta) = 0.5$$



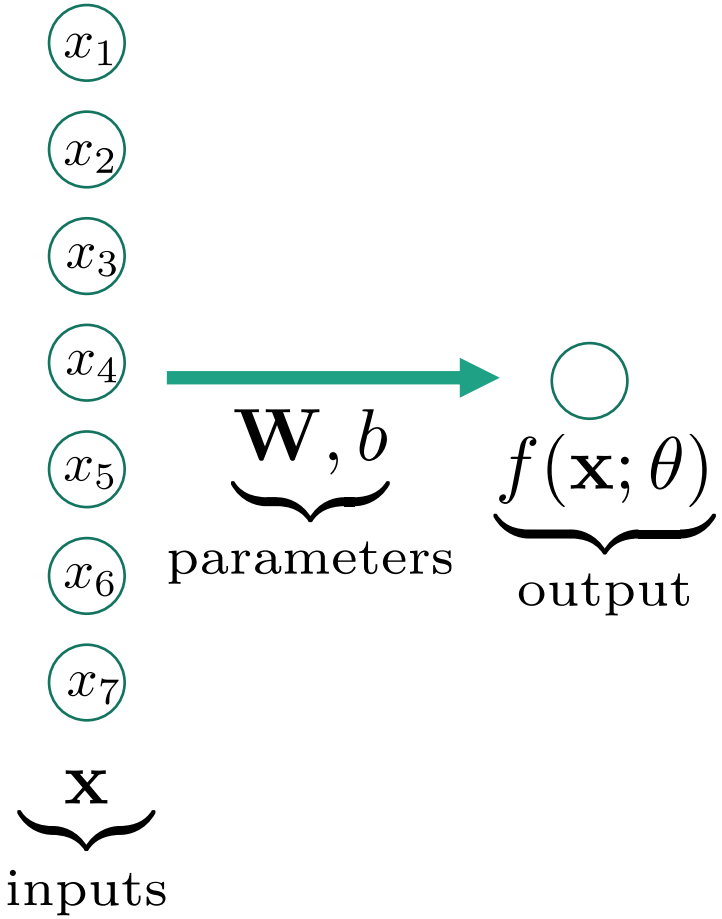
Notes on perceptron model

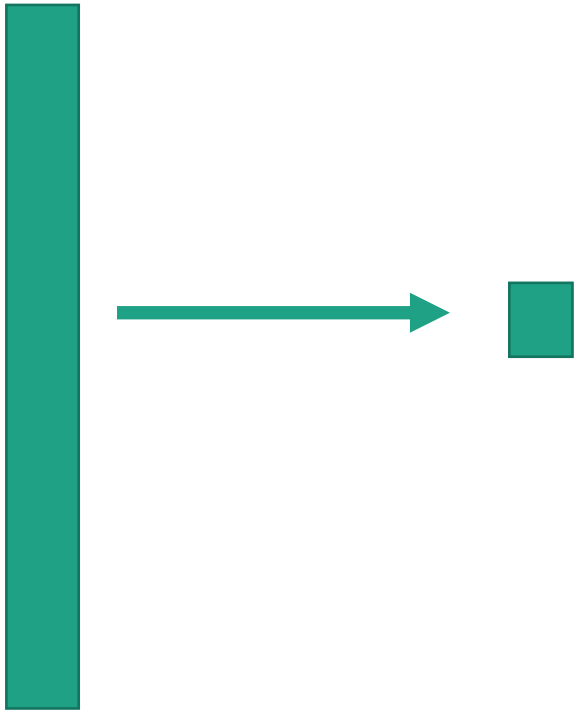
- Logistic regression is the building block
- Essentially perceptron is a linear model
- Linear decision boundary
- Cannot model more complicated decision boundaries
- Building block for more complicated models
- We have not seen training yet, will come back that
- First let’s see more complicated models

It would be better if



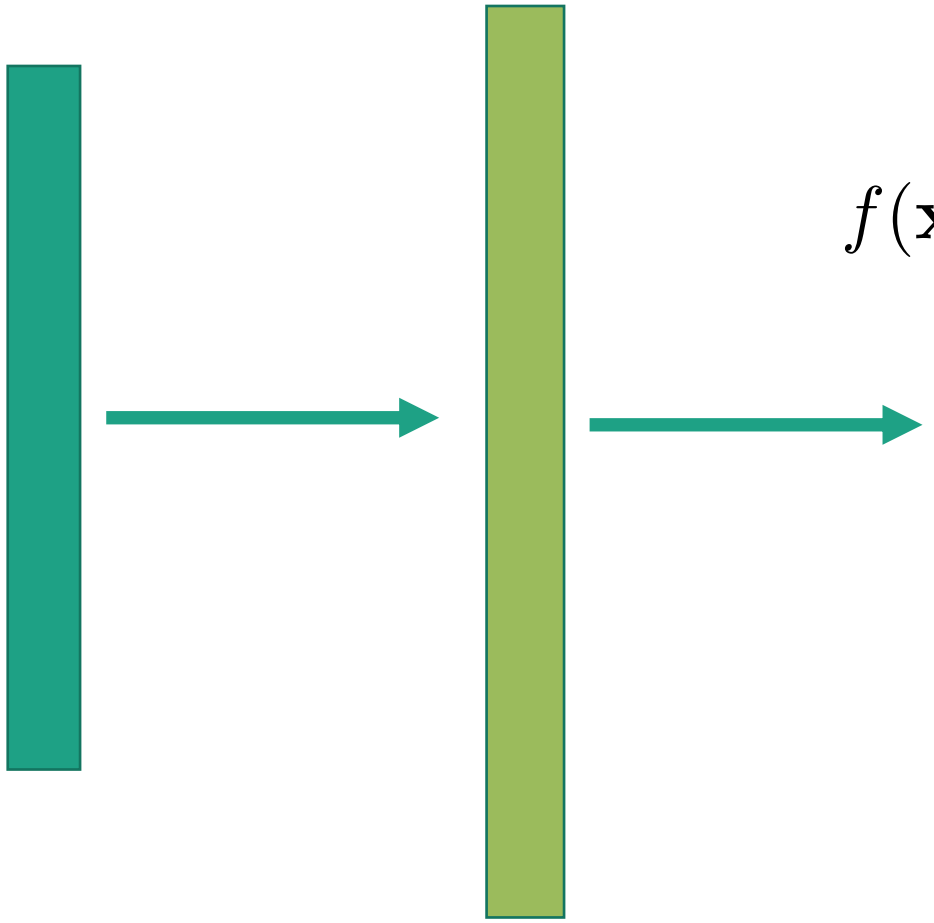
Simpler graphical representation





Simpler graphical representation

Multilayer perceptron (MLP)



$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}\mathbf{x} + b)$$

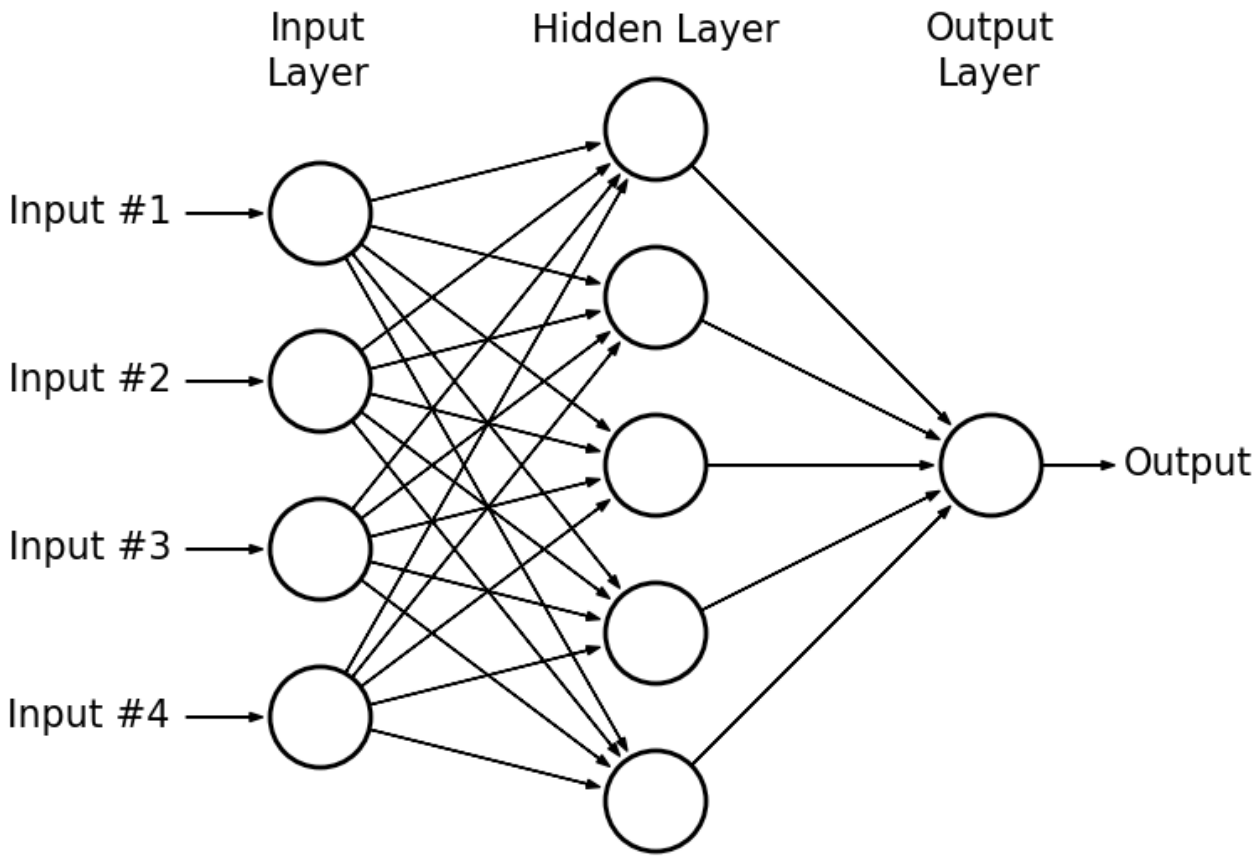
$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + b_1) + b_2)$$

$$f(\mathbf{x};\theta) = f_2(f_1(\mathbf{x};\theta_1);\theta_2)$$

$$\theta = \{\theta_1, \theta_2\} = \{\mathbf{W}_1, \mathbf{W}_2, b_1, b_2\}$$

Short-hand notation

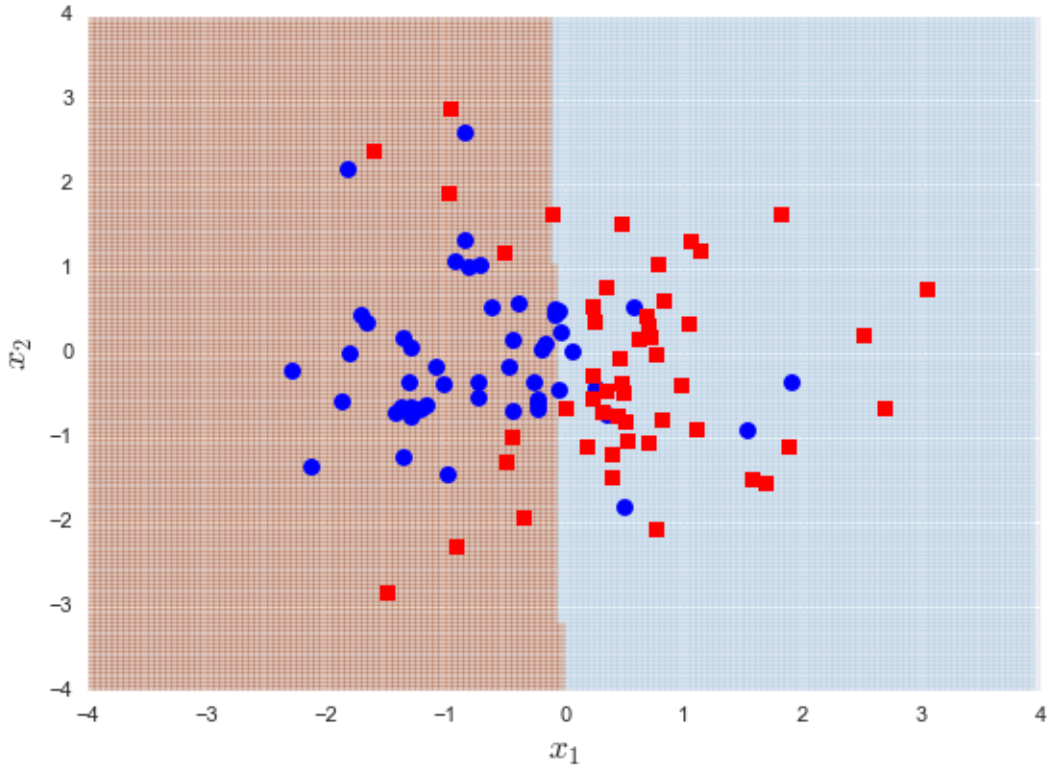
$$f(\mathbf{x};\theta) = f_2 \circ f_1 \circ \mathbf{x}$$


Input
Layer
Hidden Layer
Output
Layer
Input #1
Input #2
Input #3
Input #4
Output

Multilayer perceptron (MLP)

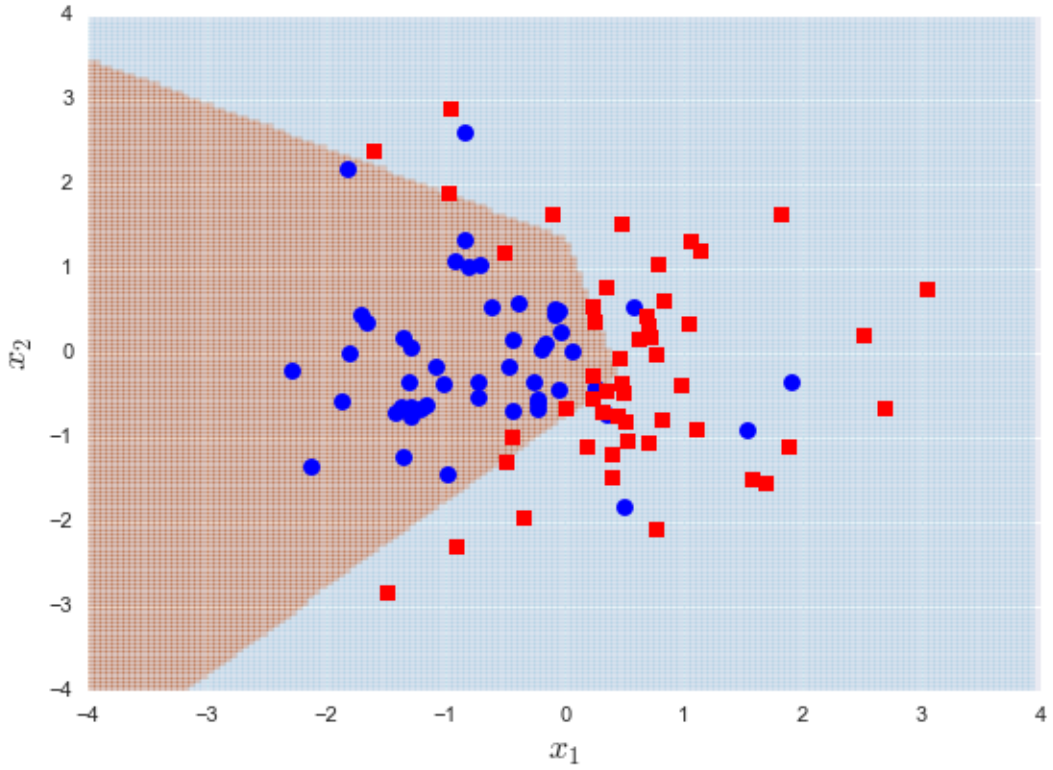
$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}\mathbf{x} + b)$$

Linear separation



$$f(\mathbf{x};\theta) = f_2(f_1(\mathbf{x};\theta_1);\theta_2)$$

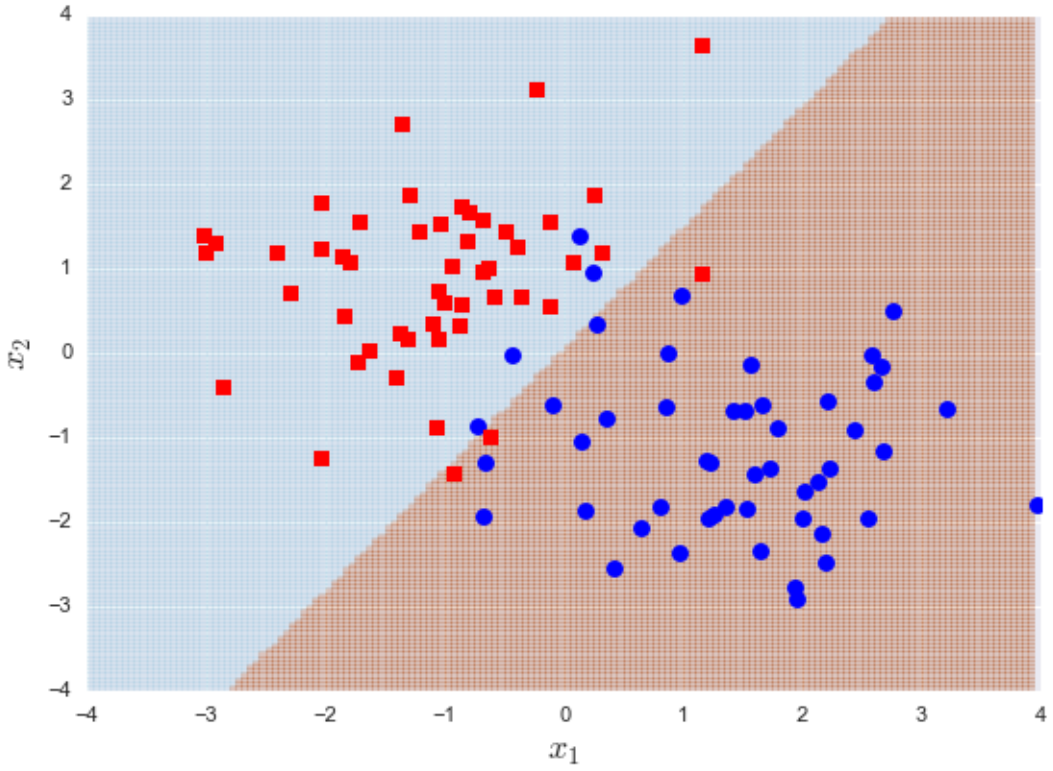
Non-linear separation



Multilayer perceptron (MLP)

$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}\mathbf{x} + b)$$

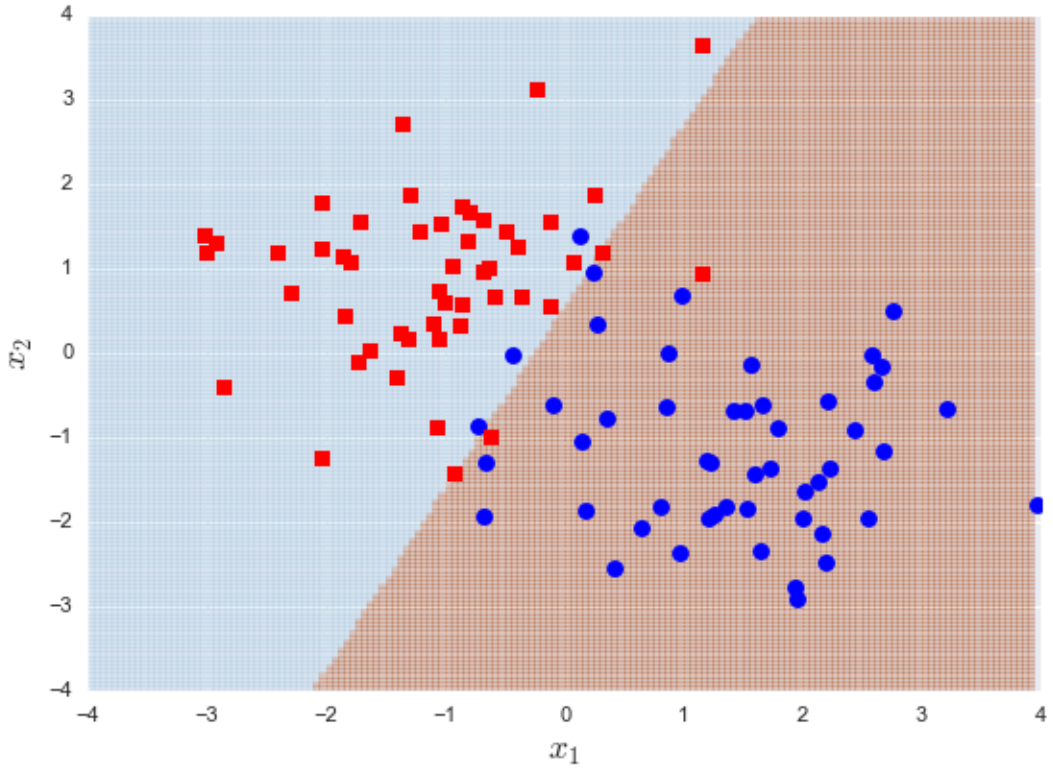
Linear separation



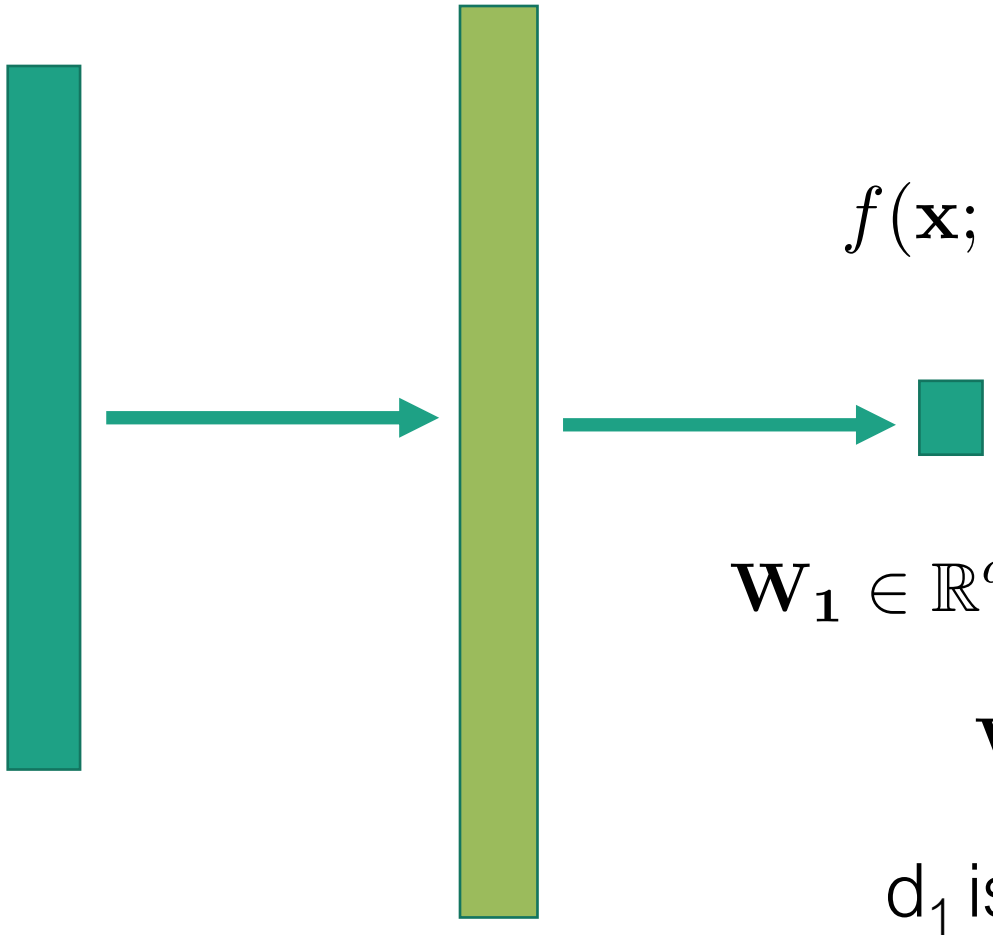
More powerful model

$$f(\mathbf{x};\theta) = f_2(f_1(\mathbf{x};\theta_1);\theta_2)$$

Able to do linear separation



MLP - width



$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}\mathbf{x} + b)$$

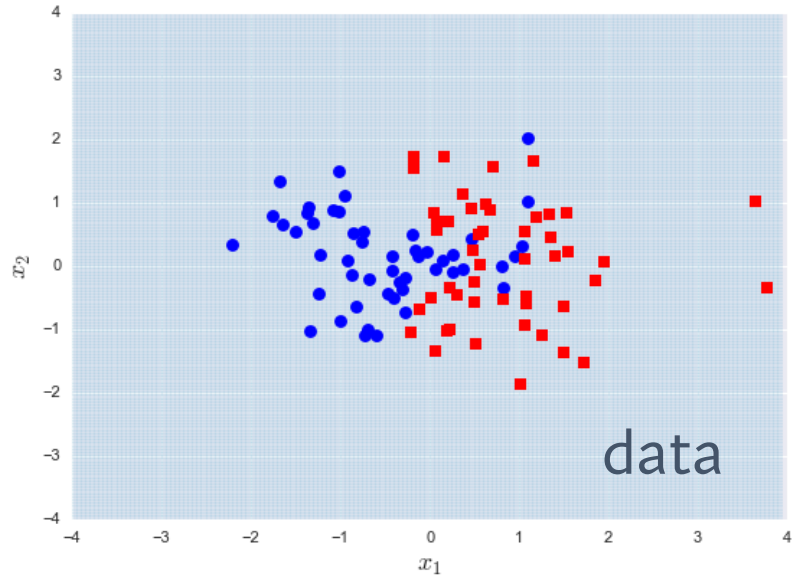
$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + b_1) + b_2)$$

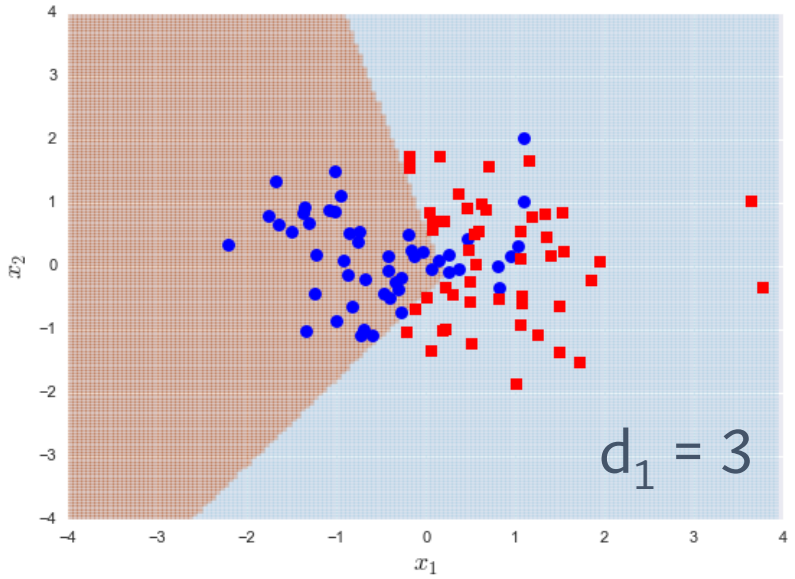
$$\mathbf{W_1} \in \mathbb{R}^{d_1 \times d} \implies \sigma(\mathbf{W}_1\mathbf{x} + b_1) \in \mathbb{R}^{d_1}$$

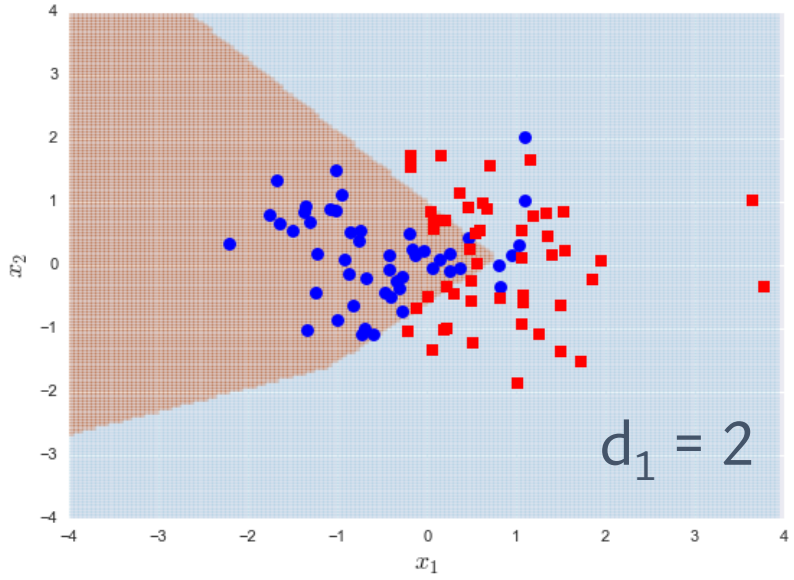
$$\mathbf{W_2} \in \mathbb{R}^{1 \times d_1} \implies f(\mathbf{x};\theta) \in \mathbb{R}$$

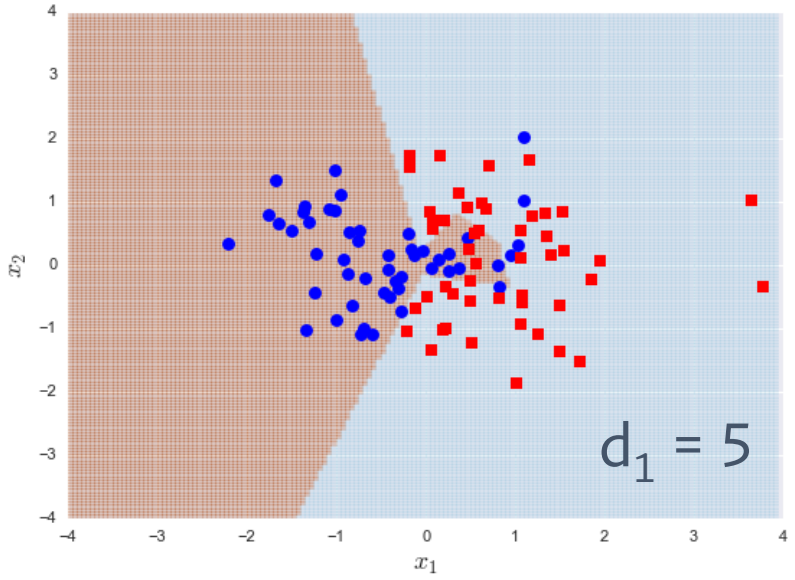
d_1 is the width of the hidden layer

Decision boundary at different width

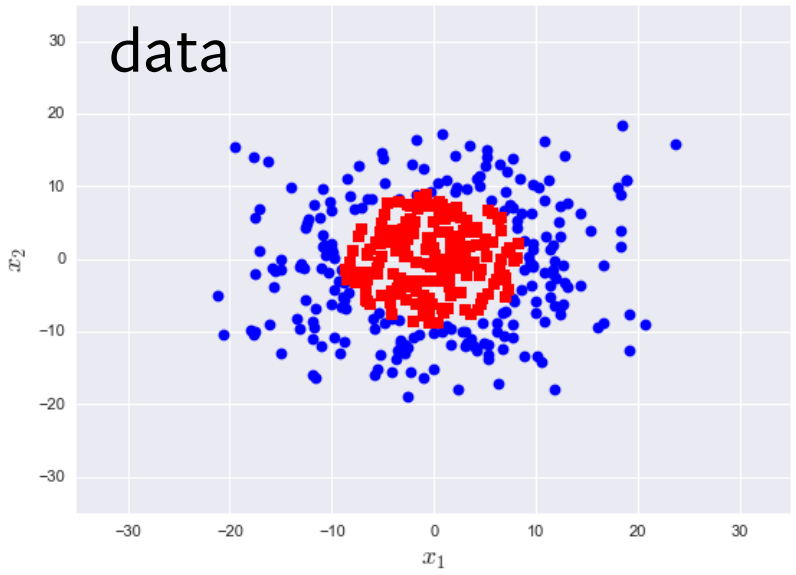


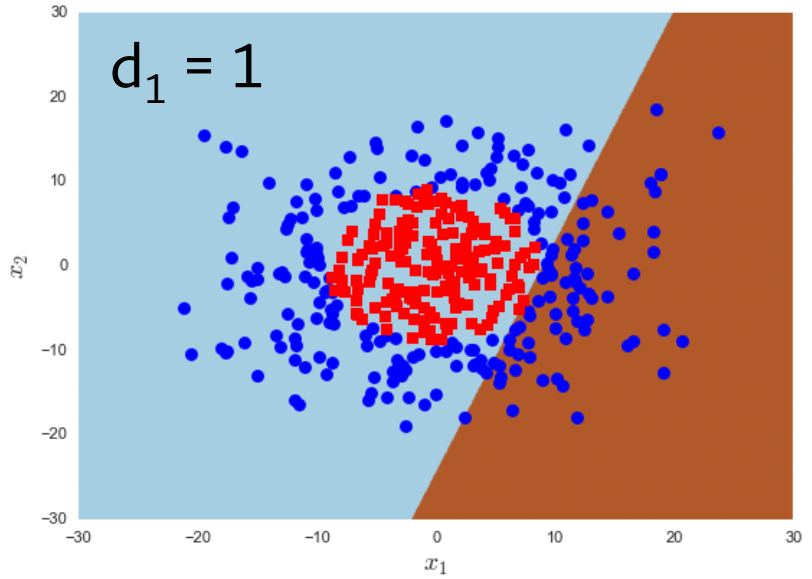


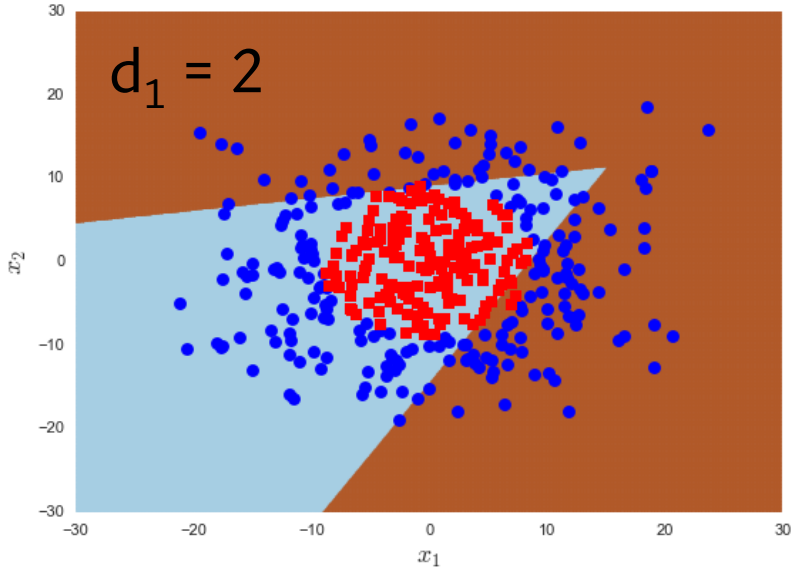


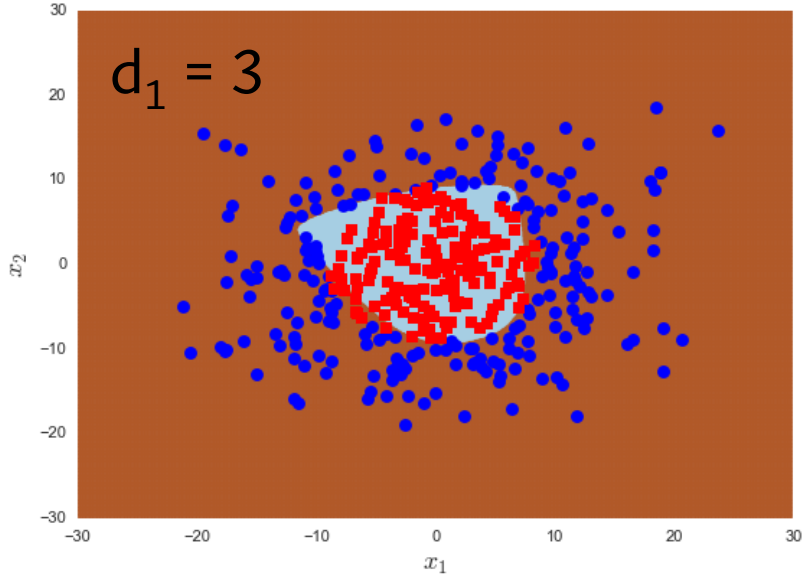


Decision boundary at different width





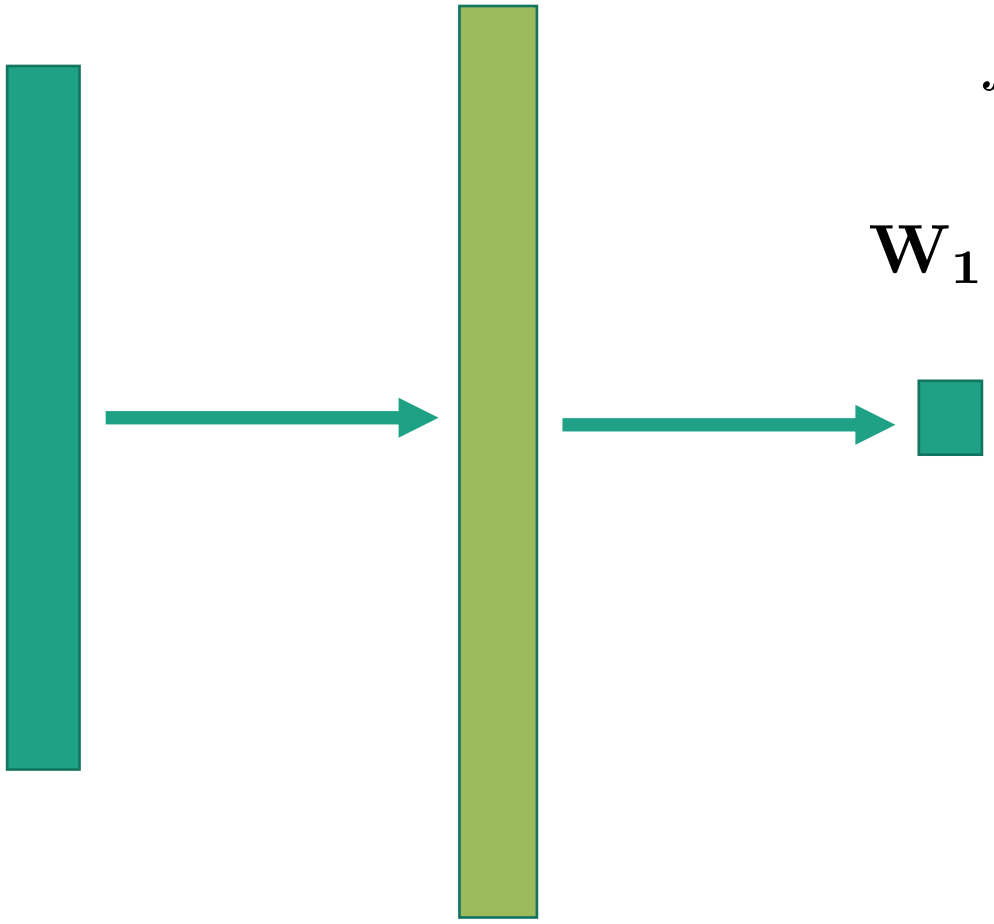




MLP – notes on width

- $d_1 = d$ -- only nonlinear transformation
- $d_1 > d$ -- mapping to a higher dimensional space
  - often it becomes easier to separate classes in higher dimensions
  - able to model complicated decision boundaries
  - Larger number of model parameters
- $d_1 < d$ – compression / bottleneck – possible information loss
  - becomes interesting for determining low-dimensional representations

Number of parameters increased



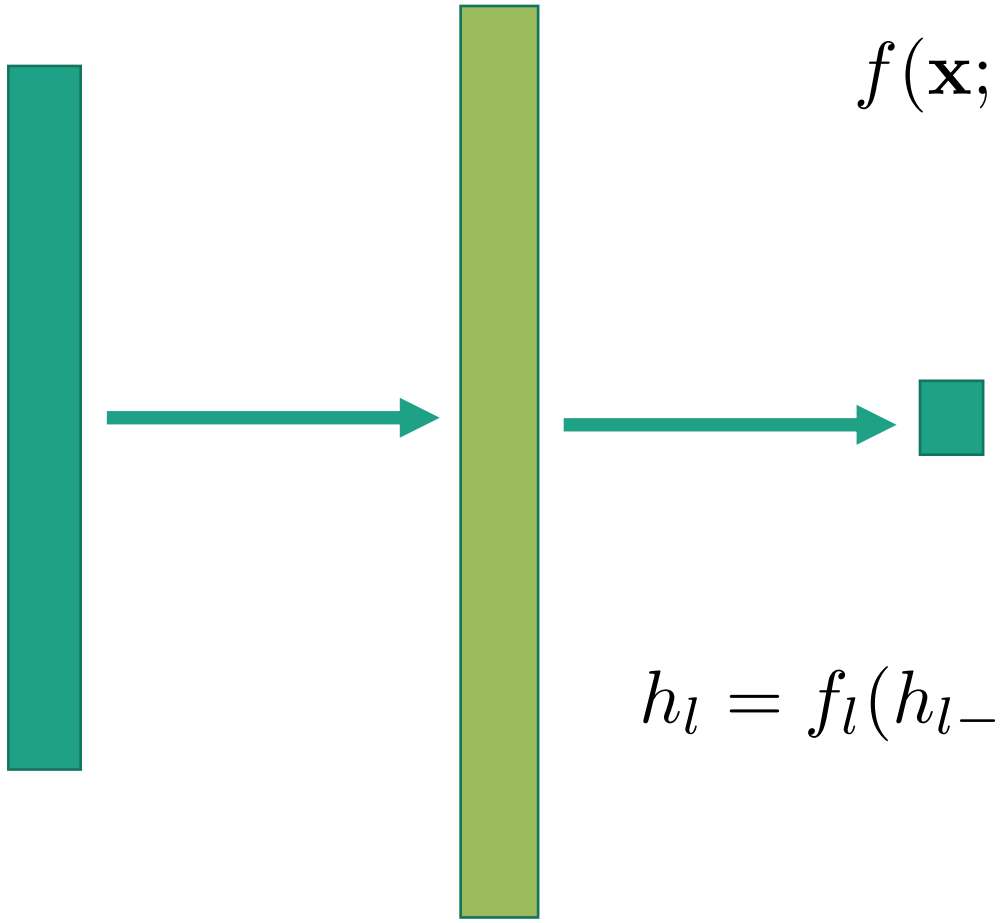
$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + b_1) + b_2)$$

$$\mathbf{W_1} \in \mathbb{R}^{d_1 \times d} \implies \sigma(\mathbf{W}_1\mathbf{x} + b_1) \in \mathbb{R}^{d_1}$$

$$\mathbf{W_2} \in \mathbb{R}^{1 \times d_1} \implies f(\mathbf{x};\theta) \in \mathbb{R}$$

W_1: has $d_1$xd parameters
b_1: is only 1 parameter
W_2: has d_1x1 parameters
b_2: is only 1 parameter
Total: $d_1$xd + d_1 + 2 parameters

MLP - depth



Network with one hidden layer

$$f(\mathbf{x};\theta) = \sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x} + b_1) + b_2)$$

Hidden layers

$$h_l = \sigma(\mathbf{W}_l h_{l-1} + b_l)$$

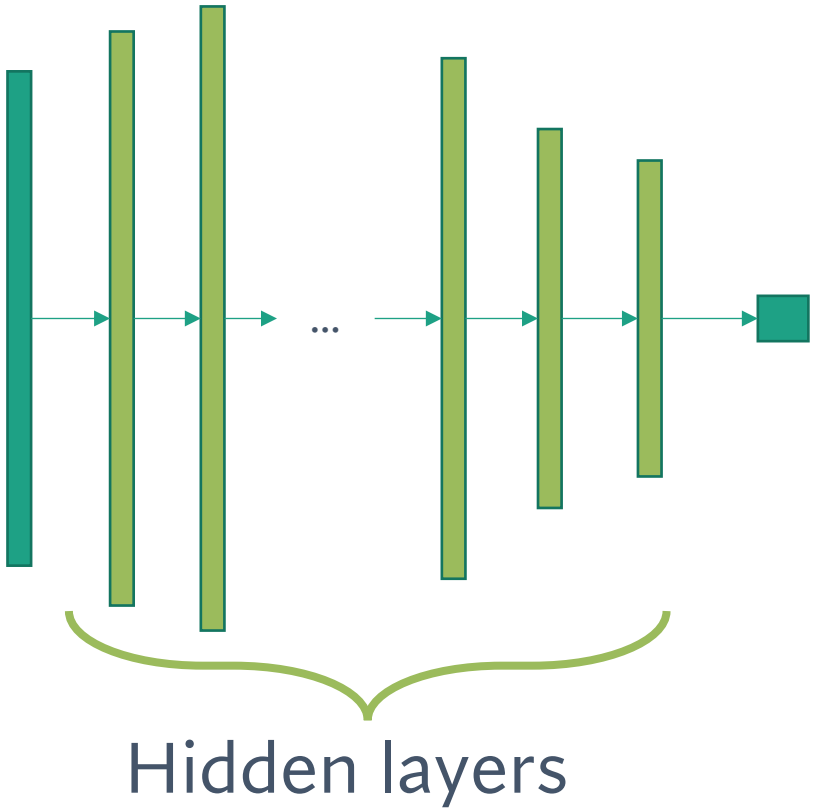
$$h_0 = \mathbf{x},\ \ h_2 = f(\mathbf{x};\theta)$$

Function view

$$h_l = f_l(h_{l-1};\theta_l) = f_l(f_{l-1}(h_{l-2},\theta_{l-1});\theta_l)$$

$$h_2 = f(\mathbf{x};\theta) = f_2 \circ f_1(\mathbf{x})$$

MLP – increasing depth



Network with L-1 hidden layer

Hidden layers

$$h_l = \sigma(\mathbf{W}_l h_{l-1} + b_l)$$

$$h_0 = \mathbf{x},\ \ h_L = f(\mathbf{x};\theta)$$

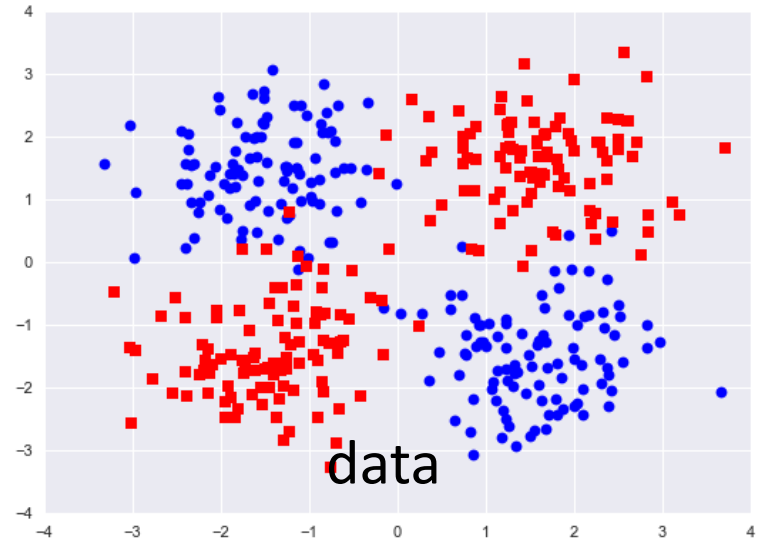
Function view

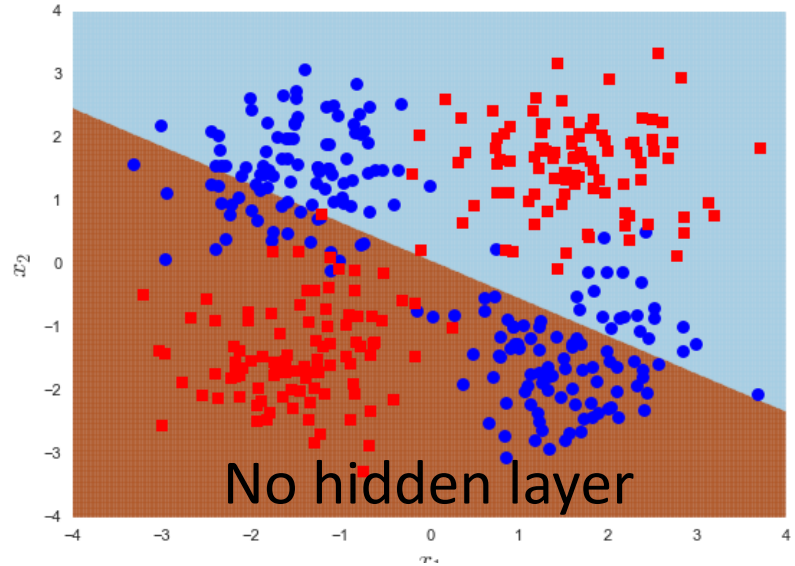
$$f_l(h_{l-1};\theta_l) = \sigma(\mathbf{W}_l h_{l-1} + b_l)$$

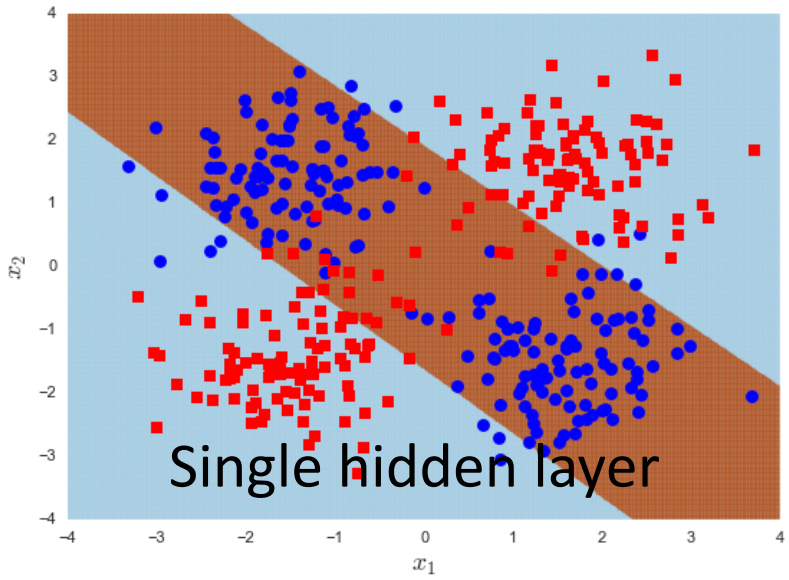
$$f(\mathbf{x};\theta) = f_L \circ \cdots \circ f_2 \circ f_1(\mathbf{x})$$

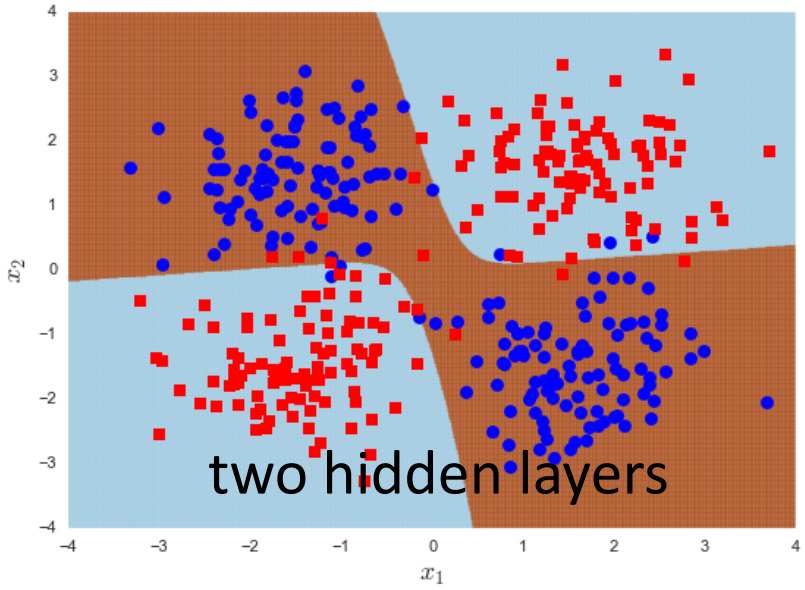
$$\theta = \{\theta_L, \ldots, \theta_1\}$$

Decision boundary at different depths [width = 2 at all depths]





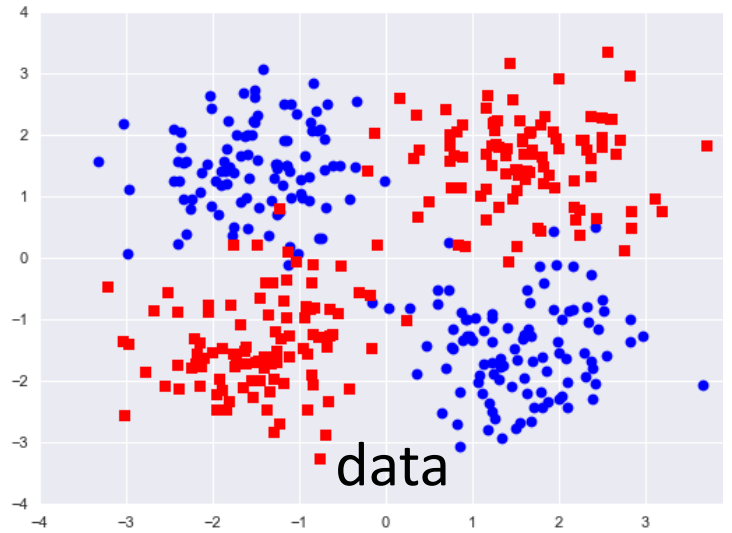


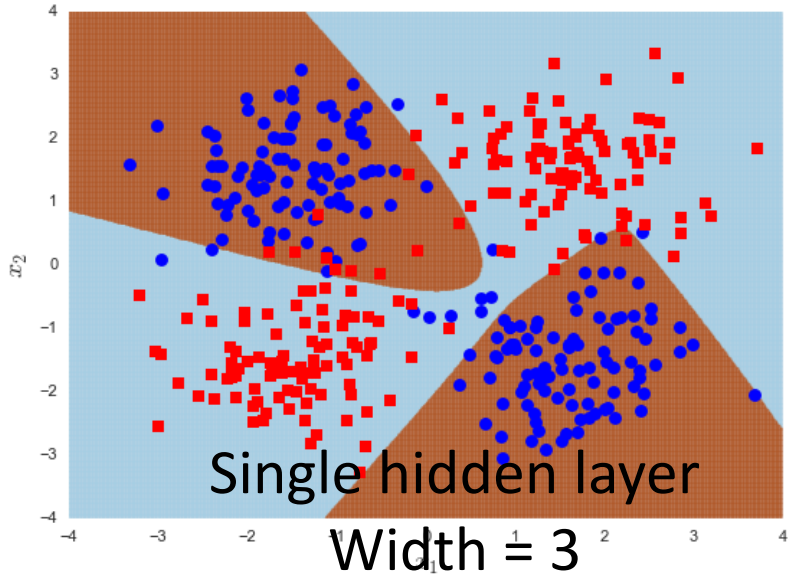


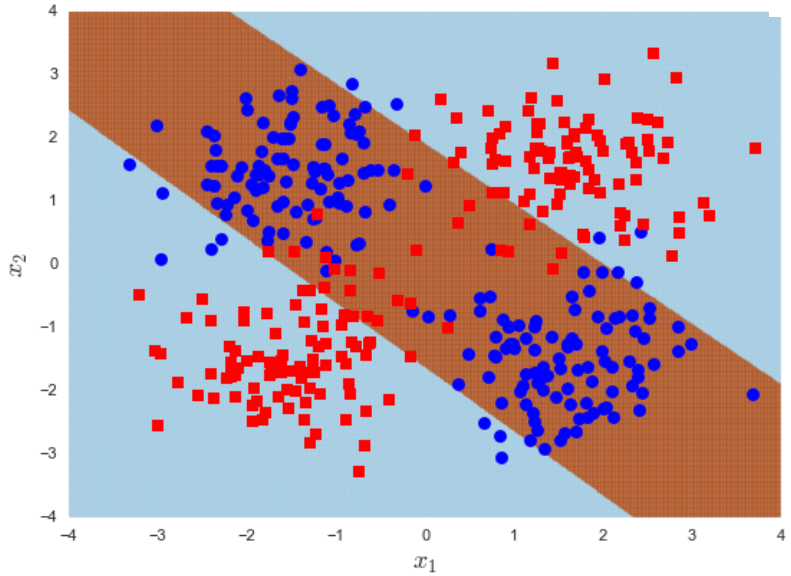
MLP – notes on depth

- No hidden layer – perceptron / logistic regression / linear
- Increasing depth
  - Allows more complicated decision boundaries
  - More powerful models
  - Leads to larger number of model parameters
  - Needs more samples to fit reliably
- May become difficult to train

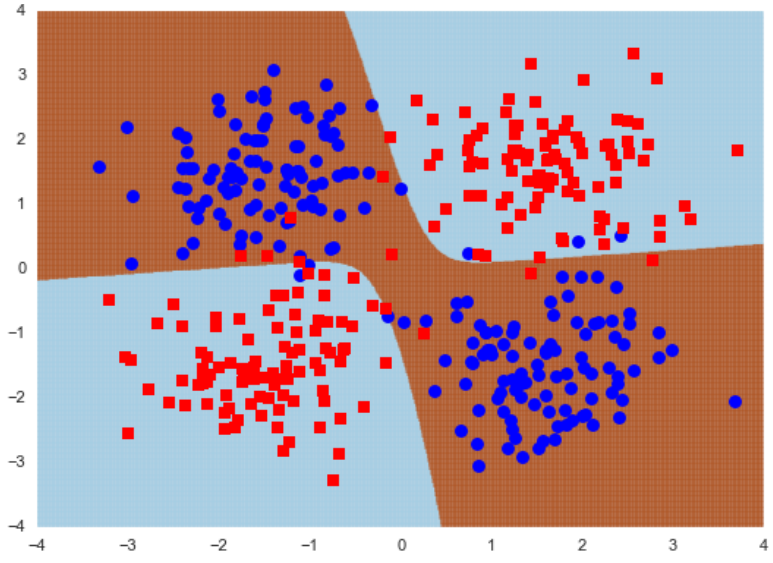
Depth / width interaction







Single hidden layer
Width = 2



Two hidden layers
Widths = 2

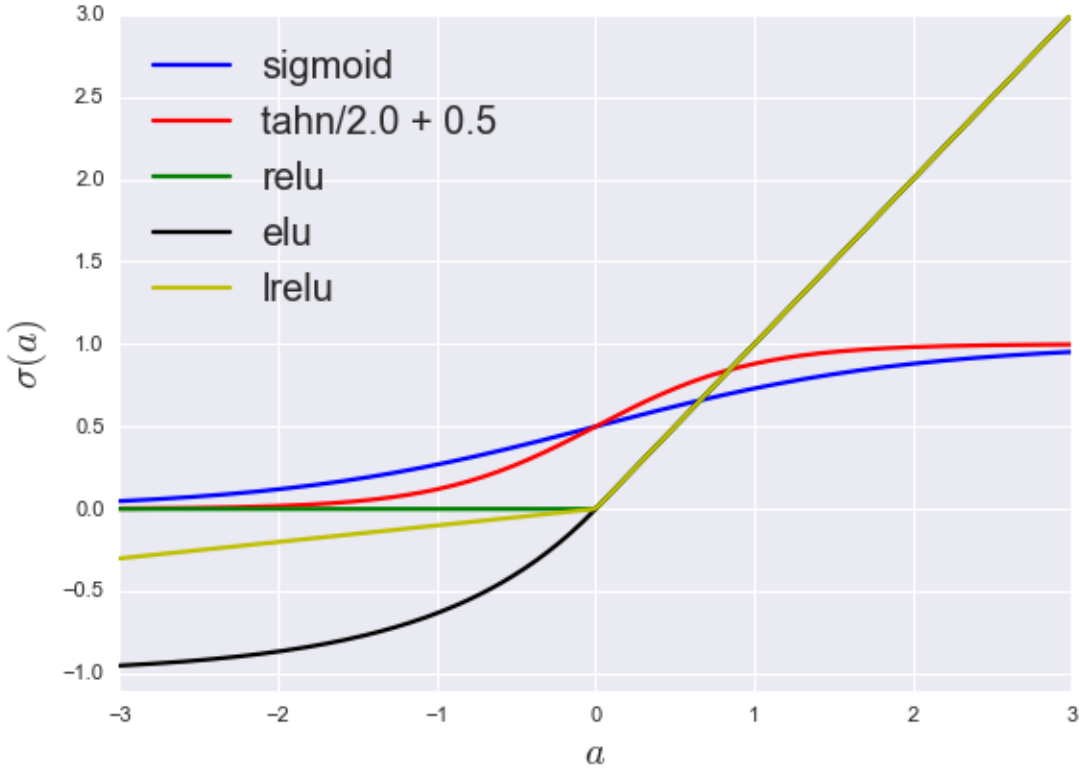
MLP – notes on depth and width

- Complicated interaction between depth and width
- Both allow modeling more complicated decision boundaries
- Choices of depth and width are *architectural* design choices.
- No accepted, widely used method for automatically determining architecture for a given problem
- Problem specific – mostly trial and error-based strategy
- A huge search space
- Engineering / intuition / art
- Prediction accuracy on a validation set

Non-linearity – more recent models

Often used in connections between internal layers

Element-wise application



Rectified Linear Unit (Relu)

$$\sigma(a) = \begin{cases} a, & a \geq 0 \\ 0, & a < 0 \end{cases}$$

Leaky Relu

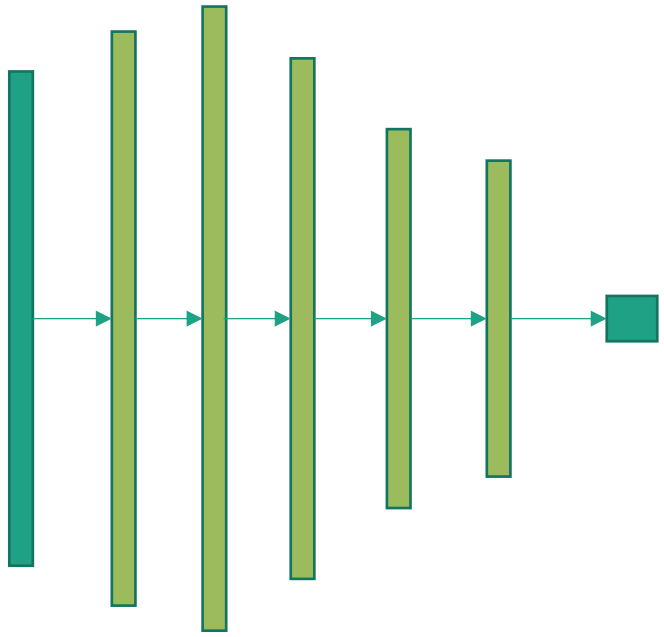
$$\sigma(a) = \begin{cases} a, & a \geq 0 \\ \alpha a, & a < 0 \end{cases}$$

Exponential Linear Unit (Elu)

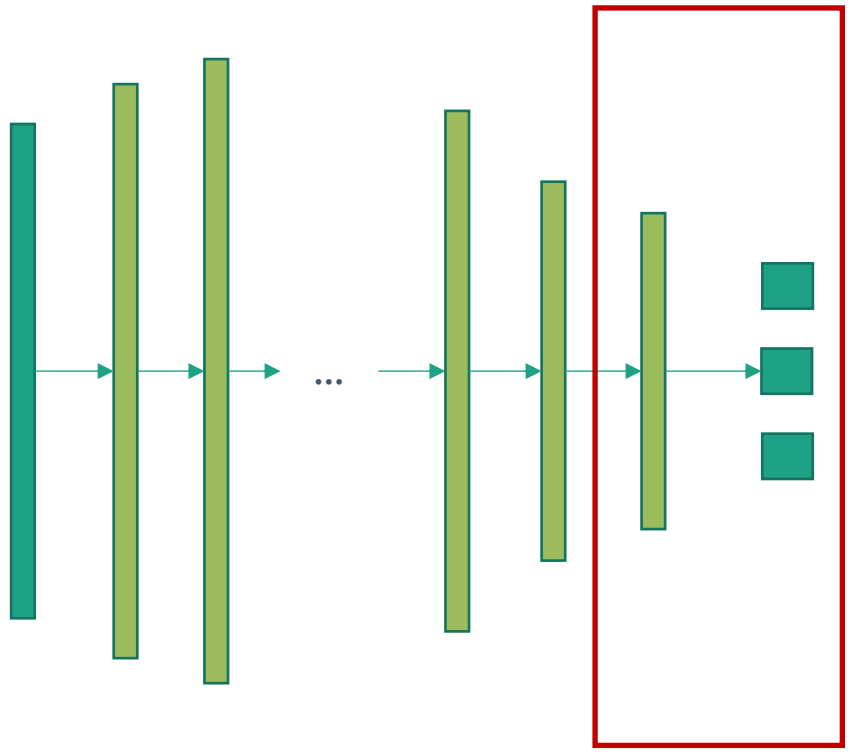
$$\sigma(a) = \begin{cases} a, & a \geq 0 \\ \exp(a) - 1, & a < 0 \end{cases}$$

Fully connected architecture

- This is called a fully connected architecture.
- Every neuron in a layer is connected to every neuron in the subsequent layer.



Fully Connected Classification Network



- Multilayer perceptron model with multiple outputs
- In the generic case, applies to multi-label classification
- Last layer of the network is constructed to make the network perform binary or multi-class classification
- So, the final layer is either a sigmoid function or a *soft-max* function that is designed for multi-label problems

Soft-max function

Binary classification: $a_L = \mathbf{W}_L h_{L-1} + b_L \in \mathbb{R}$

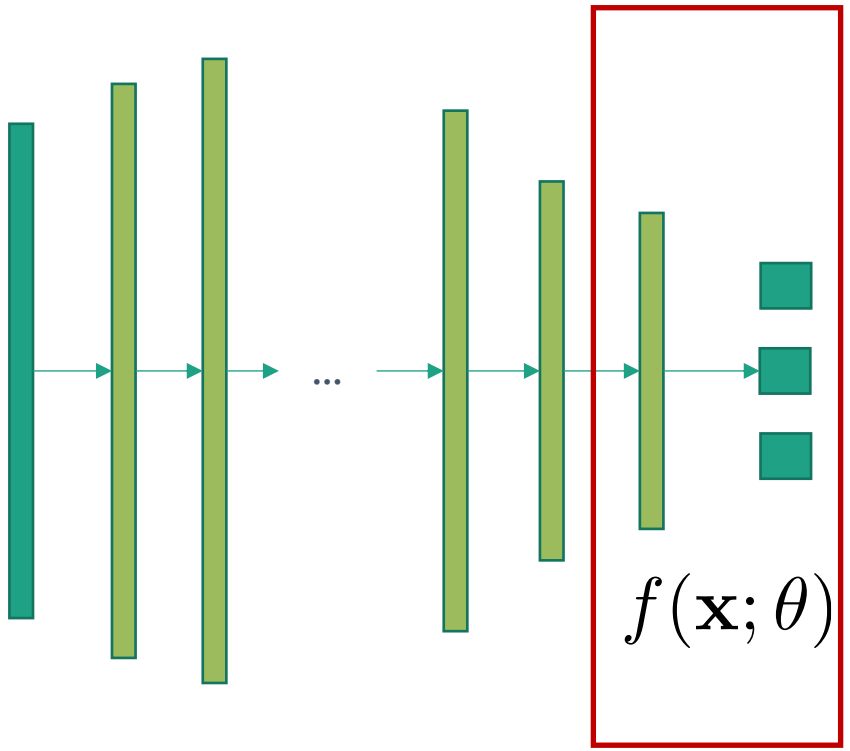
$$p(y=1) = f(\mathbf{x};\theta) = \frac{1}{1+e^{-a_L}},\ p(y=0) = 1 - p(y=1)$$

Multi-label classification – K classes: $a_L = \mathbf{W}_L h_{L-1} + b_L \in \mathbb{R}^K$

$$p(y=k) = f_k(\mathbf{x};\theta) = \frac{e^{a_{L,k}}}{\sum_{k' \in \mathcal{C}} e^{a_{L,k'}}} \quad \sum_{k \in \mathcal{C}} p(y=k) = 1$$

Soft-max function

Fully Connected Regression Network

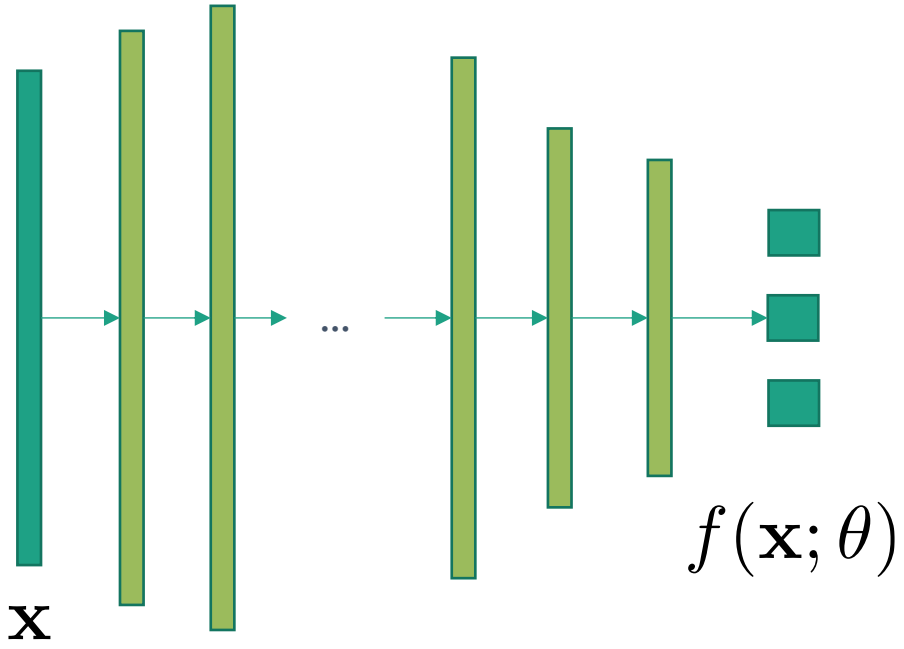


- Regression network is very similar to the classification network
- For an M dimensional regression problem

$$y \in \mathbb{R}^M \qquad f(\mathbf{x};\theta) \in \mathbb{R}^M$$

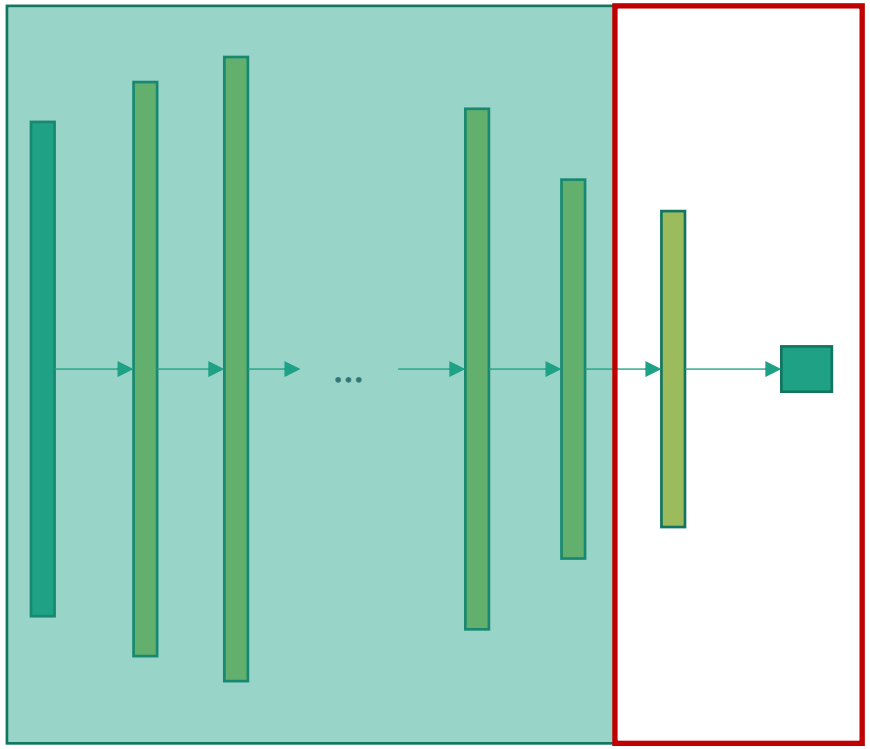
- It is common to not use any non-linear activation at the final layer

Information flow is forward while predicting



- Flow of information if *forward* when predicting
- The input is fed through the layers successively until the outputs
- *Feed-forward network architecture*

MLP – feature space view



- There is a linear model at the very final layer

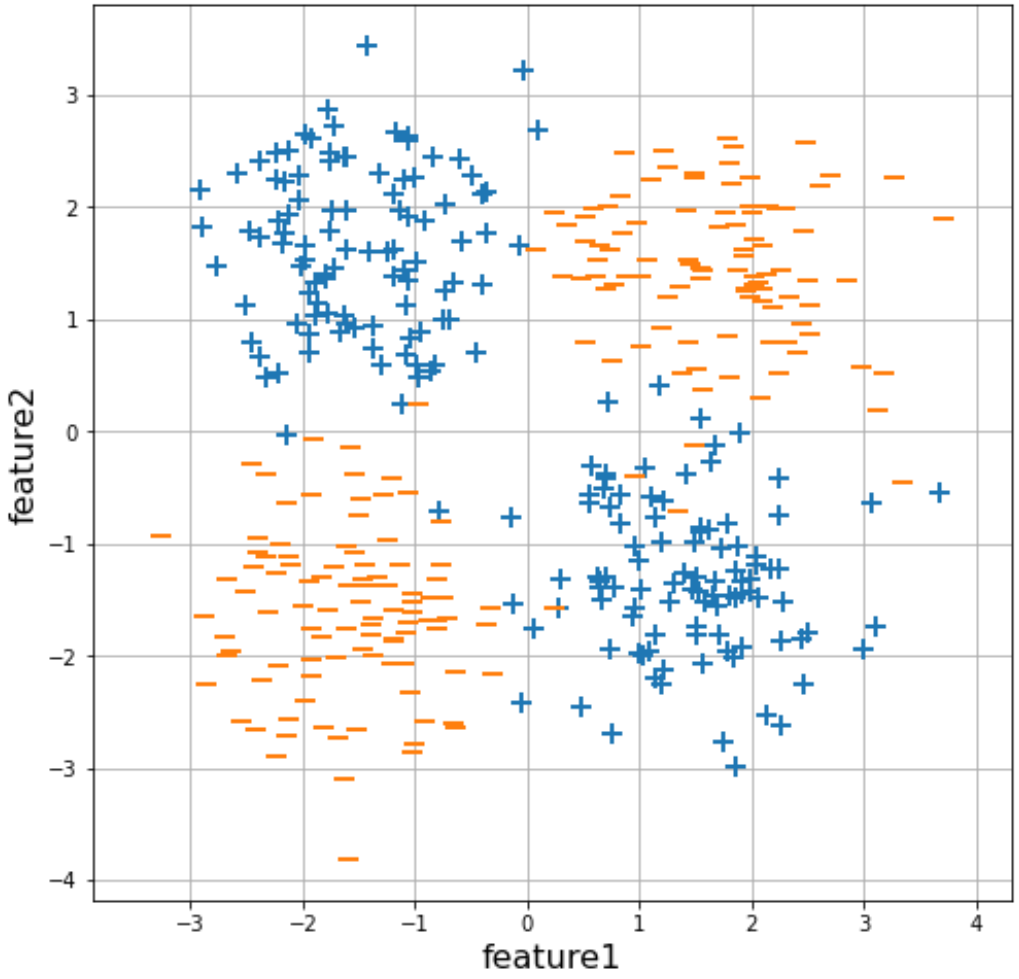
$$f(\mathbf{x};\theta) = f_L(\phi(\mathbf{x});\theta_L) = \sigma(\mathbf{W}_L\phi(\mathbf{x}) + b_L)$$

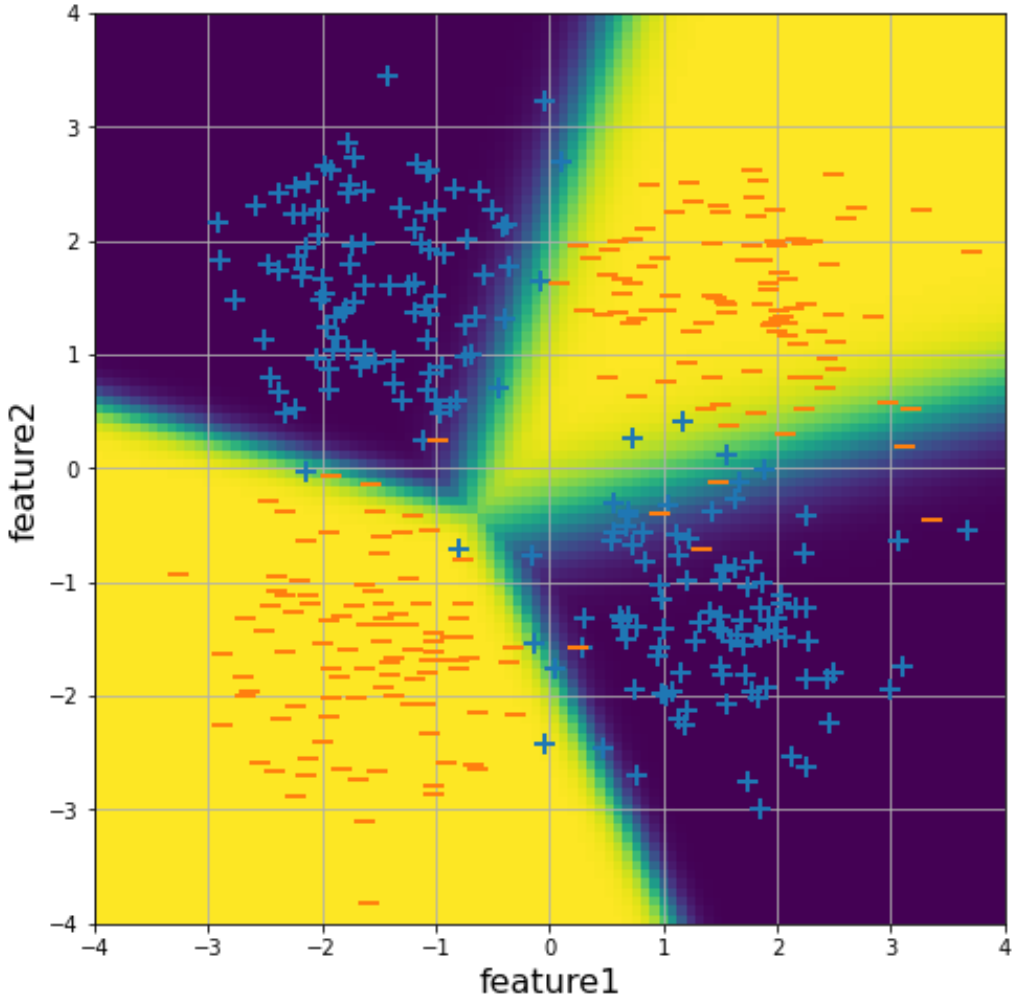
- There is a feature map that transforms the input

$$\phi(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}^{d_{L-1}}$$

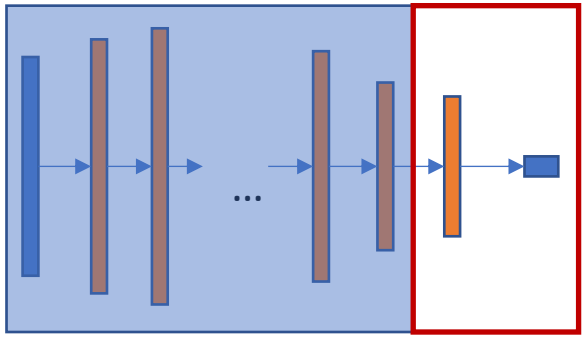
- Automatically determined non-linear feature map

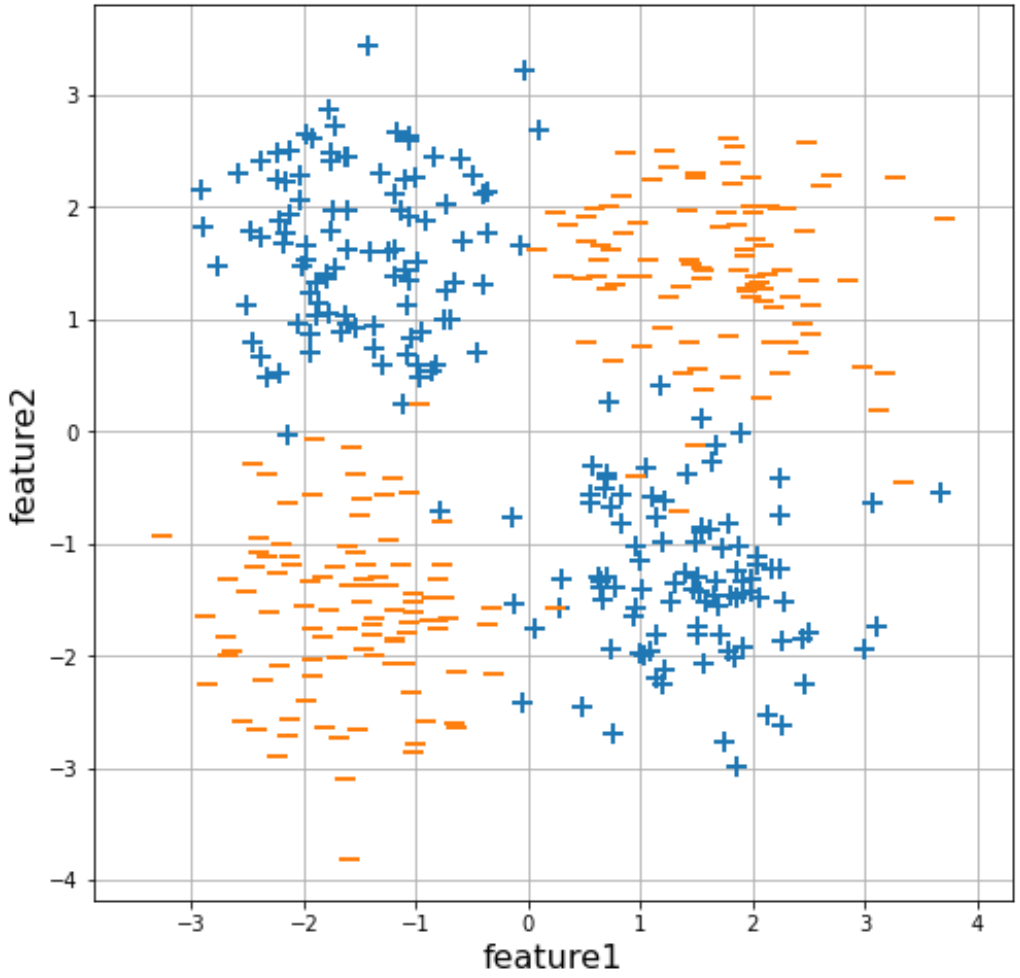
What does that mean?

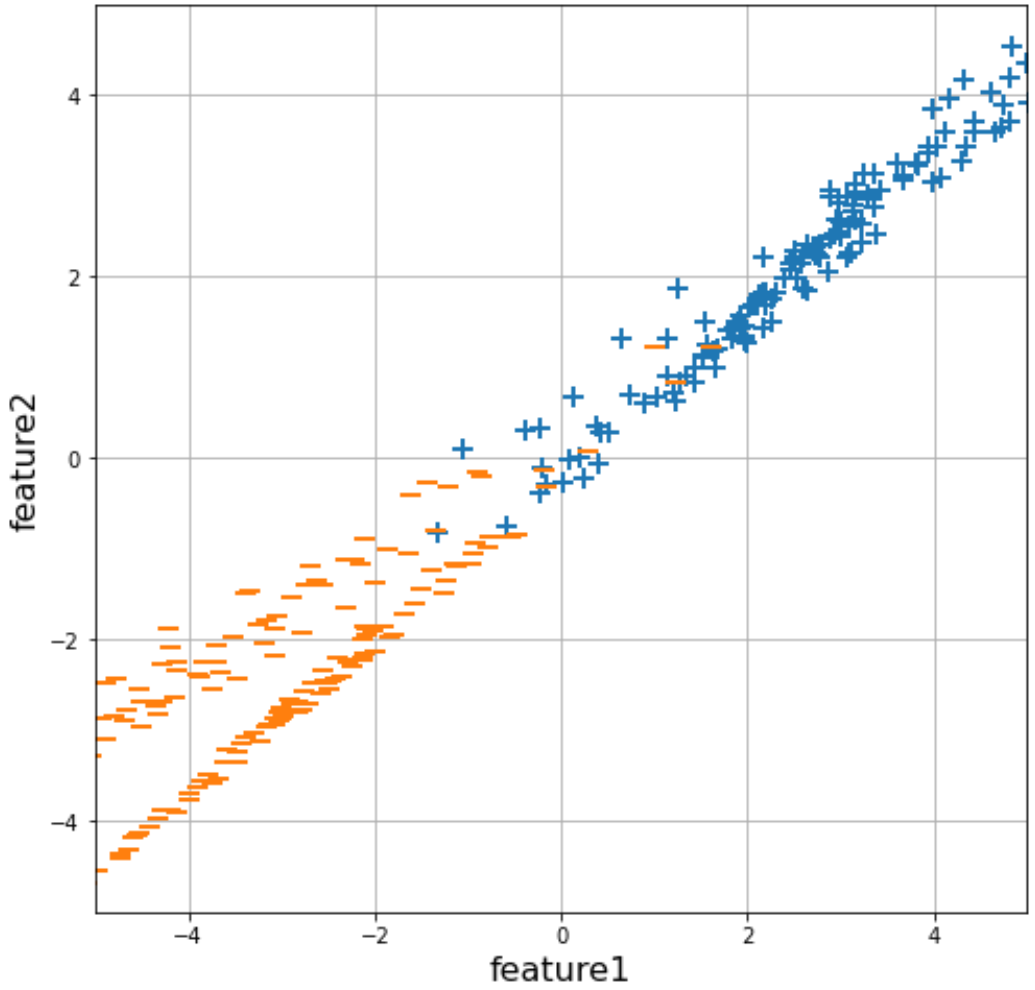




What does that mean?







MLP – why non-linear feature map

- Non-linearity is due to the activation functions $\sigma(\cdot)$
- What happens if all activations were linear functions?

$$\sigma(a) = \mathbf{V}a + c$$

$$f_l(h_{l-1}) = \mathbf{V}\left(\mathbf{W}_l h_{l-1} + b\right) + c = \tilde{\mathbf{W}} h_{l-1} + \tilde{b}$$

$$y = f_L \circ f_{L-1} \circ \cdots \circ f_1(\mathbf{x}) = \hat{\mathbf{W}}\mathbf{x} + \hat{b}$$

- The final map is effectively linear – same as a single linear model, such as logistic regression
- Non-linearity is extremely important for complicated models

The power of ANNs

- Universal approximation theorem: a feedforward network can accurately approximate any continuous function from one finite dimensional space to another, **given enough hidden units** (Hornik et al. 1989, Cybenko 1989).

- Therefore, ANNs have the potential to be universal approximators.

- However - universal approximation theorem does not provide any guarantee that training finds this representation. Subject to model tuning and computational power.

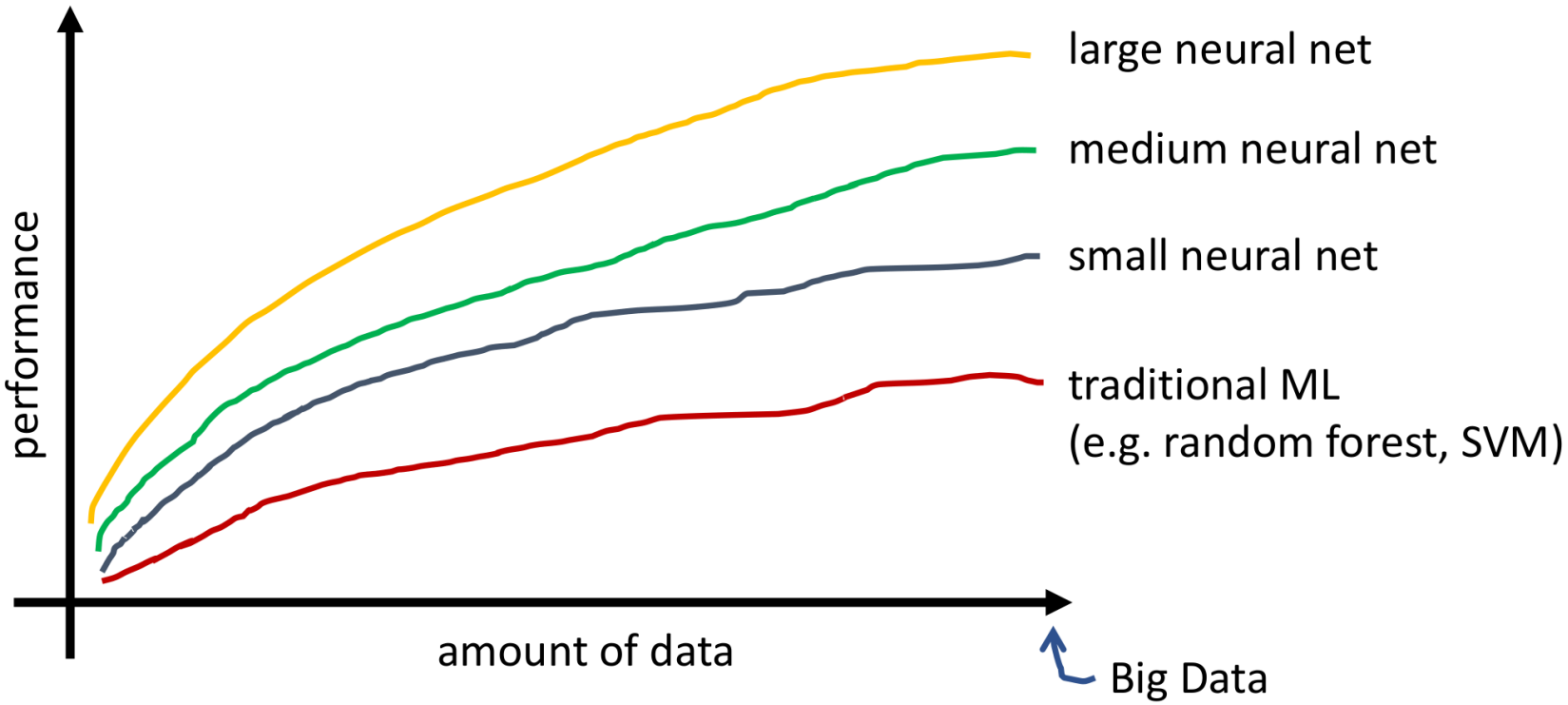
- In addition, there are many variants of ANN that are well-suited for different types of data (tableau, image, time series, etc.) without requiring data transformation. This is called end-to-end learning. Will discuss in the lecture of ‘unstructured data’.

Textbooks and tutorials

- VanderPlas, "Python data science handbook", O'Reilly, 2017, ISBN 9781491912058 (Example code)

- Geron (2nd Edition), "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", O'Reilly, 2019, ISBN 9781492032649 (Example code)

- Scikit-Learn tutorial, VanderPlas

Deep learning and data size

The performance of DL increases rapidly with the size of the data.