Module 1: Crash course in AI

INFO901

Marija Slavkovik 2022

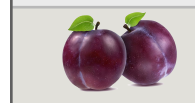
Supervised learning

- Given a training set of N example input-output pairs $(x_1,y_1), (x_2,y_2),\ldots,(x_N,y_N)$ where each y_j was generated by an unknown function $y=f(x)$, discover a function h that approximates the true function f.
- Classification - the output is label, one from a fixed, finite set of labels
- Example: is a single man in possession of a good fortune, in want of a wife? Possible labels {yes, no}
- Regression - the output is a value (real number/floating point) that needs to be predicted
- Example: how much money will this house be sold for today?

Your first ML

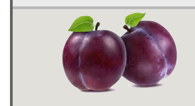
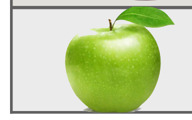
Your first ML

- You will be split into breakout sessions
- Choose 5 features phrased as yes/no questions and enter the yes/no values in the table (5 minutes)

	feature1	feature2	feature3	feature4	feature5	What is it?
						plum
						plum
						apple

Your first ML

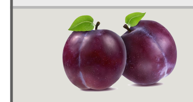
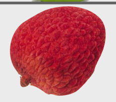
- You will be split into breakout sessions
- Choose 5 features phrased as yes/no questions and enter the yes/no values in the table (5 minutes)

	feature1	feature2	feature3	feature4	feature5	What is it?
						plum
						plum
						apple
						

- Fill in the feature values for the new fruit.
- If the new fruit has most yes/no in common with an apple, label it an apple.
- If the new fruit has most yes/no with a plum, label it a plum.

Your first ML

	feature1	feature2	feature3	feature4	feature5	What is it?
						plum
						plum
						apple
						
						

- Fill in the feature values for the new fruit.
- Count how many yes/no it has in common with an existing fruit. Label it apple if it has most answers in common with an apple. Label it plum if it has most answers in common with plum.

Finding a good hypothesis

- Supervised learning is a search through the space of possible hypothesis to find one that **performs well**.
- We measure how well a hypotheses performs in terms of **accuracy**. To do this we “give it” a **test set** of examples that are distinct from the training set, but for which we know the correct **target values** (labels)
- Accuracy - the fraction of examples for which the correct output was assigned

Overfitting - underfitting

- Supervised learning is a search through the space of possible hypothesis to find one that **performs well**… but not perfect
- The hypothesis needs to handle not only examples of input that have been used in training (“seen”) but also other (“unseen”) examples.
- Overfitting - the hypothesis has very high accuracy for training data (and performs poorly on test data)
- Underfitting -the hypothesis has very low accuracy for test data

C

- Su… find one…
- The… used in trai…
- Ov… forms poo…
- Un…

Different supervised learning methods

- kNN - k nearest neighbours *you just did that with the fruits*
- Naive Bayes clarifiers
- Linear models
- Decision trees
- Kernelized Support Vector Machines (SVM)
- Neural networks

- Which method is best depends on the data you have and if you need classification or regression.
- It should depend on which problem you apply the learnt correlation.. but math can't make you.

Decision trees

- Have existed as a method to guide decisions of people in critical situations.
- Decision trees in AI 1986: Quinlan, J. R. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Figure 18.3 Examples for the restaurant domain.

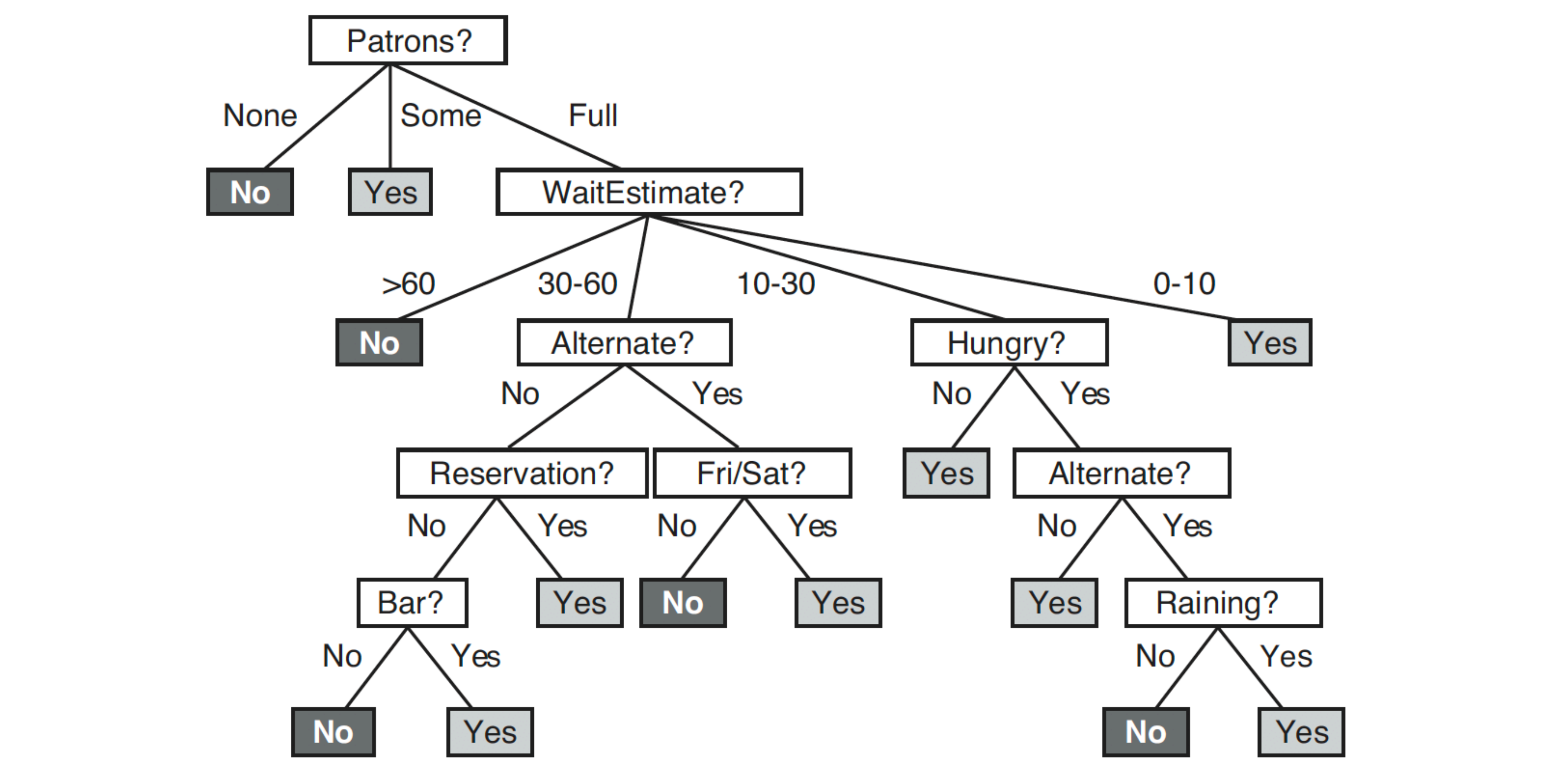


Figure 18.2 A decision tree for deciding whether to wait for a table.

Decision trees

- The goal is to build as “short” a tree as possible (perform the smallest amount of tests)
- Pro: They can handle any type of data, does not matter if the features are correlated, any amount of data (do not need big data to work)
- Con: They overfit by design
- In practice you do not really find just one tree you, you use an “ensemble of trees”
- Pro: following the label “upwards” in the tree, you find our the reasons why that particular label has been assigned (not so easy with ensemble of trees).

Linear methods

- Have been known and used in statistics before AI
- Assume the hypothesis is a linear equation
- Can only work on continuous features (feature values are numbers)
- Poor results if the features are correlated

Linear Regression

this formula is called a prediction model

$$y = w_0 * x_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + w_5 * x_5 + w_6 * x_6 + w_7 * x_7 + w_8 * x_8$$

price area bedrooms bathrooms stories mainroad guest room basement h.w.herat aircon

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning
0	13300000	7420	4	2	3	yes	no	no	no	yes
1	12250000	8960	4	4	4	yes	no	no	no	yes
2	12250000	9960	3	2	2	yes	no	yes	no	no
3	12215000	7500	4	2	2	yes	no	yes	no	yes
4	11410000	7420	4	1	2	yes	yes	yes	no	yes

- Learning = keep changing the weights until you get an error in price within acceptable bounds
- The higher the weight in the model, the more relevant the feature for determining the target value **y**

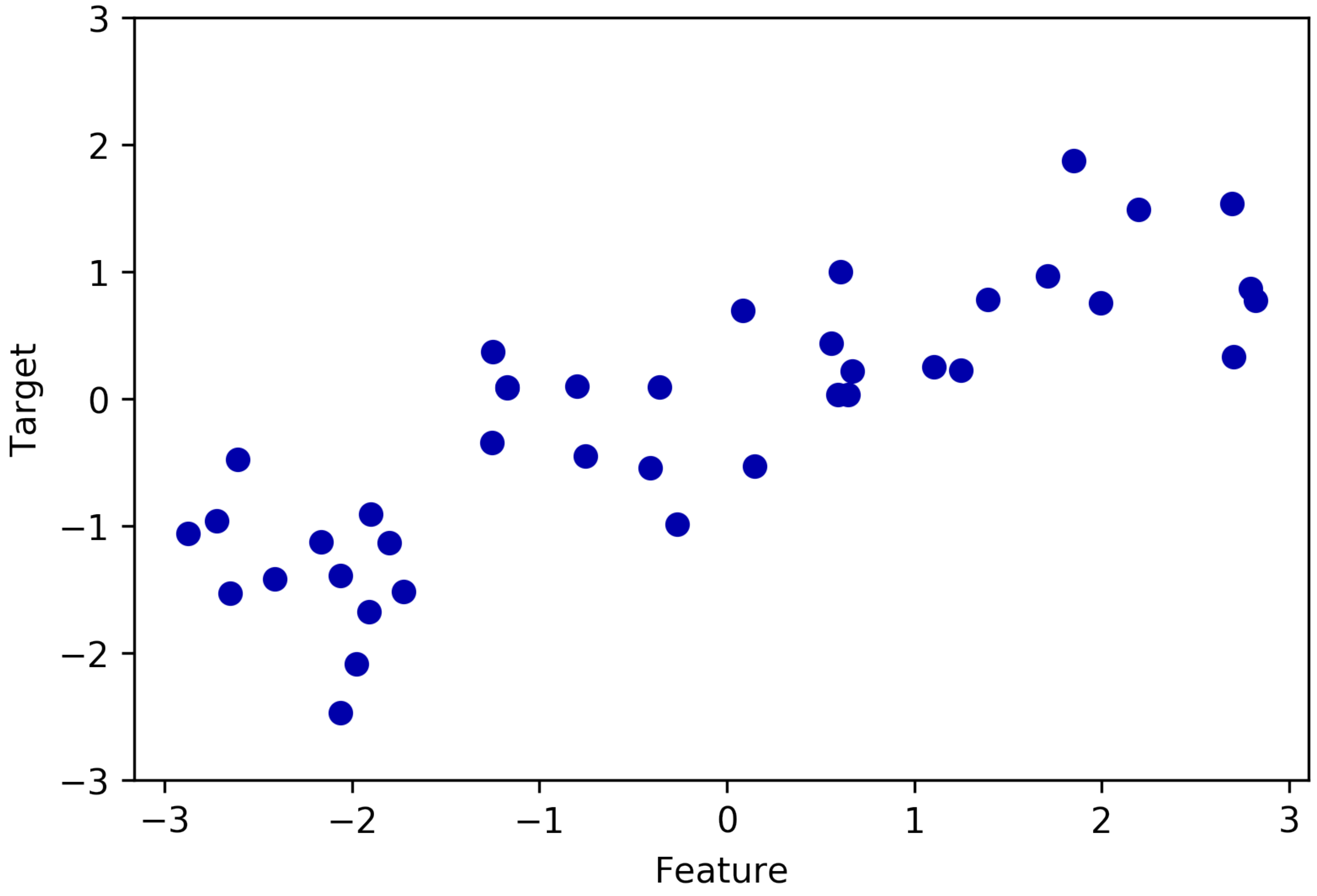
How to improve on linear regression?

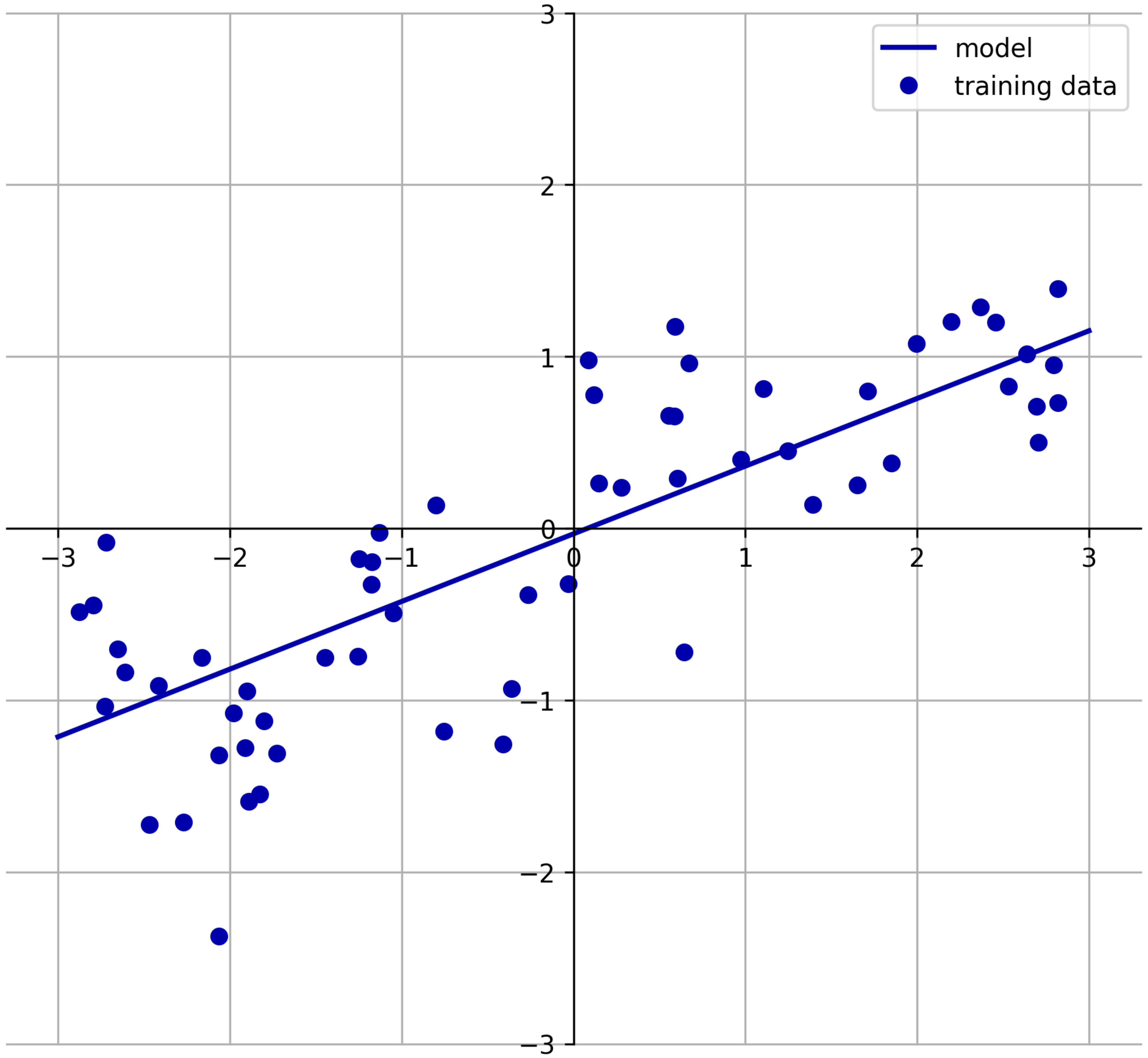
- (Kernelized) **Supported Vector Machines**
- **Neural networks**

Supported vector machines

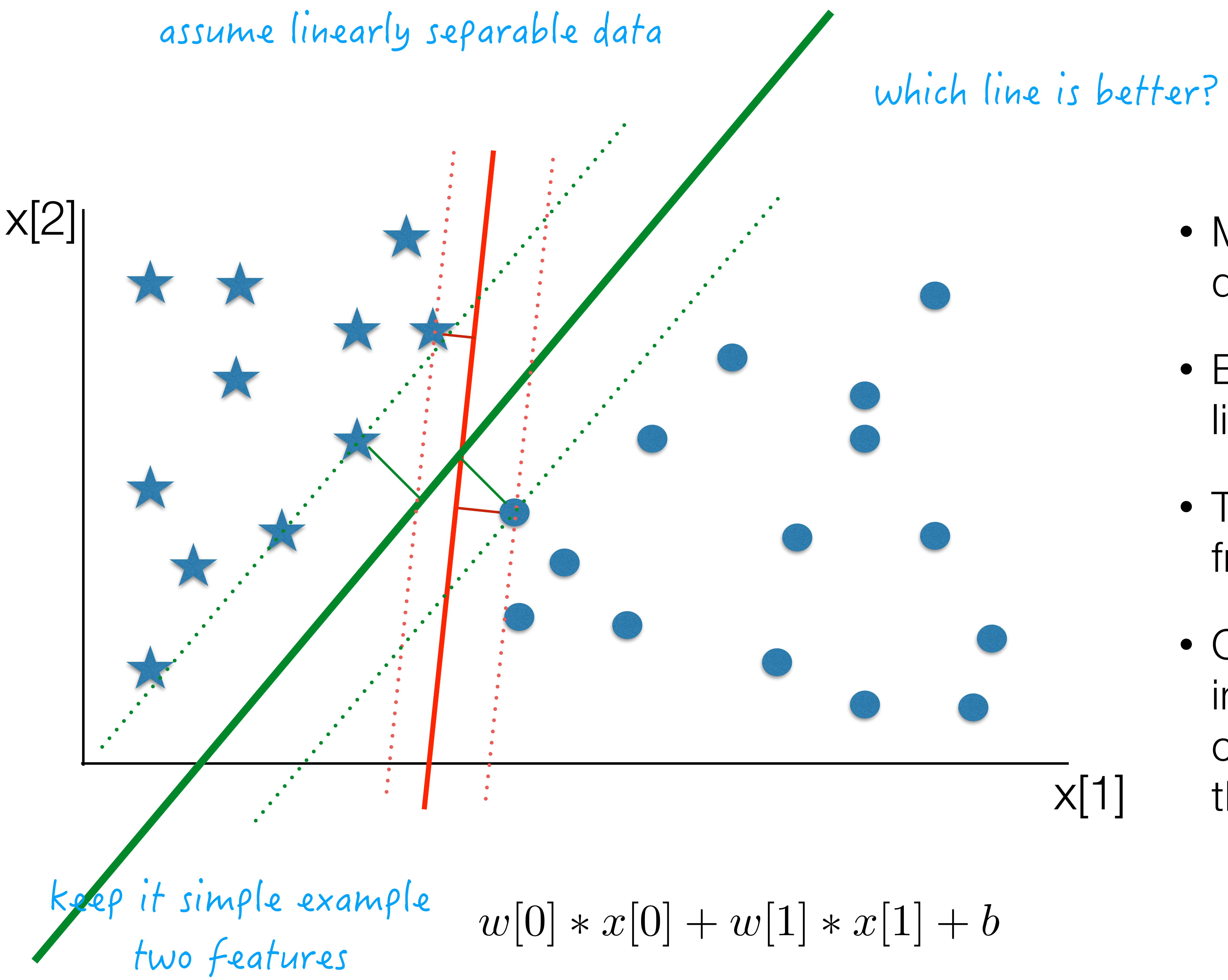
what linear regression does

$$\hat{y} = w[0] * x[0] + b$$





Supported vector machines (SVM) - linear

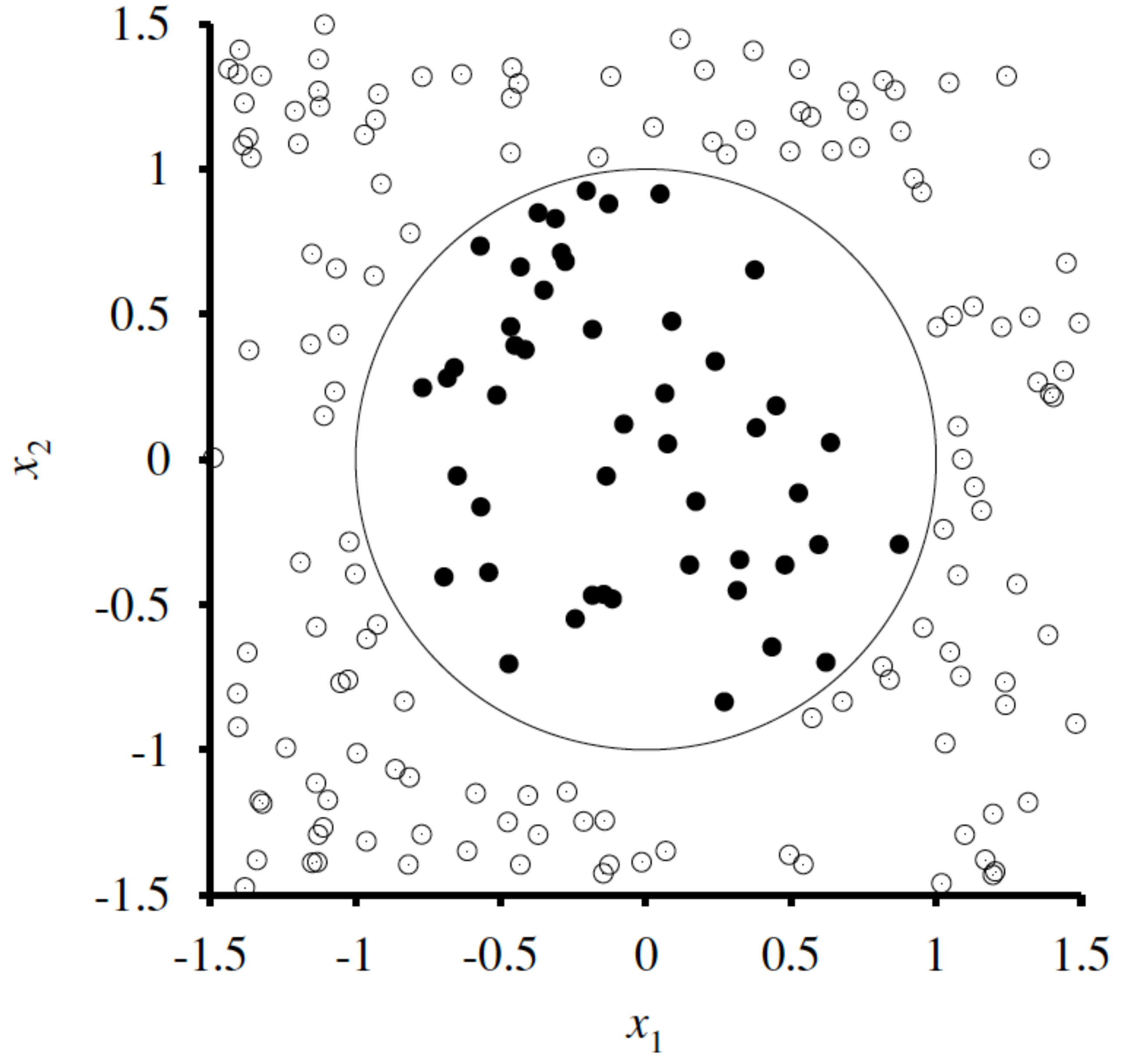


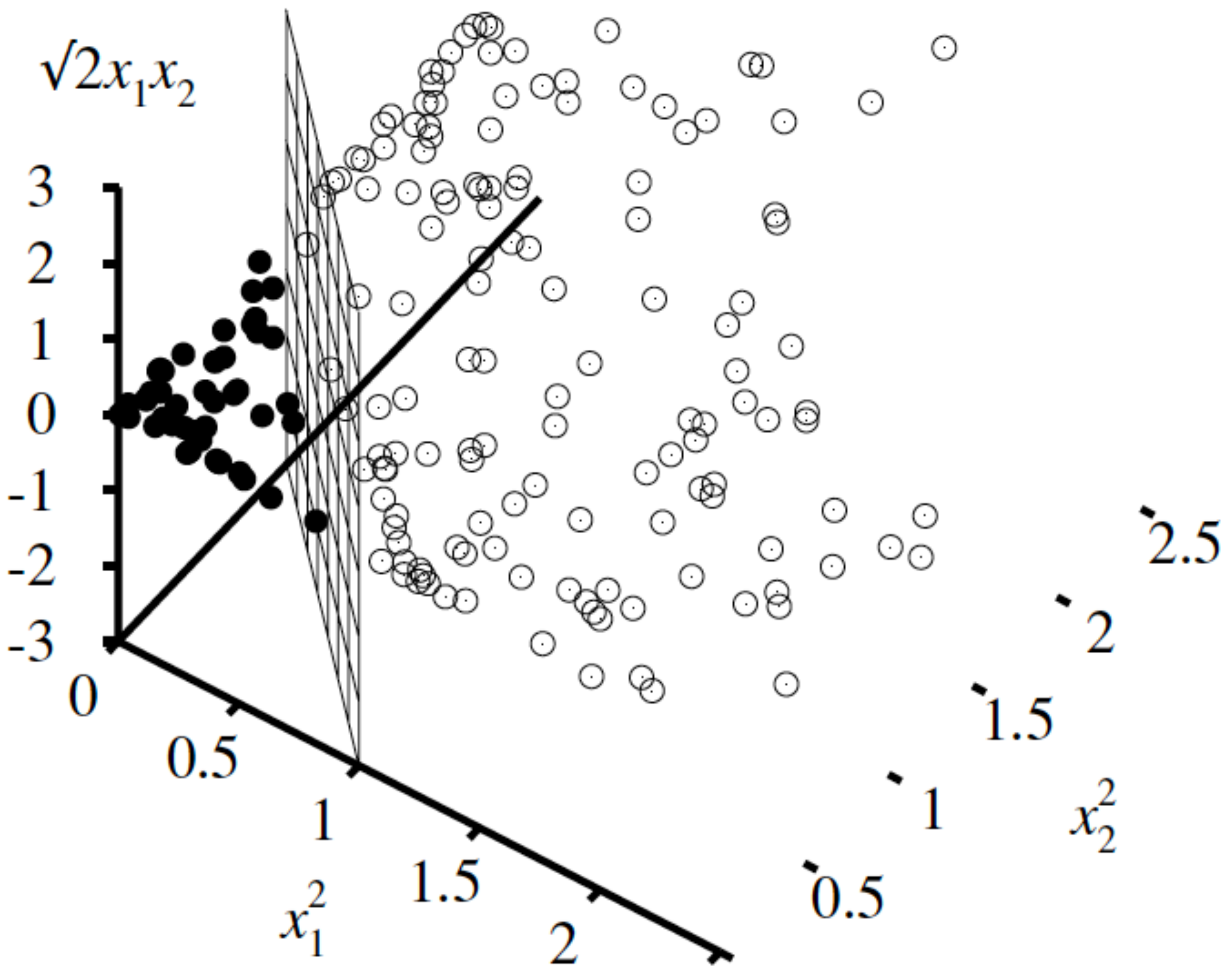
$$w[0] * x[0] + w[1] * x[1] + b$$

- Margin - the distance between the line and the closest data points from each category
- Best line - the one that separates the classes with the line that has the largest margin.
- The margin is calculated as the perpendicular distance from the line to only the closest points.
- Only these points are relevant in defining the line and in the construction of the classifier. These points are called the **support vectors**. They support or define the hyperplane.

Kernelized SVM

what if the data is not linearly separable?





- If data are mapped into a space of sufficiently high dimension, then they will almost always be linearly separable = if you look at a set of points from enough directions, you'll find a way to make them line up.

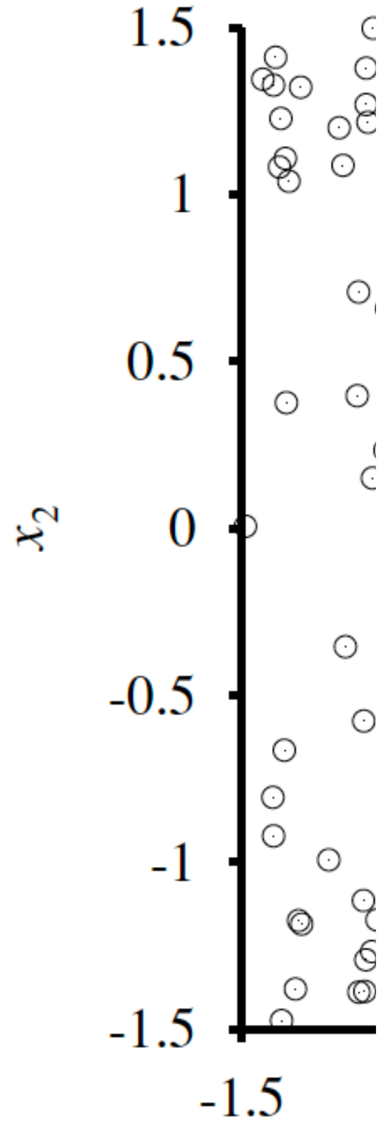
$f_1 = {x_1}^2$ $\quad$ $f_2 = {x_2}^2$ $\quad$ $f_3 = \sqrt{2}x_1x_2$

the function that transforms the feature value (ish) is called a kernel.

- Relation between feature values and classification outcome, no longer clear

Ke

wh



Kernel function

$$\underset{\alpha}{\operatorname{argmax}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k \boxed{K}(\mathbf{x}_j, \mathbf{x}_k)$$

Kernel Function

$$\hat{y} = \operatorname{sign}\left(\sum_j \alpha_j y_j K(\mathbf{x}_j, \mathbf{x})\right)$$

- The kernel is a similarity function, often but not always a distance function
- Mercer's theorem: any **positive-definite** kernel function corresponds to some **feature space** (which can be very large)
- Kernel trick: For all **x** and **x'** in the input feature space $\mathcal{X}$, certain functions can be expressed as an inner product in another feature space $\mathcal{V}$. = replacing $K(x_j, x_k)$ in the equation

f

th

- F

space of
hen they will
parable = if
rom enough
ay to make

Properties of SVM (kernelized)

- Requirement: all features to vary on a similar scale
- For SVM to work, data may need to be preprocessed
- Common preprocessing approach - all features are normalised to values between 0 and 1
- SVM models work regardless of how many features there are (dimensionality of feature space does not matter)
- SVM do not scale very well with the number of samples (Why?)
- SVM models are hard to inspect and difficult to explain why they make a particular prediction

Neural networks vs linear models

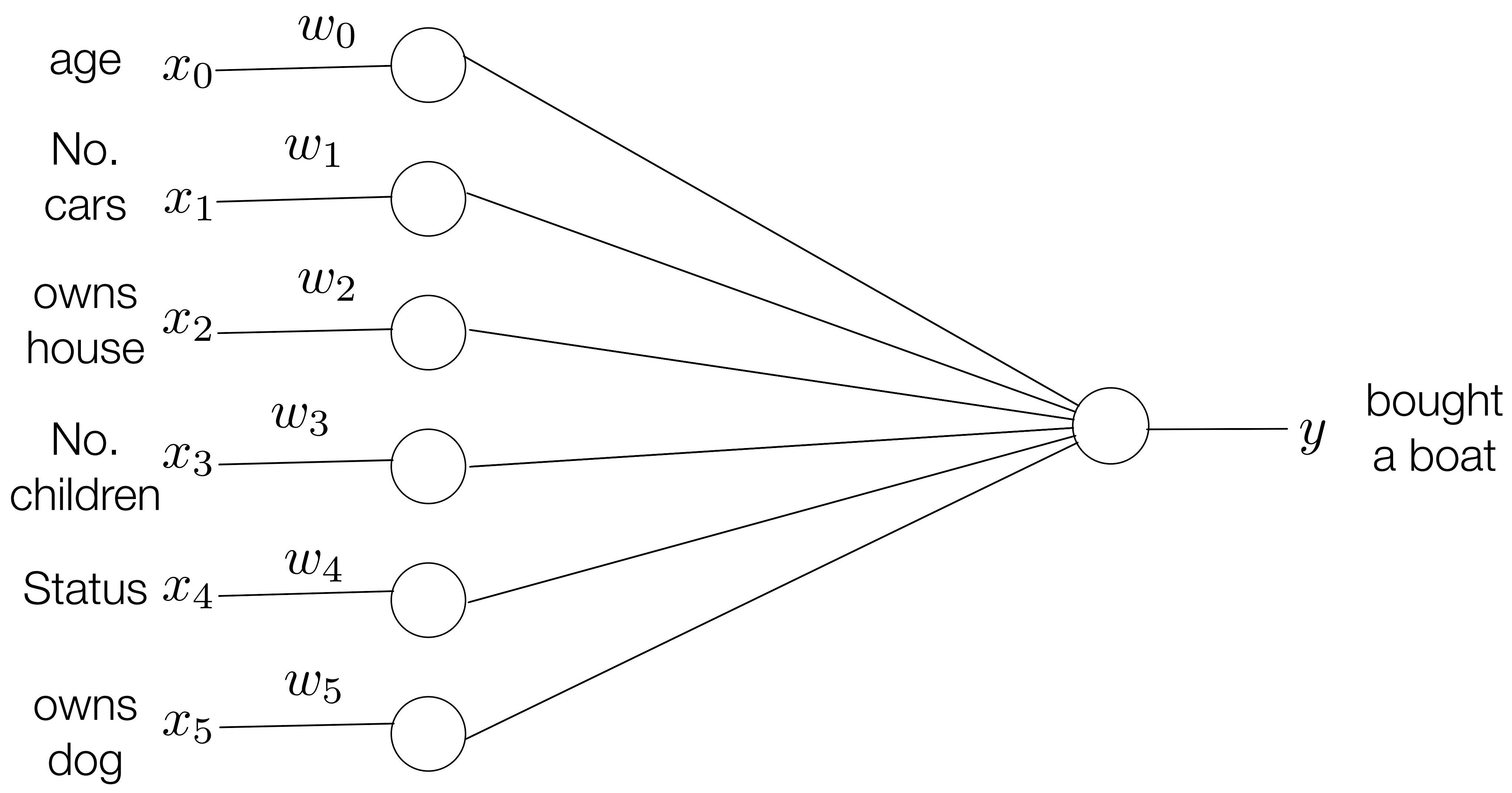
$$w_0 * x_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + w_5 * x_5 = y$$

age	No. cars	owns house	No. children	Status	owns dog	bought a boat

Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes

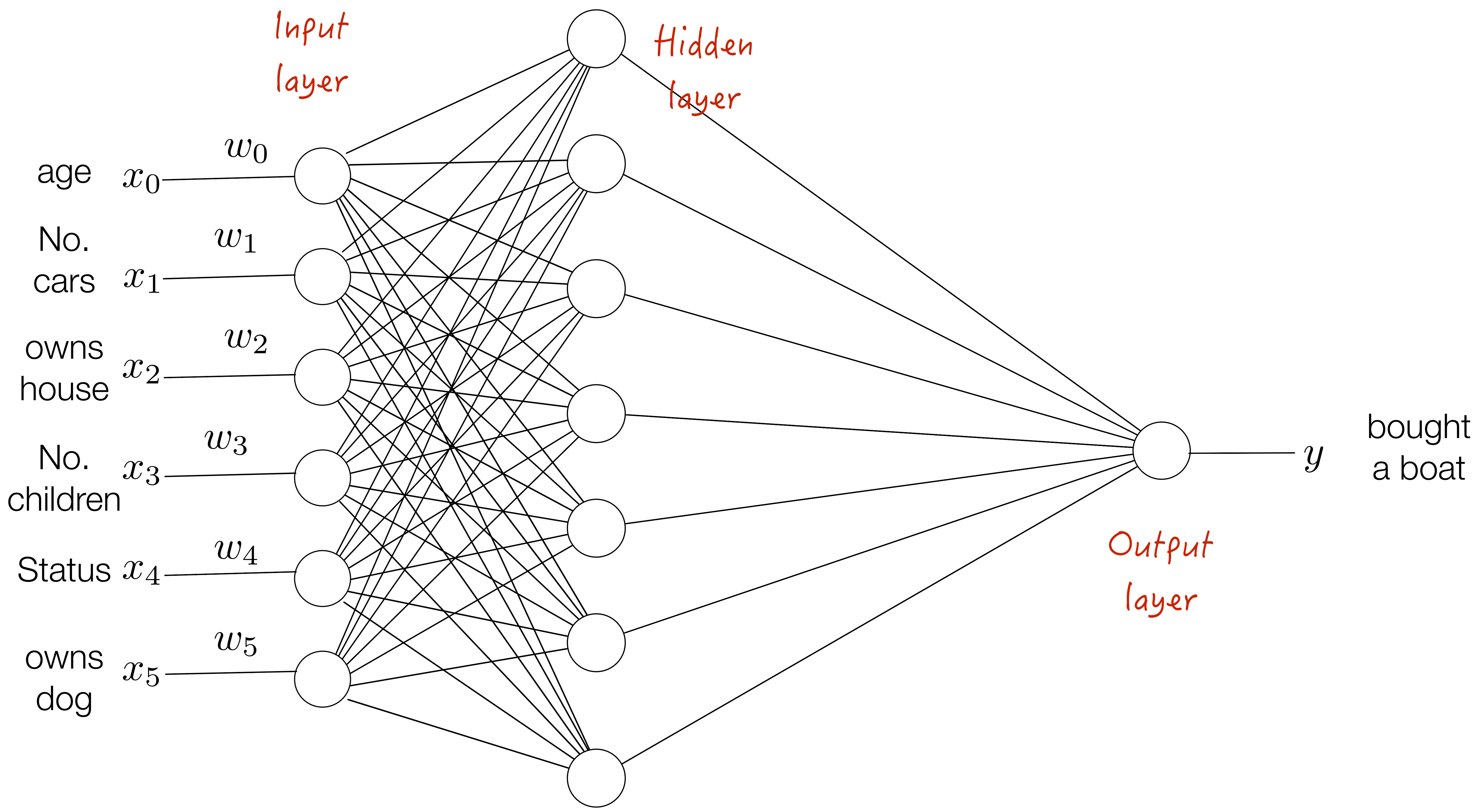
Neural networks vs linear models

$$w_0 * x_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + w_4 * x_4 + w_5 * x_5 = y$$



Neural networks vs linear models

$$w_0 * x_0 + \cdots + w_{54} * x_{54} = y$$



Deep learning
=
Using a neural network
with
a hidden layer

Neural networks

- 1943: McCulloch & Pitts: proposed the first neural model, the Binary-Threshold Neuron.
- 1957: Rosenblatt: exploiting the results by Hebb in 1949, proposes a new model of neuron able to learn from examples, the Perceptron
- 1969: Minsky & Papert: showed strong limitations of the perceptron: the interest on neural networks disappeared for many years
- 1982: Hopfield: developed a neural network capable of behaving as an associative memory.
- 1982: Kohonen: developed a competitive learning model to create Self-Organizing Maps.
- 1983: Barto, Sutton & Anderson: proposed a neural network capable of learning without supervision (Reinforcement Learning).
- 1986: Rumelhart, Hinton & Williams: formalized the process of learning by examples, defining the **Backpropagation algorithm.**
- 1986-2006: On the one hand, BP became very popular in many applications fields (engineering, physics, economy, chemistry, medicine, agriculture science, social science, etc.) as a general learning methodology to fit data sets.
- 2006-2012: LeCun, Bengio, Hinton: found solutions to overcome the difficulties in extending Backpropagation to networks with many layers. Deep learning begins.