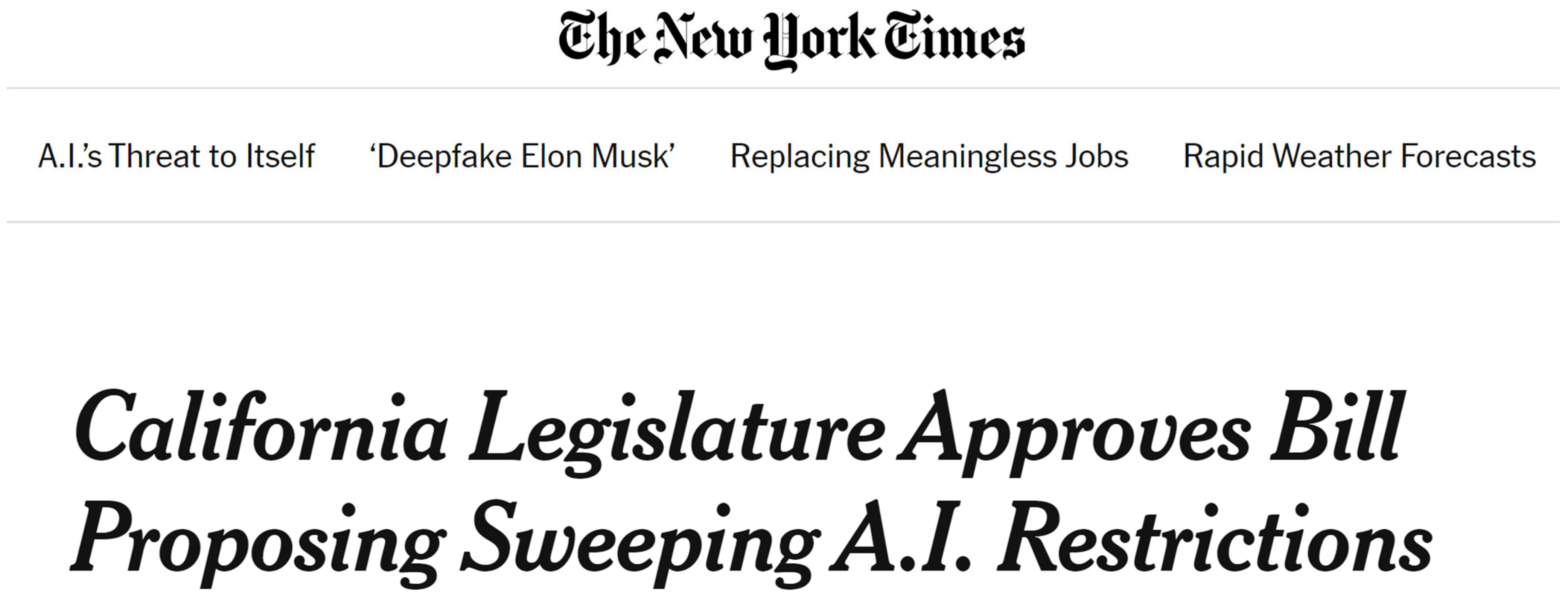
# CS 189/289?

## Today’s headline:


The New York Times

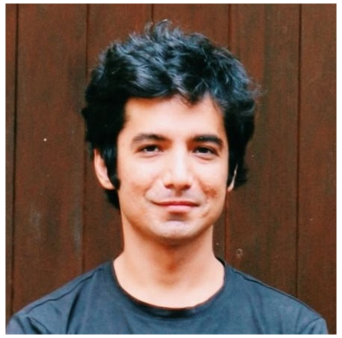
A.I.’s Threat to Itself | ‘Deepfake Elon Musk’ | Replacing Meaningless Jobs | Rapid Weather Forecasts

**California Legislature Approves Bill Proposing Sweeping A.I. Restrictions**

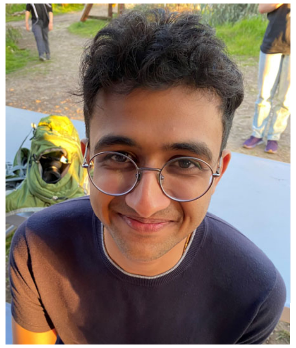
# CS 189/289

Today's lecture:

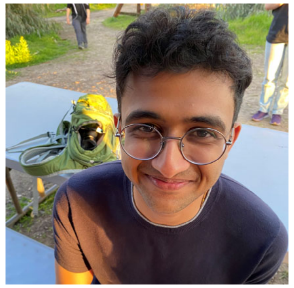
1. General Info (also read www.eecs189.org in full)
2. Introduction to ML


Prof. Saremi

Prof. Listgarten


Head TA: Suchir Agarwal

# Teaching Assistants



**Head TA**
**Suchir Agarwal**
4th Year BA
Interests: computation biology, time series forecasting

**Grace Luo**
3rd Year PhD
Interests: CV—multimodal vision and language models

**James Ni**
4th Year BS
Interests: RL & robotics

**Pierre Boyeau**
5th Year PhD
Interests: single-cell genomics

**Ruchir Rastogi**
5th Year PhD
Interests: genomics

## Class was expanded above room size

1. For the first year, we will regularly release class lecture videos on the course website.
2. All discussion sections will be in person.
3. NO ALTERNATE EXAM TIMES will be given other than what is listed as the official final exam date and time, other than DSP which gets an extended (but overlapping) version.
4. EXAMS MUST BE TAKEN IN PERSON.
5. “Ed” post with detailed exam policy coming soon.

## Midterm

Wednesday 10/16 from 7-9pm (see course website for rooms)

(DSP will overlap with this)

## Enrollment

- The class is full.
- There are no instructor-allocated enrollment codes.
- Time conflicts not allowed, prevents enrollment.
- 56/60 CE slots have been enrolled, standby for remaining 4 open slots.
- When people drop, some waitlist folks will be enrolled by the department (not by us).
- If you are on waitlist: complete all assignments as though you were enrolled.
- Questions: email cs-enrollments@berkeley.edu, or EECS enrollment drop-in office hours (look on-line for dates and times).

## Gradescope & Ed

**Ed** - https://edstem.org/us/join/RUHntB

**Gradescope** - https://www.gradescope.com/courses/814736

## Auditing

- As we are over classroom capacity, auditors should not come in-person to lectures unless they wait outside for lecture to start, and there are empty seats available.
- Can find videos and homeworks from course web page.

## Contact information

- Read www.eecs189.org before initiating any contact.
- Most questions should go to a **public Ed post first**.
- If needed, **private post on Ed** to instructors.
- Professors will have office hours after each lecture (with the professor who has been lecturing), as noted on course web page (see calendar—subject to change ).
- DSP: Krystle Simon (krystle@eecs.berkeley.edu)

## Pre-requisites

- Courses you should have taken (or equivalent):
  EE16A & EE16B & CS70 & MATH53
- Some core areas needed: linear algebra, vector calculus, probability, programming skills.
  - Discussion 0 and Homework 1 will be indicative of whether you have the necessary math knowledge to succeed in the class.

## Final grade breakdown

CS 189 & 289

- Homework: 20%
- Midterm: 35%
- Final: 45%

CS 289A will be distinguished by having extra exam questions. (No class project).

## Homeworks

- Combination of pen and paper & coding (python).
- Total of 7 homeworks.
- A subset of problems will be graded.
- See website for specific due dates.
- HW 1 is already released. Due on 9/11 at 11:59 PM.

## Homework niceties

- No late homework accepted, but…
- Can drop your two lowest homework grades.
- 80% correctness gives you a full grade on each homework.
- Intention: to encourage you to learn the material.

## Collaboration

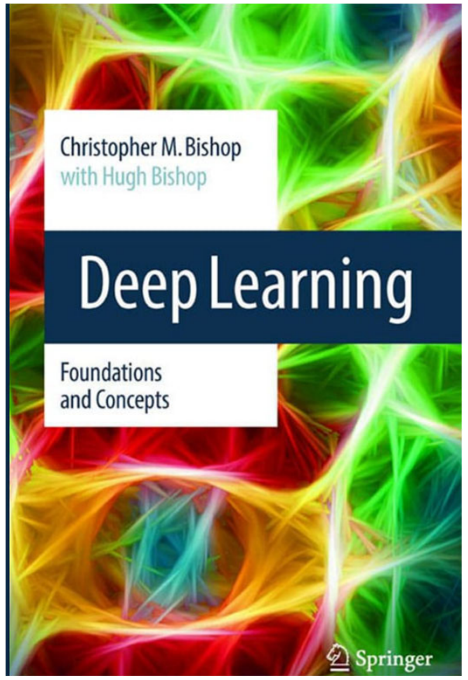
- Encouraged to work on homework problems in study groups of 2-5 people. These people must be named in your write-up.
- Must always write up the solutions on your own.
- You are not permitted to look at (or show) anyone's final written solution, even for members of your own study group.
- You may use books or online resources (not solutions from previous terms, etc.) to help solve homework problems, but you must always credit all such sources in your writeup and you must never copy material verbatim.

## Cheating

- Zero-tolerance policy for cheating.
- No online communication outside of Ed to discuss assignment content (other than formally *listed* collaborators).
- Looking at online solutions from previous semesters or other students is forbidden. Staff and software will specifically be watching for this.
- Sharing of your solution with others is forbidden.
- Full details on course website.

## Materials you are responsible for

- All Lectures and Discussions.
- Slides will be posted each lecture (course homepage)
- NEW: Assigned textbook readings:
  - Although we will not be following this textbook, nor covering all of its material, we will assign readings from it.
  - https://www.bishopbook.com/ from which a free on-line version is available.
  - Purchase from Springer, Amazon, etc.



# CS 189/289

Today's lecture:

1. General Info (also read [www.eecs189.org](http://www.eecs189.org) in full)
2. Introduction to ML

- assigned textbook reading 1.1-1.2.4 inclusive
  (do readings before the next lecture)

## Examples of learning problems

1. Recognizing digits in images.
2. Identifying tumor in an xray.
3. Classifying email as spam or not.
4. Diagnosing disease from symptoms.
5. Predicting the price of a stock 6 months from now.
6. Predicting the 3D structure of all the atoms in a protein from it’s amino acid sequence.

# Examples of learning problems

7. Predicting selling price for individual houses.
8. Netflix problem – predict  rating of a movie *j* by a customer *i*.
9. Determining credit-worthiness for a mortgage or a credit card transaction
10. Predicting 3D forces on a molecule from molecular formula.

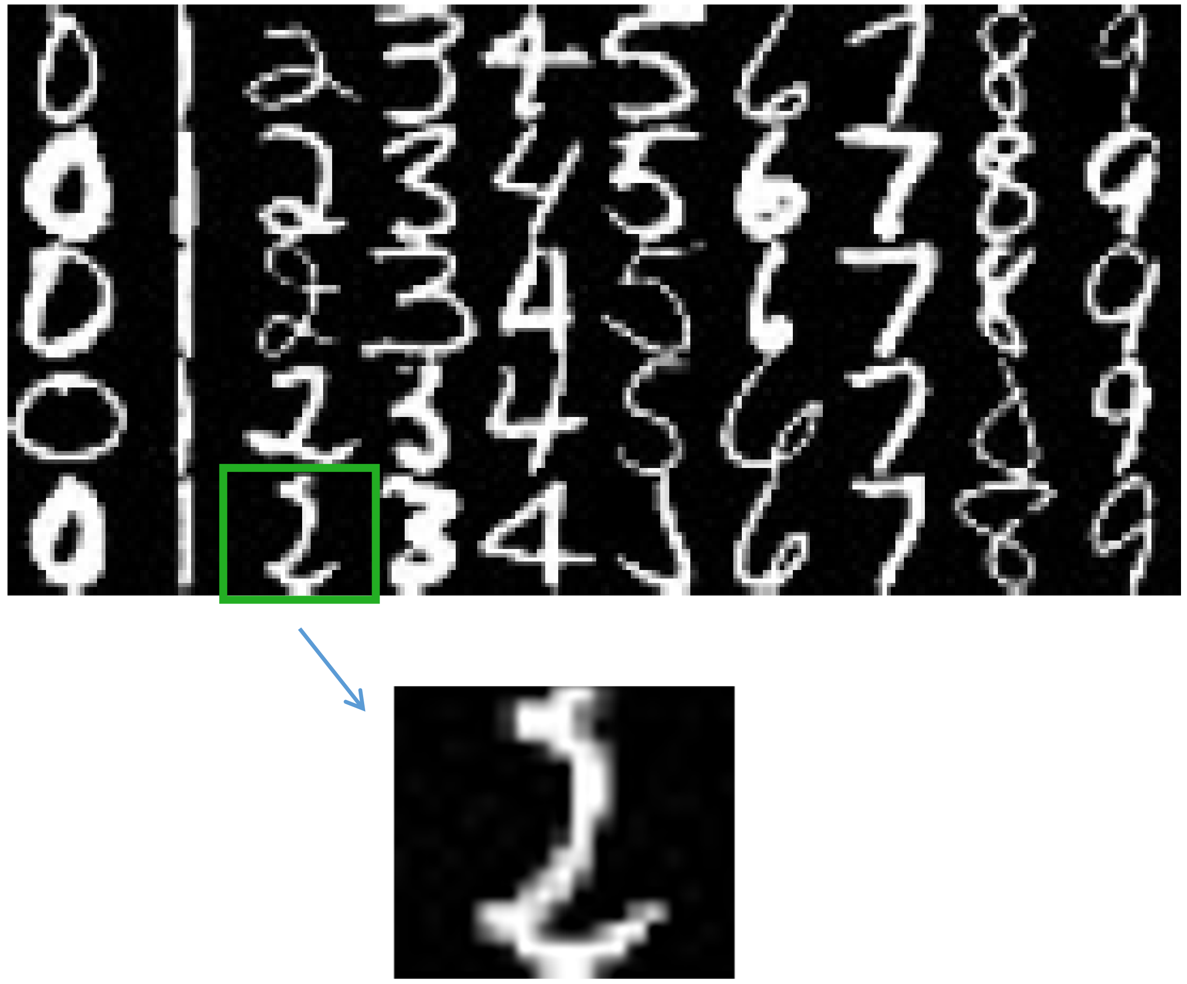
# Types of learning problems

- *Classification* problems, where the output is a label from a finite set (in the simplest case, just binary).
- *Regression* problems, where the output is real-valued.
- *Ranking* problems, where the output only ranks examples relative to one another.
- *Unsupervised* problems, such as ChatGPT, often used for *generative AI*.

*Question: what are we learning in each case?*

# Handwritten digit recognition on MNIST

- Has a *training set* of 60K examples, and a *test set* of 10K examples.
- Each digit is a 28x28 pixel grey level image.
- Error rates of best systems are under 0.5%



## The machine learning approach to object recognition

### Training time

- Compute feature vectors for positive and negative examples of image patches
- *Train* a classifier

### Test Time

- Compute feature vector on image patch
- *Evaluate* classifier

?

## The machine learning approach to object recognition

### Training time

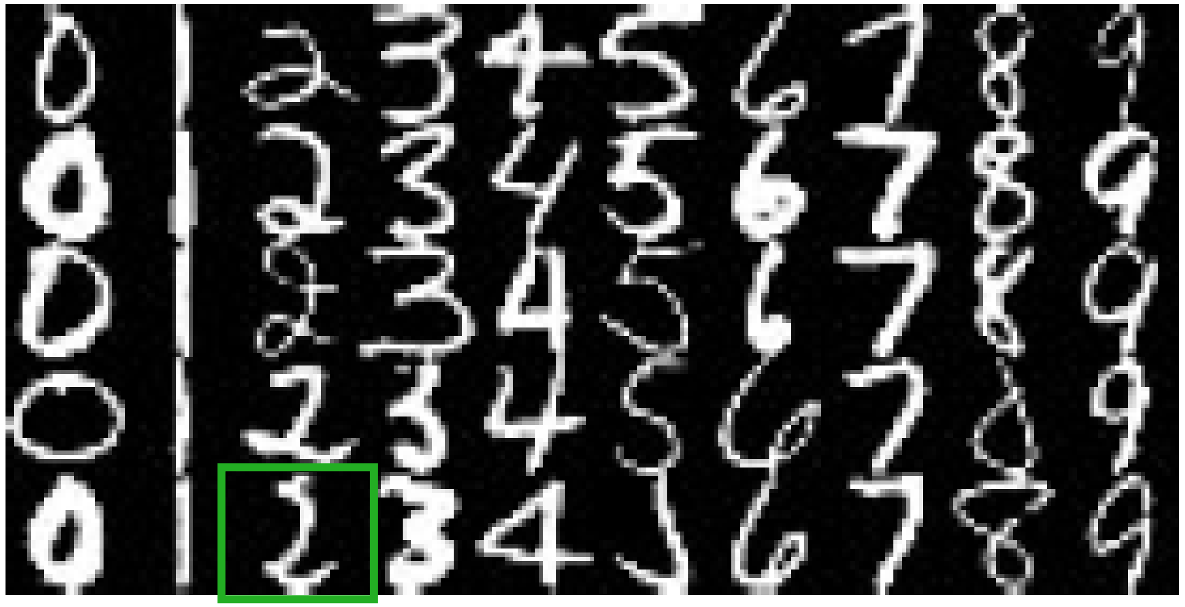
- Compute feature vectors for positive and negative examples of image patches
- *Train* a classifier

### Test Time

- Compute feature vector on image patch
- *Evaluate* classifier

### Deployment! (use this model for intended purpose)

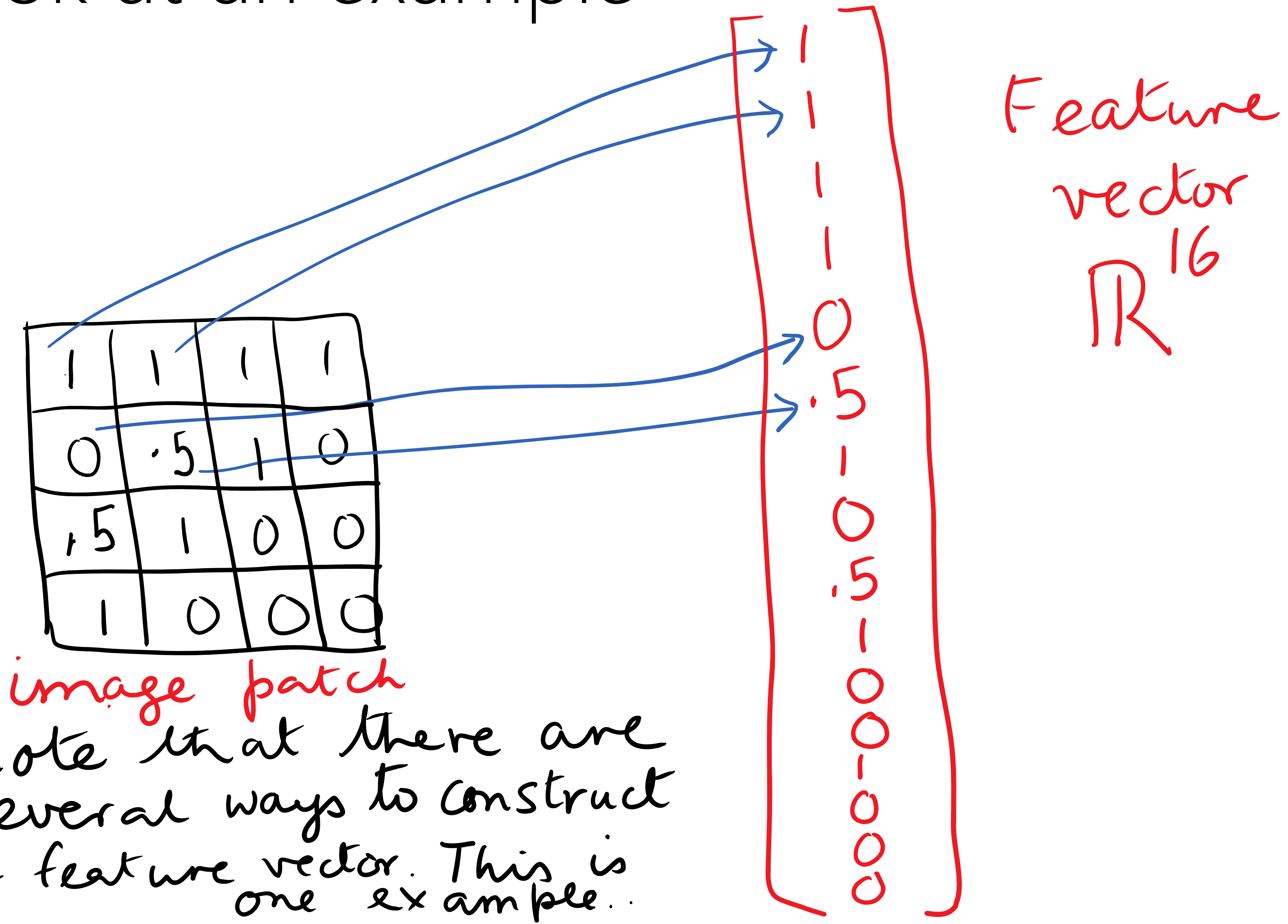
# Let's look at an example



| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | .5 | 1 | 0 |
| .5 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |

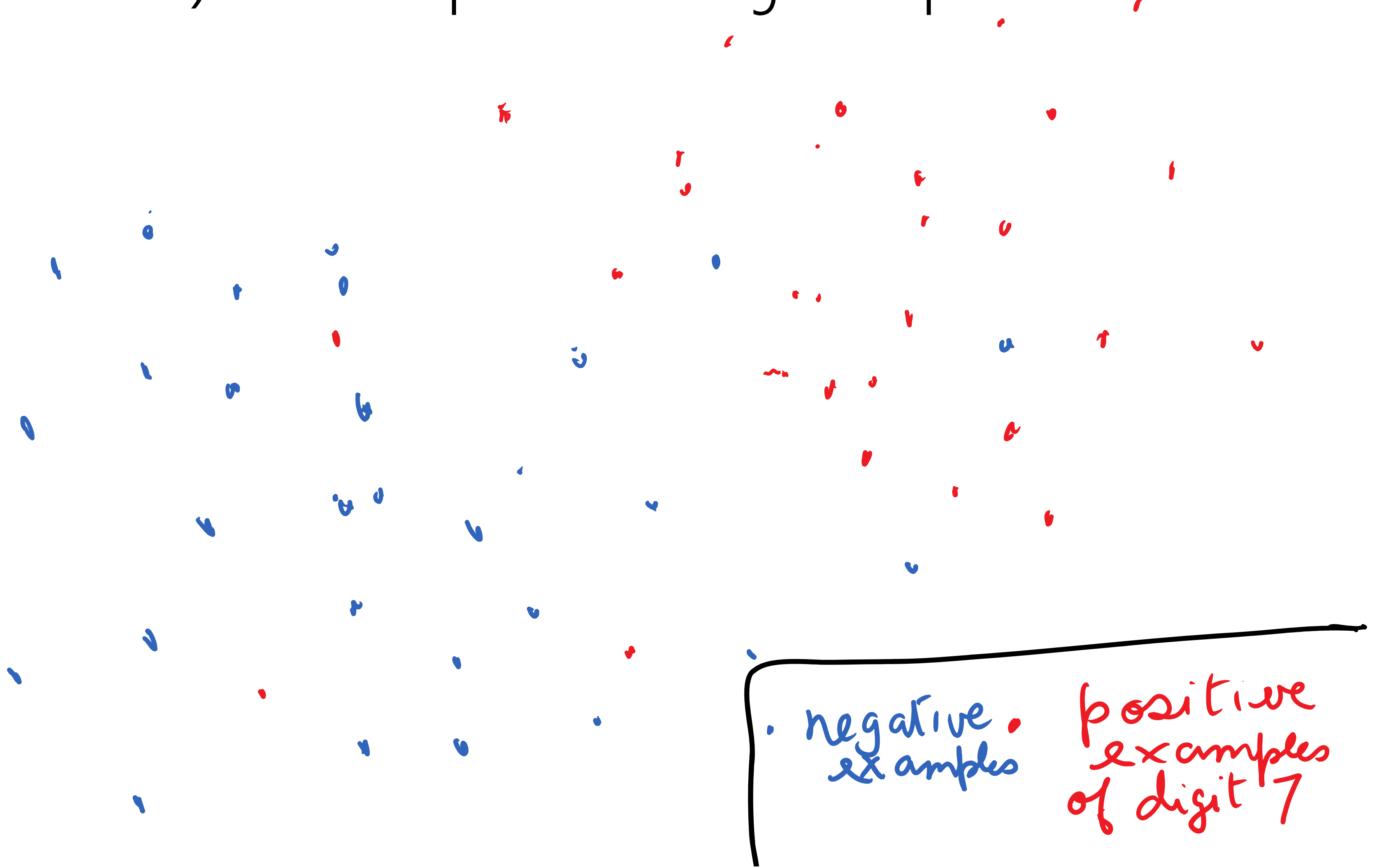
image patch

## Let's look at an example

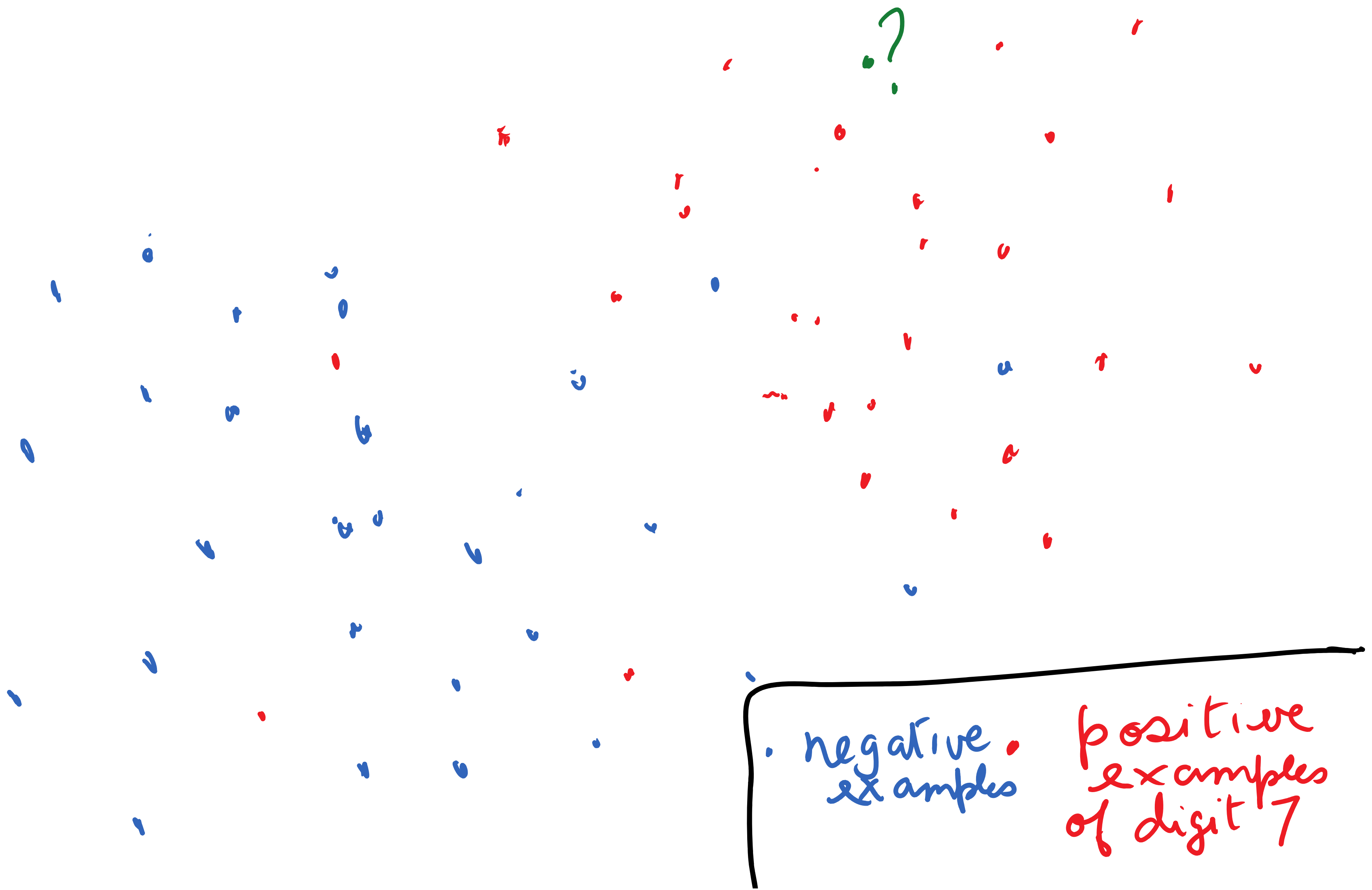


Note that there are several ways to construct a feature vector. This is one example..

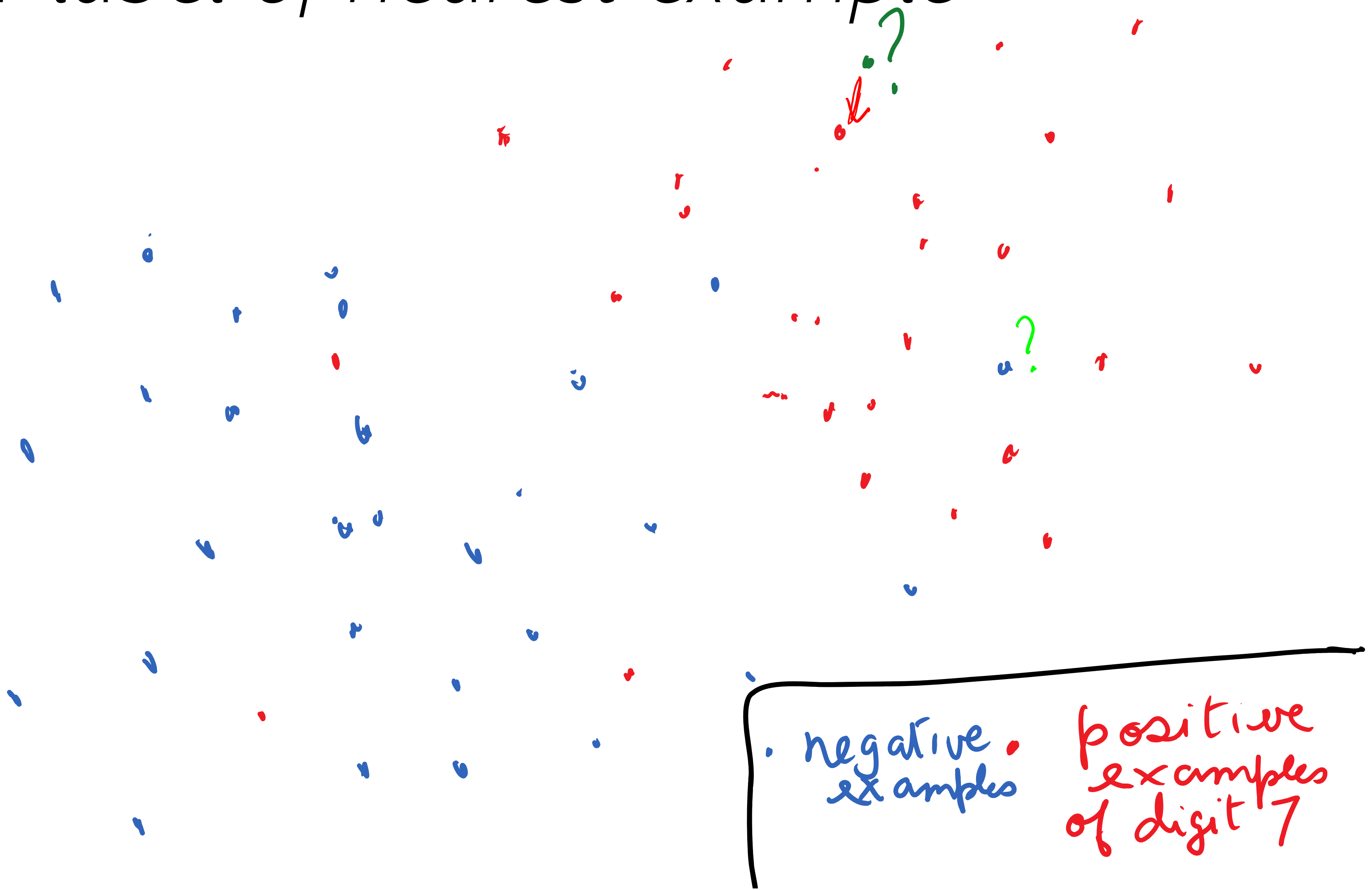
# In feature space, positive and negative (e.g. of class "7") examples are just points…



## How do we classify a new point?



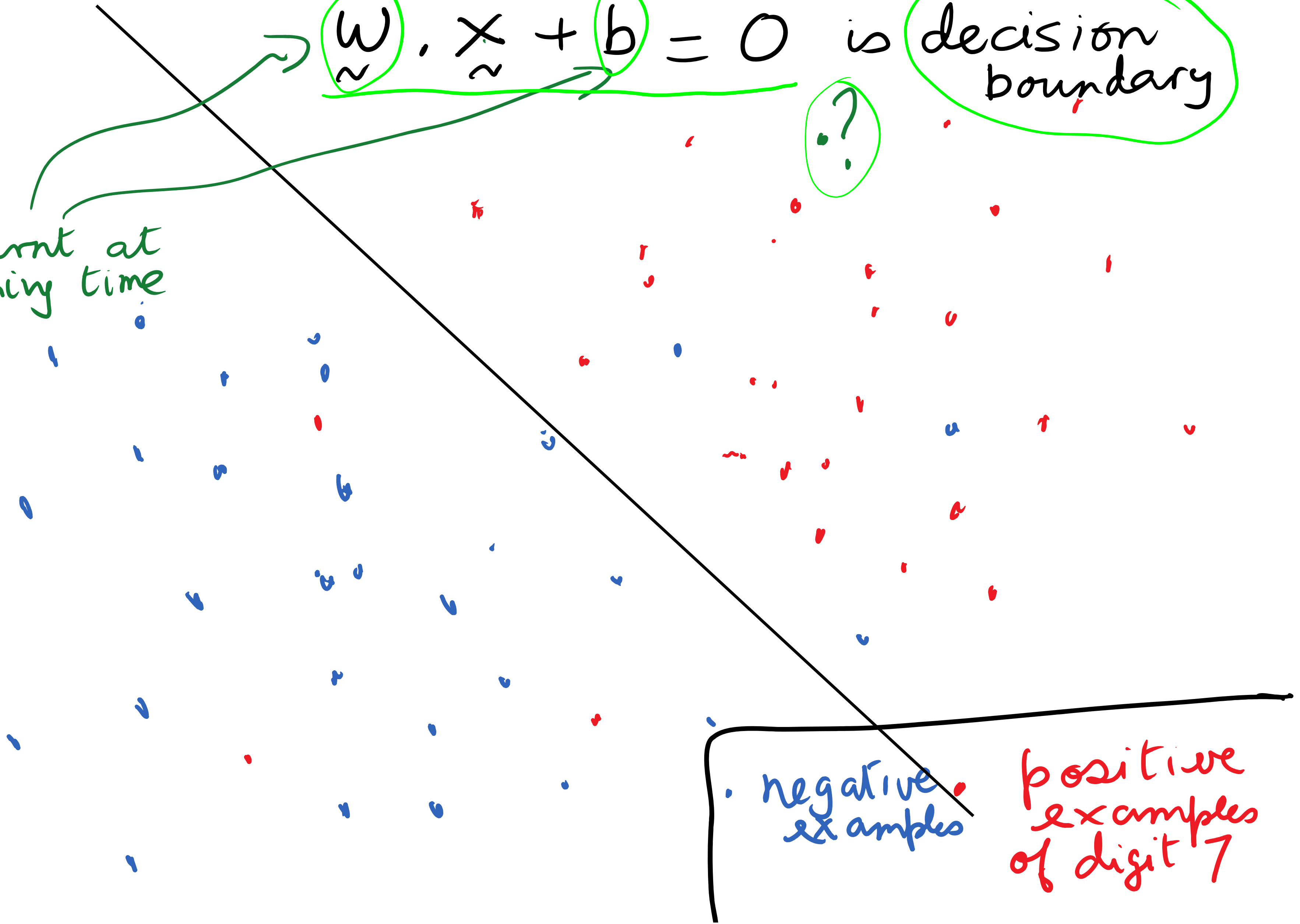
## Nearest neighbor rule:
## *"transfer label of nearest example"*



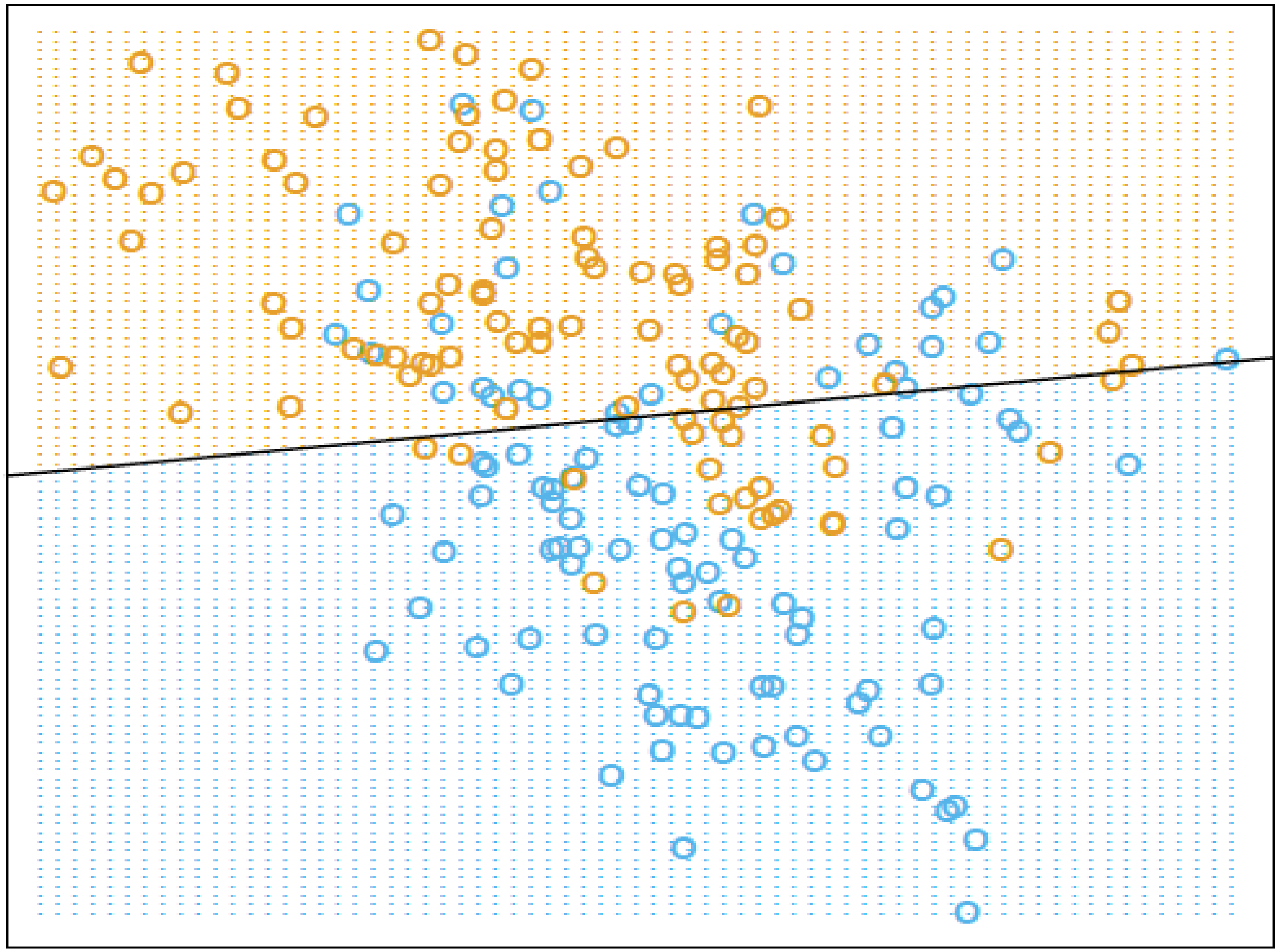
## Linear classifier rule

$\underset{\sim}{w} \cdot \underset{\sim}{x} + b = 0$ is decision boundary ?

learnt at training time



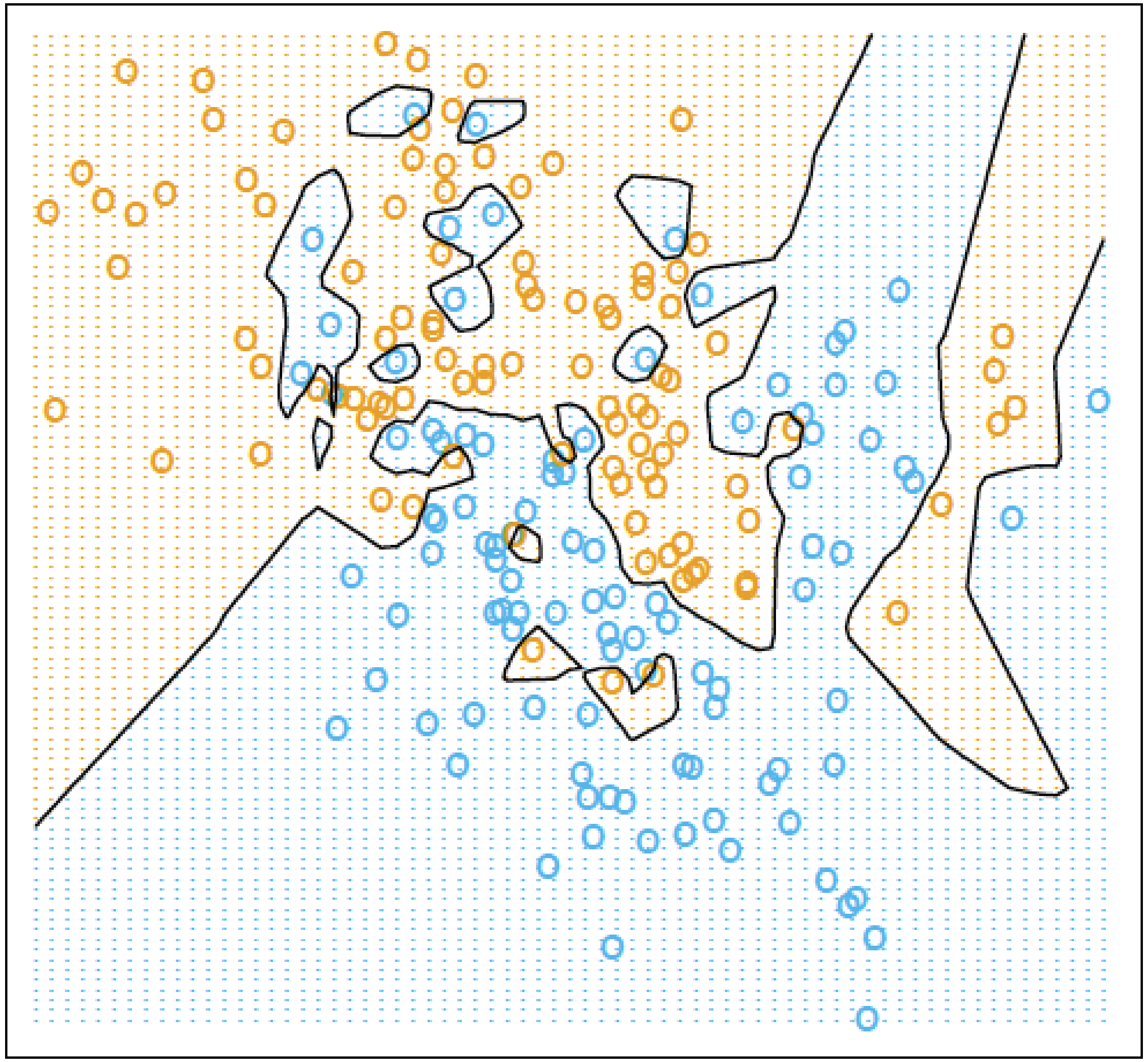
# Linear Classifier



[Source : Hastie, Tibshirani, Friedman]

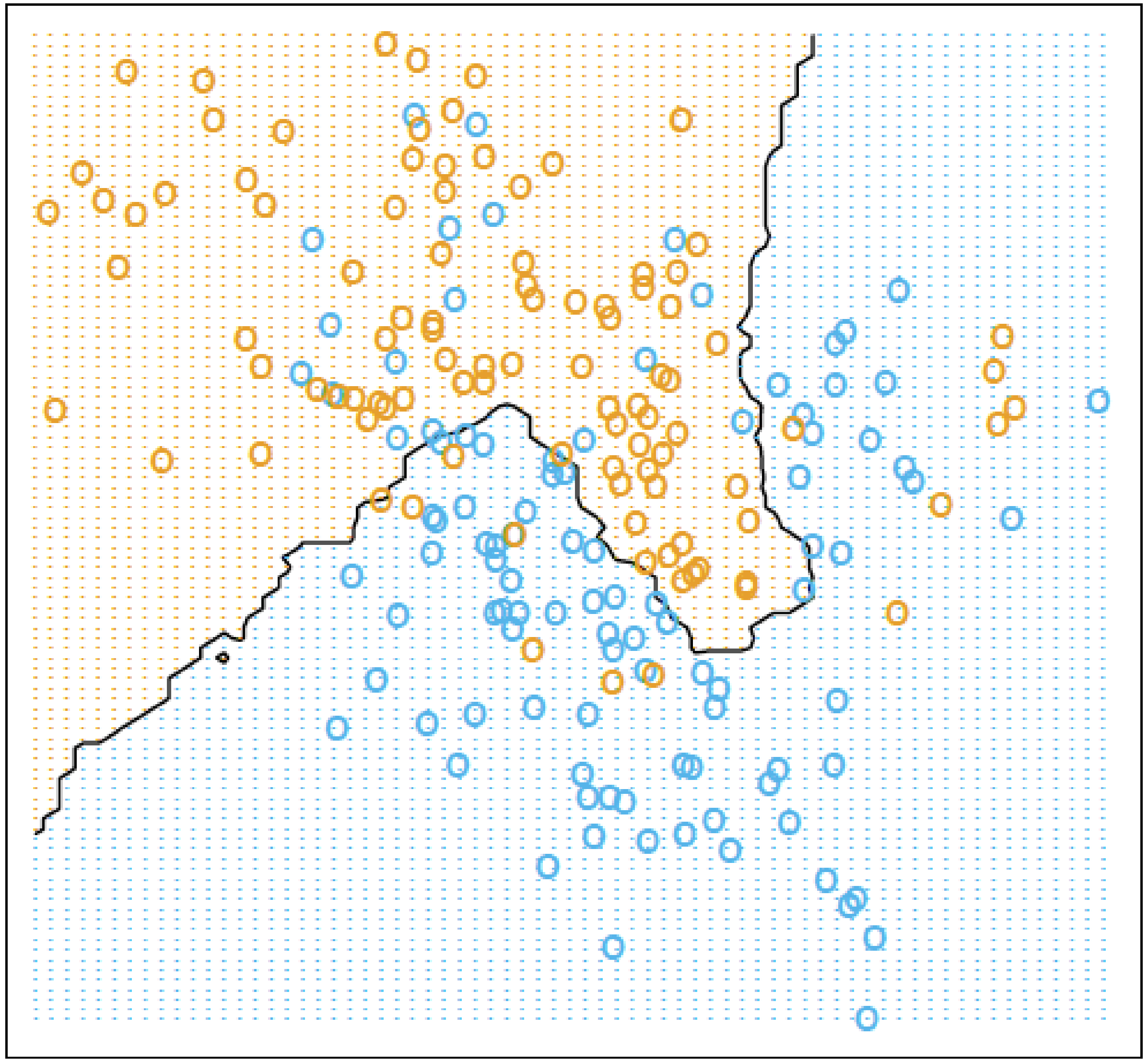
## How do we get non-linear decision boundaries?

# 1-Nearest Neighbor



[Source : Hastie, Tibshirani, Friedman]

# 15-Nearest Neighbor

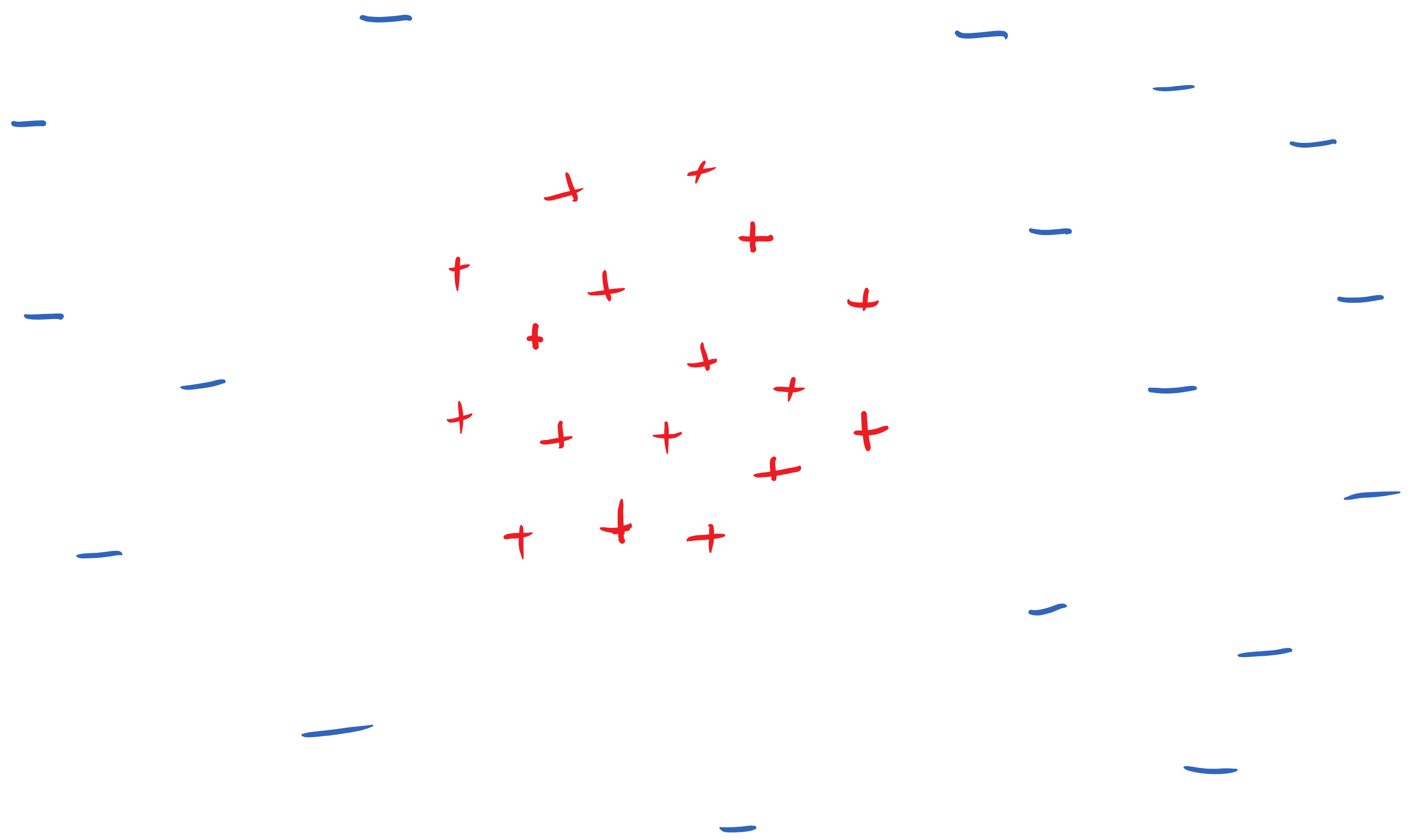


[Source : Hastie, Tibshirani, Friedman]

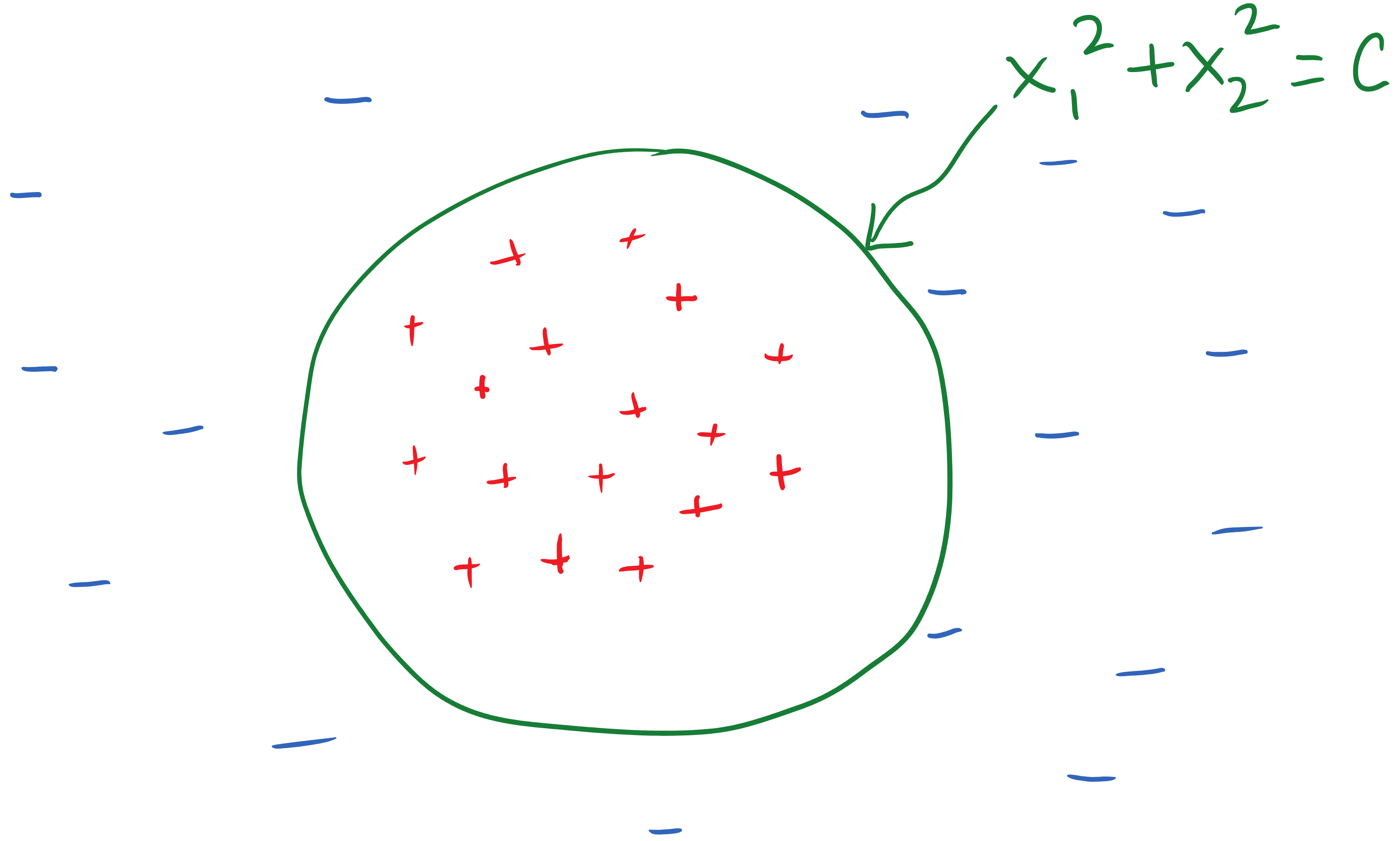
## How do we get non-linear decision boundaries?

- Nearest neighbors
- Multilayer perceptrons *aka* Neural Networks

## Suppose the positive examples lie inside a disk



## Suppose the positive examples lie inside a disk

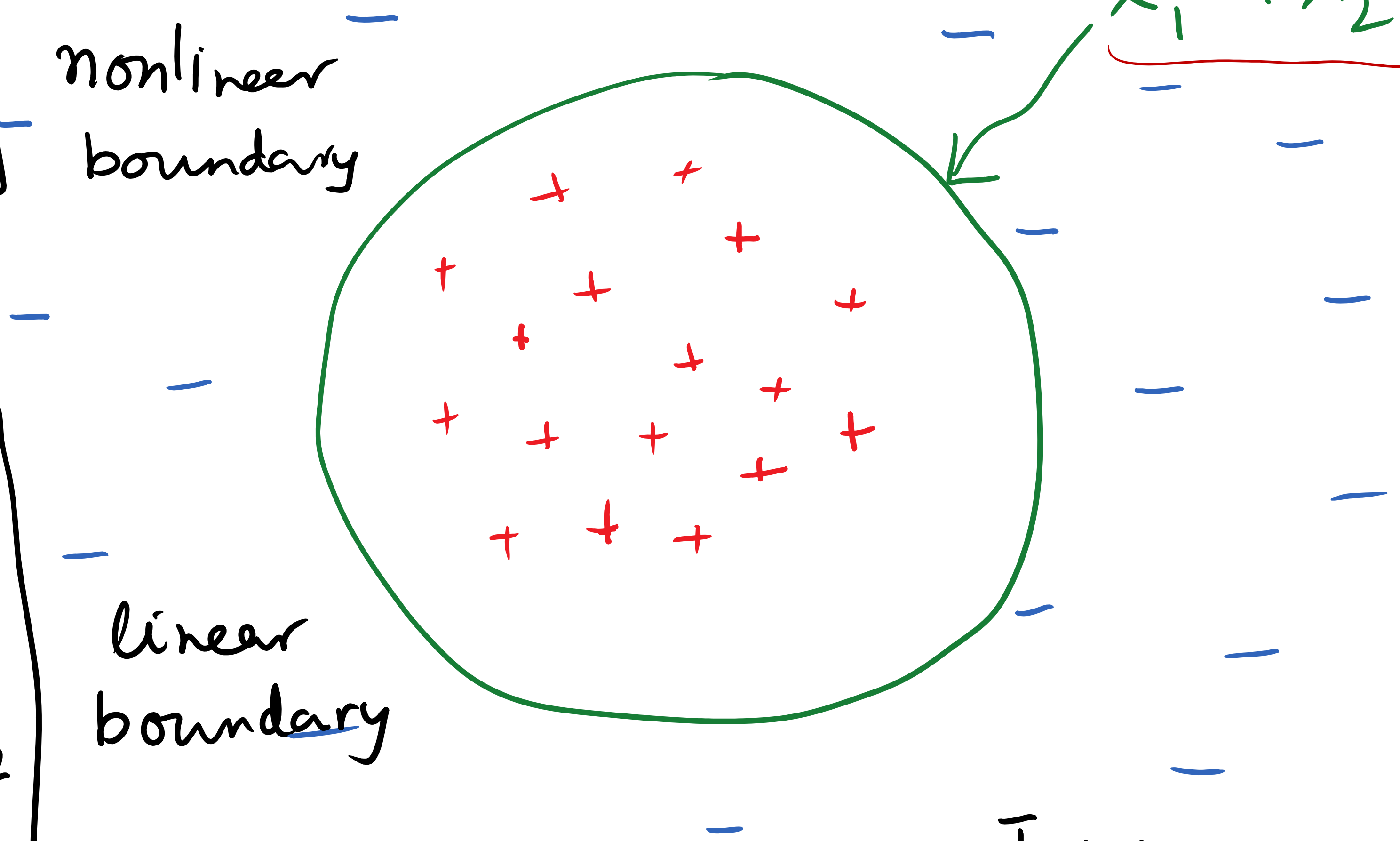


# We can construct a new higher-dimensional feature space where the boundary is linear

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

nonlinear boundary

$$x_1^2 + x_2^2 = C$$



$$\begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

linear boundary

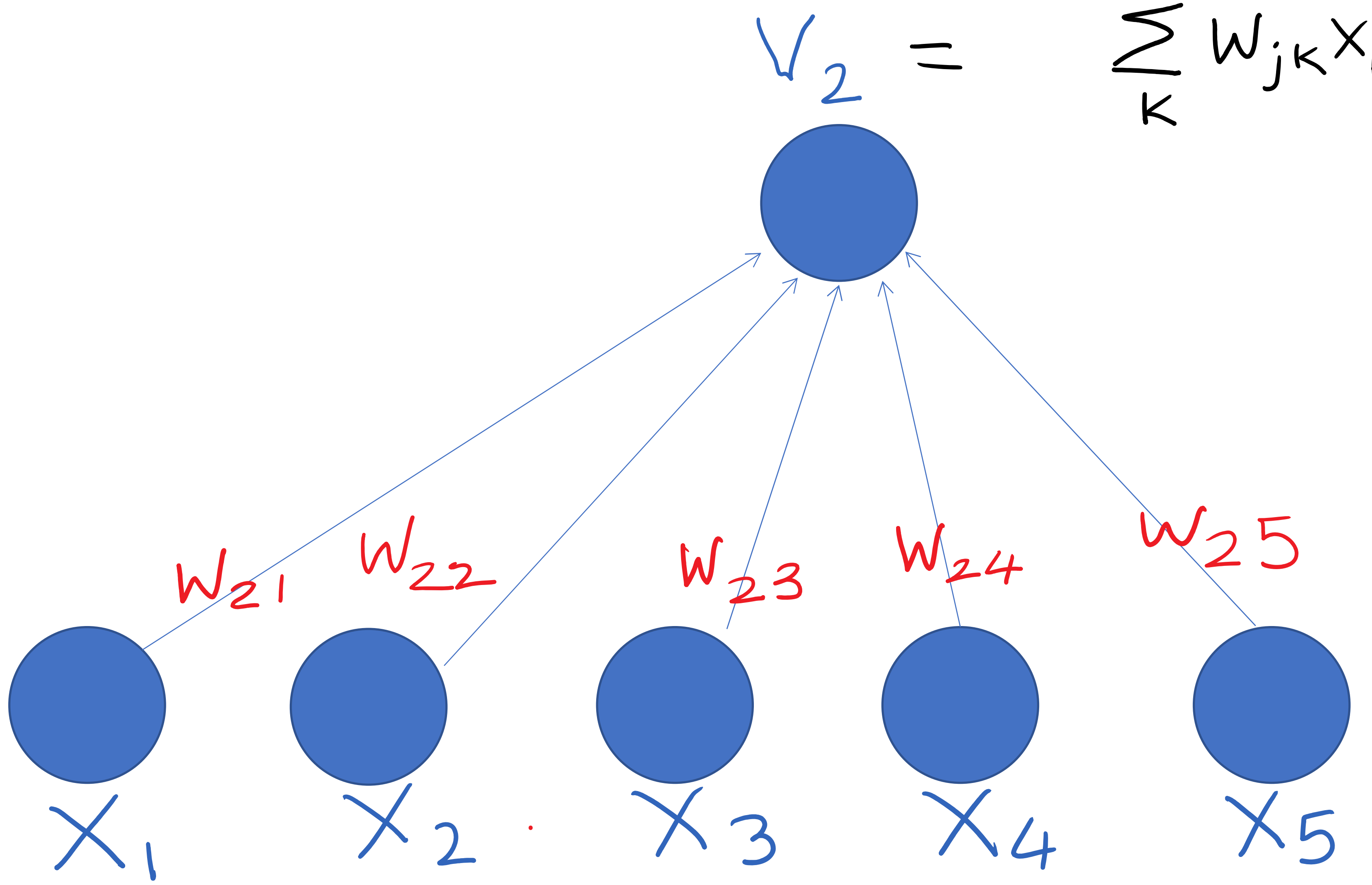
$$\underset{\sim}{x} \longrightarrow \underline{\Phi(\underset{\sim}{x})}$$

## Neural Networks

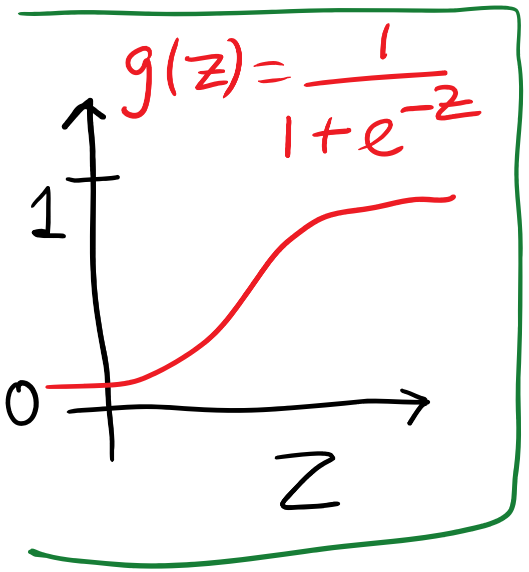
“composing simple (logistic) regression functions over and over again”

# Single-layer, single-output neural network

$$V_2 = \sum_k w_{jk} x_k$$

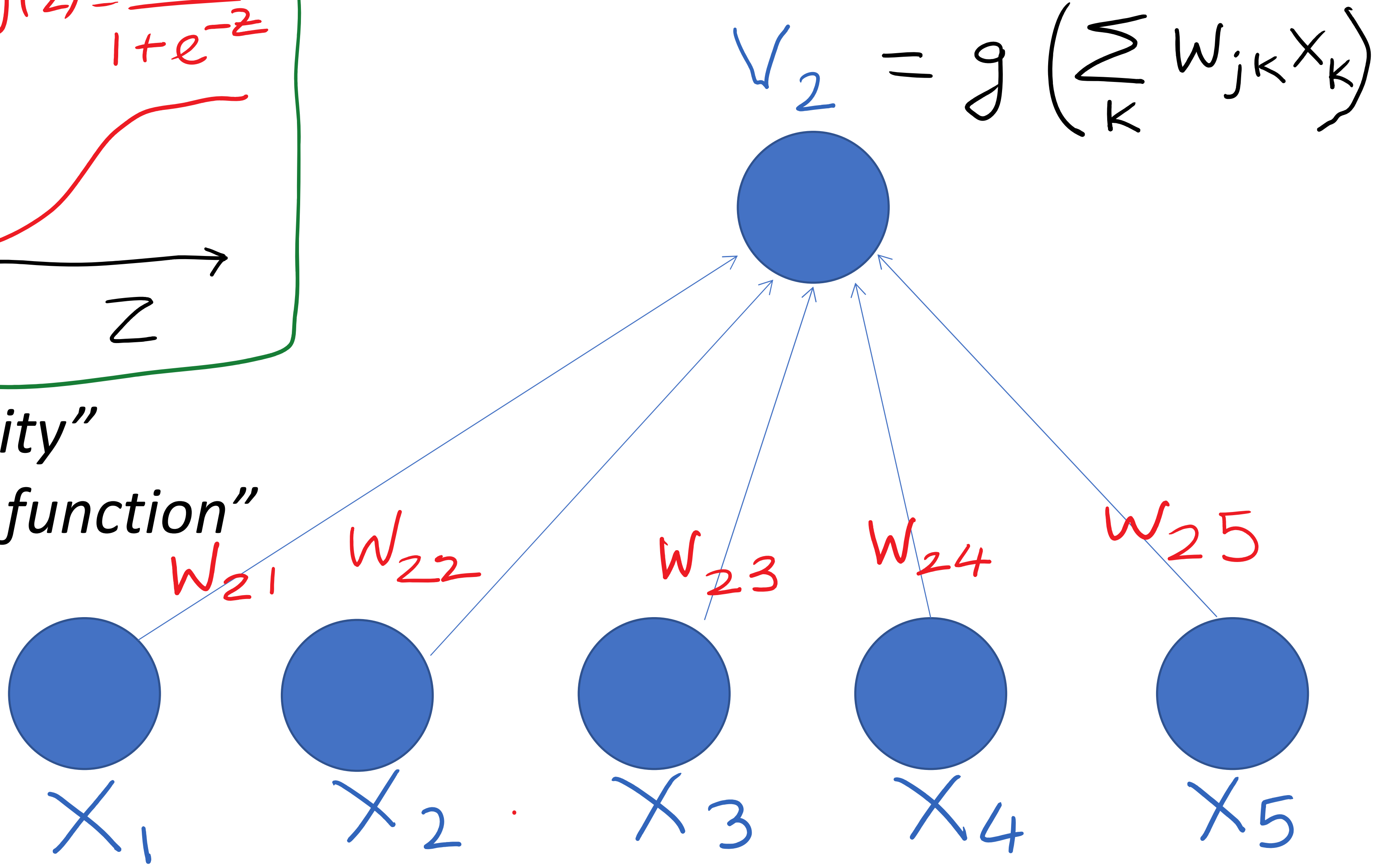


# Single-layer, *single-output* neural network



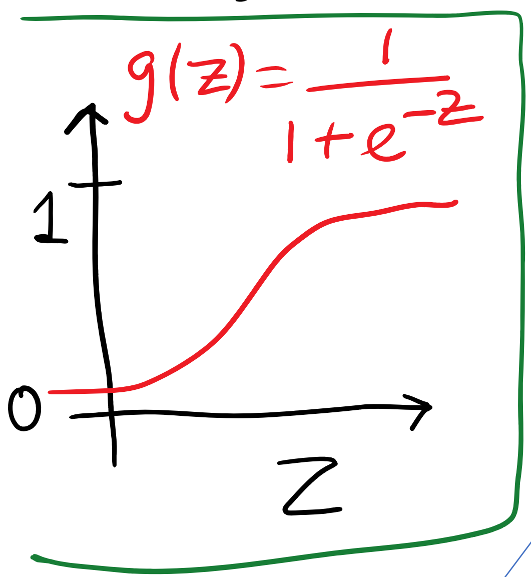
*"non-linearity"*

*"activation function"*



# Single-layer, *multi-output* neural network

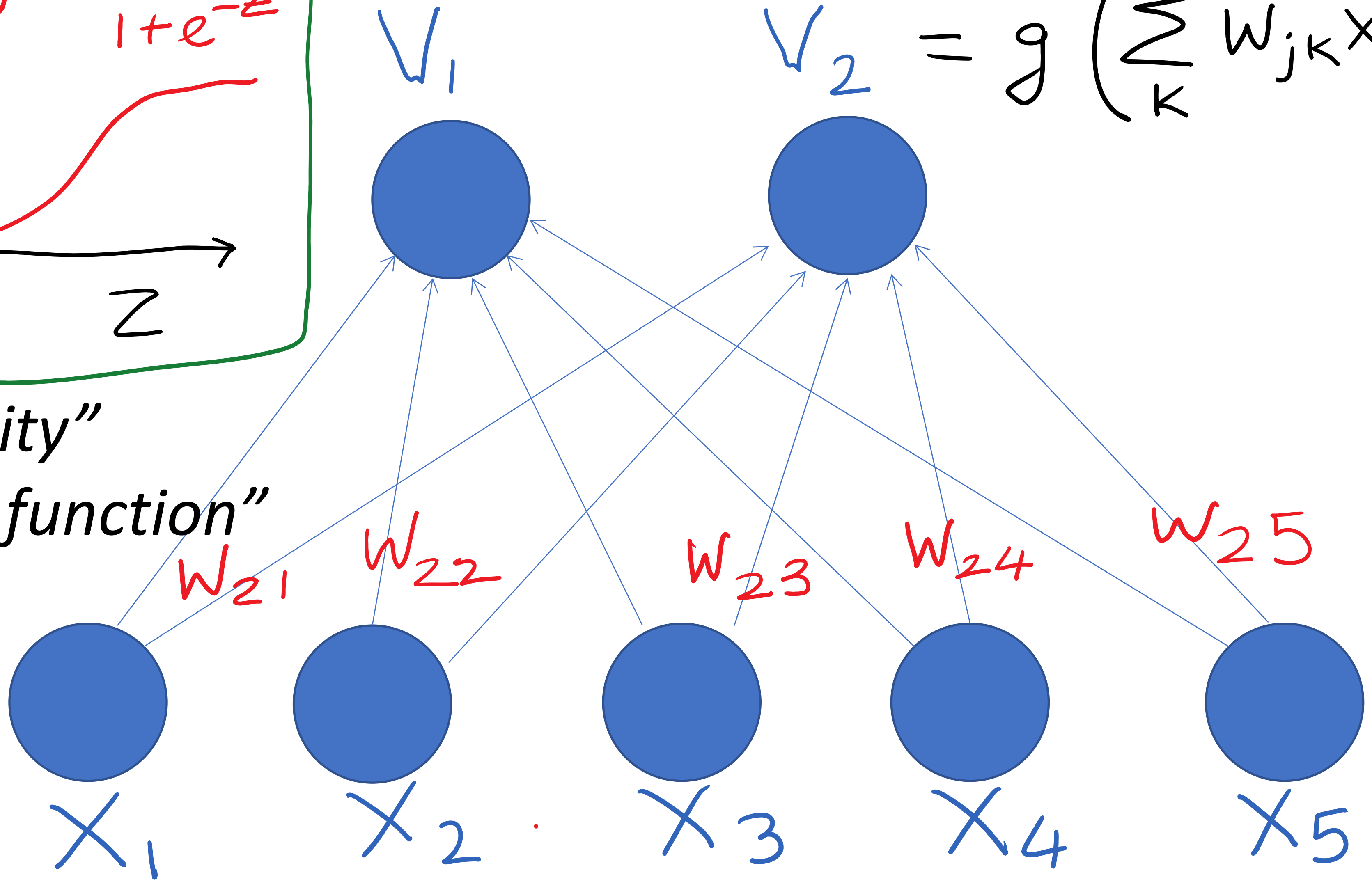
$$g(z) = \frac{1}{1+e^{-z}}$$



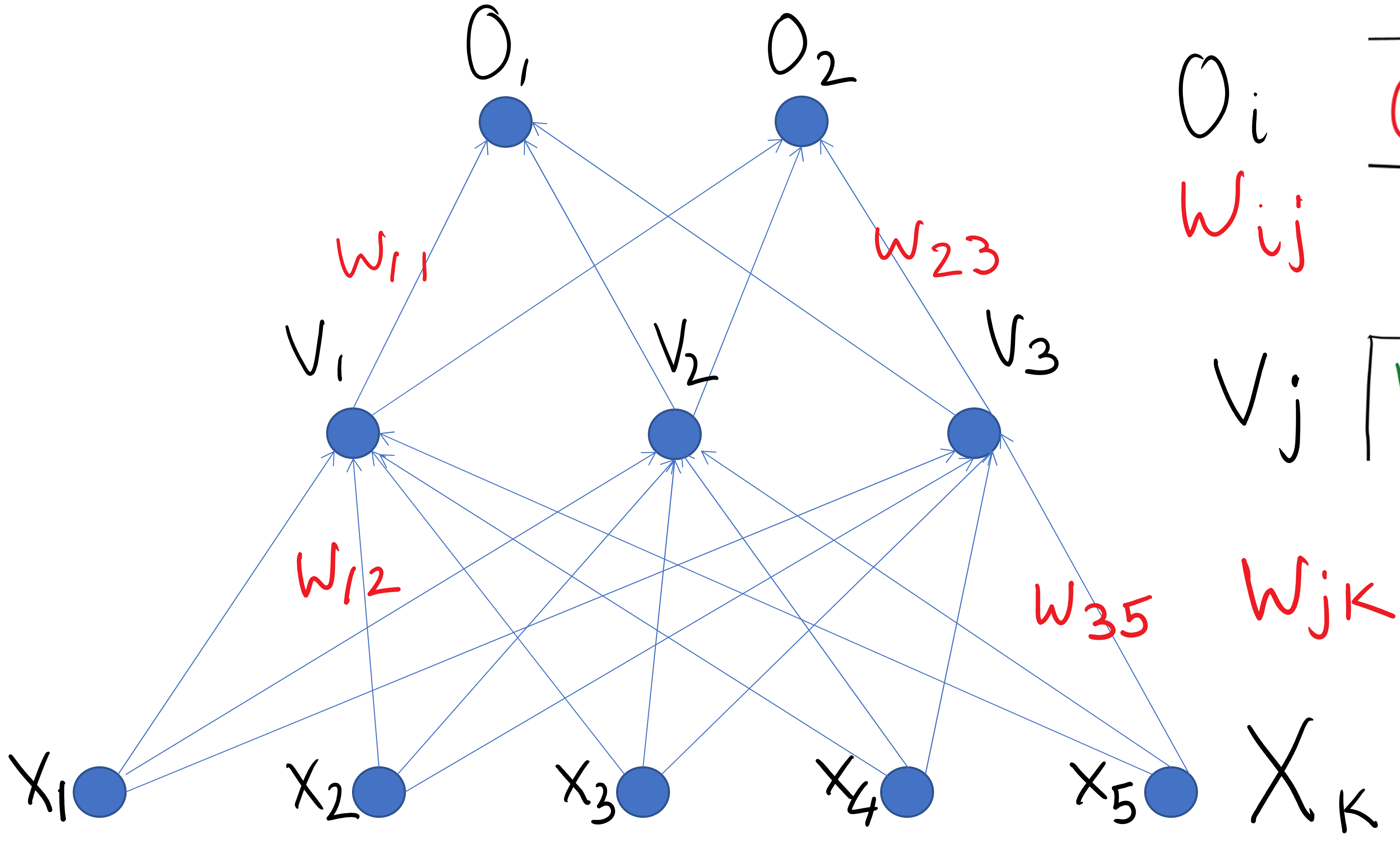
"non-linearity"

"activation function"

$$V_2 = g\left(\sum_k w_{jk} x_k\right)$$



# Two-layer neural network

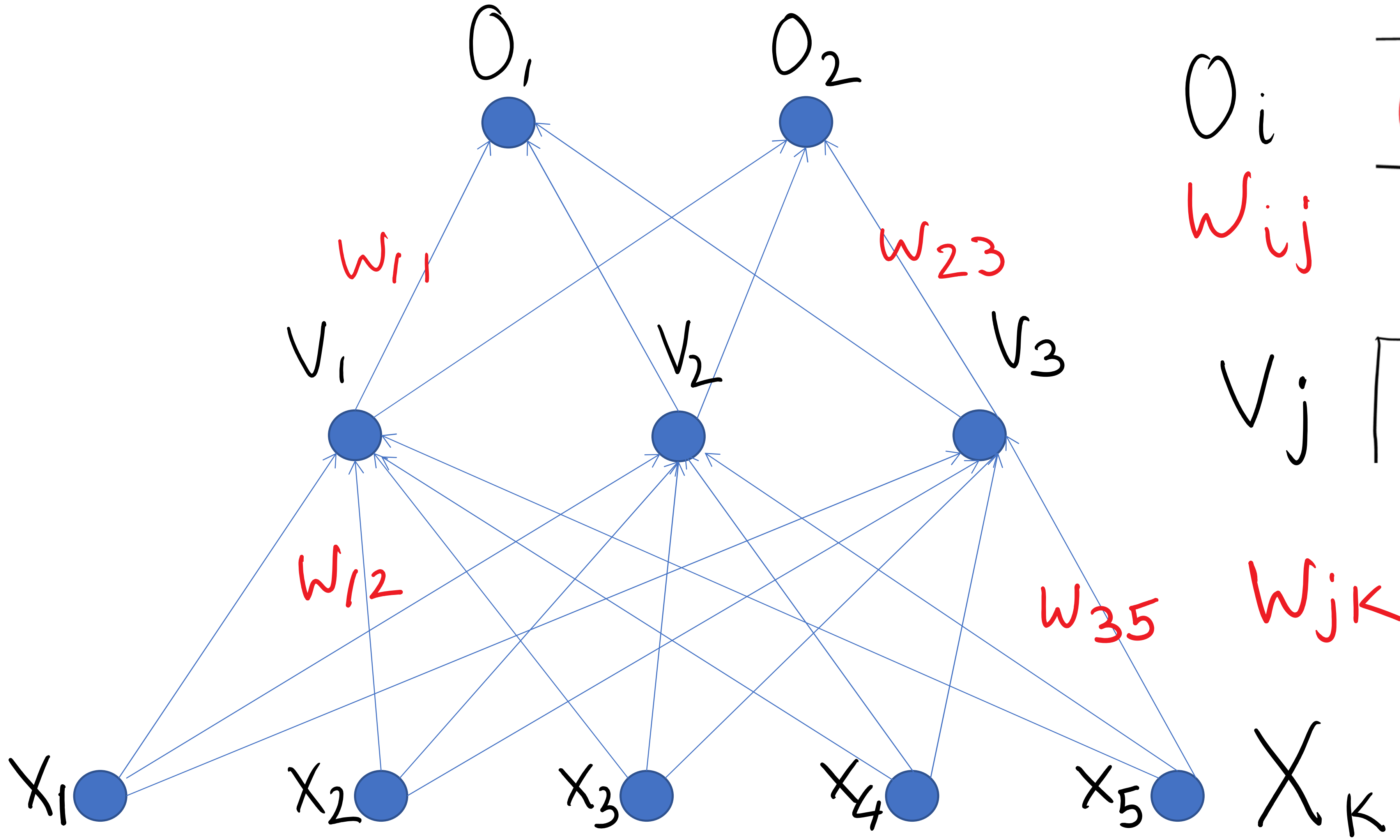


$$\boxed{O_i = g\left(\sum w_{ij} V_j\right)}$$

$$\boxed{V_j = g\left(\sum_K w_{jK} x_K\right)}$$

# Two-layer neural network

$$\boxed{O_i = g\left(\sum_j W_{ij}\, g\left(\sum_K W_{jK} x_K\right)\right)}$$

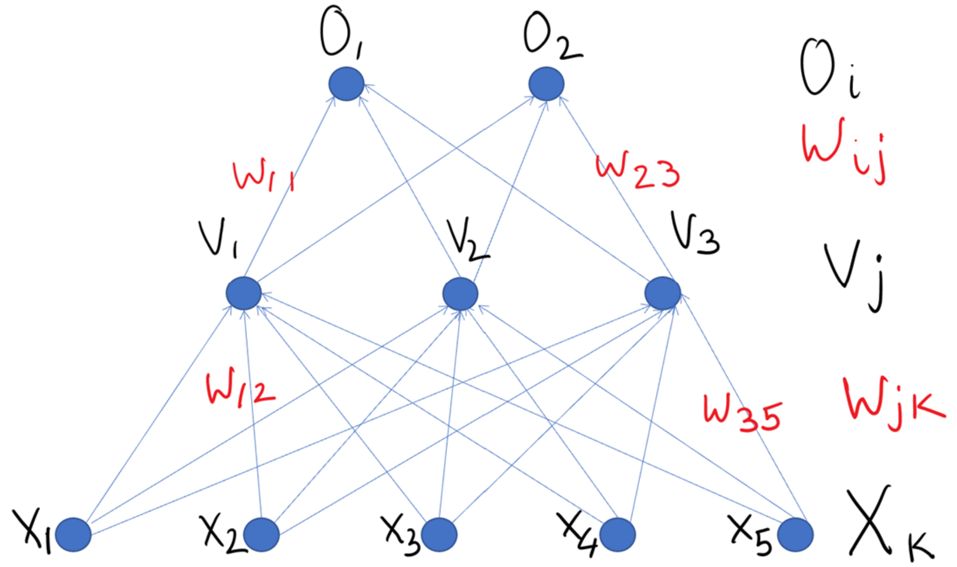


$$\boxed{O_i = g\left(\sum w_{ij} V_j\right)}$$

$$\boxed{V_j = g\left(\sum_K w_{jK} x_K\right)}$$

# Training a neural network

Goal: Find $W$ such that $O_i$ is as close as possible to $y_i$ (desired output)



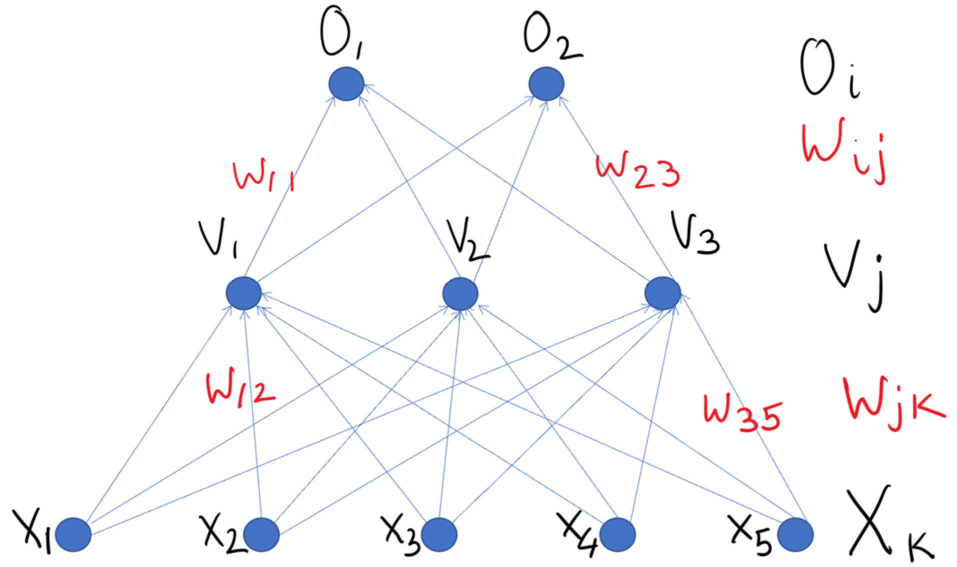
# Training a neural network

Goal: Find $W$ such that $O_i$ is as close as possible to $y_i$ (desired output)

Approach: · Define loss function $\mathcal{L}(W)$

· Compute $\nabla_W \mathcal{L}$

· $W_{new} \leftarrow W_{old} - \eta \nabla_W \mathcal{L}$



# Training a single-layer neural network

- A good choice of loss function is the likelihood, (equivalent to cross- entropy).

$$\mathcal{L} = -\sum_{\text{input data}} \left( y_i \ln O_i + (1-y_i) \ln(1-O_i) \right)$$

- Model the activation function *g* as a sigmoid

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

- Finding parameter w reduces to logistic regression!

We use *gradient descent.* $w_{new} \leftarrow w_{old} - \eta \nabla_w \mathcal{L}$

## Training a 2-layer neural network

Use the same loss function, same activation functions, and still use gradient descent. All the same principles apply!

- Compute gradient wrt all weights across all layers.
- Loss function is no longer convex, so we typically find local minima.
- Time complexity of computing the gradient is naively quadratic in the # of weights.
- The *back-propagation algorithm* is a trick that enables it to be computed in linear time.
- This works for *any* number of layers.

## How do we choose a classifier?

- If we knew the true probability distribution of the features conditioned on the classes, there is a “correct” answer – the *Bayes classifier*.
- The Bayes classifier minimizes the probability of misclassification.
- (unknown in practical situations)

# Bayes Optimal Classifier

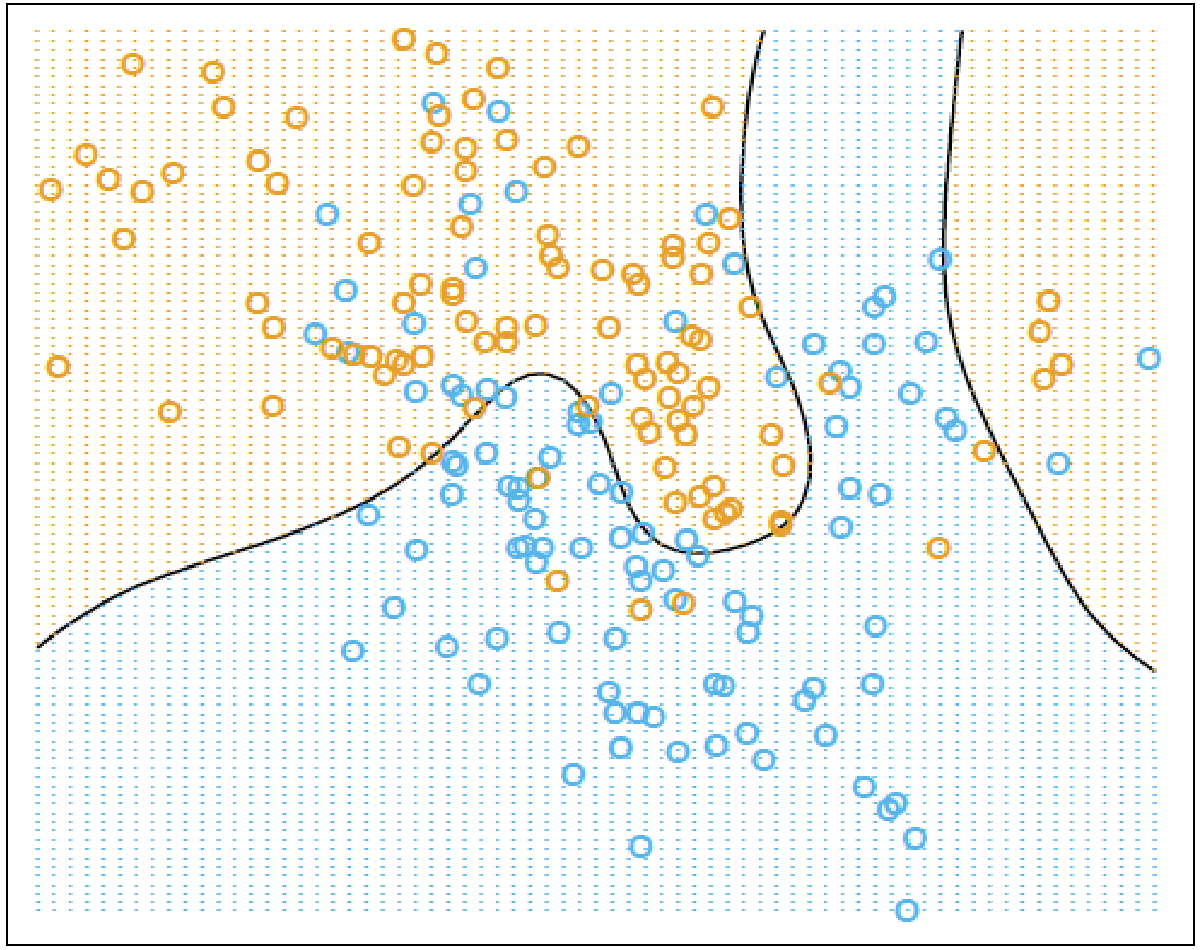


**FIGURE 2.5.** *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

[Source : Hastie, Tibshirani, Friedman]

## Two primary kinds of error

1. Training set error
   - We train a classifier to minimize training set error.
2. Test set error
   - At test time, we will take the trained classifier and use it to classify previously unseen examples for which we know the right answer. The error on these is called test set error.

## Validation and Cross-Validation

- If the test set error is much greater than training set error, this is called over-fitting.
- To avoid over-fitting, we can measure error on a held-out set of training data, called the validation set.
- We could divide the data into k-folds, use k-1 of these to train and “test”/validate on the remaining fold. This is cross-validation.