

Supervised Machine Learning:

Pattern Classification

Bayes Classifier

Classification using Reference Template Methods

- For a test example, a distance measure is computed with the reference template of each class
- The class of the reference template with least distance is assigned to the test pattern
- When mean vector and covariance matrix is used as reference template for each class, Mahalanobis distance is used
- Mahalanobis distance gives the notion that distance measure is computed between a test example and the distribution (density) of a class
 - Distribution (density) of class: All the training examples are drawn from that distribution
 - Density here is normal (Gaussian) density
- In other way, we are interested to estimate probability of class, $P(C_i | \mathbf{x})$
 - Given the test example \mathbf{x} , what is the probability that it belongs to i^{th} class (C_i)
- Solution: Bayes classifier

Bayes Classifier

- Let $C_1, C_2, \dots, C_i, \dots, C_M$ be the M classes
 - Each class has N_i number of training examples
- Given: a test example \mathbf{x}
- To Compute:
 - Probability of class, $P(C_i | \mathbf{x})$
- Bayes decision rule:
 - Prior: Prior information of a class
 - Example: Human data – Each person is represented using height and weight
 - Assume that data is collected from primary school
 - Adult: Teachers and staff
 - Child: Students
 - What is the prior information about persons in primary school?
 - Probability of Child is more than Adult
 - If the human data is collected irrespective of any location
 - Prior probabilities of Adult and Child are same

Posterior
Probability
of a class

$$P(C_i | \mathbf{x}) = \frac{p(\mathbf{x} | C_i)P(C_i)}{P(\mathbf{x})}$$

Prior

Bayes Classifier

- Let $C_1, C_2, \dots, C_i, \dots, C_M$ be the M classes
 - Each class has N_i number of training examples
- Given: a test example \mathbf{x}
- To Compute:
 - Probability of class, $P(C_i | \mathbf{x})$
- Bayes decision rule:

$$P(C_i | \mathbf{x}) = \frac{p(\mathbf{x} | C_i)P(C_i)}{P(\mathbf{x})}$$

Posterior Probability of a class **Likelihood** **Prior**
Total probability

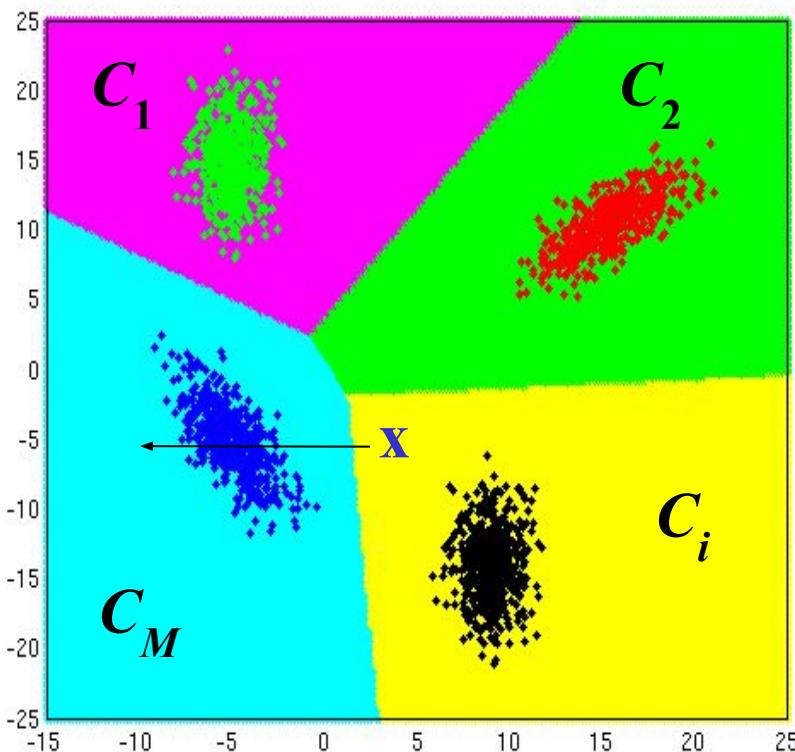
- Prior: Prior information of a class $P(C_i) = \frac{N_i}{N}$
 - where, N is total number of training examples
- Likelihood of a class: Given the **training data of a class** (C_i), what is the likelihood that \mathbf{x} is coming that class
 - It follows the distribution of the data of a class
- Total probability: Evidence/probability that \mathbf{x} exists

$$P(\mathbf{x}) = \sum_{i=1}^M p(\mathbf{x} | C_i)P(C_i)$$

- Out of all the samples, what is the probability of the sample we are looking at

$$\text{Class label for } \mathbf{x} = \arg \max_i P(C_i | \mathbf{x}) \quad i = 1, 2, \dots, M$$

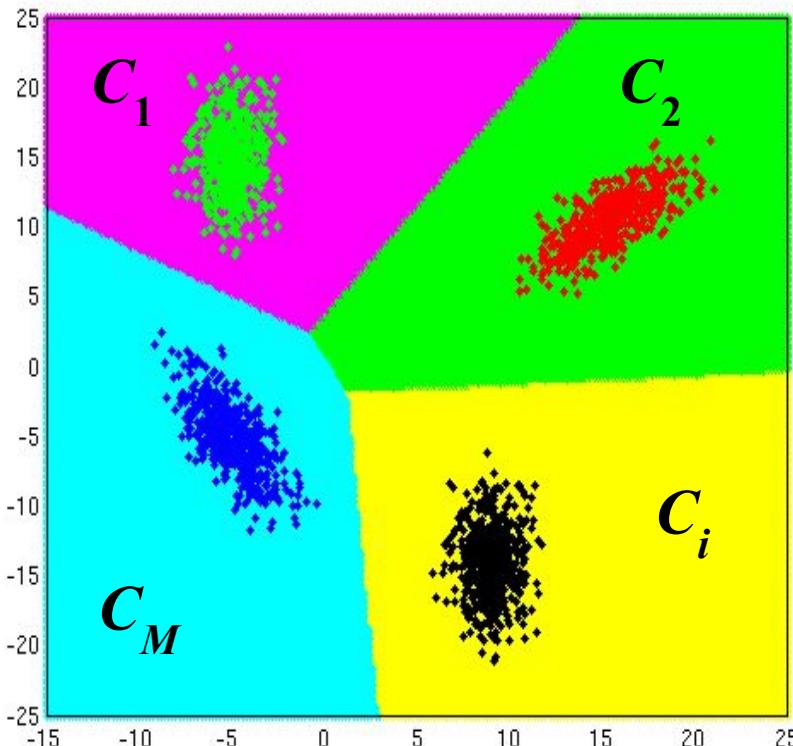
Probability Theory and Bayes Rule



- $P(A)$: Probability of an event A
- The sample space is partitioned into $C_1, C_2, \dots, C_i, \dots, C_M$ where each partitions are disjoint
 - Example:
 - Data space is sample space
 - Each class is my partitions
- Let \mathbf{x} be an event defined in sample space
 - Example: A finite data points (training data) are the event \mathbf{x}
- $P(\mathbf{x})$: Total probability i.e. joint probability of \mathbf{x} and C_i , $P(\mathbf{x}, C_i)$, for all i

$$P(\mathbf{x}) = \sum_{i=1}^M p(\mathbf{x}, C_i)$$

Probability Theory and Bayes Rule



- Conditional probability:

$$p(\mathbf{x} | C_i) = \frac{p(\mathbf{x}, C_i)}{P(C_i)} \quad (1)$$

- Rewriting (1)

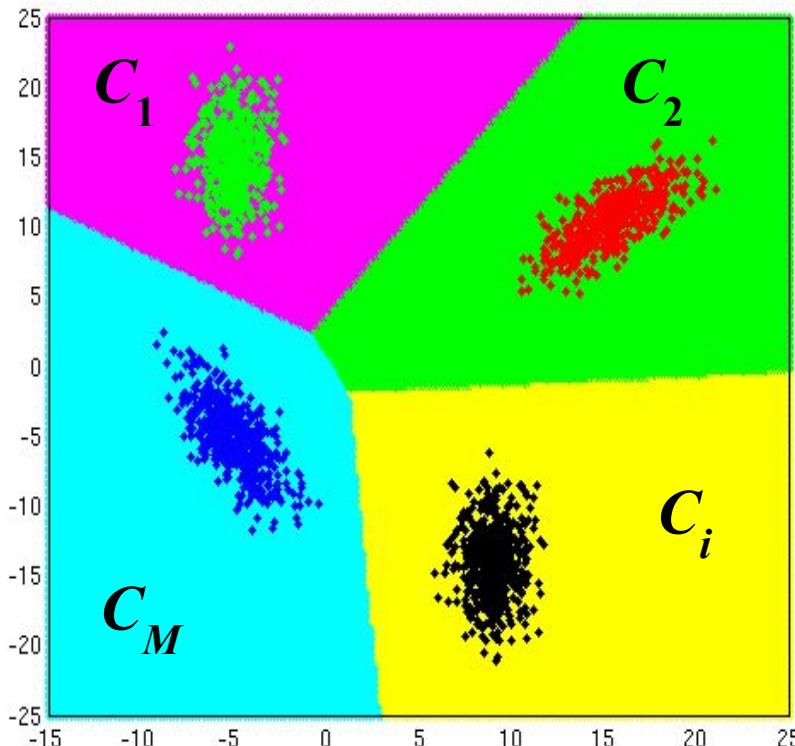
$$p(\mathbf{x}, C_i) = p(\mathbf{x} | C_i)P(C_i) \quad (3)$$

- $P(\mathbf{x})$: Total probability i.e. **joint probability** of \mathbf{x} and C_i , $P(\mathbf{x}, C_i)$, for all i

$$P(\mathbf{x}) = \sum_{i=1}^M p(\mathbf{x}, C_i) = \sum_{i=1}^M p(\mathbf{x} | C_i)P(C_i)$$

- $P(\mathbf{x})$ is **marginal probability** – probability of \mathbf{x} is obtained by marginalising over all the events C_i where $i=1,2,\dots,M$

Probability Theory and Bayes Rule



$$p(\mathbf{x} | C_i) = \frac{p(\mathbf{x}, C_i)}{P(C_i)} \quad (1)$$

$$p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$$

$$p(\mathbf{x}, C_i) = p(\mathbf{x} | C_i)P(C_i) \quad (3)$$

$$p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$$

- From (3) and (4): $p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$
- Bayes decision rule:

$$P(C_i | \mathbf{x}) = \frac{p(\mathbf{x} | C_i)P(C_i)}{P(\mathbf{x})}$$

Bayes Classifier

- Let $C_1, C_2, \dots, C_i, \dots, C_M$ be the M classes
 - Each class has N_i number of training examples
- Given: a test example \mathbf{x}
- To Compute:
 - Probability of class, $P(C_i | \mathbf{x})$
- Bayes decision rule:
 - Likelihood of a class (Class conditional density) follows the distribution of the data of a class
 - Computation of likelihood of a class (class conditional density) depends on the
 - distribution of the data (i.e. data follows some distribution) and
 - the parameters of that distribution
- Bayes decision rule can be given as $P(\theta_i | \mathbf{x}) = \frac{p(\mathbf{x} | \theta_i)P(C_i)}{P(\mathbf{x})}$
 - θ_i is the parameters of the distribution of class C_i
estimated from training data of that class

$$P(C_i | \mathbf{x}) = \frac{p(\mathbf{x} | C_i)P(C_i)}{P(\mathbf{x})}$$

Posterior Probability of a class **Likelihood** **Prior**
↓ ↓ ↓
Evidence

Parameter Estimation from Training Data: Maximum Likelihood (ML) Method

- Given: Training data for a class C_i : having N_i samples

$$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_{N_i}\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

- Data of a class C_i is sampled from a distribution, which is defined by parameter vector: $\boldsymbol{\theta}_i = [\theta_{i1}, \theta_{i2}, \dots, \theta_{iK}]^\top$ of that distribution

- Data of a class C_i is now represented by parameter vector, $\boldsymbol{\theta}_i$
- Unknown: $\boldsymbol{\theta}_i$

- Likelihood of training data (Total data likelihood) for a given $\boldsymbol{\theta}_i$:

$$p(\mathcal{D}_i | \boldsymbol{\theta}_i) = \prod_{n=1}^{N_i} p(\mathbf{x}_n | \boldsymbol{\theta}_i)$$

- Log likelihood: $L(\boldsymbol{\theta}_i) = \ln p(\mathcal{D}_i | \boldsymbol{\theta}_i) = \sum_{n=1}^{N_i} \ln p(\mathbf{x}_n | \boldsymbol{\theta}_i)$

- Advantage of applying monotonous increasing function, $\ln(\cdot)$:

- Likelihood of an example is very small value. Product of small values lead to 0. It converts product of likelihoods into sum of likelihoods
- Simplifies computation for certain forms of distribution

Parameter Estimation from Training Data: Maximum Likelihood (ML) Method

Parameter Estimation from Training Data: Maximum Likelihood (ML) Method

- Given: Training data for a class C_i : having N_i samples

$$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_{N_i}\}, \quad \mathbf{x}_n \in \mathbb{R}^d$$

- Data of a class C_i is sampled from a distribution, which is defined by parameter vector: $\boldsymbol{\theta}_i = [\theta_{i1}, \theta_{i2}, \dots, \theta_{iK}]^\top$ of that distribution

- Data of a class C_i is now represented by parameter vector, $\boldsymbol{\theta}_i$
- Unknown: $\boldsymbol{\theta}_i$

- Likelihood of training data (**Total data likelihood**) for a given $\boldsymbol{\theta}_i$:

$$p(\mathcal{D}_i | \boldsymbol{\theta}_i) = \prod_{n=1}^{N_i} p(\mathbf{x}_n | \boldsymbol{\theta}_i)$$

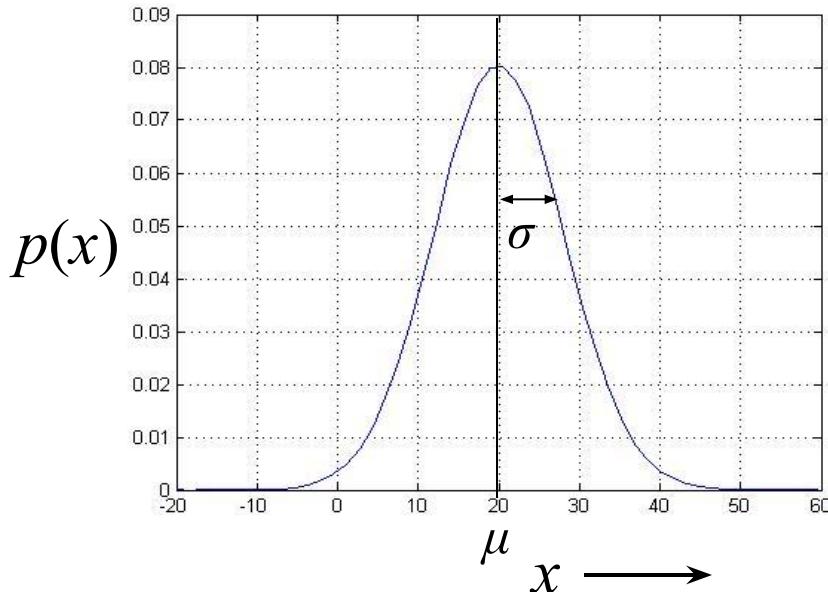
- Log likelihood: $L(\boldsymbol{\theta}_i) = \ln p(\mathcal{D}_i | \boldsymbol{\theta}_i) = \sum_{n=1}^{N_i} \ln p(\mathbf{x}_n | \boldsymbol{\theta}_i)$

- Choose the parameters for which the total data likelihood (log likelihood) is maximum:

$$\boldsymbol{\theta}_{i\text{ML}} = \arg \max_{\boldsymbol{\theta}_i} L(\boldsymbol{\theta}_i)$$

Probability Distribution

- Data of a class is represented by a probability distribution
- For a class whose data is considered to be forming a single cluster, it can be represented by a normal or Gaussian distribution
- Gaussian distribution is a unimodal distribution
 - Single mode or single peak
- Univariate Gaussian distribution:
 - Univariate data means 1-dimensional data



$$p(x) = \mathcal{N}(x | \mu, \sigma)$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- μ is the mean
- σ^2 is the variance

Probability Distribution

- Data of a class is represented by a probability distribution
- For a class whose data is considered to be forming a single cluster, it can be represented by a normal or Gaussian distribution
- Gaussian distribution is a unimodal distribution
 - Single mode or single peak
- Multivariate Gaussian distribution:
 - Multivariate data means d -dimensional data
 - *Bivariate Gaussian distribution*
 - Bivariate data means 2-dimensional data

Multivariate Gaussian Distribution

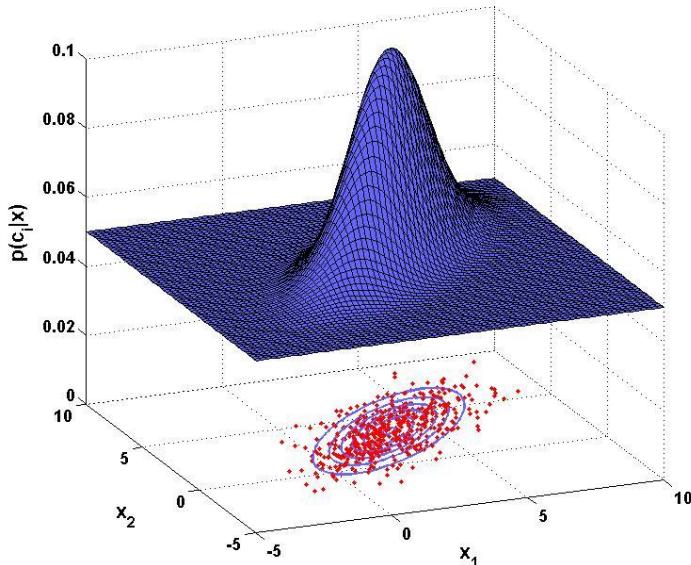
- Data in d -dimensional space

$$p(\mathbf{x}) = N(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} \underbrace{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}_{\text{Mahalanobis distance}}\right)$$

- $\boldsymbol{\mu}$ is the mean vector
- $\boldsymbol{\Sigma}$ is the covariance matrix

- Bivariate Gaussian distribution: $d=2$



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} E[x_1] \\ E[x_2] \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} E[(x_1 - \mu_1)^2] & E[(x_1 - \mu_1)(x_2 - \mu_2)] \\ E[(x_2 - \mu_2)(x_1 - \mu_1)] & E[(x_2 - \mu_2)^2] \end{bmatrix}$$

Maximum Likelihood (ML) Method for Parameter Estimation of Multivariate Gaussian Distribution

- Given: Training data for a class C_i having N_i samples

$$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \dots, \mathbf{x}_{N_i}\}, \mathbf{x}_n \in \mathbb{R}^d$$

- Data of a class C_i is coming from Gaussian distribution
 - Training data of a class C_i is represented by parameter vector: $[\boldsymbol{\mu}_i \ \boldsymbol{\Sigma}_i]^\top$, of Gaussian distribution
- Unknown: $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$
- Likelihood of training data (Total data likelihood) for a given $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$: $p(\mathcal{D} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \prod_{n=1}^{N_i} p(\mathbf{x}_n | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$
- Log likelihood: $L(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \ln p(\mathcal{D}_i | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \sum_{n=1}^{N_i} \ln p(\mathbf{x}_n | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$
- Choose the parameters for which the total data likelihood (log likelihood) is maximum:

$$\boldsymbol{\mu}_{i_{\text{ML}}}, \boldsymbol{\Sigma}_{i_{\text{ML}}} = \arg \max_{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i} L(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

ML Method for Parameter Estimation of Multivariate Gaussian Distribution

- Parameters of Gaussian distribution of class C_i : μ_i and Σ_i
- Likelihood for a single example, \mathbf{x}_n :

$$p(\mathbf{x}_n | \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x}_n - \mu_i)^\top \Sigma_i^{-1} (\mathbf{x}_n - \mu_i)\right)$$

- Log likelihood for total training data of class C_i ,

$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_i}\}$:

$$\begin{aligned} \mathcal{L}(\mu_i, \Sigma_i) &= \ln p(\mathcal{D}_i | \mu_i, \Sigma_i) = \ln \prod_{n=1}^{N_i} p(\mathbf{x}_n | \mu_i, \Sigma_i) = \sum_{n=1}^{N_i} \ln p(\mathbf{x}_n | \mu_i, \Sigma_i) \\ &= \sum_{n=1}^{N_i} -\frac{1}{2} \ln |\Sigma_i| - \frac{d}{2} \ln 2\pi - \frac{1}{2} (\mathbf{x}_n - \mu_i)^\top \Sigma_i^{-1} (\mathbf{x}_n - \mu_i) \end{aligned}$$

- Setting the derivatives of $\mathcal{L}(\mu_i, \Sigma_i)$ w.r.t. μ_i and Σ_i to zero, we get:

$$\frac{\partial \mathcal{L}(\mu_i, \Sigma_i)}{\partial \mu_i} = 0 \quad \frac{\partial \mathcal{L}(\mu_i, \Sigma_i)}{\partial \Sigma_i} = 0$$

ML Method for Parameter Estimation of Multivariate Gaussian Distribution

- Parameters of Gaussian distribution of class C_i : μ_i and Σ_i
- Likelihood for a single example, \mathbf{x}_n :

$$p(\mathbf{x}_n | \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x}_n - \mu_i)^\top \Sigma_i^{-1} (\mathbf{x}_n - \mu_i)\right)$$

- Log likelihood for total training data of class C_i ,

$\mathcal{D}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_i}\}$:

$$\begin{aligned} \mathcal{L}(\mu_i, \Sigma_i) &= \ln p(\mathcal{D}_i | \mu_i, \Sigma_i) = \ln \prod_{n=1}^{N_i} p(\mathbf{x}_n | \mu_i, \Sigma_i) = \sum_{n=1}^{N_i} \ln p(\mathbf{x}_n | \mu_i, \Sigma_i) \\ &= \sum_{n=1}^{N_i} -\frac{1}{2} \ln |\Sigma_i| - \frac{d}{2} \ln 2\pi - \frac{1}{2} (\mathbf{x}_n - \mu_i)^\top \Sigma_i^{-1} (\mathbf{x}_n - \mu_i) \end{aligned}$$

- Setting the derivatives of $\mathcal{L}(\mu_i, \Sigma_i)$ w.r.t. μ_i and Σ_i to zero, we get:

$$\mu_{i_{ML}} = \frac{1}{N_i} \sum_{n=1}^{N_i} \mathbf{x}_n \quad \Sigma_{i_{ML}} = \frac{1}{N_i} \sum_{n=1}^{N_i} (\mathbf{x}_n - \mu_{i_{ML}})(\mathbf{x}_n - \mu_{i_{ML}})^\top$$

Bayes Classifier with Unimodal Gaussian Density – Training Process

- Let $C_1, C_2, \dots, C_i, \dots, C_M$ be the M classes
- Let $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_i, \dots, \mathcal{D}_M$ be the training data for M classes
- Let each class having N_i number of training examples
- Estimate the parameters
 - $\theta_1 = [\mu_1 \Sigma_1]^T,$
 - $\theta_2 = [\mu_2 \Sigma_2]^T,$
 - $\dots,$
 - $\theta_i = [\mu_i \Sigma_i]^T,$
 - $\dots,$
 - $\theta_M = [\mu_M \Sigma_M]^T$ for each of the classes
- Number of parameters to be estimated for each class is dependent on dimensionality of the data space d
 - Number of parameters for each class: $d + (d(d+1))/2$

Bayes Classifier with Unimodal Gaussian Density – Training Process

- Let $C_1, C_2, \dots, C_i, \dots, C_M$ be the M classes
- Let $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_i, \dots, \mathcal{D}_M$ be the training data for M classes
- Compute sample mean vector and sample covariance matrix from training data of class 1, $\boldsymbol{\theta}_1 = [\boldsymbol{\mu}_1 \ \boldsymbol{\Sigma}_1]^\top$
- Compute sample mean vector and sample covariance matrix from training data of class 2, $\boldsymbol{\theta}_2 = [\boldsymbol{\mu}_2 \ \boldsymbol{\Sigma}_2]^\top$,
- ...,
- Compute sample mean vector and sample covariance matrix from training data of class M , $\boldsymbol{\theta}_M = [\boldsymbol{\mu}_M \ \boldsymbol{\Sigma}_M]^\top$

Bayes Classifier with Unimodal Gaussian Density: Classification

- For a test example \mathbf{x} :
 - likelihood of \mathbf{x} generated from each of the classes $p(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and class posterior probability $P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x})$ is computed

$$P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)}{P(\mathbf{x})}$$

Bayes Classifier with Unimodal Gaussian Density: Classification

- For a test example \mathbf{x} :
 - likelihood of \mathbf{x} generated from each of the classes $p(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ and class posterior probability $P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x})$ is computed

$$P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)}{\sum_{i=1}^M p(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)}$$

- Assign the label of class for which $P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x})$ is maximum

$$\text{Class label} = \arg \max_i P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x})$$

Bayes Classifier with Unimodal Gaussian Density: Classification

- For a test example \mathbf{x} :
 - likelihood of \mathbf{x} generated from each of the classes $p(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ or class posterior probability $P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x})$ is computed
 - Assign the label of class for which $P(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | \mathbf{x})$ is maximum

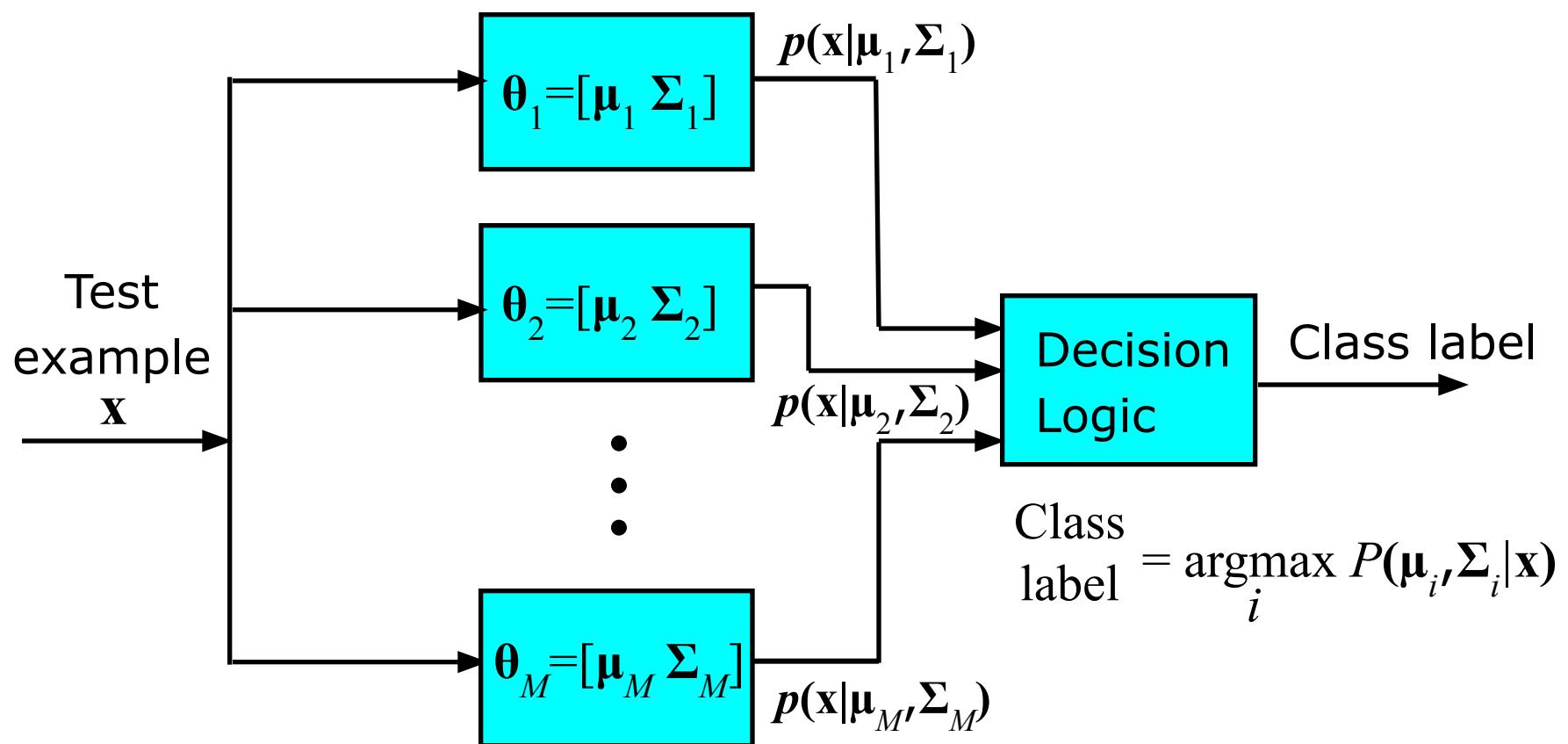


Illustration of Bayes Classifier with Unimodal Gaussian Density : Adult(1)-Child(0) Classification

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Training Phase:
 - Compute sample mean vector and sample covariance matrix from training data of class 0 (Child)
 $\mu_0 = [103.60 \ 30.66]$
 $\Sigma_0 = \begin{pmatrix} 109.38 & 61.35 \\ 61.35 & 43.54 \end{pmatrix}$
 - Prior probability for class 0 (Child):
 $P(C_0) = 10/20 = 0.5$

Illustration of Bayes Classifier with Unimodal Gaussian Density : Adult(1)-Child(0) Classification

Height	Weight	Class
90	21.5	0
95	23.67	0
100	32.45	0
116	38.21	0
98	28.43	0
108	36.32	0
104	27.38	0
112	39.28	0
121	35.8	0
92	23.56	0
152	46.8	1
178	78.9	1
163	67.45	1
173	82.9	1
154	52.6	1
168	66.2	1
183	90	1
172	82	1
156	45.3	1
161	59	1

- Training Phase:
 - Compute sample mean vector and sample covariance matrix from training data of class 0 (Child)

$$\mu_0 = [103.60 \quad 30.66]$$

$$\Sigma_0 = \begin{pmatrix} 109.38 & 61.35 \\ 61.35 & 43.54 \end{pmatrix}$$

- Prior probability for class 0 (Child):

$$P(C_0) = 10/20 = 0.5$$

- Compute sample mean vector and sample covariance matrix from training data of class 1 (Adult)

$$\mu_1 = [166.00 \quad 67.12]$$

$$\Sigma_1 = \begin{pmatrix} 110.67 & 160.53 \\ 160.53 & 255.49 \end{pmatrix}$$

- Prior probability for class 1 (Adult):

$$P(C_1) = 10/20 = 0.5$$

Illustration of Bayes Classifier with Unimodal Gaussian Density : Adult(1)-Child(0) Classification

$$\mu_0 = [103.60 \ 30.66]$$

$$\Sigma_0 = \begin{pmatrix} 109.38 & 61.35 \\ 61.35 & 43.54 \end{pmatrix}$$

Prior: $P(C_0) = 0.5$

Class
0

$$\mu_1 = [166.00 \ 67.12]$$

$$\Sigma_1 = \begin{pmatrix} 110.67 & 160.53 \\ 160.53 & 255.49 \end{pmatrix}$$

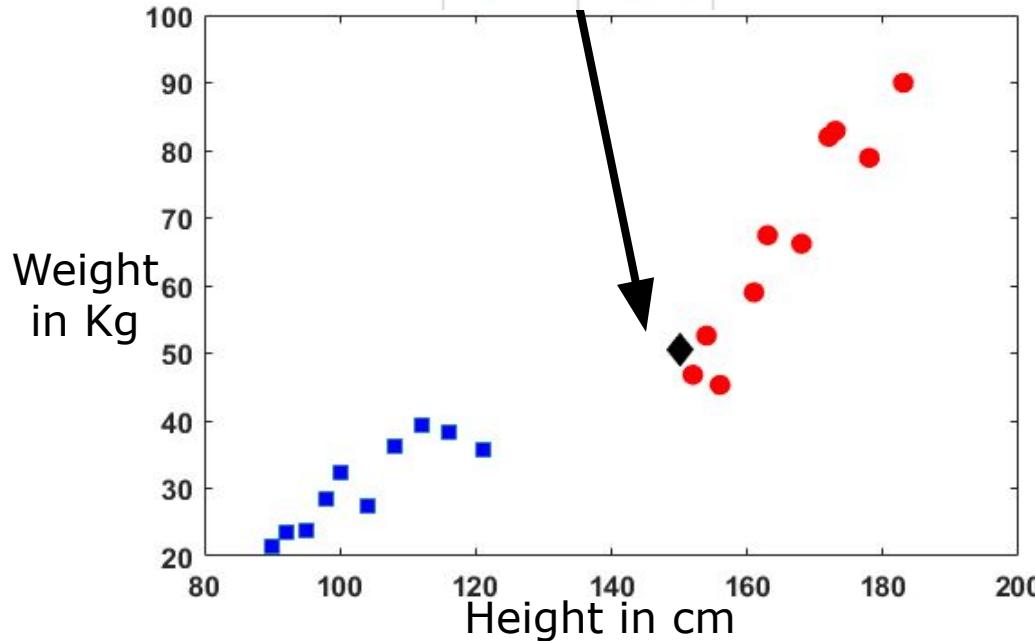
Prior: $P(C_1) = 0.5$

Class
1

- Test Phase - Classification:

Test Example, \mathbf{x} :

150	50.6
-----	------



$$p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$$

Illustration of Bayes Classifier with Unimodal Gaussian Density : Adult(1)-Child(0) Classification

$$\mu_0 = [103.60 \ 30.66]$$

$$\Sigma_0 = \begin{pmatrix} 109.38 & 61.35 \\ 61.35 & 43.54 \end{pmatrix}$$

Prior: $P(C_0) = 0.5$

Class
0

$$\mu_1 = [166.00 \ 67.12]$$

$$\Sigma_1 = \begin{pmatrix} 110.67 & 160.53 \\ 160.53 & 255.49 \end{pmatrix}$$

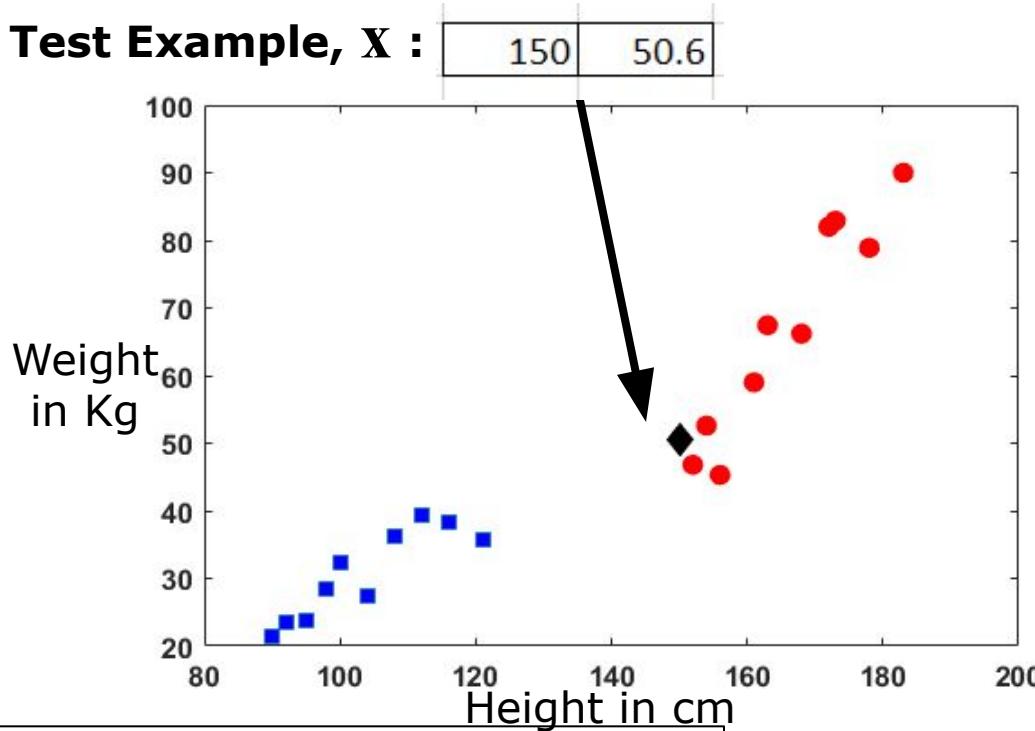
Prior: $P(C_1) = 0.5$

Class
1

- Test Phase - Classification:

Test Example, \mathbf{x} :

150	50.6
-----	------



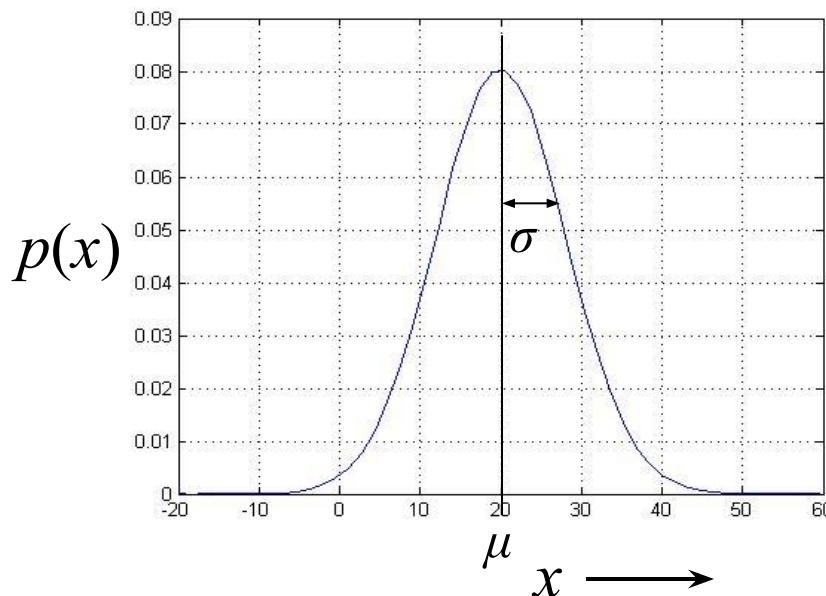
$$p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$$

- Compute a posterior probability for class 0 (Child): $p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$
- Compute a posterior probability for class 1 (Adult): $p(\mathbf{x}, C_i) = P(C_i | \mathbf{x})P(\mathbf{x}) \quad (4)$

Class label of \mathbf{x} = Adult

Summary: Bayes Classifier with Unimodal Gaussian Density

- The relation between examples and class can be captured in a statistical model
 - Bayes classifier
 - Data is represented by any distribution
 - If the underlying distribution (density) of data is known, then Bayes classifier is the minimum error classifier
- Statistical model:
 - Example distribution: Unimodal Gaussian density
 - Univariate

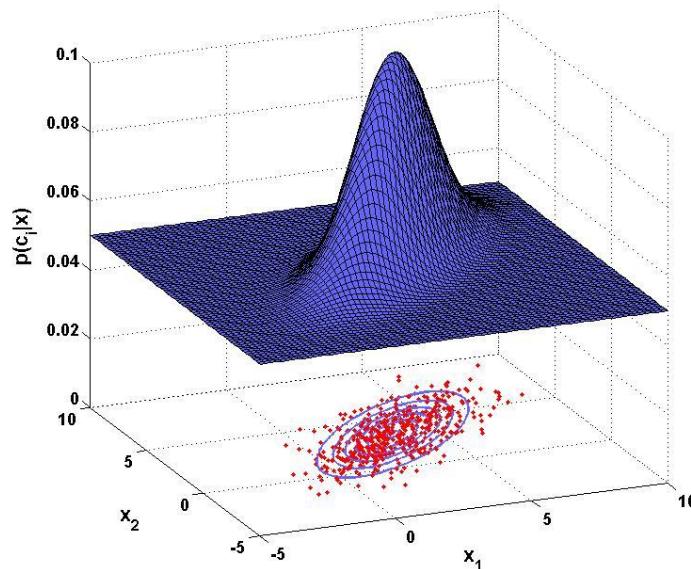


$$p(x) = N(x | \mu, \sigma)$$
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- μ is the mean
- σ^2 is the variance

Summary: Bayes Classifier with Unimodal Gaussian Density

- The relation between examples and class can be captured in a statistical model
 - Bayes classifier
 - Data is represented by any distribution
- Statistical model:
 - Example distribution: Unimodal Gaussian density
 - Univariate
 - Multivariate (*Bivariate when the dimension is 2*)



$$p(\mathbf{x}) = N(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

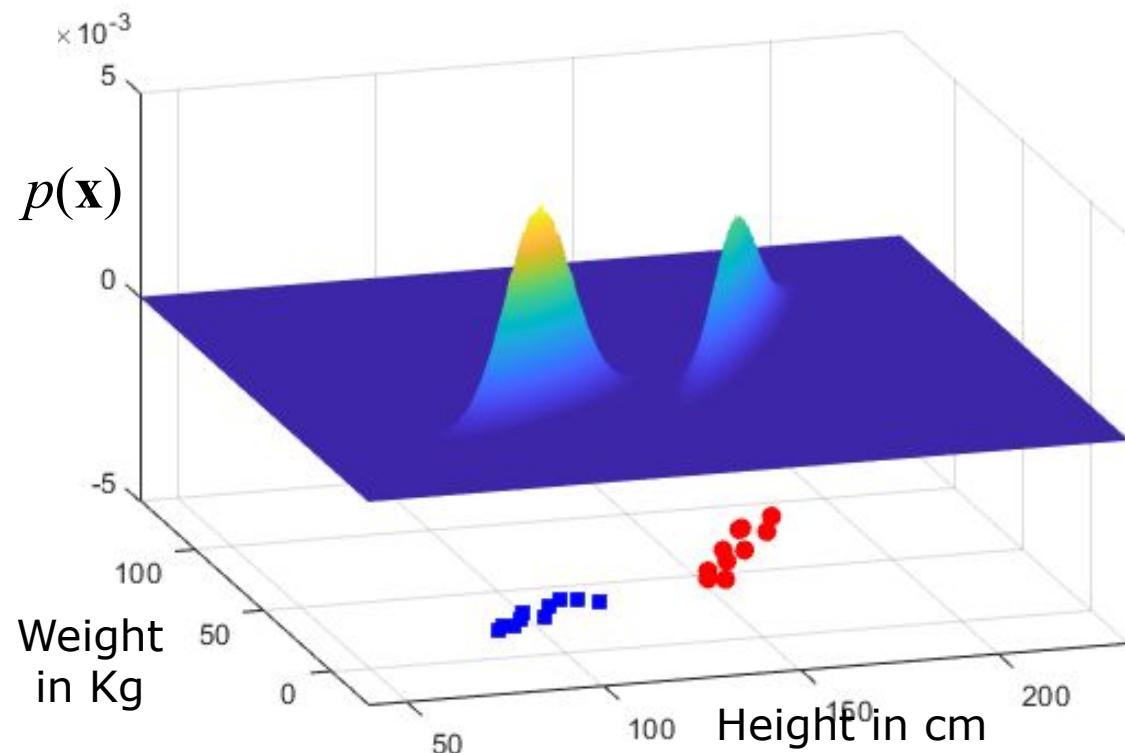
$$= \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

- $\boldsymbol{\mu}$ is the mean vector
- $\boldsymbol{\Sigma}$ is the covariance matrix

Summary: Bayes Classifier with Unimodal Gaussian Density

- The relation between examples and class can be captured in a statistical model
 - Bayes classifier
 - Data is represented by any distribution
- Statistical model:
 - Example distribution: Unimodal Gaussian density
 - Univariate
 - Multivariate



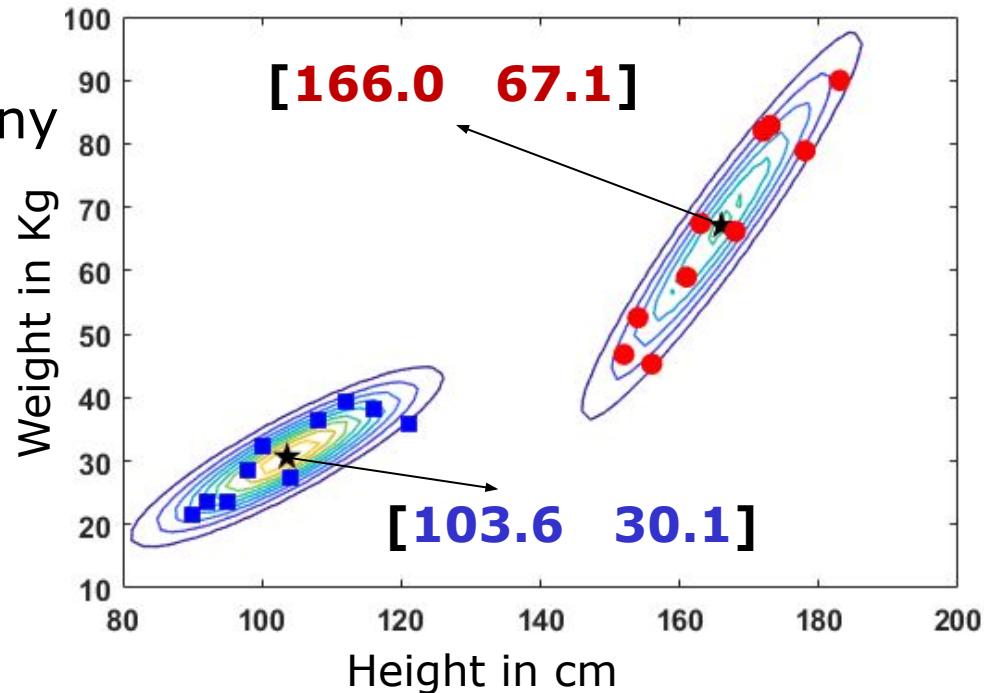
Summary: Bayes Classifier with Unimodal Gaussian Density

- The relation between examples and class can be captured in a statistical model

- Bayes classifier
- Data is represented by any

- Statistical model:

- Example distribution:
Unimodal Gaussian density
 - Univariate
 - Multivariate



- The real world data need not be unimodal
 - The shape of the density can be arbitrary
 - Bayes classifier?
- Multimodal density function

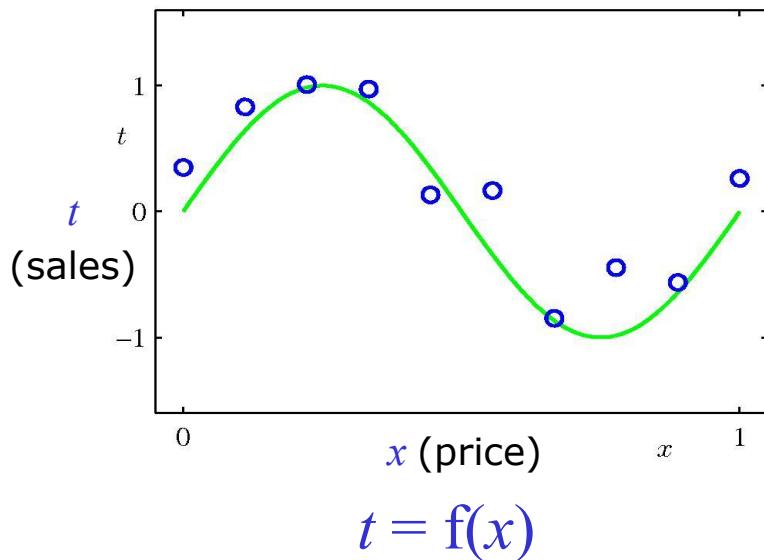
Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.
3. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

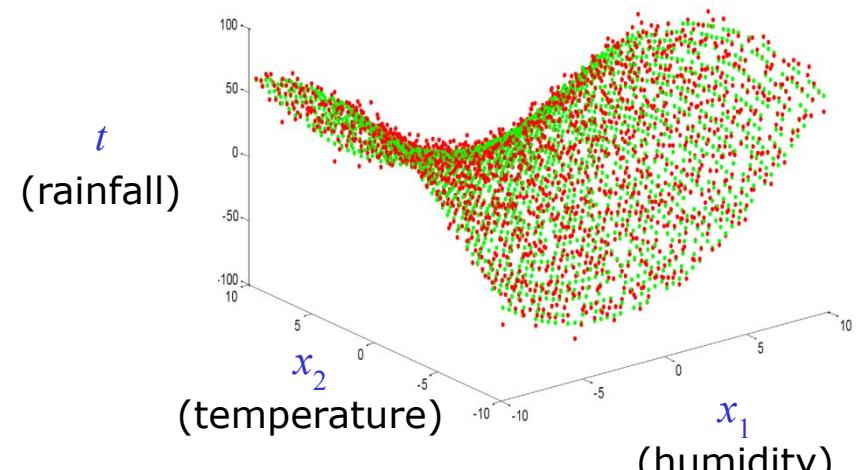
Supervised Machine Learning: Regression

Numeric Prediction (Regression)

- Numeric prediction: Task of predicting continuous (or ordered) values for given input
- Example:
 - Predicting potential sales of a new product given its price
 - Predicting amount of rainfall given the temperature and humidity in the atmosphere



– Predicting amount of rainfall given the temperature and humidity in the atmosphere

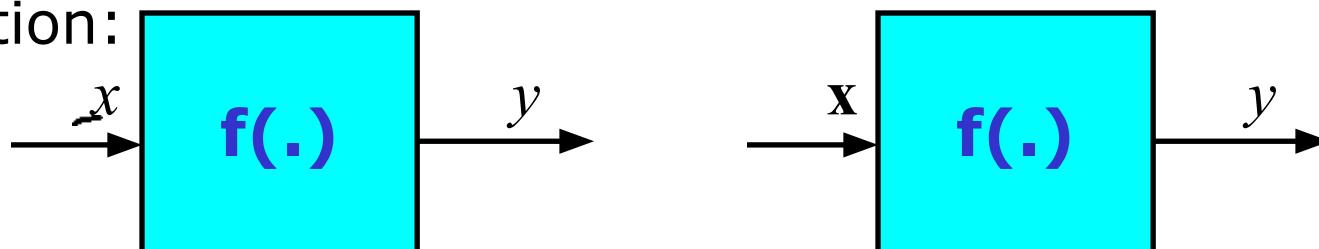


$$t = f(\mathbf{x})$$

$$\mathbf{x} = [x_1, x_2]^T$$

Numeric Prediction (Regression)

- Regression analysis is used to model the relationship between one or more independent (input) variable and a dependent (output) variable
 - Dependent variable is always continuous valued or ordered valued
 - Example: Dependent variable: Rainfall
Independent variable(s): temperature, humidity
- The values of independent variables are known
- The dependent variable is what we want to predict
- Regression analysis can be viewed as mapping function:



- Single independent variable (x)
- Single dependent variable (y)
- Multiple independent variable (\mathbf{x}) $\in \mathbb{R}^d$
- Single dependent variable (y)

Numeric Prediction (Regression)

- Regression is a two step process
 - Step1: Building a regression model
 - Learning from data (training phase)
 - Supervised learning: In supervised learning, each example is a *pair* consisting of an input example (independent variables) and a desired output value (dependent variable)
 - Regression model is built by analysing or learning from a training data set made up of one or more independent variables and their dependent labels
$$y_n = f(\mathbf{x}_n)$$
 - \mathbf{x}_n is the n^{th} input example and y_n is the corresponding output variable
 - Step2: Using regression model for prediction
 - Testing phase
 - Predicting dependent variable
- Accuracy of a predictor:
 - How well a given predictor can predict for new values
- Target of learning techniques: Good generalization ability

Illustration of Training Set: Salary Prediction

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

Independent variable: Years of experience

Dependent variable: Salary

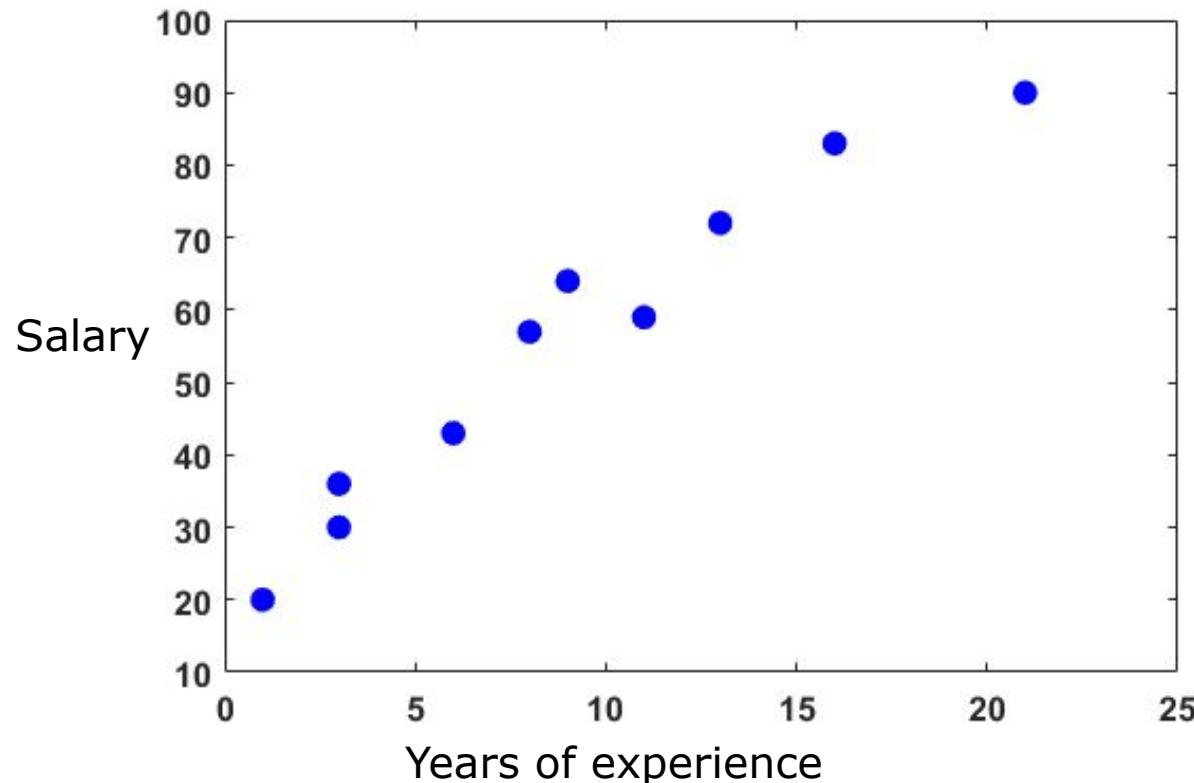


Illustration of Training Set: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- Independent variable: Humidity, Pressure
- Dependent variable: Temperature (Temp)

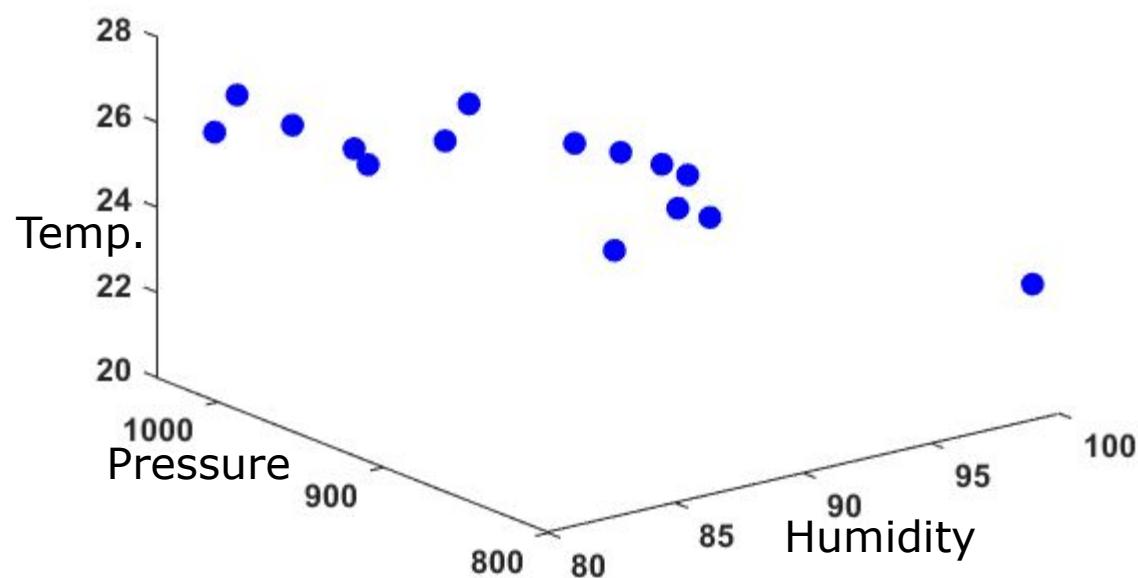


Illustration of Training Set: Wine Quality Prediction [1]

Fixed Acidity (x_1)	Volatile Acidity (x_2)	Citric acid (x_3)	Residual Sugar (x_4)	Chlorides (x_5)	Free SO ₂ (x_6)	Total SO ₂ (x_7)	Density (x_8)	pH (x_9)	Sulphates (x_{10})	Alcohol (x_{11})	Quality (y)
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5.42
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5.57
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5.17
11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6.65
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5.68
7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5.63
7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5.32
7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7.16
7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7.2
7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5.18

- Number of independent variable: 10
- Dependent variable: Quality

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. "Modeling wine preferences by data mining from physicochemical properties," In Decision Support Systems, Elsevier, vol. 47, issue 4, pp. 547-553, 2009.

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

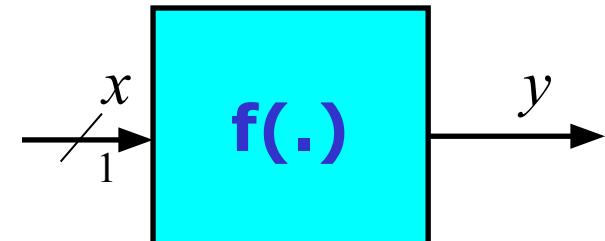
Supervised Machine Learning:

Regression

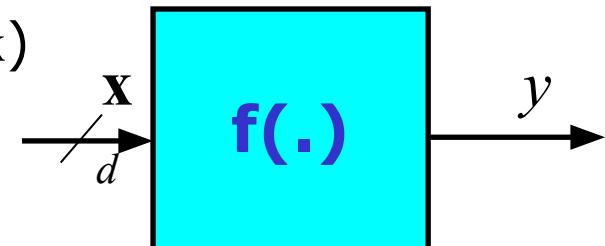
Linear Regression

Linear Regression

- Linear approach to model the relationship between a scalar response, (y) (or dependent variable) and one or more predictor variables, (x or \mathbf{x}) (or independent variables)
- The output is going to be the linear function of input (one or more independent variables)
- Simple linear regression (straight-line regression):
 - Single independent variable (x)
 - Single dependent variable (y)
 - *Fitting a straight-line*



- Multiple linear regression:
 - two or more independent variable (\mathbf{x})
 - Single dependent variable (y)
 - *Fitting a hyperplane (linear surface)*



Straight-Line (Simple Linear) Regression

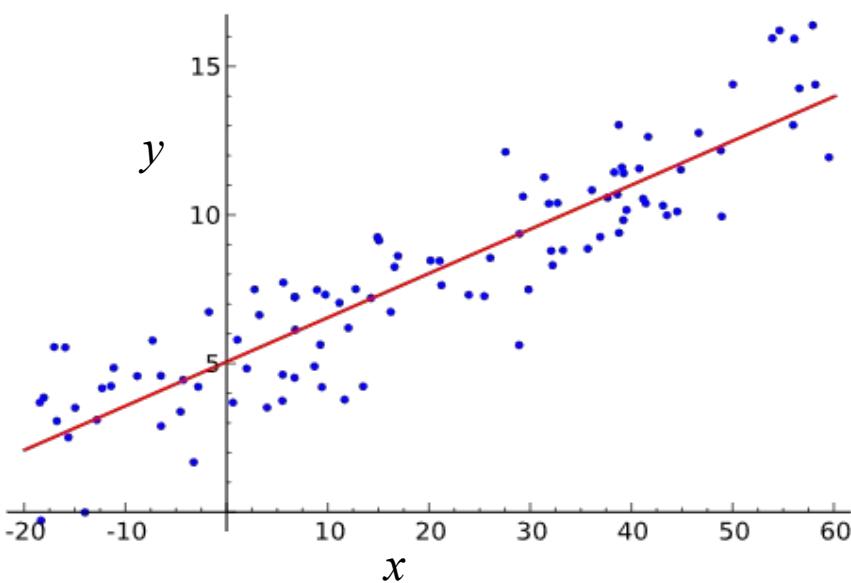
- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
 - x_n : n^{th} input example (independent variable)
 - y_n : Dependent variable (output) corresponding to n^{th} independent variable
- Example: Predicting the salary given the year of experience

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

- Independent variable:
 - Years of experience
- Dependent variable:
 - Salary

Straight-Line (Simple Linear) Regression

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
 - x_n : n^{th} input example (independent variable)
 - y_n : Dependent variable (output) corresponding to n^{th} independent variable
- Function governing the relationship between input and output: $y_n = f(x_n, w, w_0) = w x_n + w_0$
 - The coefficients w_0 and w are parameters of straight-line (regression coefficients)
 - **Unknown**



- Function $f(x_n, w, w_0)$ is a linear function of x_n and it is a linear function of coefficients w and w_0
 - **Linear model for regression**
- The values for the coefficients will be determined by fitting the linear function (straight-line) to the training data

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w x_n + w_0$$

$$(\underline{\hat{y}}_n - \underline{y}_n)^2 \quad \forall n = 1, 2, \dots, N$$

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w x_n + w_0$$

$$E(w, w_0) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w x_n + w_0$$

$$E(w, w_0) = \frac{1}{2} \sum_{n=1}^N (f(x_n, w, w_0) - y_n)^2$$

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w x_n + w_0$$

$$\underset{w, w_0}{\text{minimize}} E(w, w_0) = \frac{1}{2} \sum_{n=1}^N (f(x_n, w, w_0) - y_n)^2$$

- Minimize the error such that the coefficients w_0 and w represent the parameter of line that best fit the training data

Straight-Line (Simple Linear) Regression: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of w and w_0

$$\hat{y}_n = f(x_n, w, w_0) = w x_n + w_0$$

$$\underset{w, w_0}{\text{minimize}} E(w, w_0) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

- The derivatives of error function with respect to the coefficients will be linear in the elements of w and w_0
- Hence the minimization of the error function has unique solution and found in closed form

Straight-Line (Simple Linear) Regression: Training Phase

- Cost function for optimization:

$$E(w, w_0) = \frac{1}{2} \sum_{n=1}^N (f(x_n, w, w_0) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(w, w_0)}{\partial w} = 0 \quad \frac{\partial E(w, w_0)}{\partial w_0} = 0$

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (w x_n + w_0 - y_n)^2}{\partial w} = 0$$

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (w x_n + w_0 - y_n)^2}{\partial w_0} = 0$$

:

- Solving this give optimal \hat{w} and \hat{w}_0 as

$$\hat{w} = \frac{\sum_{n=1}^N (x_n - \mu_x)(y_n - \mu_y)}{\sum_{n=1}^N (x_n - \mu_x)^2}$$

$$\hat{w}_0 = \mu_y - \hat{w}\mu_x$$

- μ_x : sample mean of independent variable x
- μ_y : sample mean of dependent variable y

Straight-Line (Simple Linear) Regression: Testing Phase

- For any test example x , the predicted value is given by:

$$\hat{y} = f(x, \hat{w}, \hat{w}_0) = \hat{w}x + \hat{w}_0$$

- For any \hat{w} and \hat{w}_0 are the optimal parameters of the line learnt during training

Evaluation Metrics for Regression: Squared Error and Mean Squared Error

- The prediction accuracy is measured in terms of squared error:
$$E = (\hat{y} - y)^2$$
 - y : actual value
 - \hat{y} : predicted value
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error (RMSE):

$$E_{\text{RMSE}} = \sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (\hat{y}_n - y_n)^2}$$

- RMSE expressed in % as:

$$E_{\text{RMSE}}(\%) = \frac{\sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (\hat{y}_n - y_n)^2}}{\frac{1}{N_t} \sum_{n=1}^{N_t} y_n} * 100$$

Illustration of Simple Linear Regression: Salary Prediction - Training

Years of experience (x)	Salary (in Rs 1000) (y)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

$$\hat{w} = \frac{\sum_{n=1}^N (x_n - \mu_x)(y_n - \mu_y)}{\sum_{n=1}^N (x_n - \mu_x)^2}$$
$$\hat{w}_0 = \mu_y - \hat{w}\mu_x$$

- μ_x : 9.1
- μ_y : 55.4
- \hat{w} : 3.54
- \hat{w}_0 : 23.21

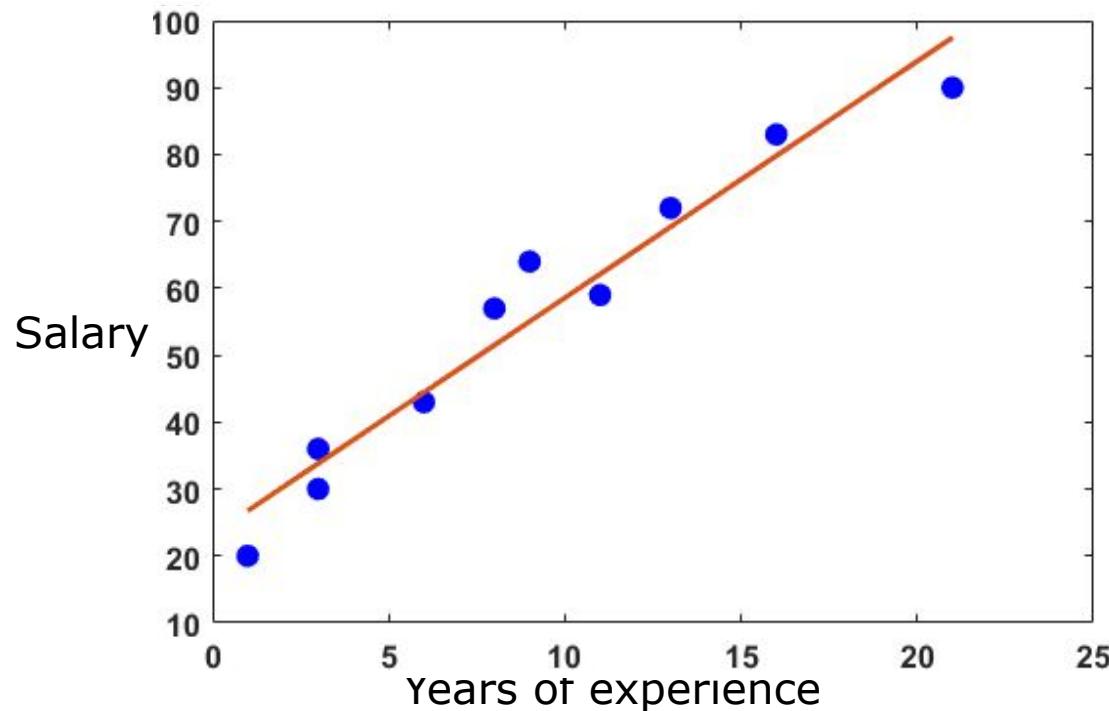
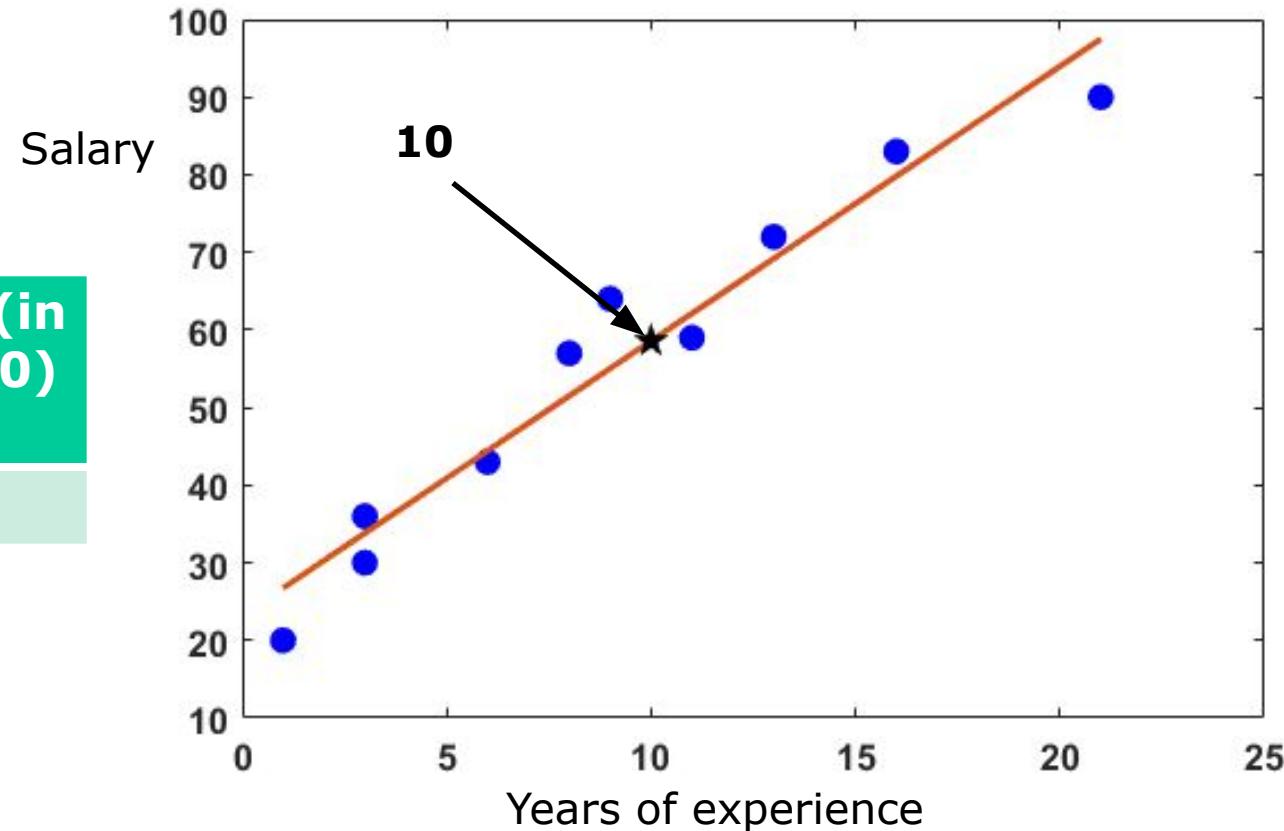


Illustration of Simple Linear Regression: Salary Prediction - Test

- $\hat{w}: 3.54$
- $\hat{w}_0: 23.21$

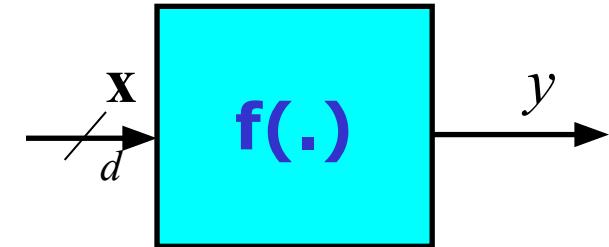
Years of experience (x)	Salary (in Rs 1000) (y)
10	-



- Predicted salary: 58.584
- Actual salary: 58.000
- Squared error: 0.34

Multiple Linear Regression

- Multiple linear regression:
 - Two or more independent variable (\mathbf{x})
 - Single dependent variable (y)
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
 - d : dimension of input example (number of independent variables)
 - \mathbf{x}_n : n^{th} input example (d independent variables)
 - y_n : Dependent variable (output) corresponding to n^{th} input example
- Function governing the relationship between input and output:
$$y_n = f(\mathbf{x}_n, \mathbf{w}) = w_d x_{nd} + \dots + w_2 x_{n2} + w_1 x_{n1} + w_0 = \sum_{i=0}^d w_i x_{ni} = \mathbf{w}^\top \mathbf{x}_n$$
 - The coefficients w_0, w_1, \dots, w_d are collectively denoted by the vector \mathbf{w} - **Unknown**
- Function $f(\mathbf{x}_n, \mathbf{w})$ is a linear function of \mathbf{x}_n and it is a linear function of coefficients \mathbf{w}
 - **Linear model for regression**



Linear Regression: Linear Function Approximation

- Linear function:
 - 2 input variable case (3-dimensional space): The mapping function is a **plane** specified by

$$y = f(\mathbf{x}, \mathbf{w}) = w_2 x_2 + w_1 x_1 + w_0 = 0$$

where $\mathbf{w} = [w_0, w_1, w_2]^T$ and $\mathbf{x} = [1, x_1, x_2]^T$

- d input variable case ($d+1$ -dimensional space): The mapping function is a **hyperplane** specified by

$$y = f(\mathbf{x}, \mathbf{w}) = w_d x_d + \dots + w_2 x_2 + w_1 x_1 + w_0 = \sum_{i=0}^d w_i x_i = \mathbf{w}^T \mathbf{x} = 0$$

where $\mathbf{w} = [w_0, w_1, \dots, w_d]^T$ and $\mathbf{x} = [1, x_1, \dots, x_d]^T$

Multiple Linear Regression: Training Phase

- The values for the coefficients will be determined by fitting the linear function to the training data
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(\mathbf{x}_n, \mathbf{w})$, in the training set for any given value of \mathbf{w}

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w}) = \mathbf{w}^\top \mathbf{x}_n + w_0 = \sum_{i=0}^d w_i x_i$$

$$\text{minimize } E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

- The error function is a
 - quadratic function of the coefficients \mathbf{w} and
 - The derivatives of error function with respect to the coefficients will be linear in the elements of \mathbf{w}
- Hence the minimization of the error function has unique solution and found in closed form

Multiple Linear Regression: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (f(\mathbf{x}_n, \mathbf{w}) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N \left(\sum_{i=0}^d w_i x_{ni} - y_n \right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{x}_n - y_n)^2}{\partial \mathbf{w}} = \mathbf{0}$$

Multiple Linear Regression: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (f(\mathbf{x}_n, \mathbf{w}) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$
- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{x}_n - y_n)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}}$$

– Assumption: $d < N$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nd} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{Nd} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{bmatrix}$$

\mathbf{X} is data matrix

Multiple Linear Regression: Testing Phase

- Optimal coefficient vector \mathbf{w} is given by

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\hat{\mathbf{w}} = \mathbf{X}^+ \mathbf{y}$$

where $\mathbf{X}^+ = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ is the pseudo inverse of matrix \mathbf{X}

- For any test example \mathbf{x} , the predicted value is given by:

$$\hat{y} = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\top \mathbf{x} = \sum_{i=0}^d \hat{w}_i x_i$$

- The prediction accuracy is measured in terms of squared error: $E = (\hat{y} - y)^2$
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (\hat{y}_n - y_n)^2}$$

Illustration of Multiple Linear Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- Training:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

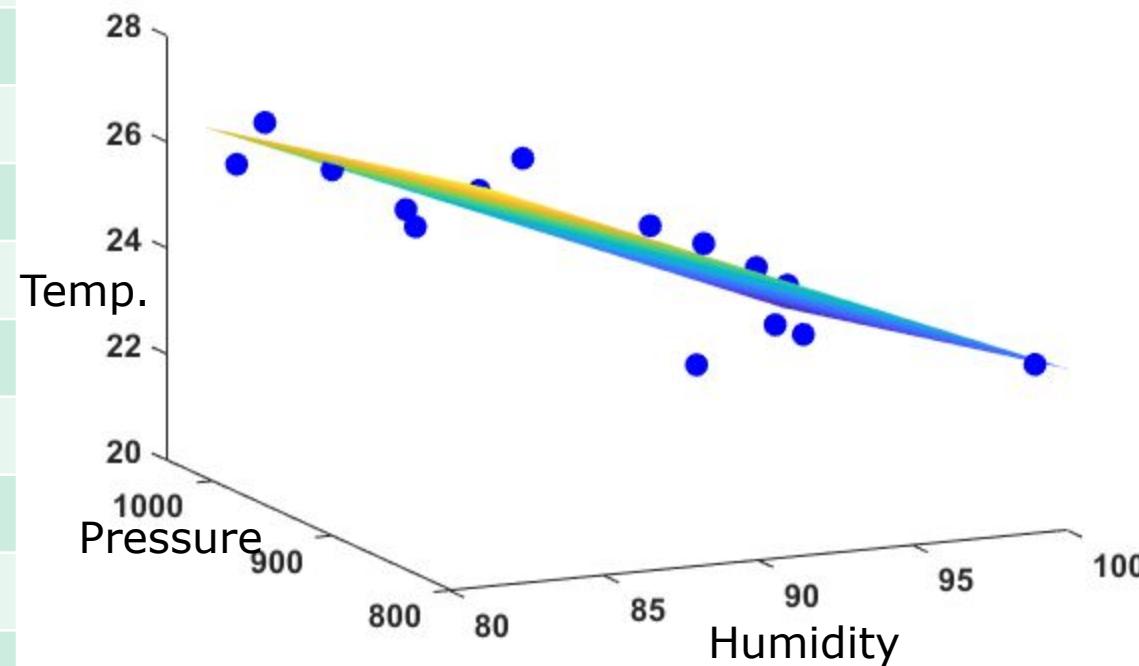
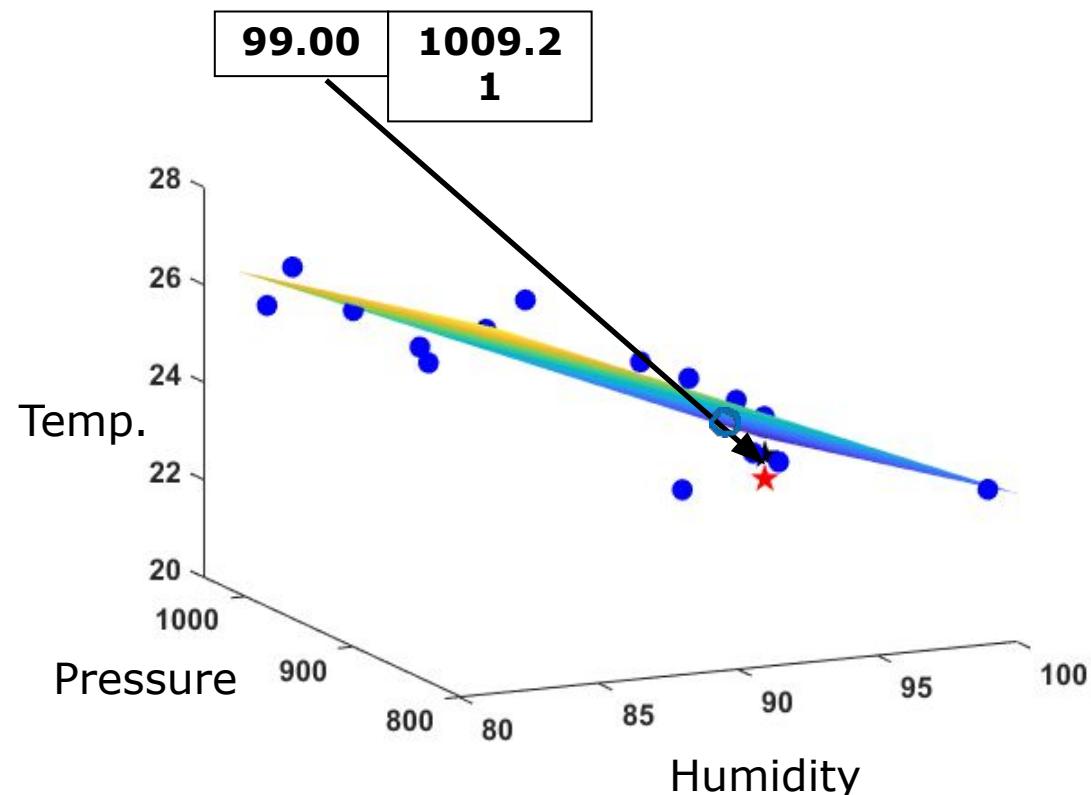


Illustration of Multiple Linear Regression: Temperature Prediction - Test

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

Humidity (x_1)	Pressure (x_2)	Temp (y)
99.00	1009.21	-

$$y = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^T \mathbf{x}$$

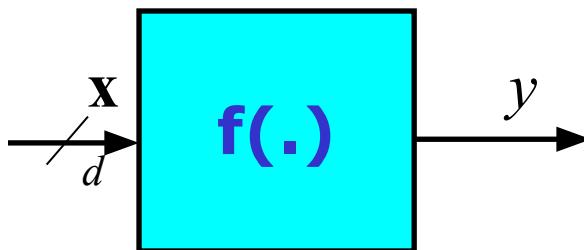


- Predicted temperature: 21.72
- Actual temperature: 21.24
- Squared error: 0.2347

Application of Regression: A Method to Handle Missing Values

- Use most probable value to fill the missing value:
 - Use regression techniques to predict the missing value (regression imputation)
 - Let x_1, x_2, \dots, x_d be a set of d attributes
 - Regression (multivariate): The n^{th} value is predicted as

$$y_n = f(x_{n1}, x_{n2}, \dots, x_{nd})$$



- Simple or Multiple Linear regression:

$$y_n = w_1 x_{n1} + w_2 x_{n2} + \dots + w_d x_{nd}$$

- Popular strategy
- It uses the most information from the present data to predict the missing values
- It preserves the relationship with other variables

Application of Regression: A Method to Handle Missing Values

- Training process:
 - Let y be the attribute, whose missing values to be predicted
 - Training examples: All $\mathbf{x}=[x_1, x_2, \dots, x_d]^T$, a set of d dependent attributes for which the independent variable y is available
 - The values for the coefficients will be determined by fitting the linear function to the training data

1	Dates	Temperature	Humidity	Rain
2	08-07-2018	25.46875	82.1875	6.75
3	09-07-2018	26.19298	83.1491	1761.75
4	10-07-2018	25.17021	85.3404	652.5
5	11-07-2018	NaN	87.6866	963
6	12-07-2018	24.06923	87.6462	254.25
7	13-07-2018	21.20779	95.9481	339.75
8	15-07-2018	23.48571	96.1714	38.25
9	18-07-2018	NaN	98.5897	29.25
10	19-07-2018	25.09346	88.3271	4.5
11	20-07-2018	25.39423	90.4327	112.5
12	21-07-2018	NaN	94.5378	735.75
13	22-07-2018	22.5098	99	607.5
14	23-07-2018	22.904	98	717.75
15	24-07-2018	NaN	99	513
16	25-07-2018	23.18182	98.9697	195.75
17	26-07-2018	21.24222	99	474.75

- Dependent variable: Temperature
- Independent variables: Humidity and Rainfall

Application of Regression: A Method to Handle Missing Values

- Testing process (Prediction):

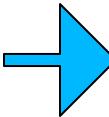
- Optimal coefficient vector w is given by

$$\hat{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- For any test example x , the predicted value is given by:

$$\hat{y} = f(\mathbf{x}, \hat{w}) = \hat{w}^T \mathbf{x} = \sum_{i=0}^d \hat{w}_i x_i$$

	Dates	Temperature	Humidity	Rain
1	08-07-2018	25.46875	82.1875	6.75
2	09-07-2018	26.19298	83.1491	1761.75
3	10-07-2018	25.17021	85.3404	652.5
4	11-07-2018	NaN	87.6866	963
5	12-07-2018	24.06923	87.6462	254.25
6	13-07-2018	21.20779	95.9481	339.75
7	15-07-2018	23.48571	96.1714	38.25
8	18-07-2018	NaN	98.5897	29.25
9	19-07-2018	25.09346	88.3271	4.5
10	20-07-2018	25.39423	90.4327	112.5
11	21-07-2018	NaN	94.5378	735.75
12	22-07-2018	22.5098	99	607.5
13	23-07-2018	22.904	98	717.75
14	24-07-2018	NaN	99	513
15	25-07-2018	23.18182	98.9697	195.75
16	26-07-2018	21.24272	99	174.75



	Dates	Temperature	Humidity	Rain
1	08-07-2018	25.46875	82.1875	6.75
2	09-07-2018	26.19298	83.1491	1761.75
3	10-07-2018	25.17021	85.3404	652.5
4	11-07-2018	24.2	87.6866	963
5	12-07-2018	24.06923	87.6462	254.25
6	13-07-2018	21.20779	95.9481	339.75
7	15-07-2018	23.48571	96.1714	38.25
8	18-07-2018	21.5	98.5897	29.25
9	19-07-2018	25.09346	88.3271	4.5
10	20-07-2018	25.39423	90.4327	112.5
11	21-07-2018	23.7	94.5378	735.75
12	22-07-2018	22.5098	99	607.5
13	23-07-2018	22.904	98	717.75
14	24-07-2018	21.6	99	513
15	25-07-2018	23.18182	98.9697	195.75
16	26-07-2018	21.24272	99	174.75

Summary: Regression

- Regression analysis is used to model the relationship between one or more independent (**predictor**) variable and a dependent (**response**) variable
- Response is some function of one or more input variables
- **Linear regression:** Response is linear function of one or more input variables
 - If the response is linear function of one input variable, then it is **simple linear regression** (**straight-line fitting**)
 - If the response is linear function of two or more input variable, then it is **multiple linear regression** (**linear surface fitting** or **hyperplane fitting**)

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

Supervised Machine Learning:

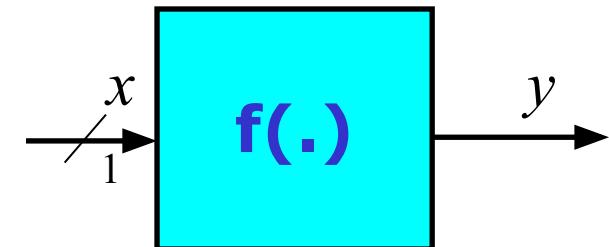
Regression

Nonlinear Regression

Nonlinear Regression

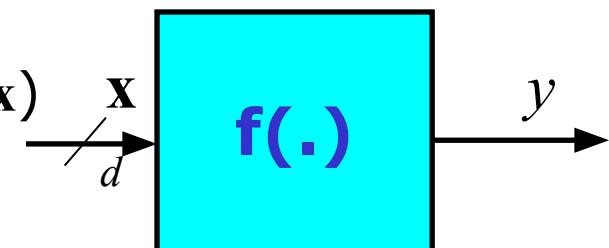
- Nonlinear approach to model the relationship between a scalar response, (y) (or **dependent** variable) and one or more predictor variables, (x or \mathbf{x}) (or **independent** variables)
- The response is going to be the **nonlinear function** of input (one or more independent variables)
- Simple nonlinear regression (**Polynomial curve fitting, Neural Network**):

- Single independent variable (x)
- Single dependent variable (y)
- *Fitting a curve*



- Multiple nonlinear regression (**Polynomial regression, Neural Network**):

- Two or more independent variable (\mathbf{x})
- Single dependent variable (y)
- *Fitting a surface*

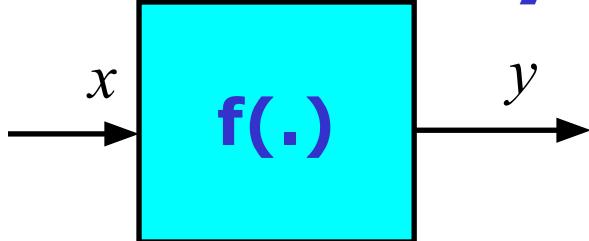


Supervised Machine Learning:

Regression

Polynomial Curve Fitting

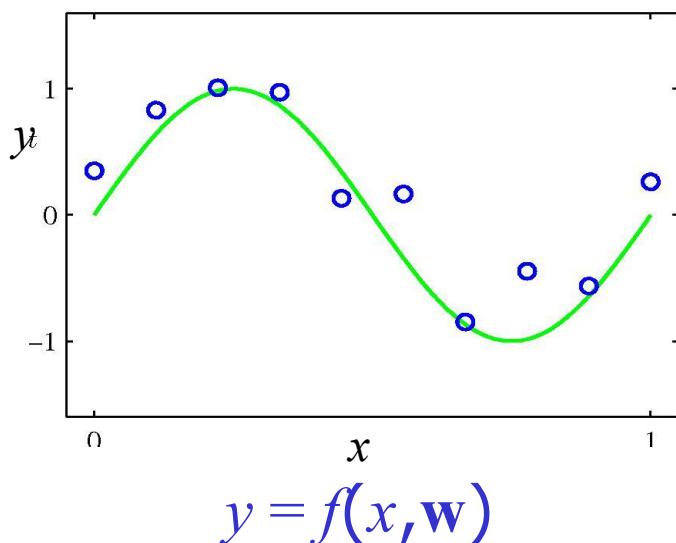
Polynomial Curve Fitting



- Given:-Training data:
 $D = \{x_n, y_n\}_{n=1}^N, x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a **polynomial function of degree p** :

$$y_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \dots + w_p x_n^p = \sum_{j=0}^p w_j x_n^j$$

- Here, $1, x_n, x_n^2, x_n^3, \dots, x_n^p$ are the monomials of polynomial up to degree p
- The coefficients $\mathbf{w} = [w_0, w_1, \dots, w_p]$ are parameters of polynomial curve (**regression coefficients**) - **Unknown**
- Polynomial function $f(x_n, \mathbf{w})$ is a **nonlinear function** of x_n and
- Function $f(x_n, \mathbf{w})$ is a **linear function** of coefficients \mathbf{w}
 - **Linear model for regression**



Polynomial Curve Fitting: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of \underline{w}

$$\hat{y}_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \dots + w_p x_n^p$$

$$\underset{\mathbf{w}}{\text{minimize}} E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

- Minimize the error such that the coefficients w represent the parameter of polynomial curve that best fit the training data

Polynomial Curve Fitting: Training Phase

- Given:- Training data: $D = \{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^1$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(x_n, w, w_0)$, in the training set for any given value of w

$$\hat{y}_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \dots + w_p x_n^p$$

$$\underset{\mathbf{w}}{\text{minimize}} E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

- The error function is a
 - quadratic function of the coefficients w and
 - The derivatives of error function with respect to the coefficients will be linear in the elements of w
- Hence the minimization of the error function has unique solution and found in closed form

Polynomial Curve Fitting: Training Phase

$$\hat{y}_n = f(x_n, \mathbf{w}) = w_0 + w_1 x_n + w_2 x_n^2 + \dots + w_p x_n^p = \sum_{j=0}^p w_j x_n^j$$

- Let's consider: $x_n \quad x_n^2 \quad x_n^3 \quad \dots \quad x_n^p \quad p$ is degree of polynomial

$$\downarrow \quad \downarrow \quad \downarrow \quad \dots \quad \downarrow$$

$$z_{n1} \quad z_{n2} \quad z_{n3} \quad \dots \quad z_{np}$$

$$\hat{y}_n = f(\mathbf{z}_n, \mathbf{w}) = w_0 + w_1 z_{n1} + w_2 z_{n2} + \dots + w_p z_{np}$$

$$\hat{y}_n = f(\mathbf{z}_n, \mathbf{w}) = \sum_{j=0}^p w_j z_{nj} = \mathbf{w}^\top \mathbf{z}_n$$

where $\mathbf{w} = [w_0, w_1, \dots, w_p]^\top$ and $\mathbf{z}_n = [1, z_{n1}, \dots, z_{np}]^\top$

Polynomial Curve Fitting: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (f(\mathbf{z}_n, \mathbf{w}) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$
- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N \left(\sum_{j=0}^p w_j z_{nj} - y_n \right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{z}_n - y_n)^2}{\partial \mathbf{w}} = \mathbf{0}$$

Polynomial Curve Fitting: Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (f(\mathbf{z}_n, \mathbf{w}) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$
- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \mathbf{z}_n - y_n)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\hat{\mathbf{w}} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y}$$

– Assumption: $p < N$

\mathbf{Z} is Vandermonde matrix

$$\mathbf{Z} = \begin{bmatrix} 1 & z_{11} & z_{12} & \dots & z_{1p} \\ 1 & z_{21} & z_{22} & \dots & z_{2p} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{n1} & z_{n2} & \dots & z_{np} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{N1} & z_{N2} & \dots & z_{Np} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ \vdots \\ y_N \end{bmatrix}$$

where, $z_{nj} = x_n^j$

Polynomial Curve Fitting: Testing

- Optimal coefficient vector \mathbf{w} is given by

$$\hat{\mathbf{w}} = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top \mathbf{y}$$

$$\hat{\mathbf{w}} = \mathbf{Z}^+ \mathbf{y}$$

where $\mathbf{Z}^+ = (\mathbf{Z}^\top \mathbf{Z})^{-1} \mathbf{Z}^\top$ is the pseudo inverse of matrix \mathbf{Z}

- For any test example x , the predicted value is given by:

$$\hat{y} = f(x, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\top \mathbf{z} = \sum_{j=0}^p \hat{w}_i x^j$$

- The prediction accuracy is measured in terms of squared error: $E = (\hat{y} - y)^2$
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (\hat{y}_n - y_n)^2}$$

Determining p , Degree of Polynomial

- This is determined experimentally
- Starting with $p=1$, test set is used to estimate the accuracy, in terms of error, of the regression model
 - Note: The polynomial degree $p=1$ is equivalent to simple linear regression (straight-line regression)
- This process is repeated each time by incrementing p
- The regression model with p that gives the minimum error on test set may be selected

Illustration of Polynomial Curve Fitting: Humidity Prediction - Training

Temp (x)	Humidity (y)
25.47	82.19
26.19	83.15
25.17	85.34
24.30	87.69
24.07	87.65
21.21	95.95
23.49	96.17
21.79	98.59
25.09	88.33
25.39	90.43
23.89	94.54
22.51	99.00
22.90	98.00
21.72	99.00
23.18	98.97

- Degree of polynomial p : 1

$$\hat{\mathbf{w}} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{y} \quad \mathbf{Z} \text{ is } 15 \times 2 \text{ matrix}$$

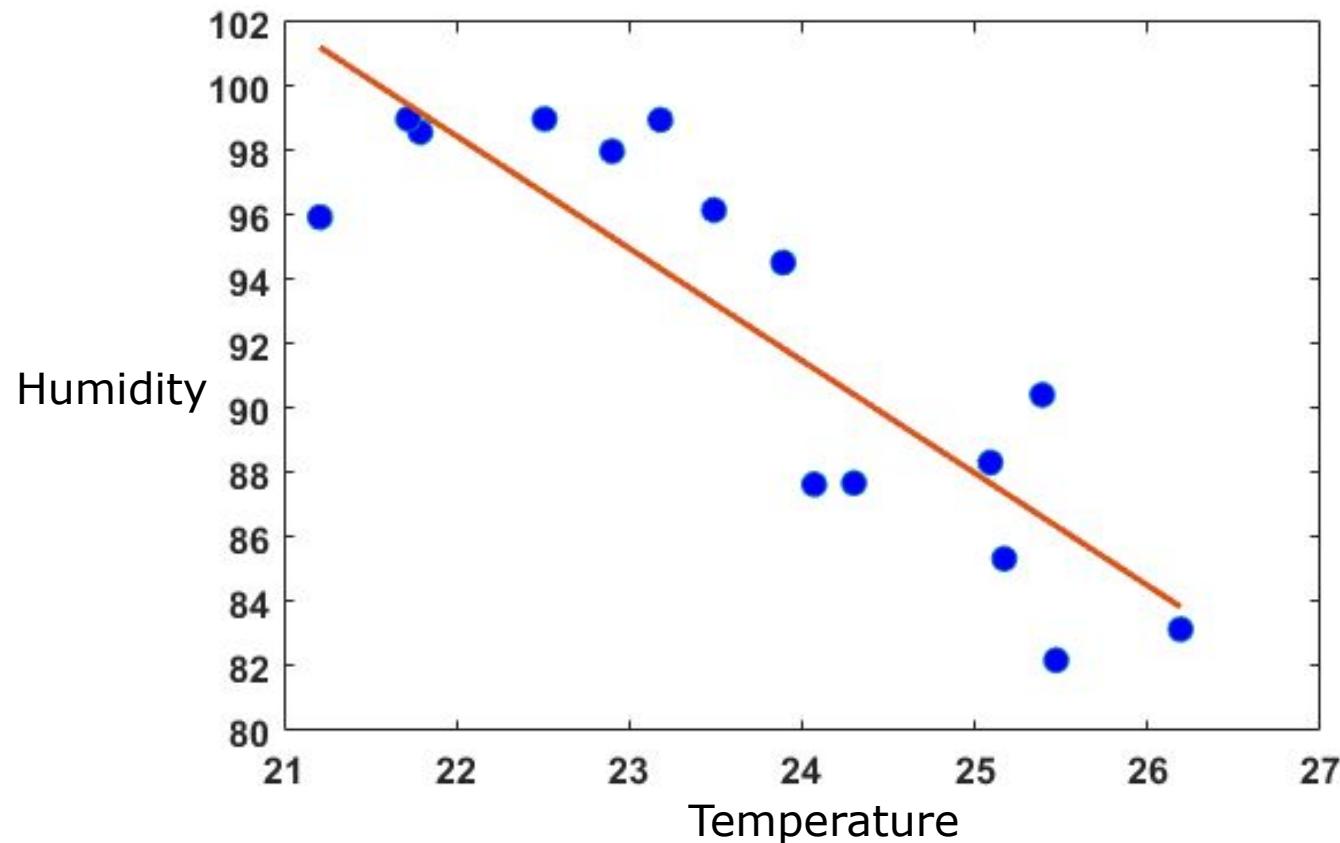
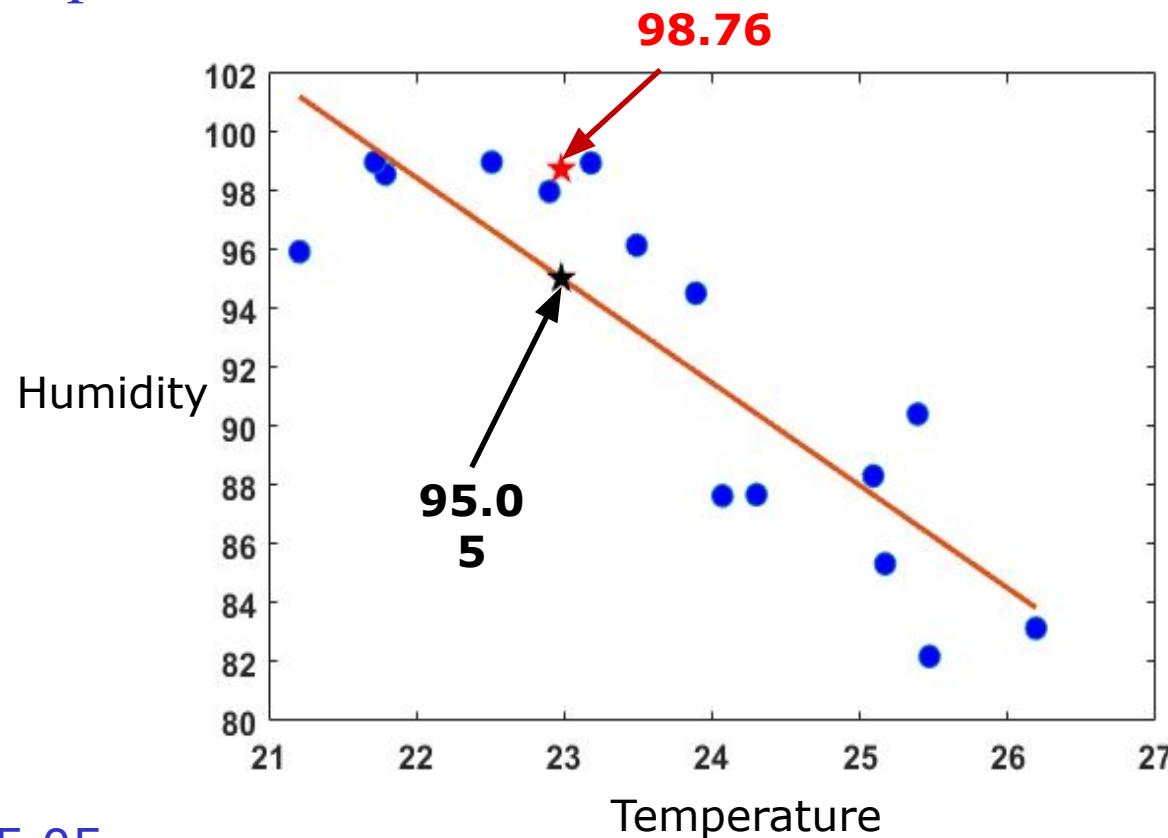


Illustration of Polynomial Curve Fitting: Humidity Prediction - Test

- Degree of polynomial p : 1

Temp (x)	Humidity (y)
22.98	--



- Predicted humidity: 95.05
- Actual humidity: 98.76
- Squared error: 13.77

Illustration of Polynomial Curve Fitting: Humidity Prediction - Training

Temp (x)	Humidity (y)
25.47	82.19
26.19	83.15
25.17	85.34
24.30	87.69
24.07	87.65
21.21	95.95
23.49	96.17
21.79	98.59
25.09	88.33
25.39	90.43
23.89	94.54
22.51	99.00
22.90	98.00
21.72	99.00
23.18	98.97

- Degree of polynomial p : 2

$$\hat{\mathbf{w}} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{y} \quad \mathbf{Z} \text{ is } 15 \times 3 \text{ matrix}$$

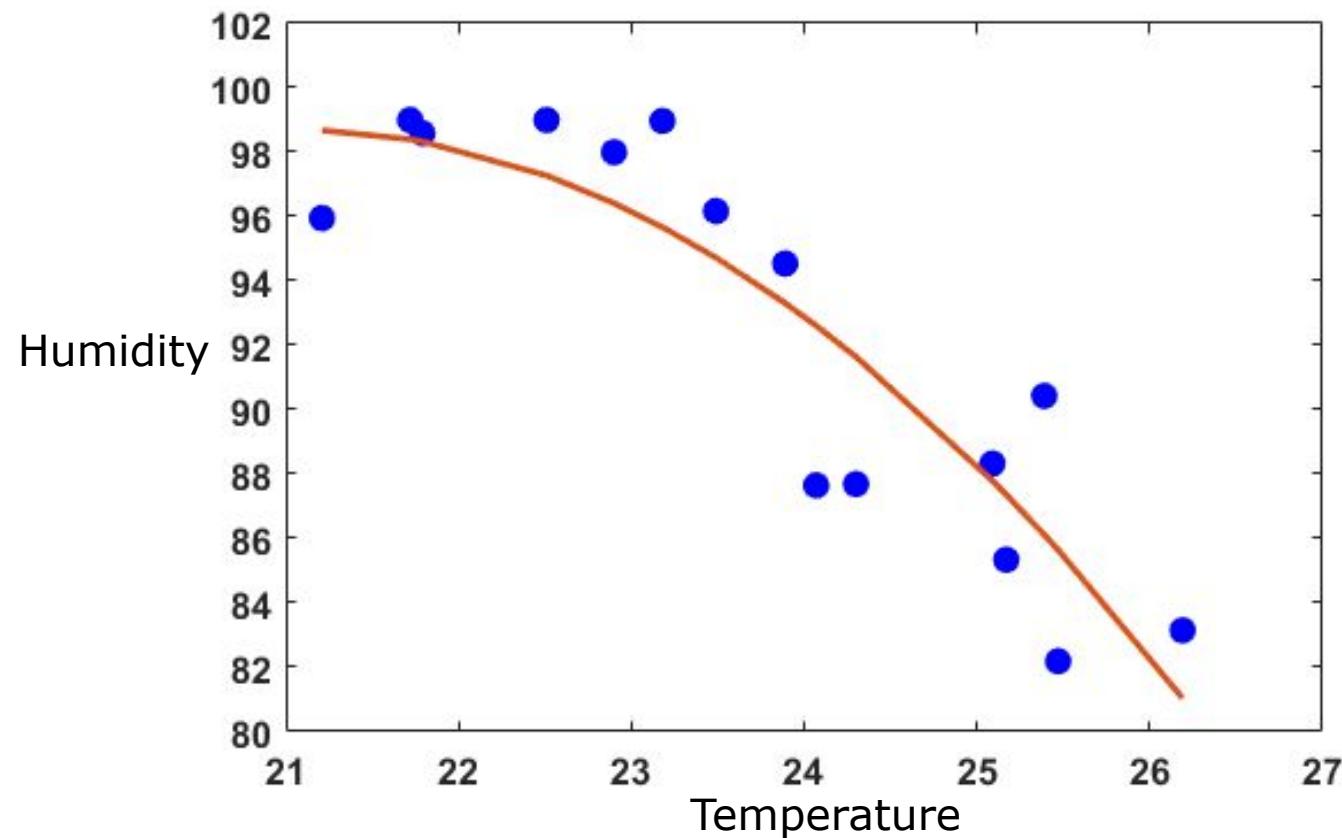
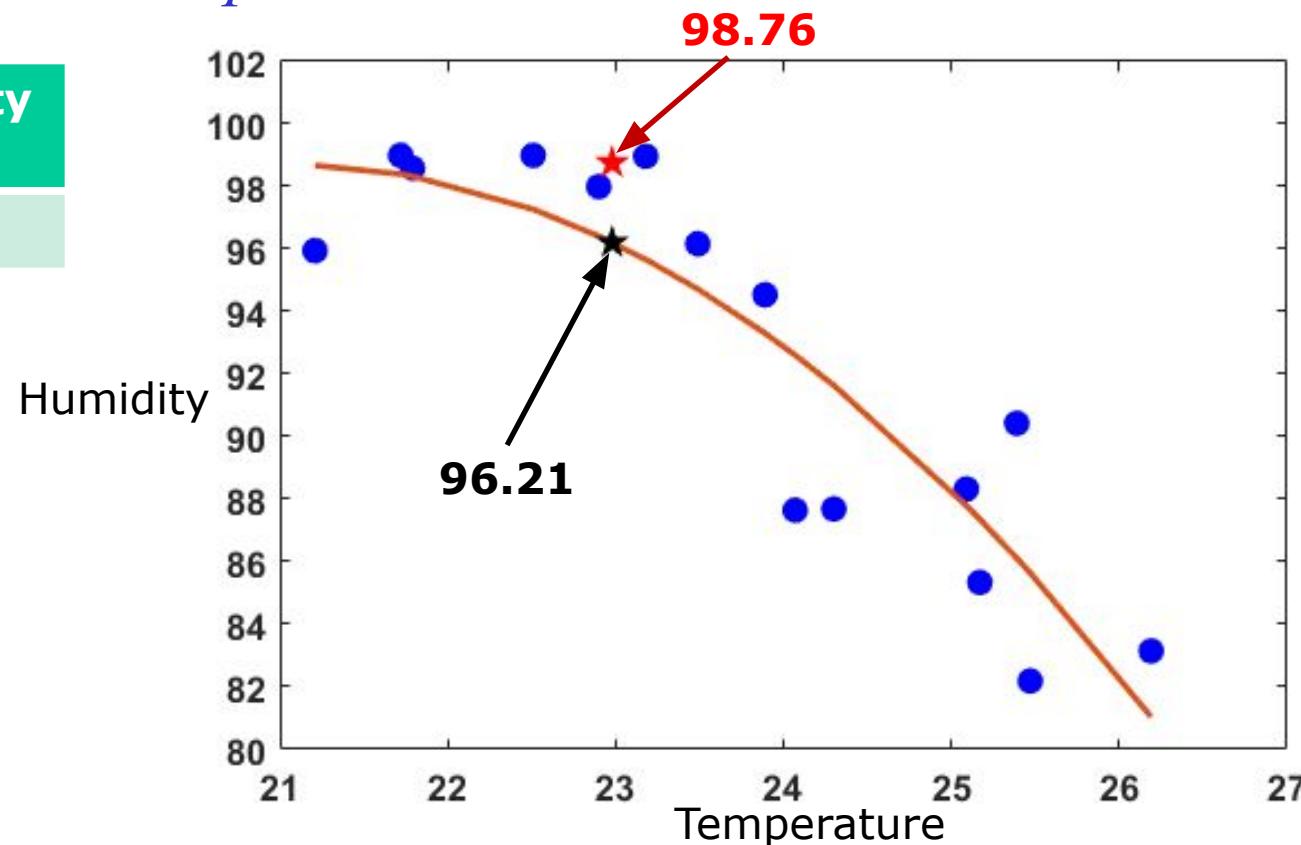


Illustration of Polynomial Curve Fitting: Humidity Prediction - Test

- Degree of polynomial p : 2

Temp (x)	Humidity (y)
22.98	--



- Predicted humidity: 96.21
- Actual humidity: 98.76
- Squared error: 06.49

Illustration of Polynomial Curve Fitting: Humidity Prediction - Training

Temp (x)	Humidity (y)
25.47	82.19
26.19	83.15
25.17	85.34
24.30	87.69
24.07	87.65
21.21	95.95
23.49	96.17
21.79	98.59
25.09	88.33
25.39	90.43
23.89	94.54
22.51	99.00
22.90	98.00
21.72	99.00
23.18	98.97

- Degree of polynomial p : 3

$$\hat{\mathbf{w}} = (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{y} \quad \mathbf{Z} \text{ is } 15 \times 4 \text{ matrix}$$

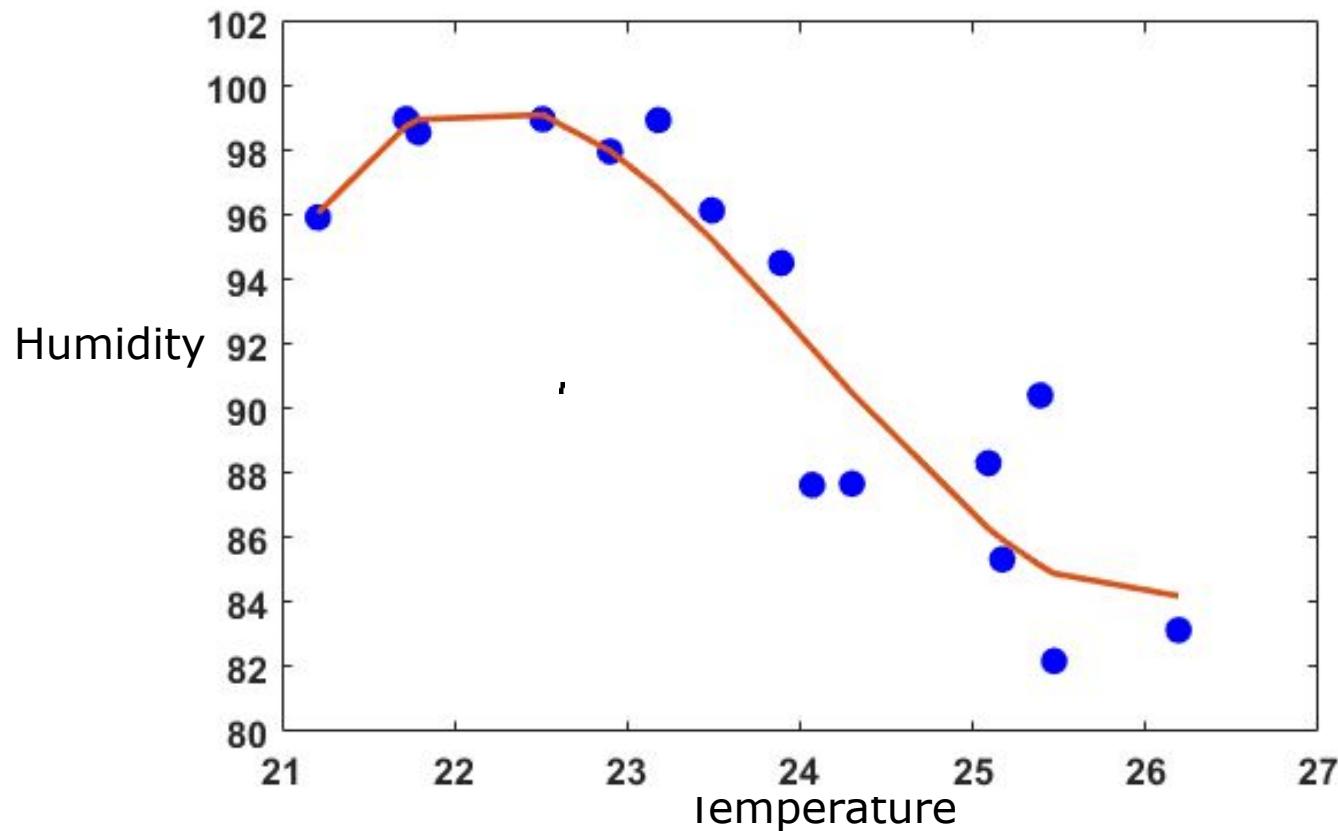
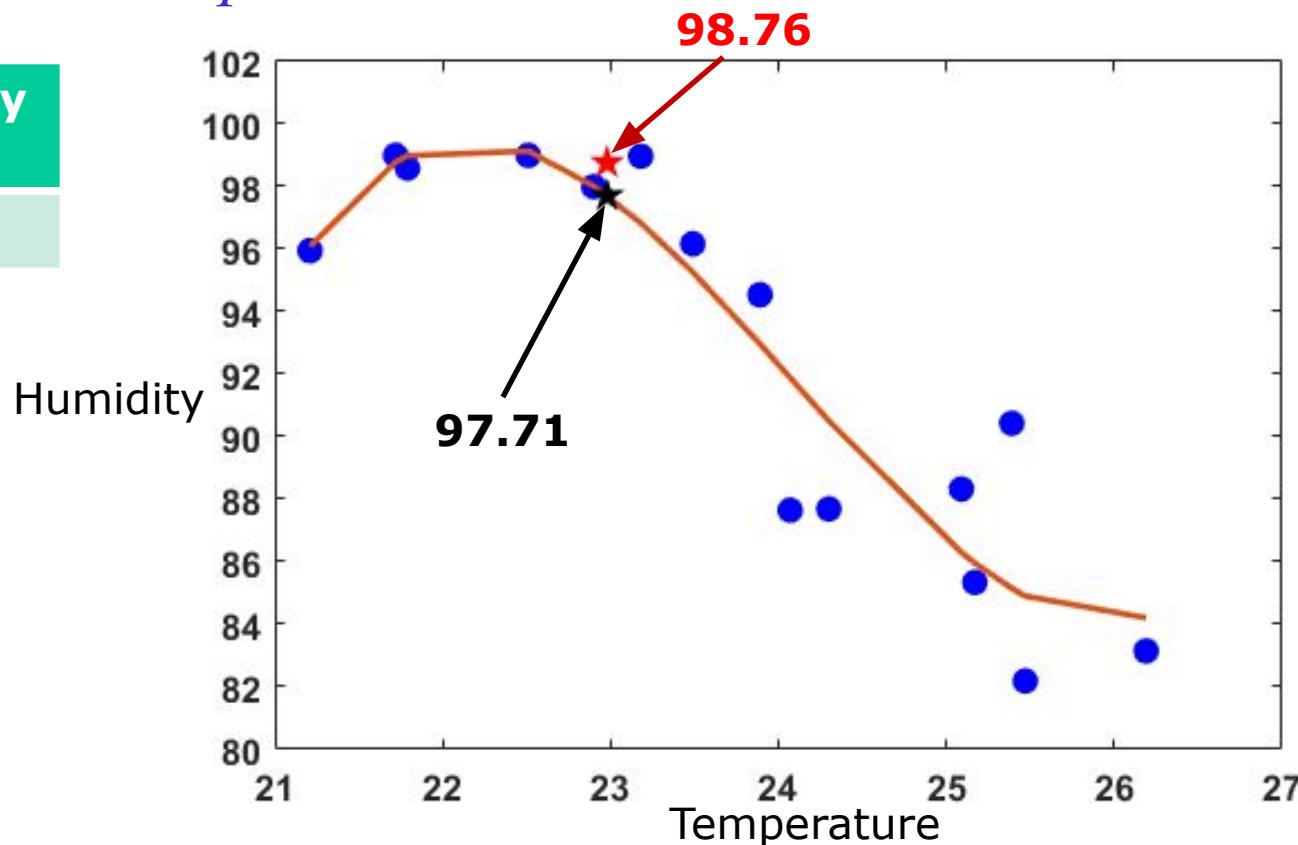


Illustration of Polynomial Curve Fitting: Humidity Prediction - Test

- Degree of polynomial p : 3

Temp (x)	Humidity (y)
22.98	--

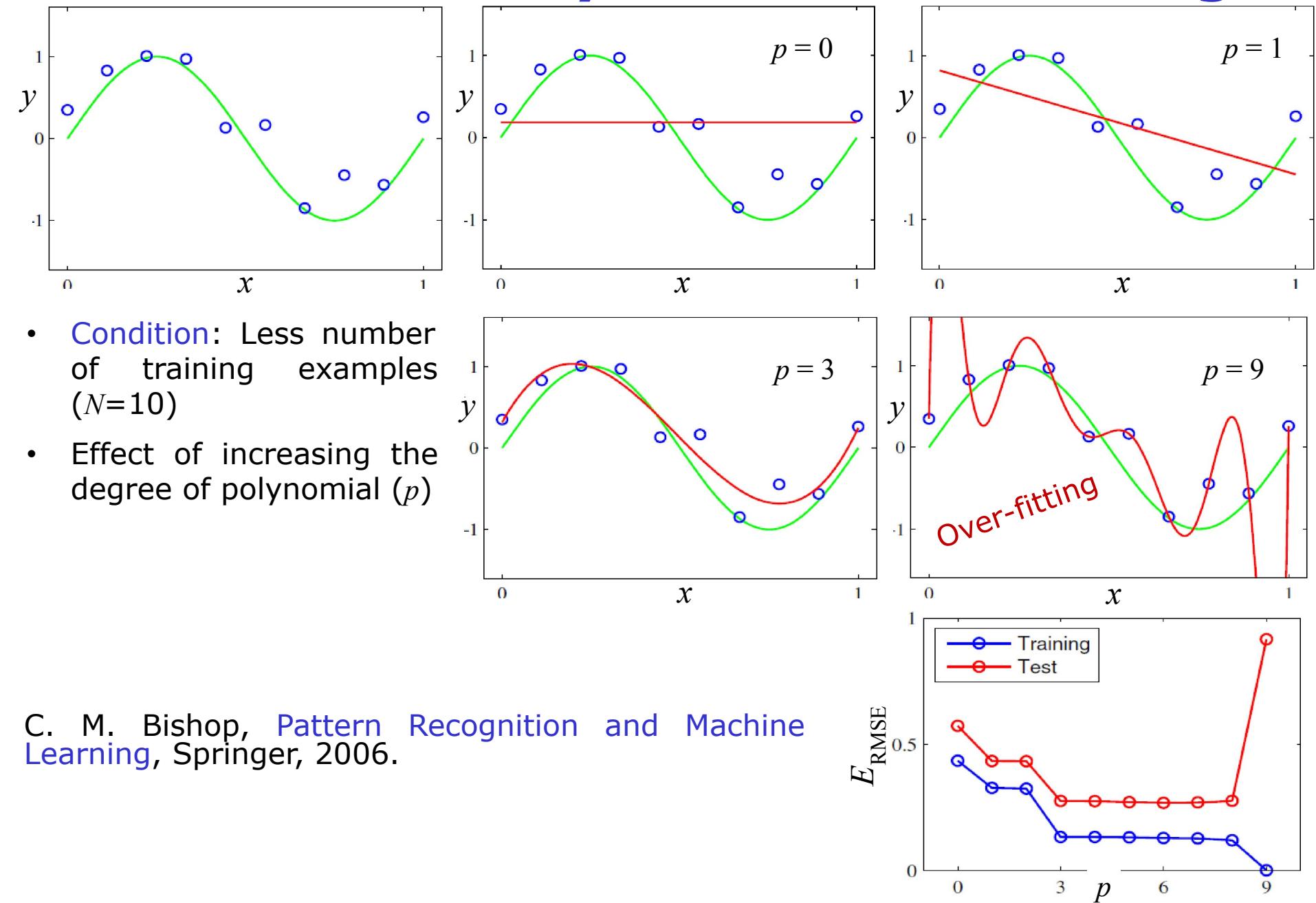


- Predicted humidity: 97.71
- Actual humidity: 98.76
- Squared error: 01.11

$$\underline{p < N}$$

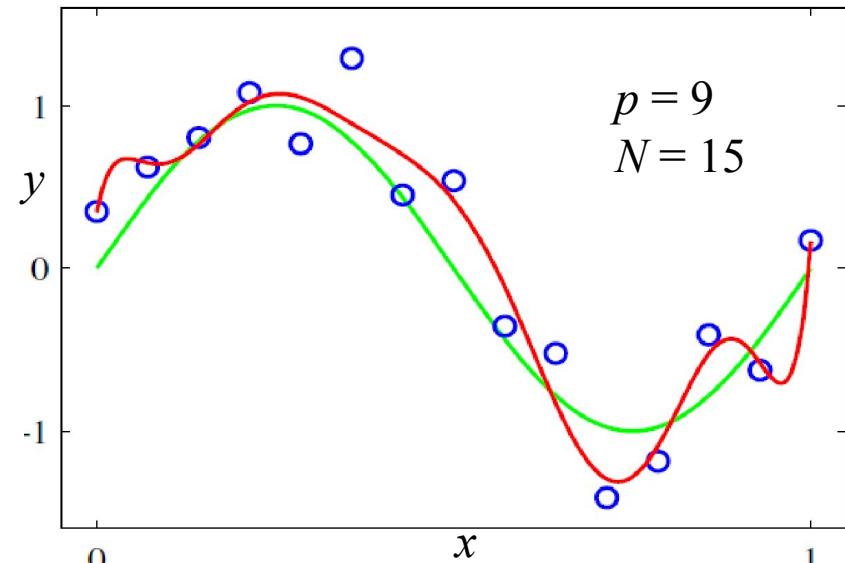
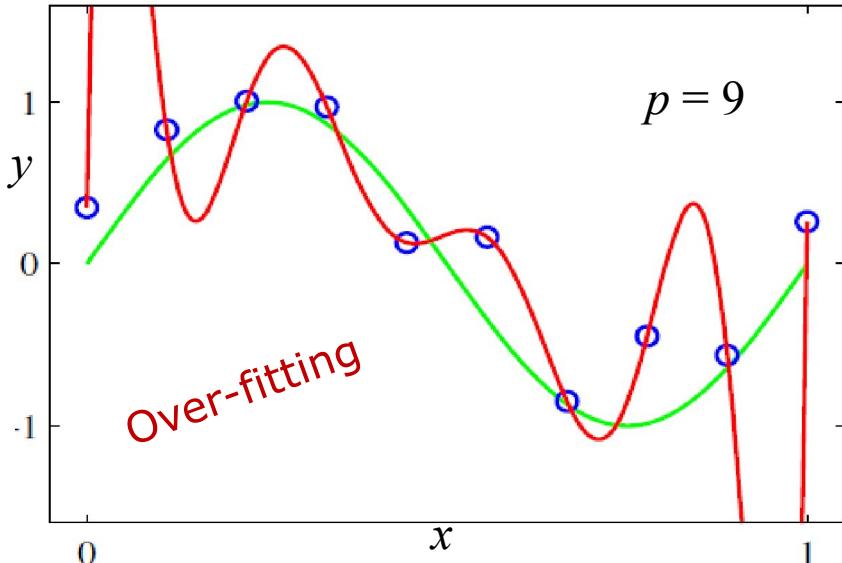
$$\underline{r > N}$$

Illustration: Polynomial Curve Fitting

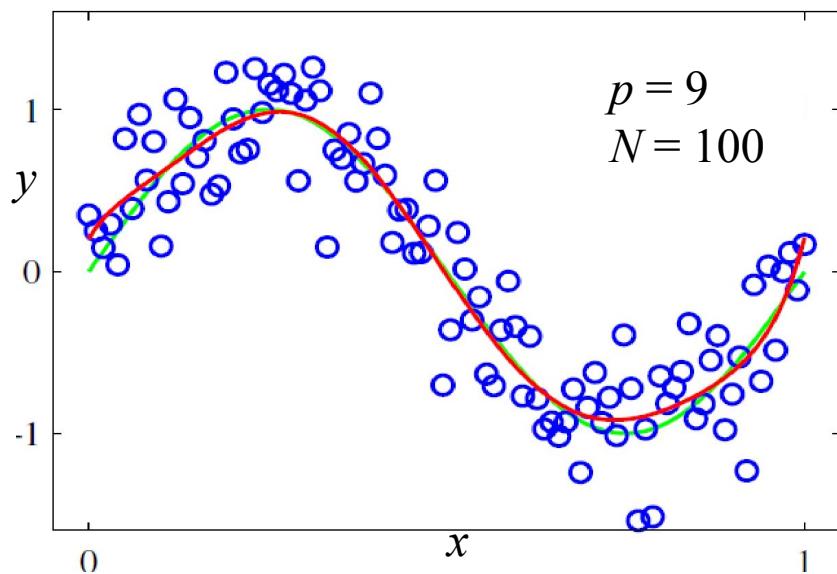


C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Illustration: Polynomial Curve Fitting



- Increasing the size of the data set **reduces** the over-fitting problem



Supervised Machine Learning:

Regression

Polynomial Regression

Nonlinear Regression: Polynomial Regression

- Polynomial regression:
 - Two or more independent variable (\mathbf{x}) $\xrightarrow[d]{\quad}$
 - Single dependent variable (y)
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p :

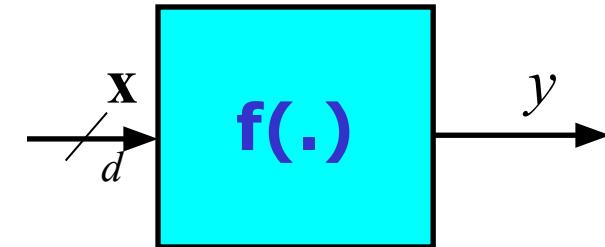
$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

- m is the number of monomials of polynomial up to degree p
- $\varphi_j(\mathbf{x}_n)$ is the j th monomial of degree p for \mathbf{x}_n
- For 2-dimensional input, $\mathbf{x}_n = [x_{n1}, x_{n2}]^\top$ and degree, $p=2$

$$\boldsymbol{\varphi}(\mathbf{x}_n) = [\varphi_0(\mathbf{x}_n), \varphi_1(\mathbf{x}_n), \varphi_2(\mathbf{x}_n), \varphi_3(\mathbf{x}_n), \varphi_4(\mathbf{x}_n), \varphi_5(\mathbf{x}_n)]^\top$$

$$\boldsymbol{\varphi}(\mathbf{x}_n) = [1, x_{n1}, x_{n2}, x_{n1}^2, x_{n2}^2, x_{n1}x_{n2}]^\top$$

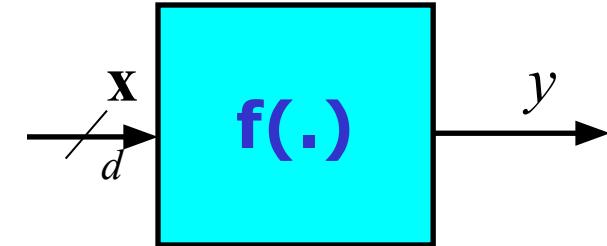
$$m = 6$$



Nonlinear Regression: Polynomial Regression

- Polynomial regression:

- Two or more independent variable (\mathbf{x})
- Single dependent variable (y)



- Given:- **Training data**: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a **polynomial function of degree p** :

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

- m is the number of monomials of polynomial up to degree p
- $\varphi_j(\mathbf{x}_n)$ is the j th monomial of degree p for \mathbf{x}_n
- For 2-dimensional input, $\mathbf{x}_n = [x_{n1}, x_{n2}]^\top$ and degree, $p=2$

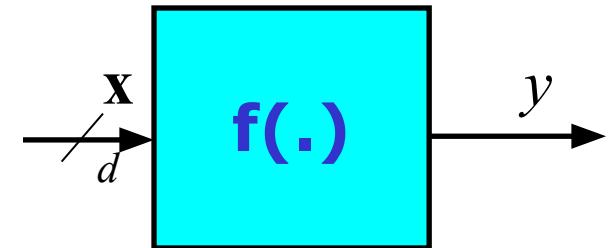
$$y_n = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = w_0 + w_1 x_{n1} + w_2 \overline{x}_{n2} + w_3 x_{n1}^2 + w_4 x_{n2}^2 + w_5 x_{n1} x_{n2}$$

$$y_n = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = w_0 \varphi_0(\mathbf{x}_n) + w_1 \varphi_1(\mathbf{x}_n) + w_2 \varphi_2(\mathbf{x}_n) + w_3 \varphi_3(\mathbf{x}_n) + w_4 \varphi_4(\mathbf{x}_n) + w_5 \varphi_5(\mathbf{x}_n)$$

Nonlinear Regression: Polynomial Regression

- Polynomial regression:

- Two or more independent variable (\mathbf{x})
- Single dependent variable (y)



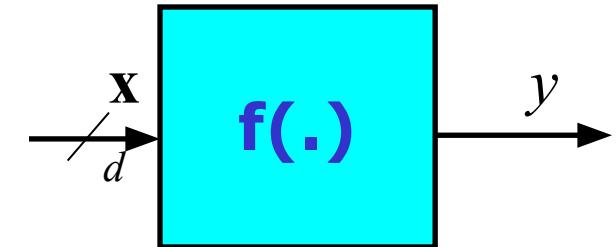
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a **polynomial function of degree p** :

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

- m is the number of monomials of polynomial up to degree p
- $\varphi_j(\mathbf{x}_n)$ is the j th monomial of degree p for \mathbf{x}_n
- For 2-dimensional input, $\mathbf{x}=[x_1, x_2]^T$ and degree, $p=3$

Nonlinear Regression: Polynomial Regression

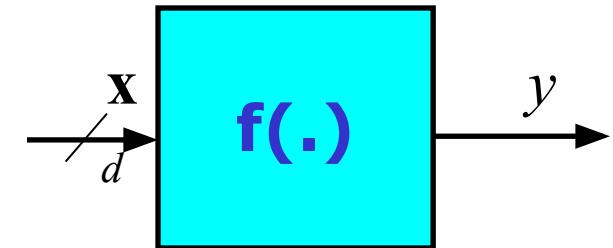
- Polynomial regression:
 - Two or more independent variable (\mathbf{x})
 - Single dependent variable (y)
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p :
$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$
 - m is the number of monomials of polynomial up to degree p
 - $\varphi_j(\mathbf{x}_n)$ is the j th monomial of degree p for \mathbf{x}_n
- For 3-dimensional input, $\mathbf{x} = [x_1, x_2, x_3]^T$ and degree, $p=2$



Nonlinear Regression: Polynomial Regression

- Polynomial regression:

- One or more independent variable (\mathbf{x})
- Single dependent variable (y)



- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a **polynomial function of degree p** :

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

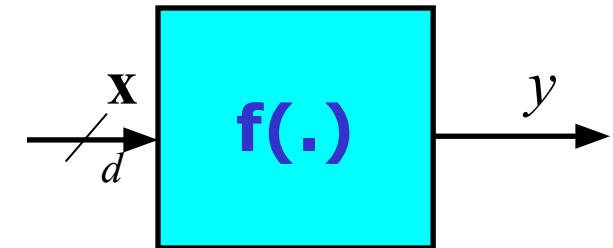
- m is the number of monomials of polynomial up to degree p
- $\varphi_j(\mathbf{x}_n)$ is the j th monomial of degree p for \mathbf{x}_n

The number of monomials m for the polynomial of degree p and the dimension of d is given by $m = \frac{(d+p)!}{d! p!}$

Nonlinear Regression: Polynomial Regression

- Polynomial regression:

- Two or more independent variable (\mathbf{x})
- Single dependent variable (y)



- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a **polynomial function of degree p** :

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

- m is the number of monomials of polynomial up to degree p
- $\varphi_j(\mathbf{x}_n)$ is the j th monomial of degree p for \mathbf{x}_n

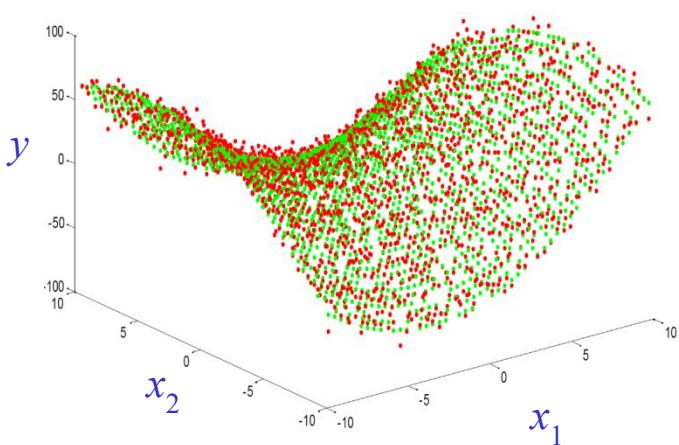
$m = \frac{(d+p)!}{d! p!}$ **Example:** Let the dimension of input variable is $d=6$ and the polynomial of degree $p=3$

- The number of monomials $m = 84$

Nonlinear Regression: Polynomial Regression

- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Function governing the relationship between input and output given by a polynomial function of degree p :

$$y_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$



$$y = f(\mathbf{x}_n, \mathbf{w})$$

$$\mathbf{x} = [x_1, x_2]^\top$$

Fitting a surface

- The coefficients $\mathbf{w} = [w_0, w_1, \dots, w_{m-1}]$ are parameters of surface (polynomial function) (regression coefficients) - **Unknown**
- Polynomial function $f(\mathbf{x}_n, \mathbf{w})$ is a nonlinear function of \mathbf{x}_n and
- Function $f(\mathbf{x}_n, \mathbf{w})$ is a linear function of coefficients \mathbf{w}
 - **Linear model for regression**

Polynomial Regression: Training Phase

- The values for the coefficients will be determined by fitting the linear function to the training data
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(\mathbf{x}_n, \mathbf{w})$, in the training set for any given value of \mathbf{w}

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$
$$\underset{\mathbf{w}}{\text{minimize}} E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

- Minimize the error such that the coefficients \mathbf{w} represent the parameter of polynomial curve that best fit the training data

Polynomial Regression: Training Phase

- The values for the coefficients will be determined by fitting the linear function to the training data
- Given:- Training data: $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}^1$
- Method of least squares:** Minimizes the sum of the squared error between
 - all the actual data (y_n) i.e. actual dependent variable and
 - the estimate of line (predicted dependent variable (\hat{y}_n)) i.e. the function $f(\mathbf{x}_n, \mathbf{w})$, in the training set for any given value of \mathbf{w}

$$\hat{y}_n = f(\mathbf{x}_n, \mathbf{w}) = f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

$$\text{minimize } E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

- The error function is a
 - quadratic function of the coefficients \mathbf{w} and
 - The derivatives of error function with respect to the coefficients will be linear in the elements of \mathbf{w}
- Hence the minimization of the error function has unique solution and found in closed form

Polynomial Regression : Training Phase

$$\hat{y}_n = f(\underline{\mathbf{x}}_n, \mathbf{w})$$
$$\hat{y}_n = f(\underline{\Phi}(\mathbf{x}_n), \mathbf{w})$$

$$\hat{y}_n = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n)$$

$$\hat{y}_n = \mathbf{w}^\top \underline{\Phi}(\mathbf{x}_n)$$

where $\mathbf{w} = [w_0, w_1, \dots, w_{m-1}]^\top$ and

$$\underline{\Phi}(\mathbf{x}_n) = [\varphi_0(\mathbf{x}_n), \varphi_1(\mathbf{x}_n), \varphi_2(\mathbf{x}_n), \dots, \varphi_{m-1}(\mathbf{x}_n)]^\top$$

Polynomial Regression : Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N \left(\sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x}_n) - y_n \right)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}_n) - y_n)^2}{\partial \mathbf{w}} = \mathbf{0}$$

Polynomial Regression : Training Phase

- Cost function for optimization:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (f(\boldsymbol{\varphi}(\mathbf{x}_n), \mathbf{w}) - y_n)^2$$

- Conditions for optimality: $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = 0$

- Application of optimality conditions gives optimal $\hat{\mathbf{w}}$:

$$\frac{\partial \frac{1}{2} \sum_{n=1}^N (\mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}_n) - y_n)^2}{\partial \mathbf{w}} = \mathbf{0}$$

$$\hat{\mathbf{w}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{y}$$

- Assumption: $m < N$

$$\boldsymbol{\Phi} = \begin{bmatrix} \varphi_0(\mathbf{x}_1) & \varphi_1(\mathbf{x}_1) & \dots & \varphi_{m-1}(\mathbf{x}_1) \\ \varphi_0(\mathbf{x}_2) & \varphi_1(\mathbf{x}_2) & \dots & \varphi_{m-1}(\mathbf{x}_2) \\ \hline \vdots & \vdots & \ddots & \vdots \\ \varphi_0(\mathbf{x}_n) & \varphi_1(\mathbf{x}_n) & \dots & \varphi_{m-1}(\mathbf{x}_n) \\ \hline \vdots & \vdots & \ddots & \vdots \\ \varphi_0(\mathbf{x}_N) & \varphi_1(\mathbf{x}_N) & \dots & \varphi_{m-1}(\mathbf{x}_N) \end{bmatrix}$$

Polynomial Regression: Testing

- Optimal coefficient vector \mathbf{w} is given by

$$\hat{\mathbf{w}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

$$\hat{\mathbf{w}} = \Phi^+ \mathbf{y}$$

where $\Phi^+ = (\Phi^\top \Phi)^{-1} \Phi^\top$ is the pseudo inverse of matrix Φ

- For any test example \mathbf{x} , the predicted value is given by:

$$\hat{y} = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\top \varphi(\mathbf{x}) = \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x})$$

- The prediction accuracy is measured in terms of squared error: $E = (\hat{y} - y)^2$
- Let N_t be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{N_t} \sum_{n=1}^{N_t} (\hat{y}_n - y_n)^2}$$

Determining p , Degree of Polynomial

- This is determined experimentally
- Starting with $p=1$, test set is used to estimate the accuracy, in terms of error, of the regression model
 - Note: The polynomial degree $p=1$ is equivalent to multiple linear regression
- This process is repeated each time by incrementing p
- The regression model with p that gives the minimum error on test set may be selected

Illustration of Polynomial Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

$d=2$

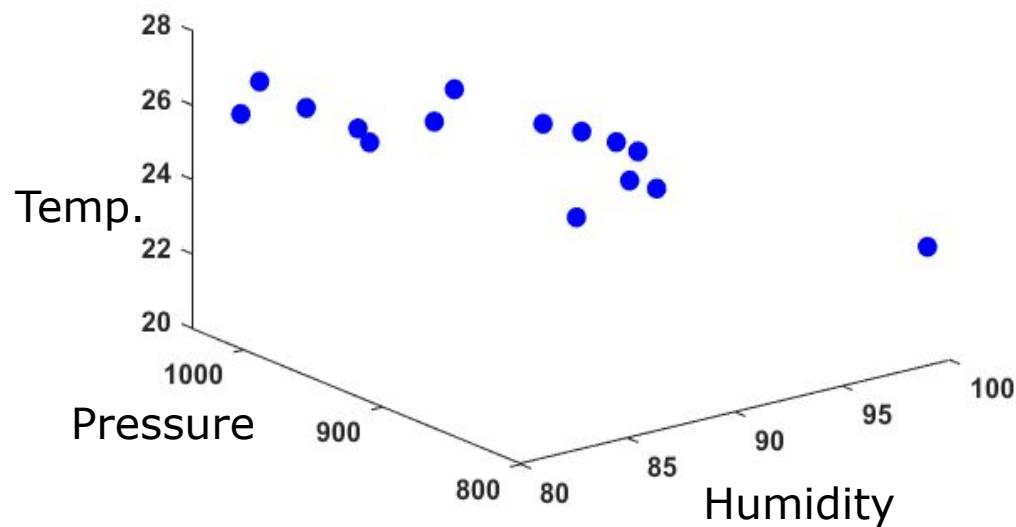


Illustration of Polynomial Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- **Training:**

- Polynomial Degree $p = 3$

$$\hat{\mathbf{w}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

Φ is 15×10 matrix

Number of monomials, $m = \frac{(d+p)!}{d! p!} = \frac{(2+3)!}{2! * 3!} = 10$

Illustration of Polynomial Regression: Temperature Prediction

Humidity (x_1)	Pressure (x_2)	Temp (y)
82.19	1036.35	25.47
83.15	1037.60	26.19
85.34	1037.89	25.17
87.69	1036.86	24.30
87.65	1027.83	24.07
95.95	1006.92	21.21
96.17	1006.57	23.49
98.59	1009.42	21.79
88.33	991.65	25.09
90.43	1009.66	25.39
94.54	1009.27	23.89
99.00	1009.80	22.51
98.00	1009.90	22.90
99.00	996.29	21.72
98.97	800.00	23.18

- Training:
- Polynomial Degree $p = 3$
$$\hat{\mathbf{w}} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$$

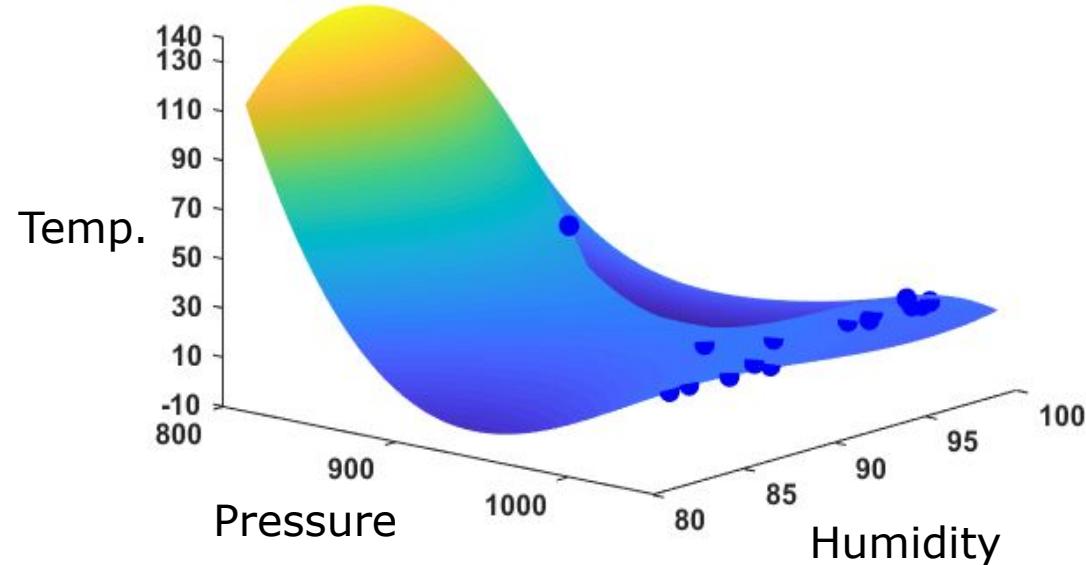


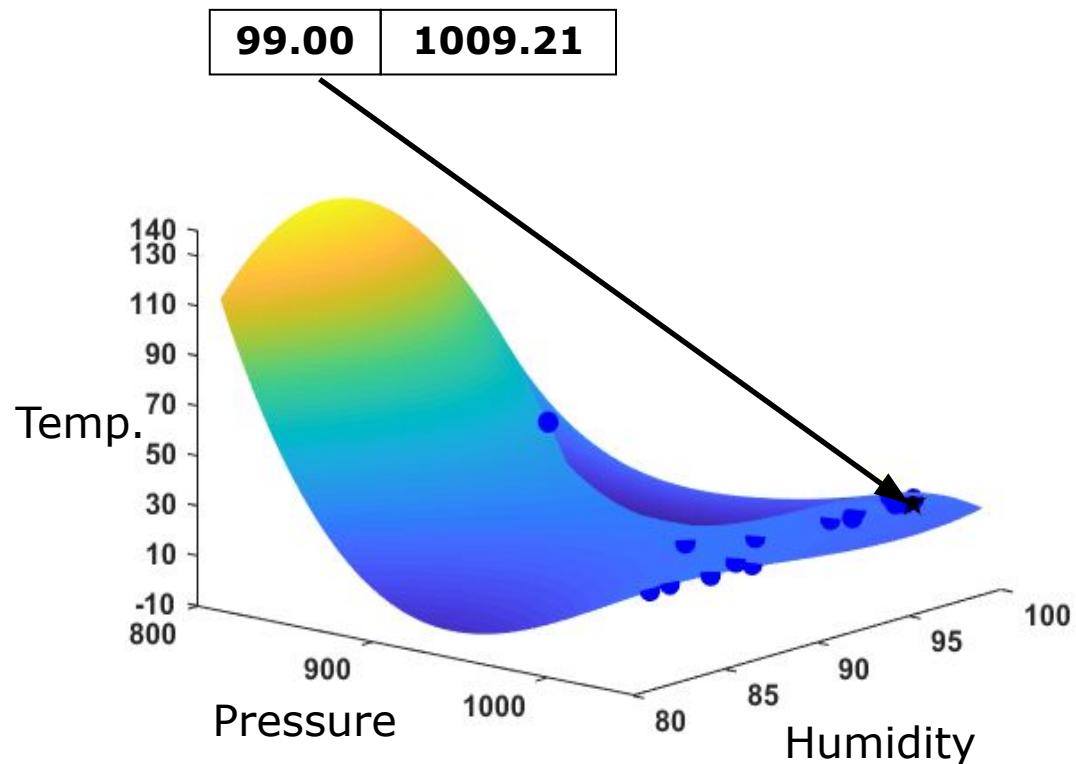
Illustration of Polynomial Regression: Temperature Prediction - Test

- Degree of polynomial $p = 3$

$$\hat{\mathbf{w}} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$

Humidity (x_1)	Pressure (x_2)	Temp (y)
99.00	1009.21	-

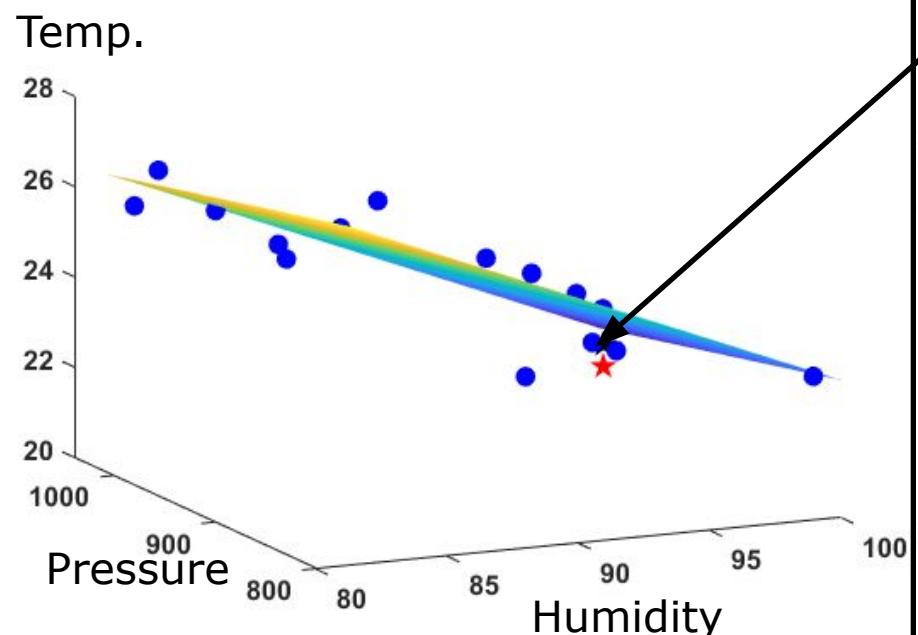
$$\begin{aligned}\hat{y} &= f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{\mathbf{w}}^\top \boldsymbol{\varphi}(\mathbf{x}) \\ &= \sum_{j=0}^{m-1} w_j \varphi_j(\mathbf{x})\end{aligned}$$



- Predicted Temperature: 21.05
- Actual Temperature: 21.24
- Squared error: **0.035**

Multiple Linear Regression vs Polynomial Regression Temperature Prediction

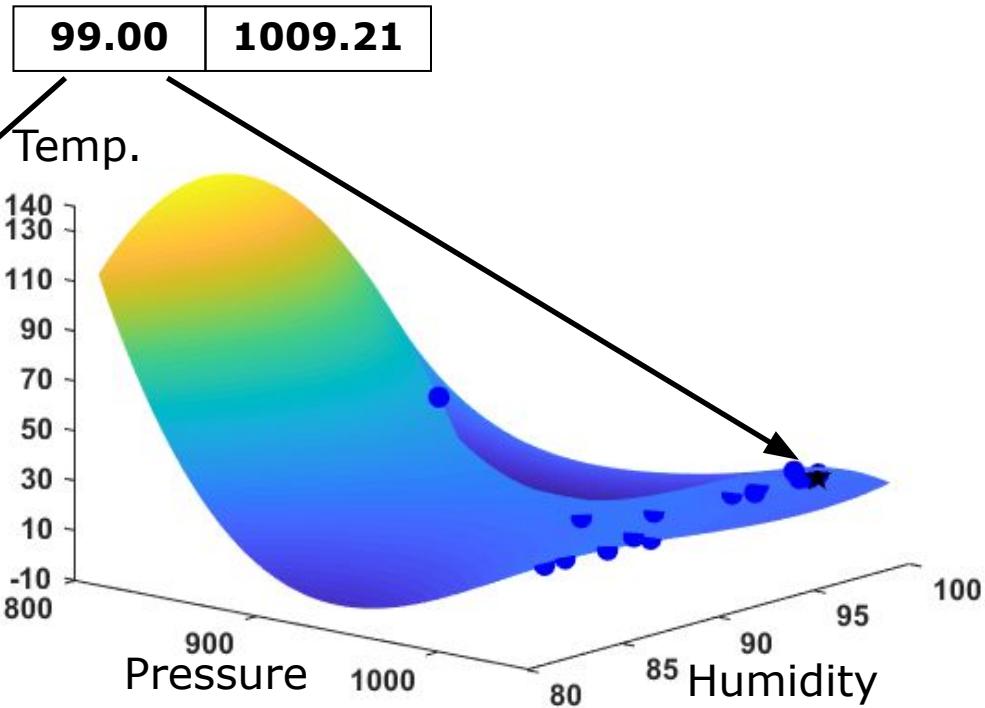
- Multiple Linear Regression



- Predicted Temperature: 21.72
- Actual Temperature: 21.24
- Squared error: **0.2347**

- Polynomial Regression

- Degree of polynomial $p = 3$



- Predicted Temperature: 21.05
- Actual Temperature: 21.24
- Squared error: **0.035**

Summary: Regression

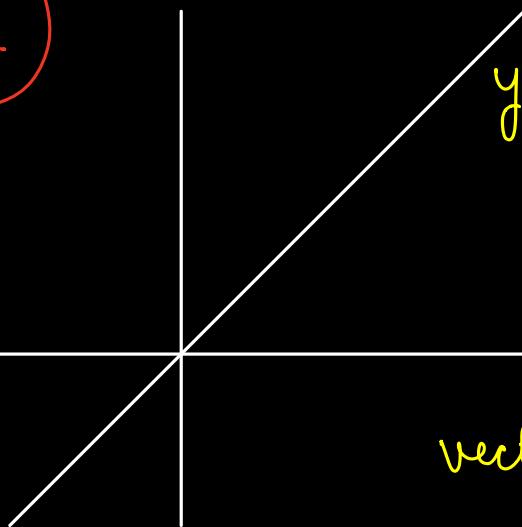
- Regression analysis is used to model the relationship between one or more independent (**predictor**) variable and a dependent (**response**) variable
- Response is some function of one or more input variables
- **Linear regression:** Response is linear function of one or more input variables
 - If the response is linear function of one input variable, then it is simple linear regression (**straight-line fitting**)
 - If the response is linear function of two or more input variable, then it is multiple linear regression (**linear surface fitting** or **hyperplane fitting**)
- **Nonlinear regression:** Response is nonlinear function of one or more input variables
 - **Polynomial regression:** Response is nonlinear function approximated using polynomial function up to degree p of one or more input variables
 - When the degree of polynomial (p) is 1, then it is linear regression

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

Linear equation in Matrix form

I



$$y = mx + c \Rightarrow ax + by = c$$

$$w_1x + w_2y = w_0$$

OR we can write it in the vector form.

$$X = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\therefore \boxed{W^T X = 0}$$

What is physical significance of m or Slope or $\frac{dy}{dx}$ or Derivative

$$W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

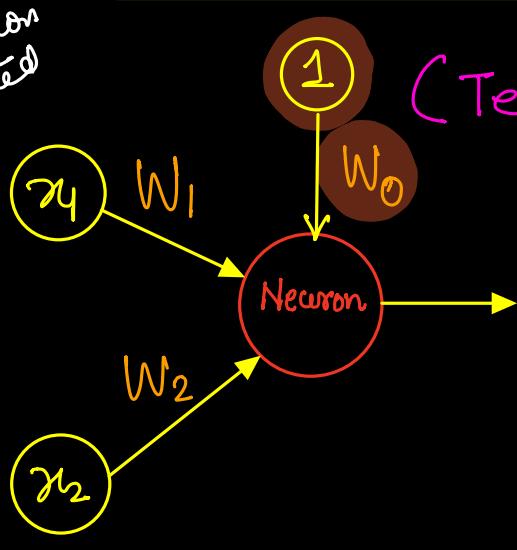
Given a line say $(y = mx + c)$ & a point (x_1, y_1)

How to know its side.

$$y_1 - m x_1 - c \geq 0$$

$$\text{OR } \boxed{y_1 - m x_1 - c < 0}$$

This computation can be abstracted as a neuron.



(Termed as bias)

This neuron just works as an aggregator and do 2 things

- Compute $w_0 + w_1 x_1 + w_2 x_2$
- fires if it is ≥ 0

Derivative

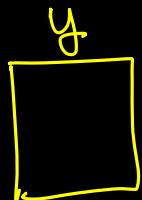
$$m = \frac{dy}{dx}$$

which is rate of change of y wrt x

↓↓

Just thinks x as a knob and we

How much is Δy when $\Delta x = 1$



need to know how much it can effect y .

its sensitivity wrt to the value of y

III

What is a function?

It is just a mapping b/w input & o/p.

$$y = f(x) \quad (1D \text{ function})$$

Plotted in 2D space

Can only be differentiated in one direction

$$\left[\frac{dy}{dx} \right]$$

(Multivariate
Calculus)

$$y = f(x_1, x_2)$$

Plotted in 3D space.

Can be differentiated in 2 directions

$$\left[\frac{\partial y}{\partial x_1} \text{ & } \frac{\partial y}{\partial x_2} \right]$$

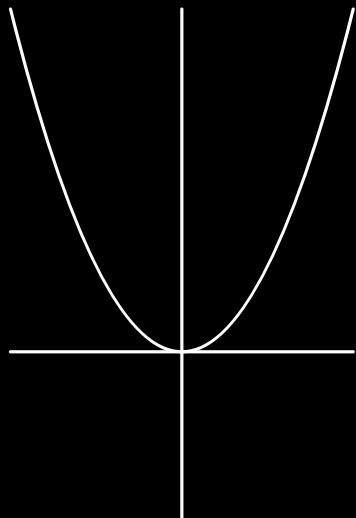
If differentiated wrt vector $x = [x_1 \ x_2]$

But why derivatives are essentially required.

IV

Given any function $f(x)$ how can you maximize or minimize. [Optimization]

Say $y = f(x) = x^2$ (Quadratic fn)



$$\frac{dy}{dx} = 2x \quad \therefore 2x = 0 \\ \Rightarrow x = 0$$

where we can get extremes.

If $y = (x+1)^2$ $\frac{dy}{dx} = 2(x+1) = 0$
 $(x = -1)$

(How to know maxima OR minima)

$$ax + by + cz = d \quad (\text{This is a plane in 3D})$$

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 = 0 \quad (\text{What is this})$$

$$\sum_{i=0}^n w_i x_i = 0 \quad (\text{what is this?}) \quad (x_0 = 1)$$

(Hyper-plane)

Multidimensional
linear function.

Given a multidimensional fn say (F) defined in \mathbb{R}^N

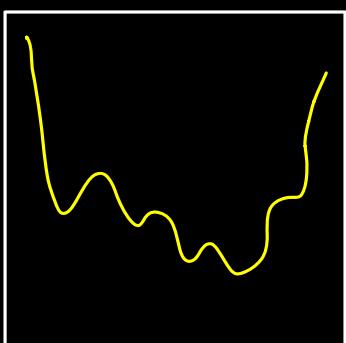
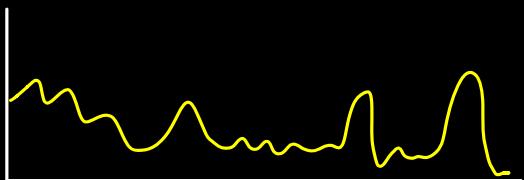
How can we minimize it. This (F) can be a loss function or objective function. (Closed-form)

In some (N) dimensional (say $w_0, w_1, w_2, \dots, w_{N-1}$) parametric space assume that some loss function (F) is defined. We require :

$$w^* = \underset{W}{\operatorname{arg\min}} (F)$$

Just differentiate F wst $w \Rightarrow \frac{dF}{dw}$

This can be done easily if (F) can be represented as some close form formula (as discussed above)



But most of the functions are not well defined as some formula and are represented in a discrete manner.

$$f(x_1), f(x_2), f(x_3), \dots$$

OR $f(x_1, x_2, x_3)$

$$x_1, x_2, x_3$$

Discrete functions

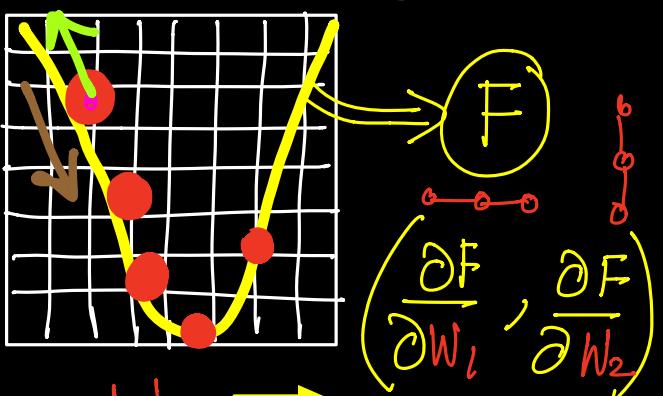
Now in such a scenario how can we optimize a an objective function.

In discrete space we have an iterative solution.

$$W_1^{\text{New}} = W_1^{\text{Old}} - \eta \frac{\partial F}{\partial W_1}$$

$$W_2^{\text{New}} = W_2^{\text{Old}} - \eta \frac{\partial F}{\partial W_2}$$

Gradient decent algorithm for optimization (used as optimizer)



Parametric Search for optimization. [Update till Convergence]

SGD, ADAM, AdaGrad etc.

Analytically Computing gradient :

$$\lim_{h \rightarrow 0} \frac{f(x_i + h) - f(x_i)}{h} \quad \boxed{\text{A}}$$

0	0	0
x_{i-1}	x_i	x_{i+1}

$$\boxed{\text{B}}$$

$x_{i-1,j-1}$	$x_{i,j-1}$	$x_{i,j+1}$
$x_{i,j-1}$	$x_{i,j}$	$x_{i,j+1}$
$x_{i+1,j-1}$	$x_{i+1,j}$	$x_{i+1,j+1}$

for (x_i), gradient in horizontal direction can be : $x_{i+1} - x_{i-1}$

for ($x_{i,j}$) gradient in horizontal :

$$x_{i,j+1} - x_{i,j-1} \text{ OR}$$

Averaging all 3 Rows

This is gradient w.r.t a neighbourhood.

VII

Hence in that high dimensional space



(N-dim)

- ⊗ These are (N) directions to move on.
- ⊗ we are in search of optimal set of parameters (W^*) that can minimize the objective fn (F).
- ⊗ We need to use Gradient decent iterative algorithm to optimize over (F) in \mathbb{R}^N .

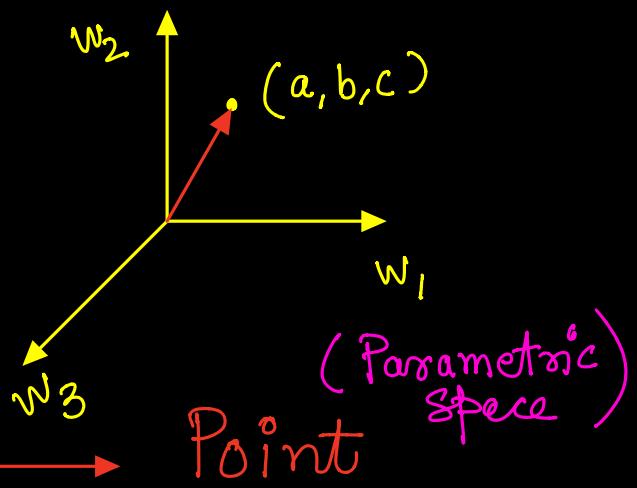
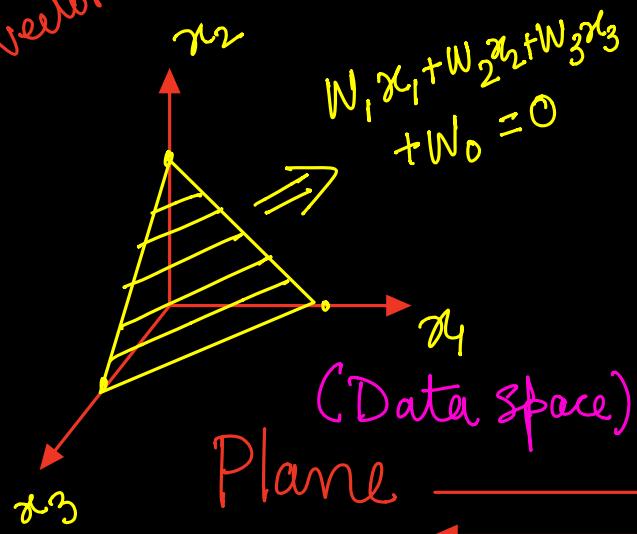
→ Initialize (W) vector $\in \mathbb{R}^N$, randomly.
 → Update $\forall W_i^* = W_i^{\text{old}} + \Delta W_i$

$$\text{where } \left\{ \Delta W_i = -\eta \frac{\partial F}{\partial W_i} \right\}$$

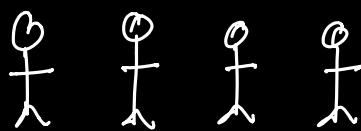
$WX = 0$
 A h-plane in
 vector form is \mathbb{R}^N

Each $[W]$ vector $\in \mathbb{R}^N$, and can be seen as a h-plane in the space of (X) .

\mathbb{R}^N



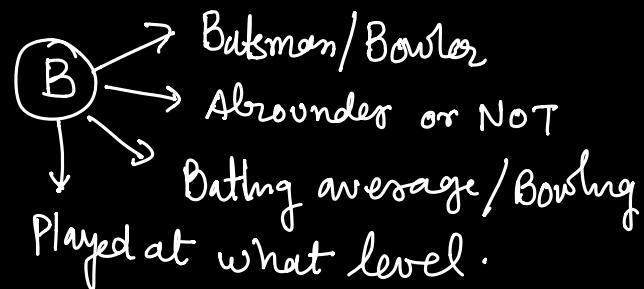
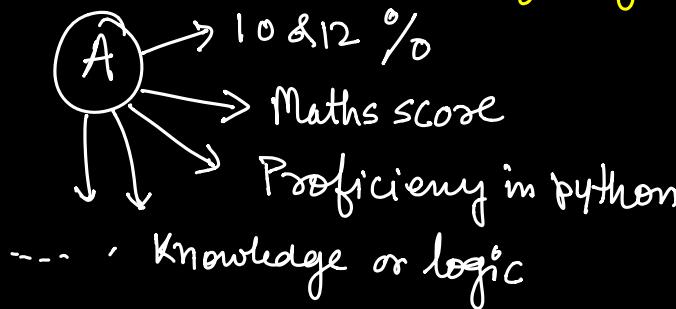
Problem of binary classification



(1000 persons have applied
for
 ① Join by research group
 ② Join my cricket team.

Name ③ (How many attributes to be considered)?
Mobile
Qualification as well as what?
Gender

- Central question
- ④ During the interview like (father's Name, Village, Uncle, family, friend, 3D gesture, face structure - - - - -)
 what questions we need to ask.
- ⑤ What is the weightage of that question.

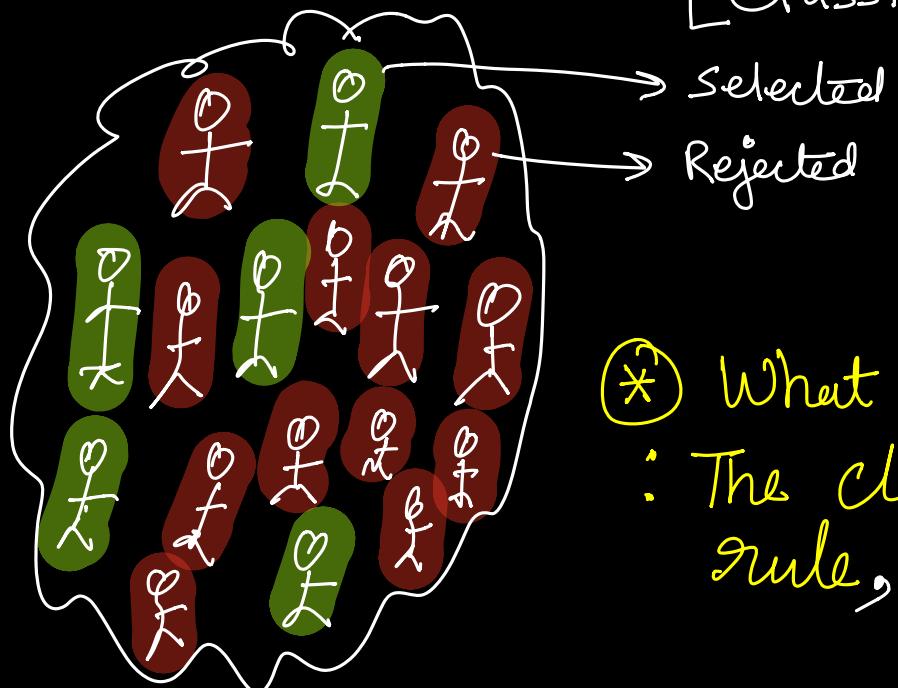


feature selection ⑥ One can observe that the questions that we need to ask are depending upon the objective or Task in hand (Problem Statement)

Question and their importance keep on changing with respect to the problem in hand [Feature Engg]

Now as an interviewer, I will ask questions to all the applicants. Get their answers (features) and accept/reject (Based on some Criterion)

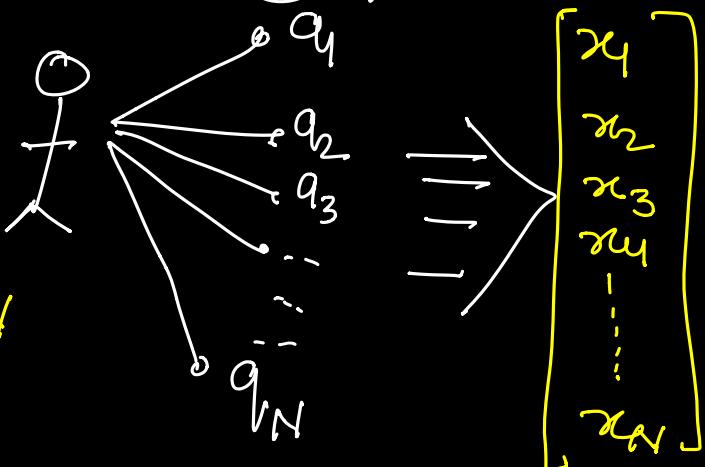
[Classification decision]



(Out of 1000
100 Selected &
900 got rejected)

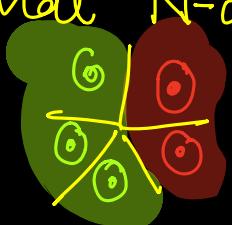
* What we need to learn:
: The classification decision rule, given the questions.

* Assuming we ask N questions to each applicant:



* Basically each data got projected onto \mathbb{R}^N space: data to an N -dim vector

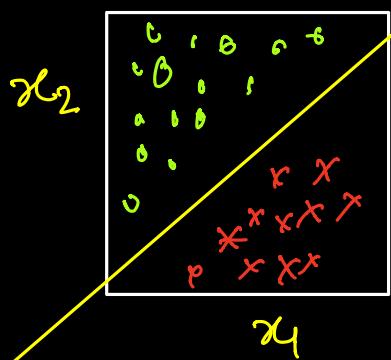
* In that N -d space (\mathbb{R}^N) it will be seen as a point.



Typical (N) values are 128/256 & much bigger.

* We need to ask relevant questions, so as Feature extraction to project data points into (\mathbb{R}^N) in such a way that in that space, accepted & rejected Candidates are well apart.

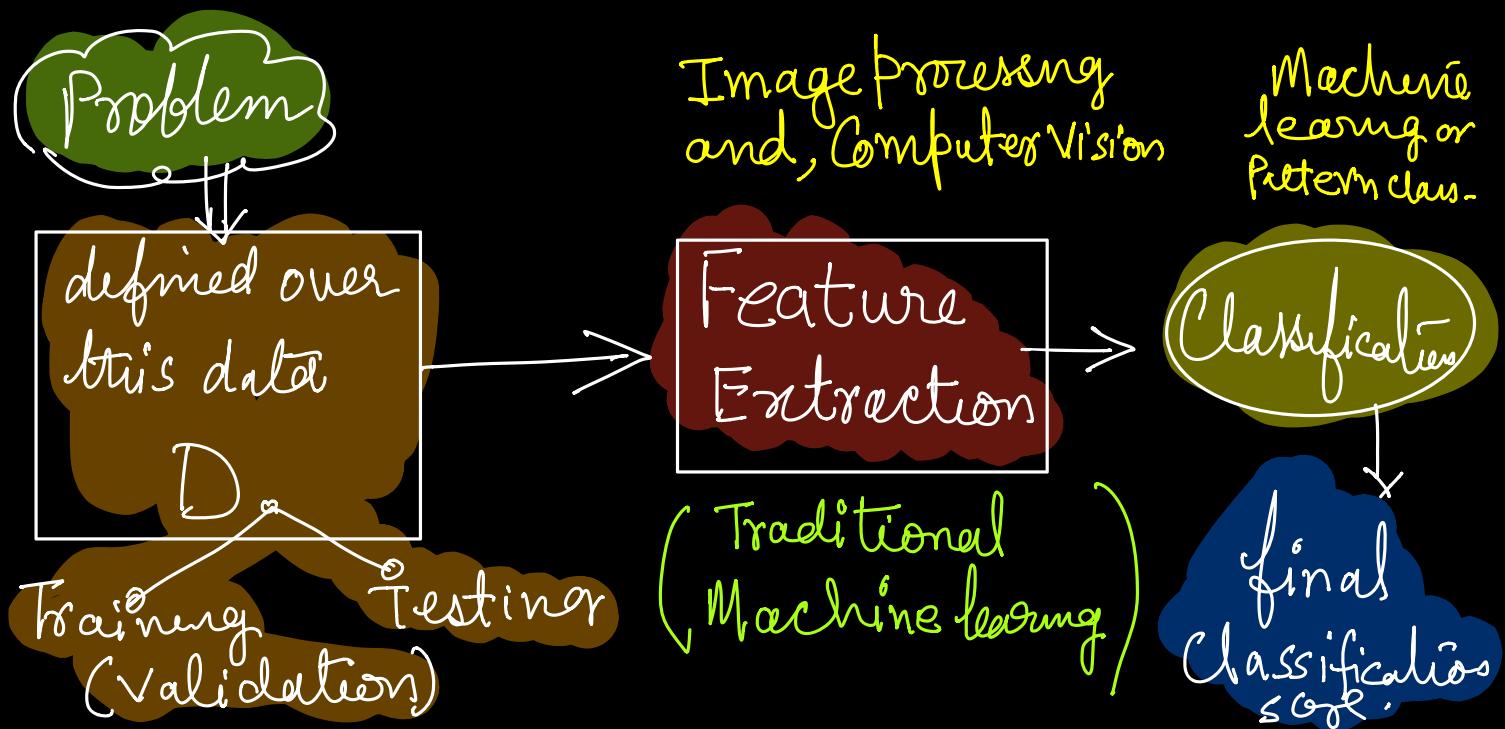
* Assuming we have asked only 2 questions
→ data got projected in 2D space.



Classifier need to learn this Separating line or H-plane Called Decision boundary.

ML to DL

The full flow of a classification system can be





Similar to the above mentioned functional Optimization, we need to search for the best h-plane in the feature space $X = \{x_1, x_2, \dots, x_N\} \in \mathbb{R}^N$

which will be

$$w_0, w_1, w_2, \dots, w_N \left. \begin{array}{l} \text{Coefficients} \\ \text{of the h-plane.} \end{array} \right\}$$

Bias

But what is going to be the function that we need to optimize \Rightarrow loss fn

- MSE or MAE (for Regression)
- Cross Entropy (Classification)

Supervised Learning:

These loss fn, for each given i/p data (X) assumes a correct o/p (Y) say ground truth. Using the learned logic, (X) will be converted into the predicted o/p (Y'). Loss functions need to estimate the dissimilarity/deviation b/w Y & Y' $\Rightarrow \{LF(Y, Y')\}$

In case of regression Y & Y' may be a single value
for classification Y & $Y' \Rightarrow$ Probability scores for all the classes.



(A) Regression need to learn

(θ) (Set of parameters) such that

$f_{\theta}(X_i) = Y'_i$ need to be as close as possible to Y_i . Loss fns can be :

$$L_2 \text{ Norm (a)} \quad MSE(Y_i, Y'_i) = \frac{1}{2} (Y_i - Y'_i)^2$$

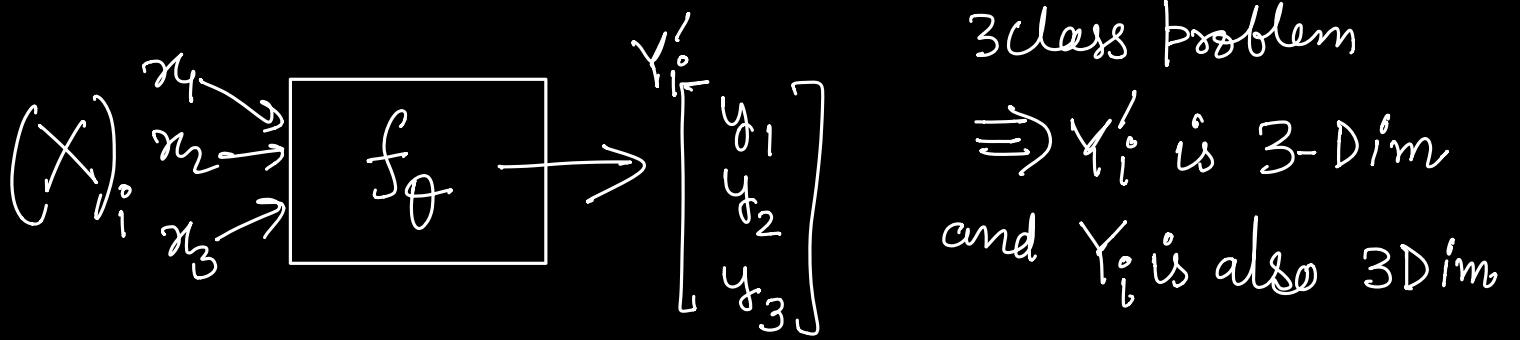
$$L_1 \text{ Norm (b)} \quad MAE(Y_i, Y'_i) = |Y_i - Y'_i|$$

(B) Classification : Y_i 's are represented as 1-Hot encoding.

if 10 class classification then

each $Y_i \Rightarrow$ 10-dim (binary vector)

$$\begin{bmatrix} \text{Cat} \\ \text{dog} \\ \text{car} \\ \vdots \\ \vdots \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \begin{array}{l} \text{implies } Y_i \text{ belongs to} \\ \text{Cat (only one place)} \\ \text{is set} \end{array}$$



$\forall i$ X_i we have their corresponding labels (Y_i) (3D)

Loss fn : Cross entropy (Y_i, Y'_i)

$$\Rightarrow Y_i = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \Rightarrow \sum_{i=1}^3 -y_{i0} \log(y'_{i0})$$

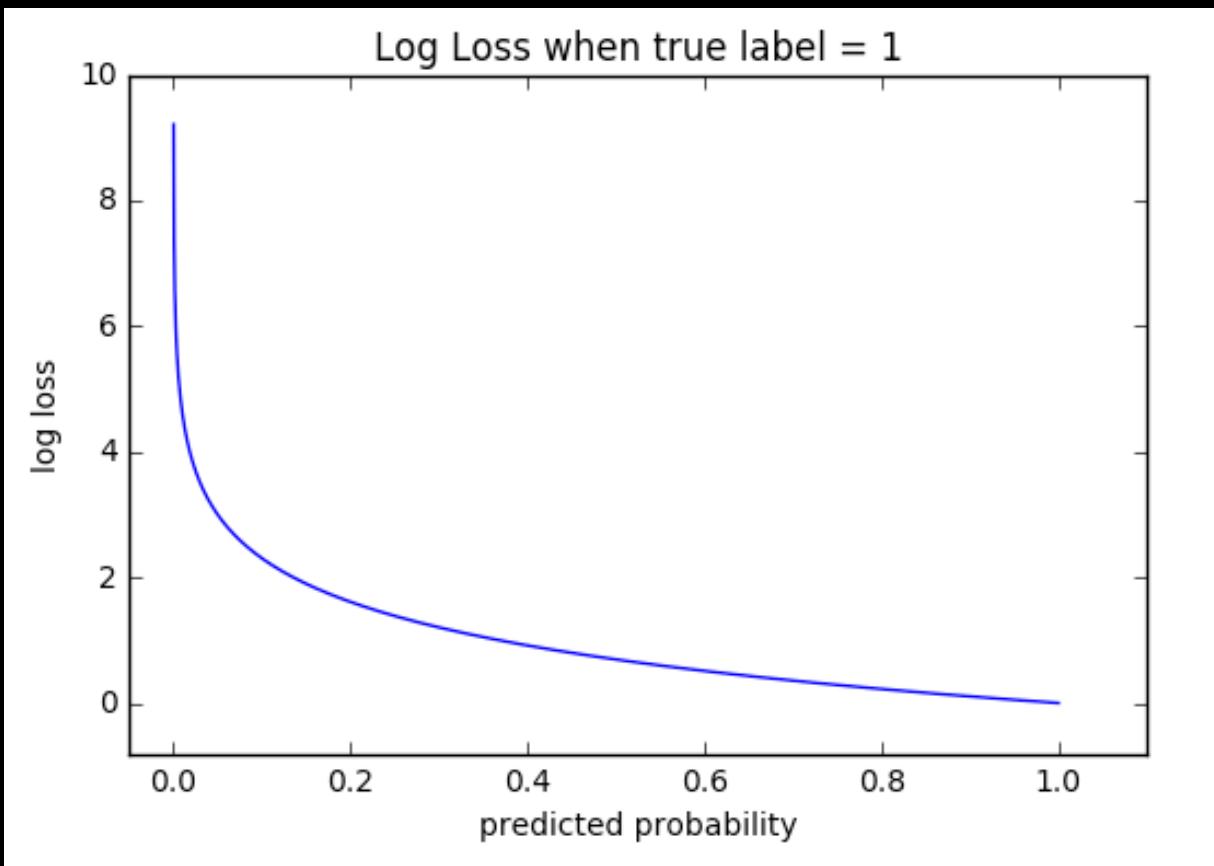
y P

 $1-y$ $1-P$

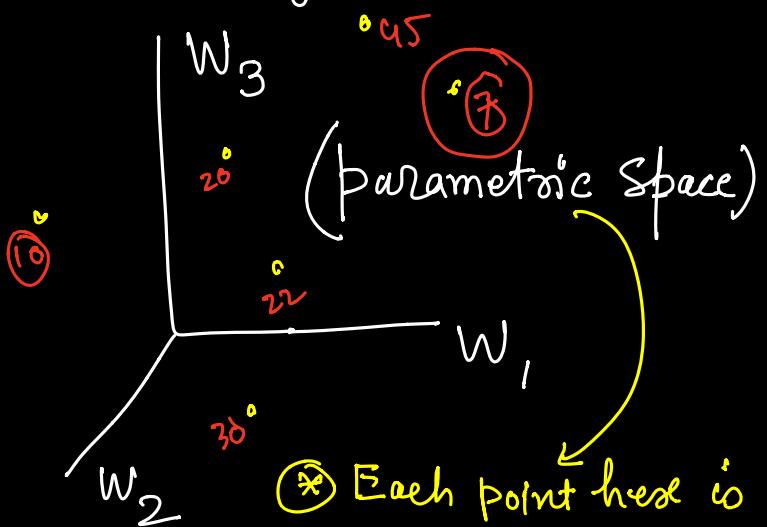
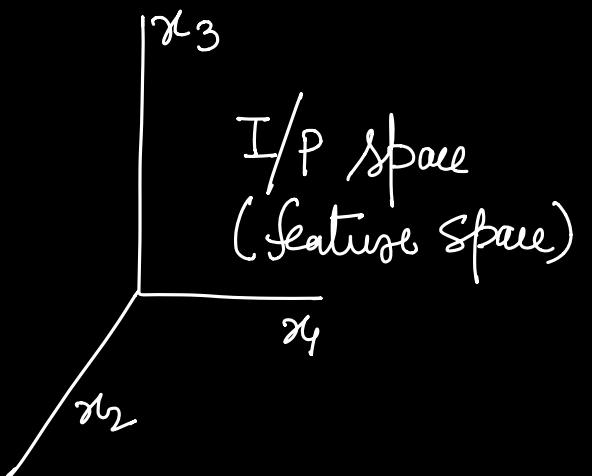
\Rightarrow Binary cross entropy

$$-(y \log P + (1-y) \log(1-P))$$

$$This \Rightarrow -(y_1 \log(y'_1) + y_2 \log(y'_2) + y_3 \log(y'_3))$$



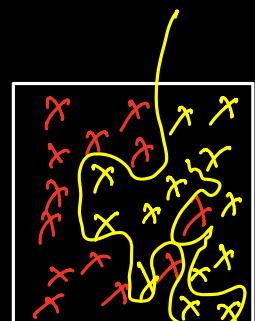
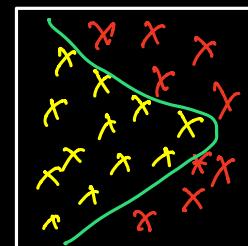
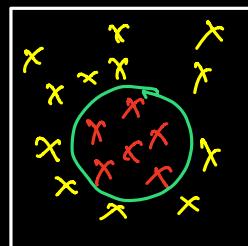
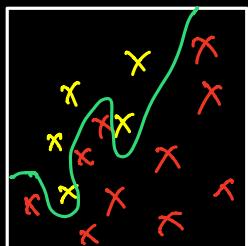
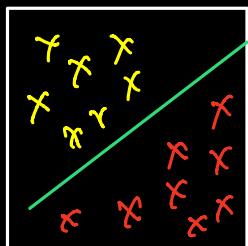
* So, now we are having 2 spaces



* we need to explore the parametric space in order to optimize the Cost fn efficiently

(This is done by applying GD or SGD)

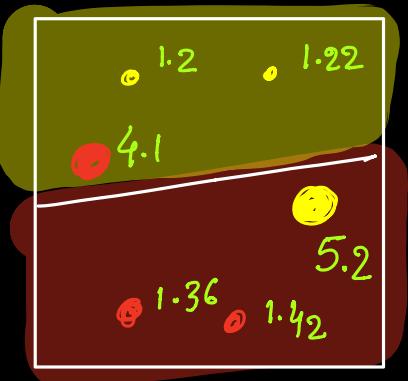
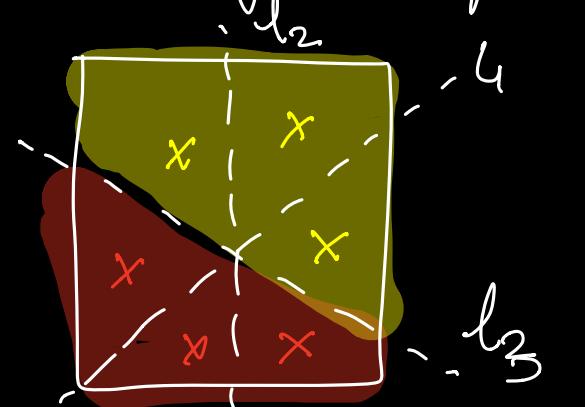
* One neuron can be used to learn one such h-plane if our data is linearly separable. But mostly data is not linearly separable. Then how to learn non-linear boundaries.



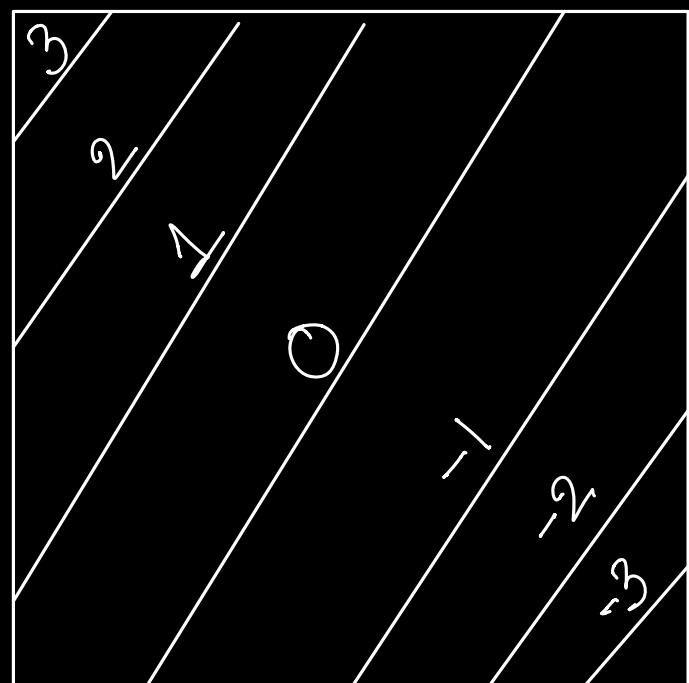
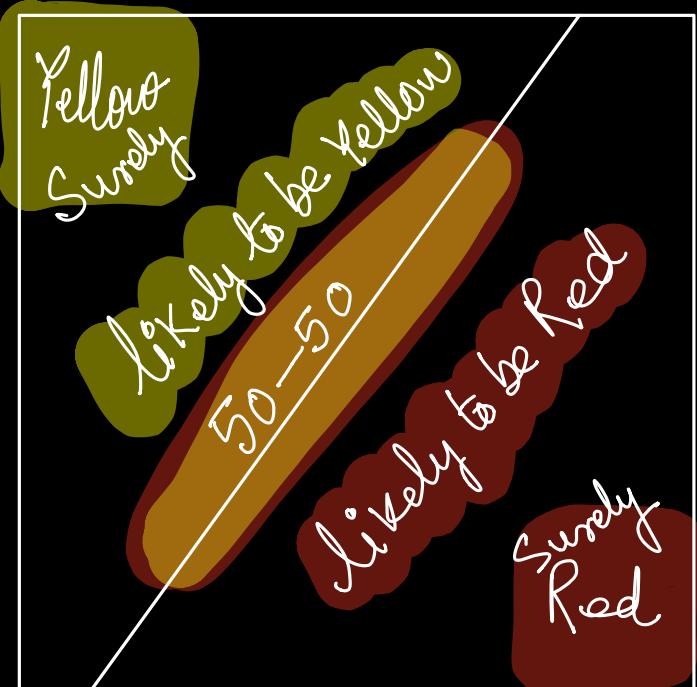
We require more neurons, may be in multiple layers.

Intuition of Neural Networks

- ✳) Neural networks can be giants and may have millions of neurons & thousands of layers.
- ✳) Essentially they are partitioning the feature space with an objective of minimizing misclassification.
Basically Splitting the data
 - one can start from some very bad h-plane.
 - Need to update it parameter so as to minimize the objective fn (mis-classification)
 - We can utilize SGD & GD iteratively.
But there is an issue \Rightarrow Loss fn need to be differentiable
(Here it is discrete)
- ✳) We can use logistic loss fn ?
 - Each and every data is Penalized. Correctly classified then less penalty & mis-classified then more penalty
 - Error $\Rightarrow 1.2 + 1.22 + 4.1 + 5.2 + 1.36 + 1.42$ [continuous]

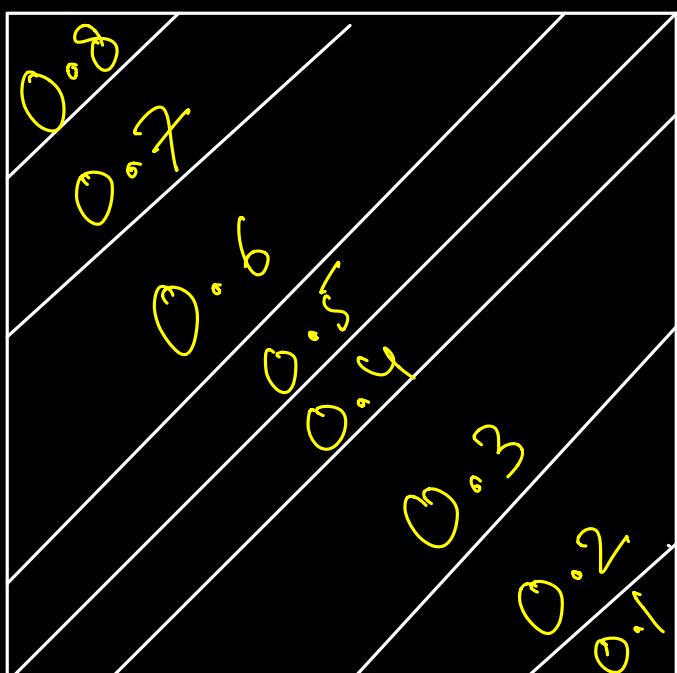
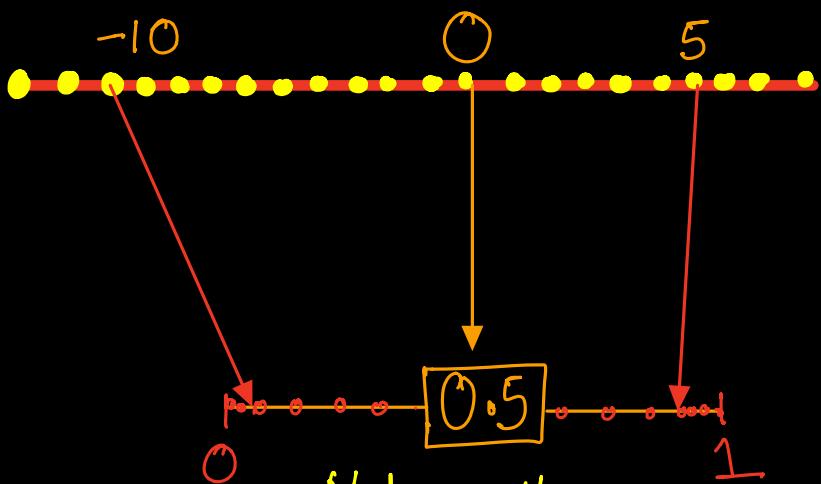


Now the question is how to get such loss fn?



It can be done using
Sigmoid fn :

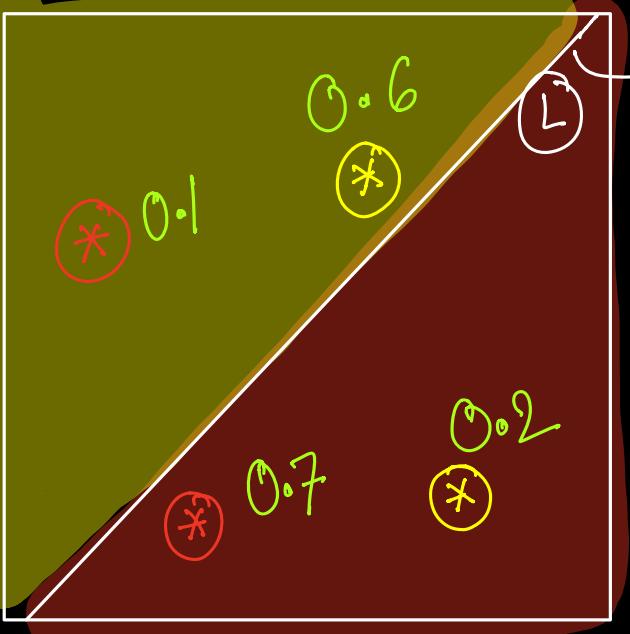
$$S(x) = \frac{1}{1+e^{-x}}$$



Not equally spaced \Rightarrow Basically a non-linear mapping

They are called Activation function, (Nonlinear)
like, Sigmoid, Tanh, ReLU, LReLU, PReLU - - - .

Let us consider this Example.

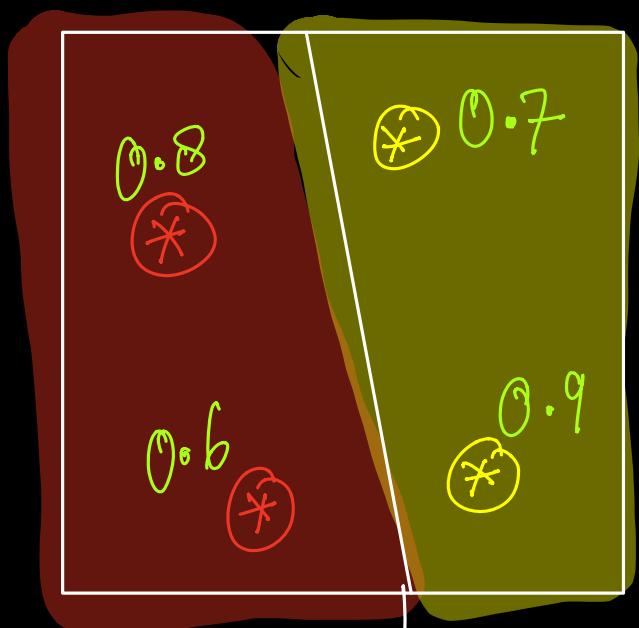


→ Decision boundary (L) separating Yellow & Red Region.

- 2 Classes Yellow / Red
- Shown values are the probability that they belong to the correct class wrt given decision boundary (L).

∴ Joint probability that all points got correctly classified by (L) : $0.6 \times 0.2 \times 0.1 \times 0.7 \Rightarrow 0.0084$
(This can be termed as likelihood.)

∴ Likelihood can be used as a gain fn & need to be maximized.



for above shown boundary

$$0.6 \times 0.2 \times 0.1 \times 0.7$$

$$= 0.0084$$

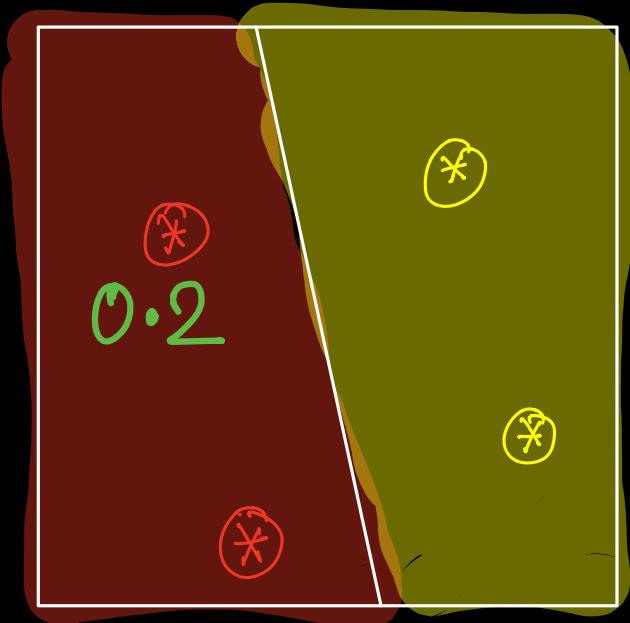
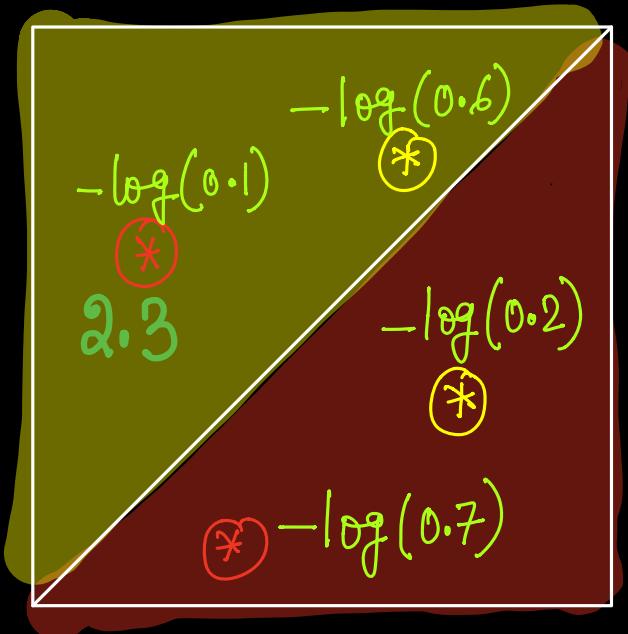
for L'

(maximized)

$$0.7 \times 0.9 \times 0.8 \times 0.6 = 0.3024$$

- These issue is it is multiplication
- The values are very small.

Multiplication → Addition & value scaling.



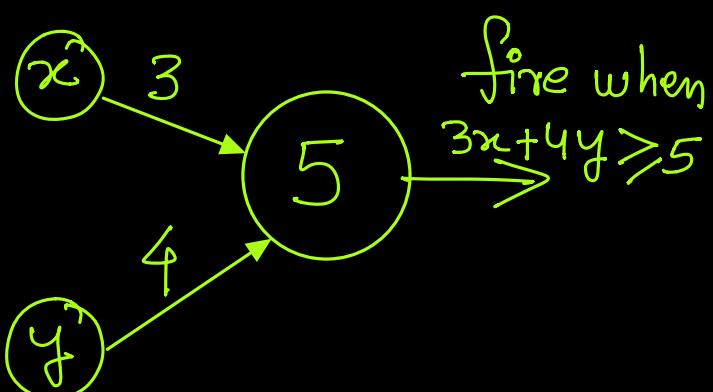
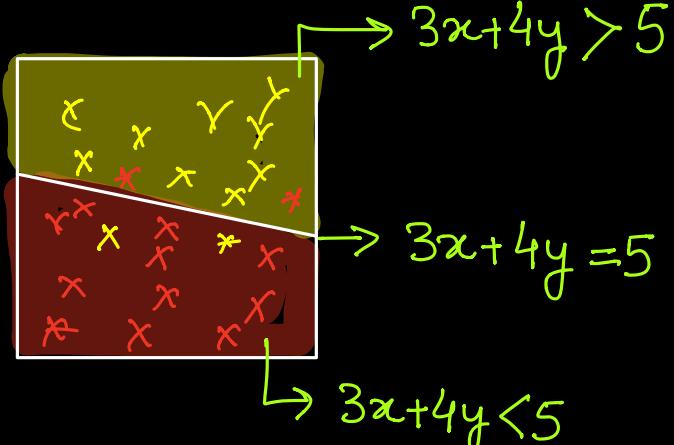
$$0.6 * 0.2 * 0.1 * 0.7 \\ = (0.0084)$$

$$0.7 * 0.9 * 0.8 * 0.6 = 0.3024 \\ \text{(Maximization of likelihood)}$$

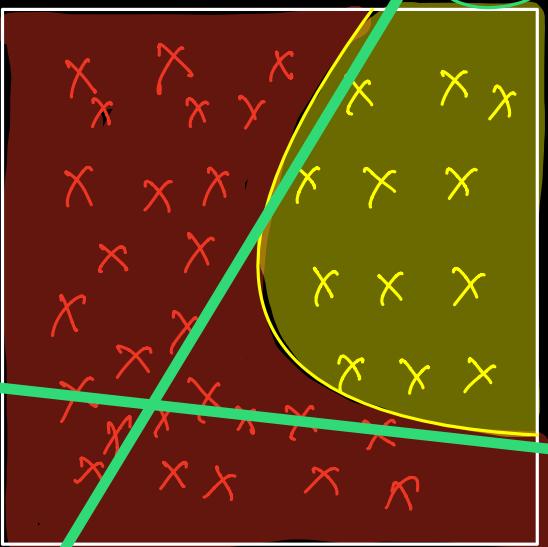
$$-\log(0.6) - \log(0.2) - \log(0.1) \\ - \log(0.7) = 4.8 \\ \downarrow \\ (2.3)$$

$$-\log(0.7) - \log(0.9) \\ - \log(0.8) - \log(0.6) \\ \downarrow \\ (0.2)$$

Log likelihood minimization. \Rightarrow Optimize & get the linear boundary.



Single neuron can learn decision boundary if data is linearly separable.

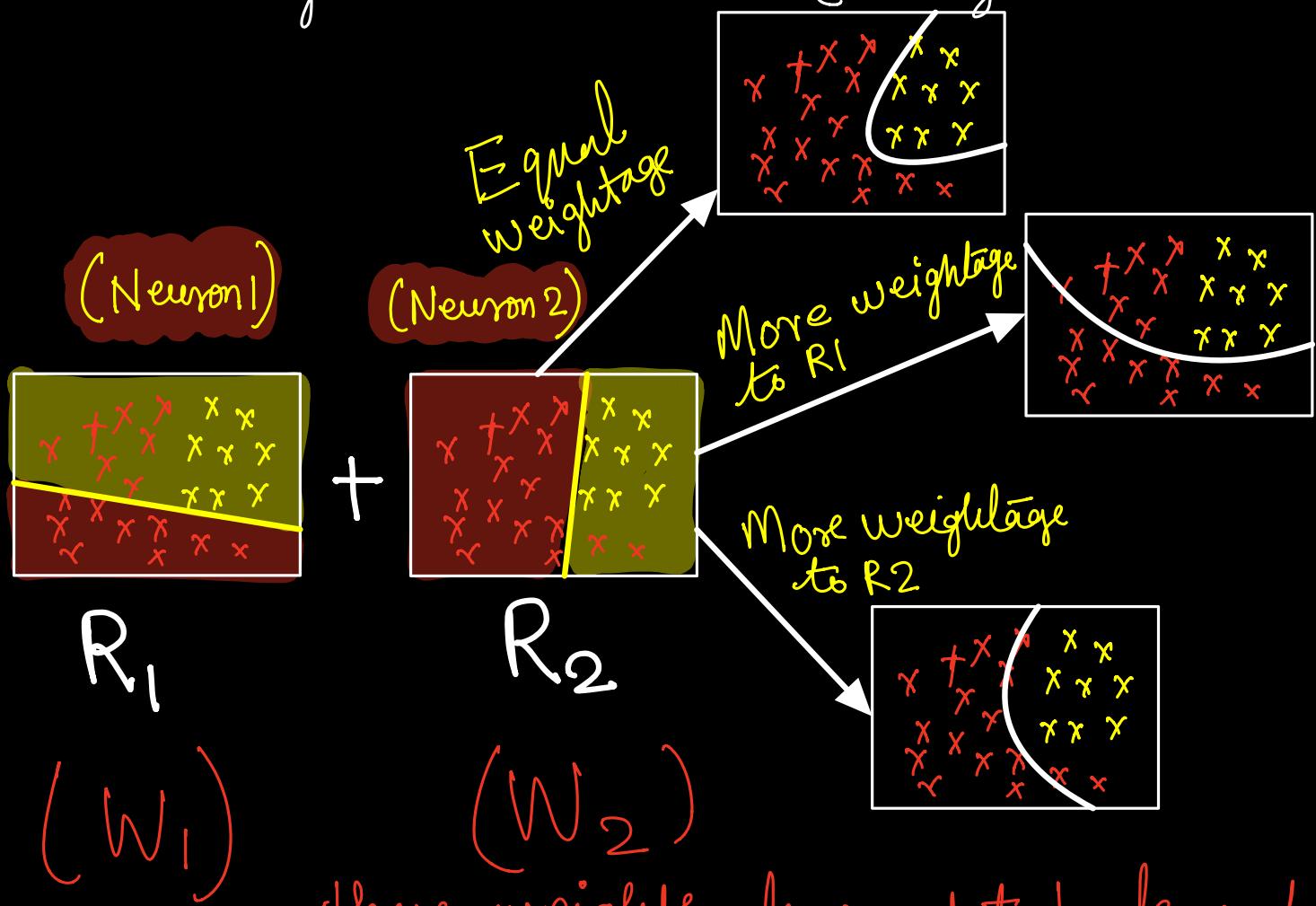


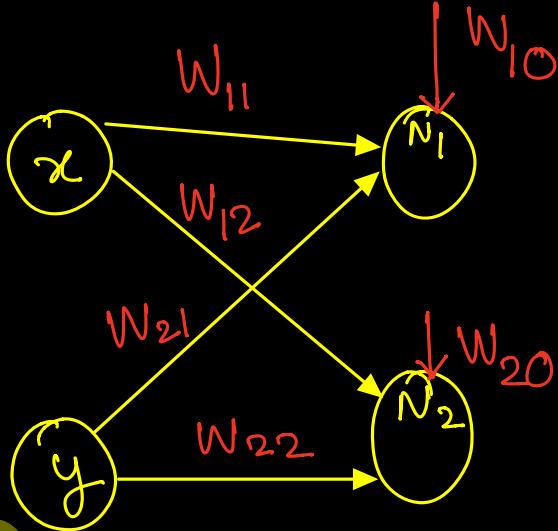
(L_1)

How to learn the non linearity?

\Rightarrow We need to learn the non-linear boundaries in a modular fashion & hierarchically, (using a layered N/w)

First we need to understand Region Combination (weighted)

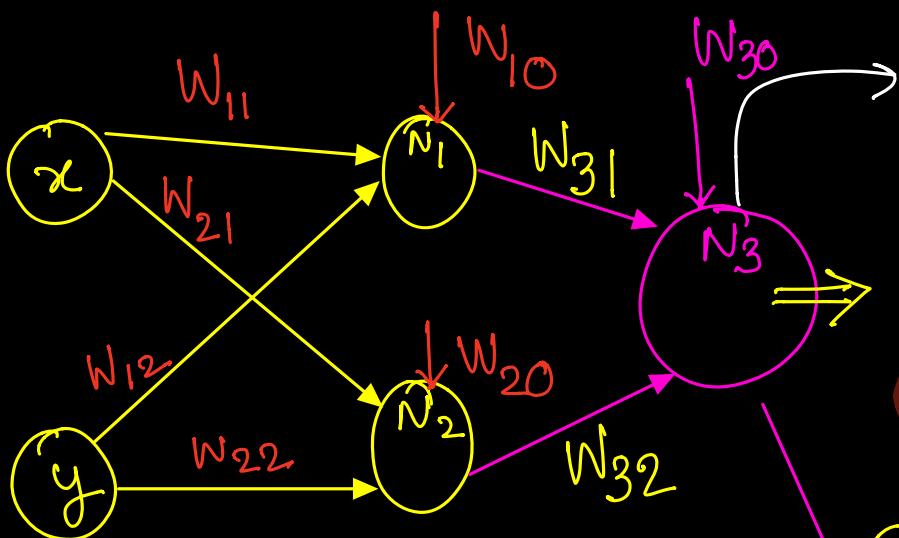




→ This neuron is learning one h-plane (learning Region1)

→ This neuron is learning another h-plane. (learning Region2)

$w_{ij} \Rightarrow i^{\text{th}} \text{ neuron, } j^{\text{th}} \text{ input}$



This neuron is learning how to merge 2 regions, (Basically 2 decision merging)

④ (N_3) not seeing the actual IP.

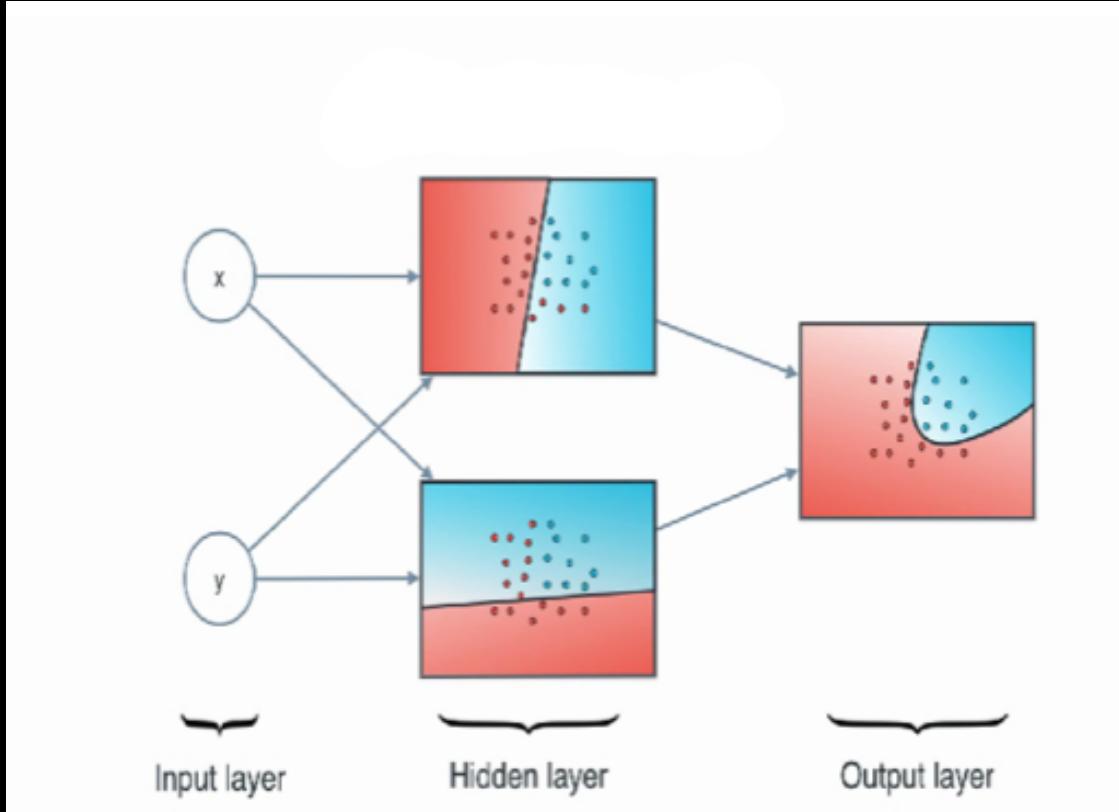
* Objectives of N_1 & N_2 (?)

$$\begin{matrix} \text{neuron-1} \\ \text{neuron-2} \end{matrix} \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} w_{10} \\ w_{20} \end{bmatrix}$$

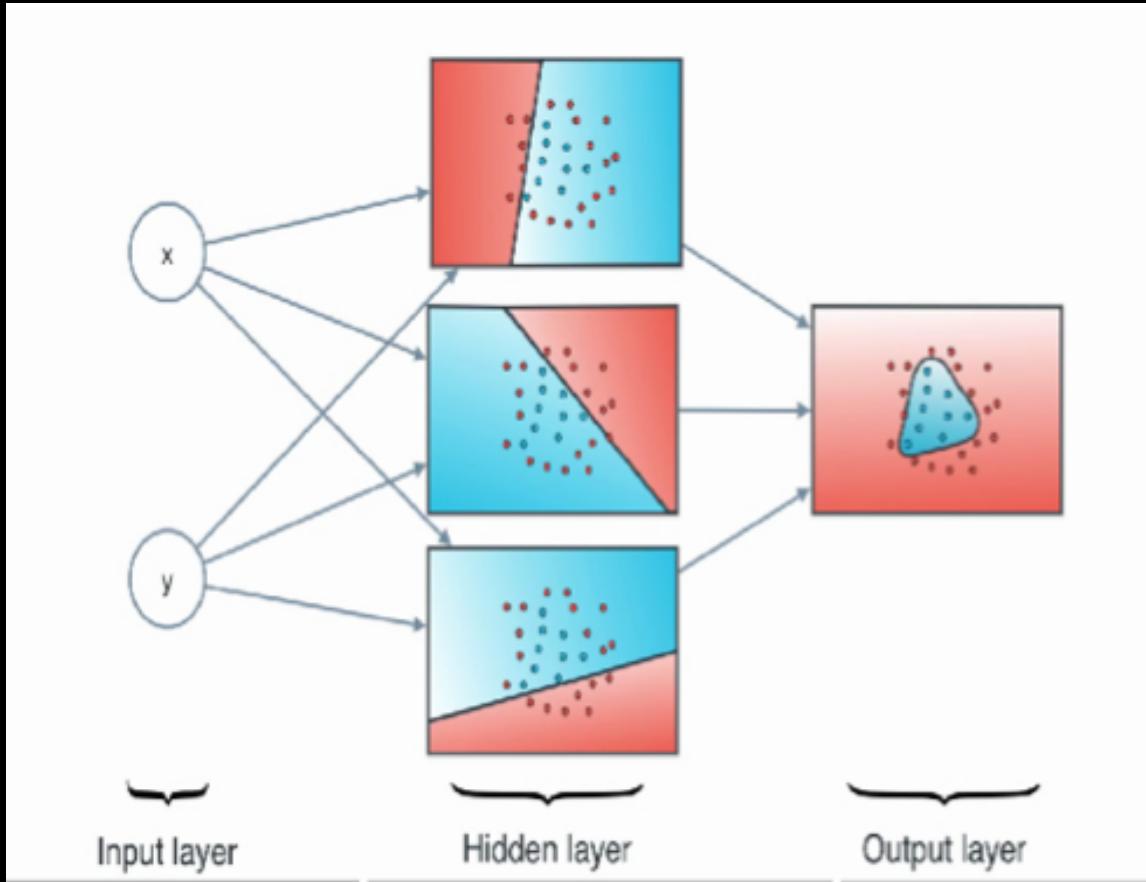
$$\Rightarrow \begin{bmatrix} w_{11}x + w_{12}y + w_{10} \\ w_{21}x + w_{22}y + w_{20} \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

$$\begin{bmatrix} w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

$$\Rightarrow w_{31}I_1 + w_{32}I_2 + w_{30}$$

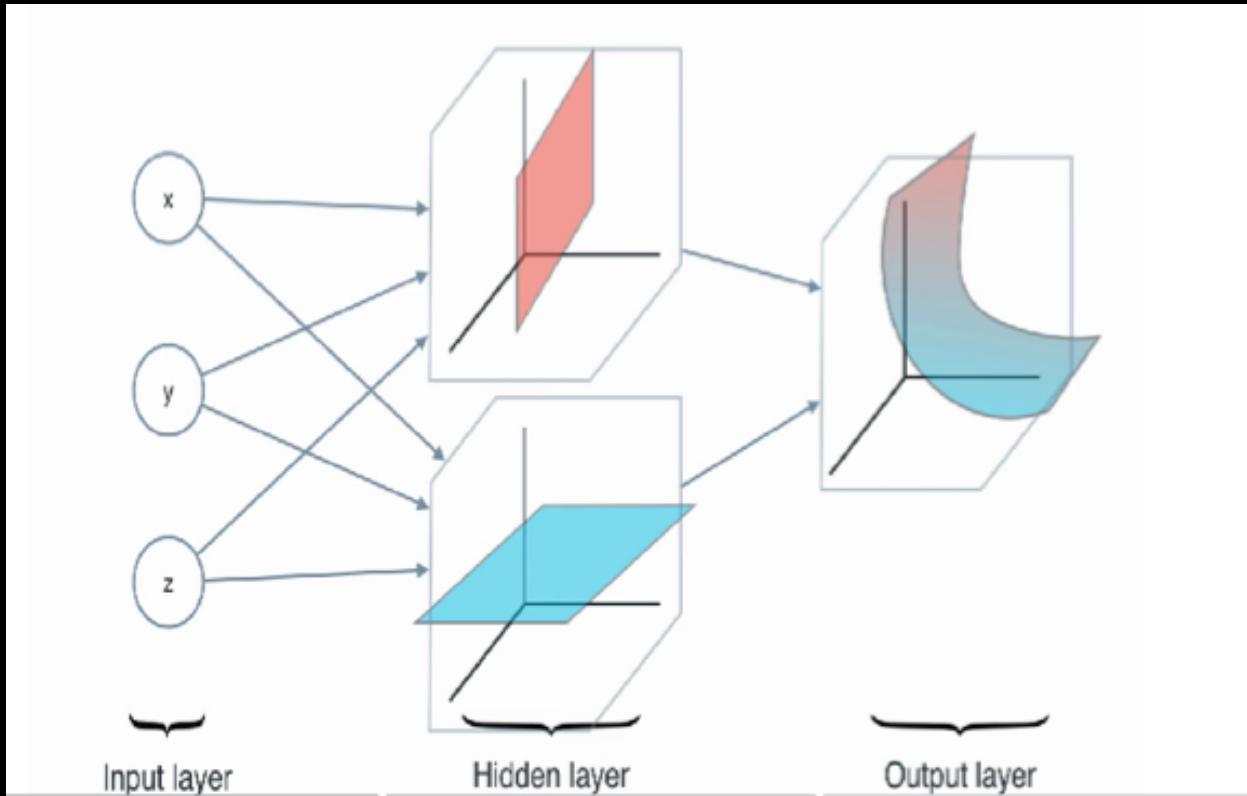


Region Merging

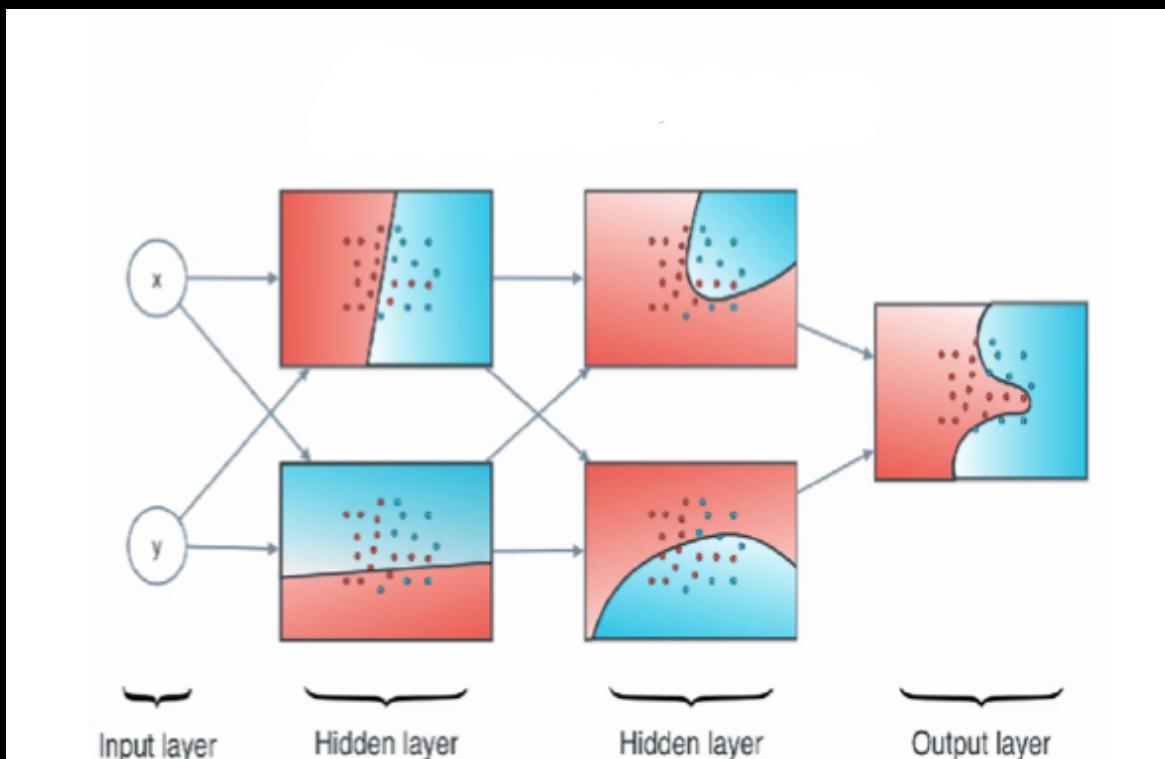


Adding
More
neurons
in a
layer

Scaling Vertically

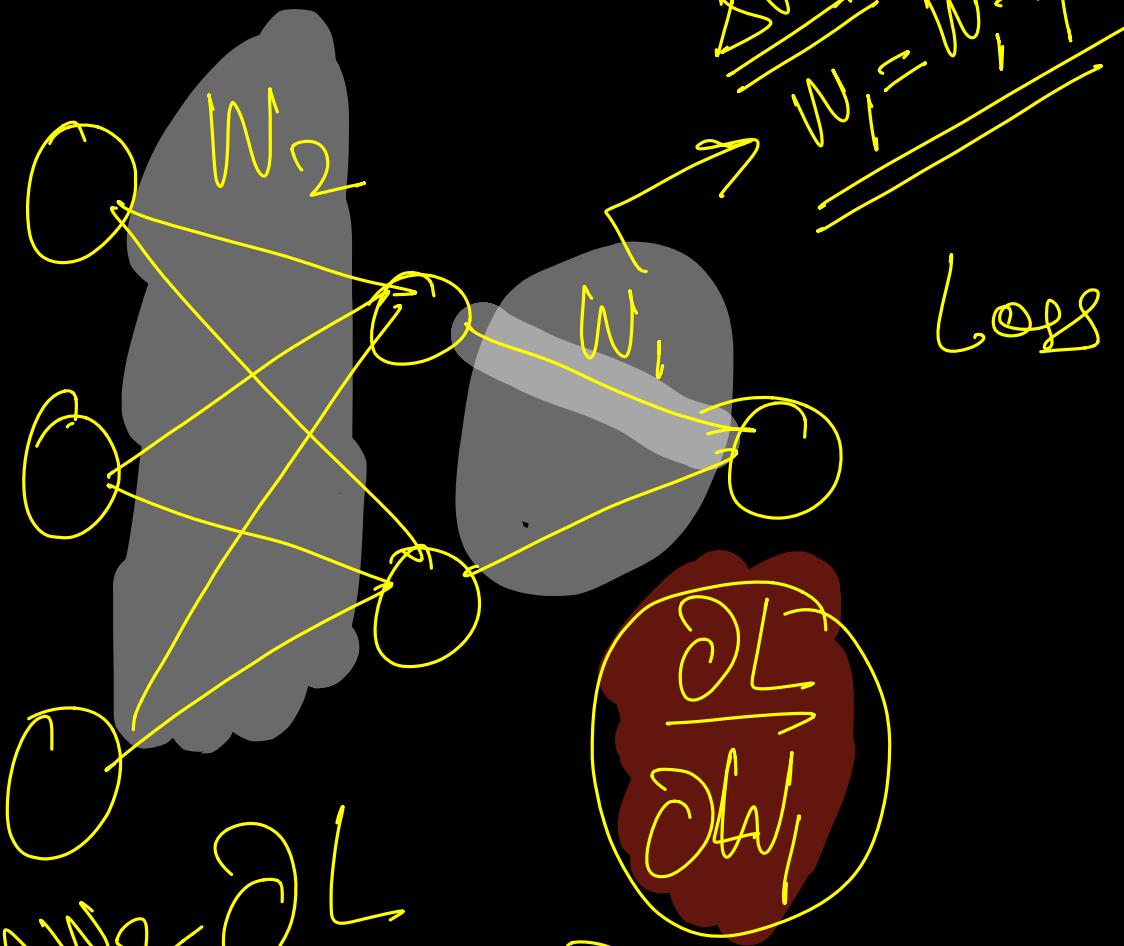


Input dimension > 2



Scaling horizontally.

... N + ... - NDN₁



$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial W_2} = \frac{\partial L}{\partial y} \cdot W_2$$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial W_1} = \frac{\partial L}{\partial y} \cdot W_1$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b_1} = \frac{\partial L}{\partial y}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b_2} = \frac{\partial L}{\partial y}$$

DEEP LEARNING AND ITS APPLICATIONS

Fully Connected Layer

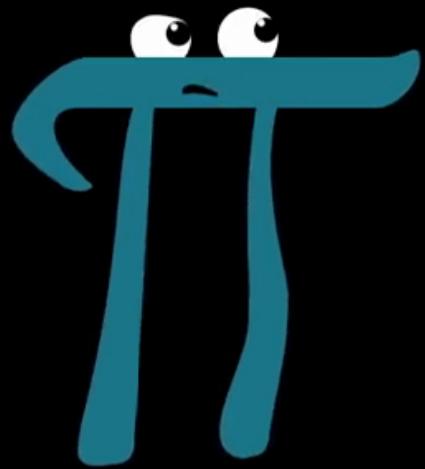
Aditya Nigam, Assistant Professor

School of Computing and Electrical Engineering (SCEE)
Indian Institute of Technology, Mandi

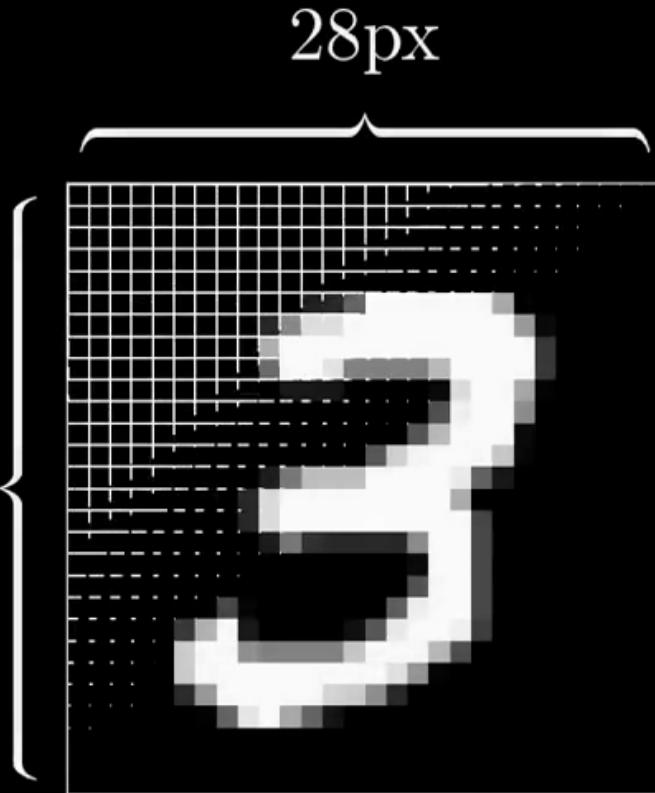
<http://faculty.iitmandi.ac.in/~aditya/> aditya@iitmandi.ac.in



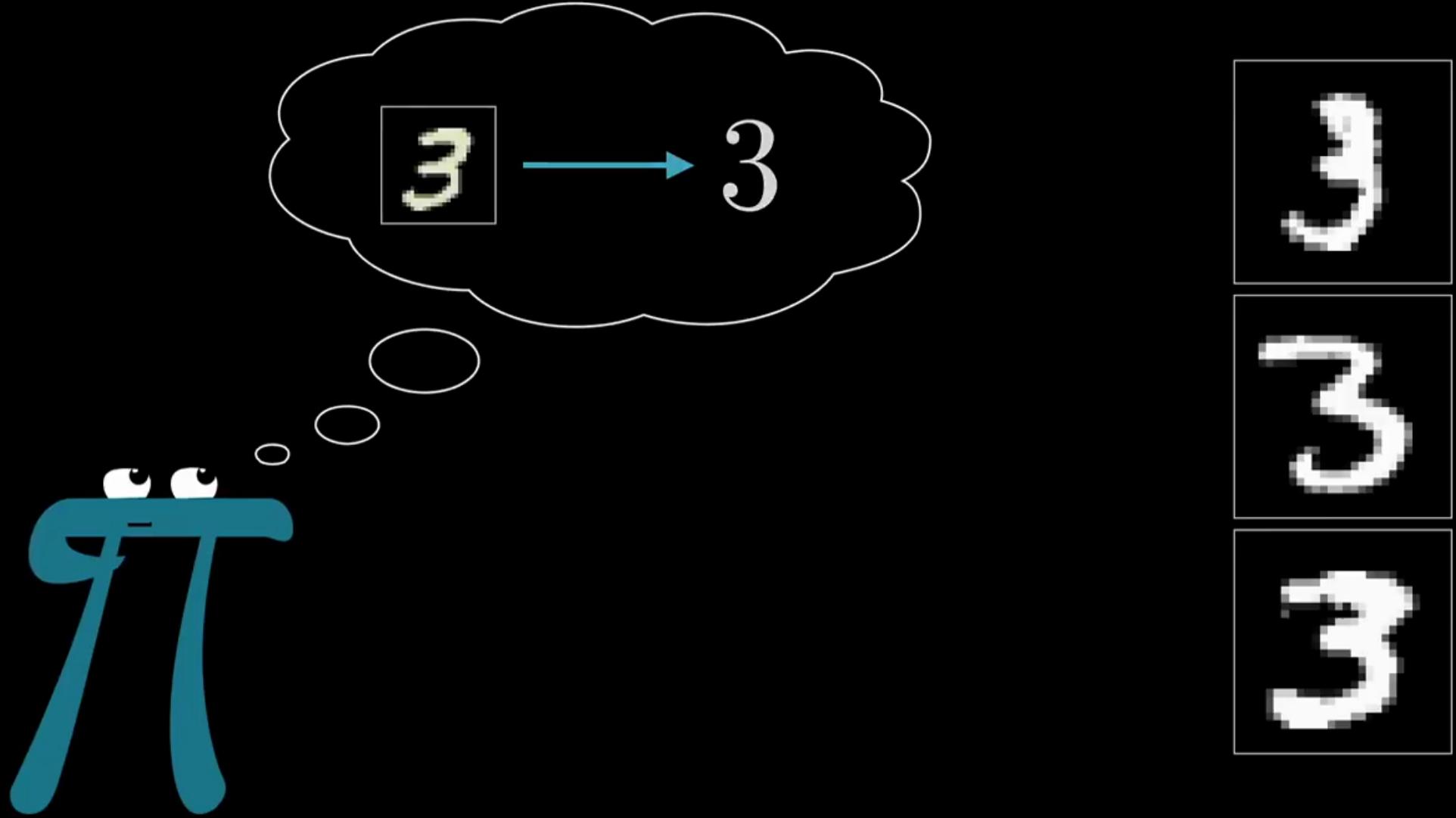
(*Slides : But what *is* a Neural Network? Chapter 1, deep learning by 3Blue1Brown)
<https://www.youtube.com/watch?v=aircAruvnKk&index=6&list=PL7glvrW4cSVNjYKRxS7okEEhuZs8UpvpE&t=964s>

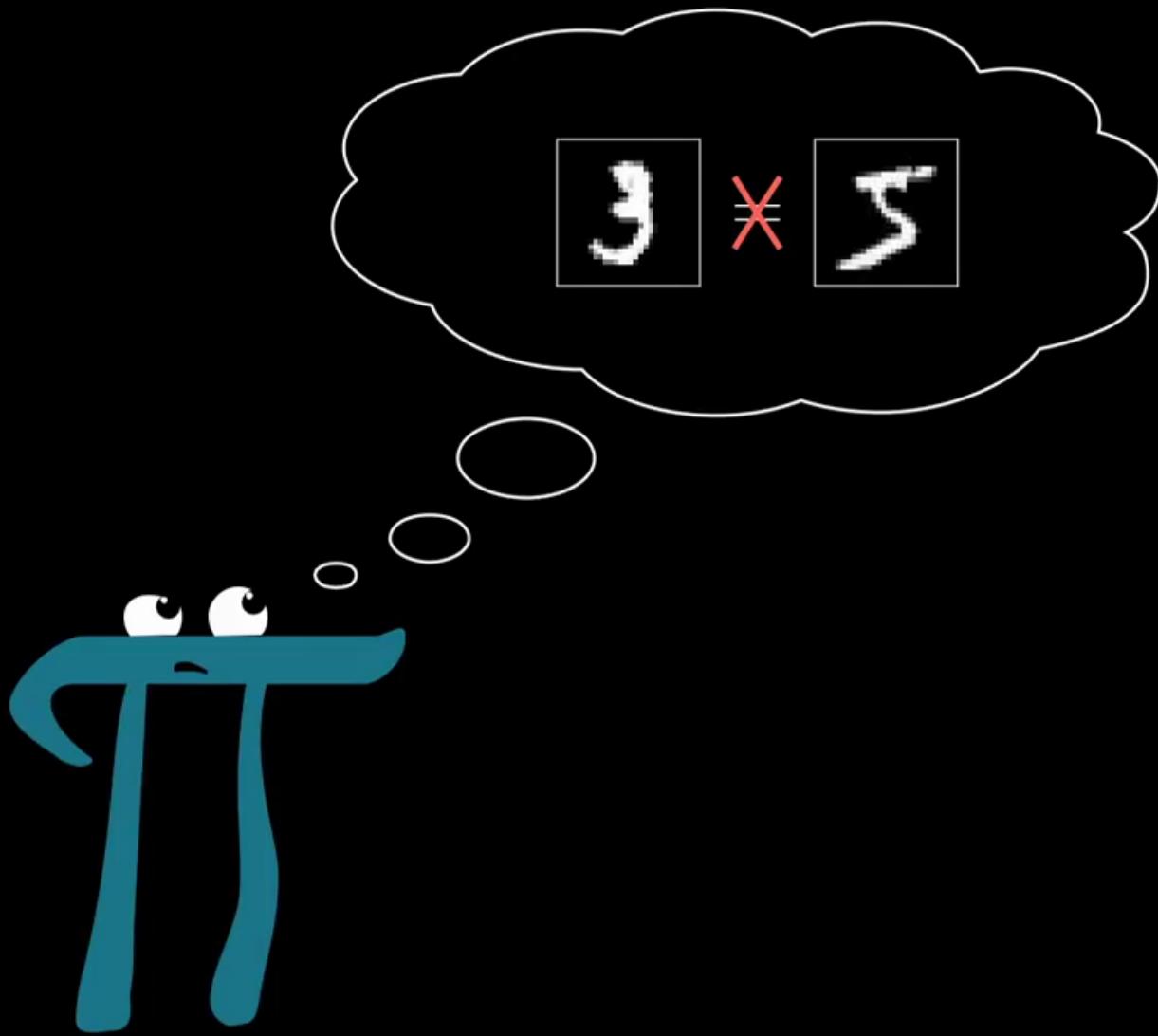


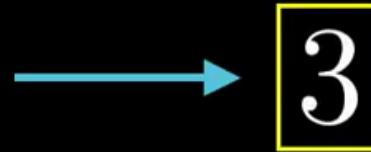
28px

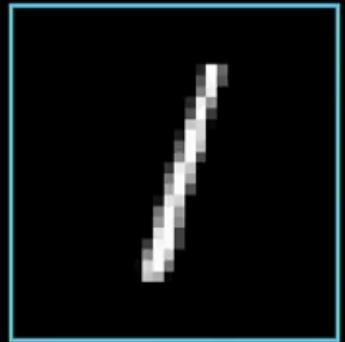


28px

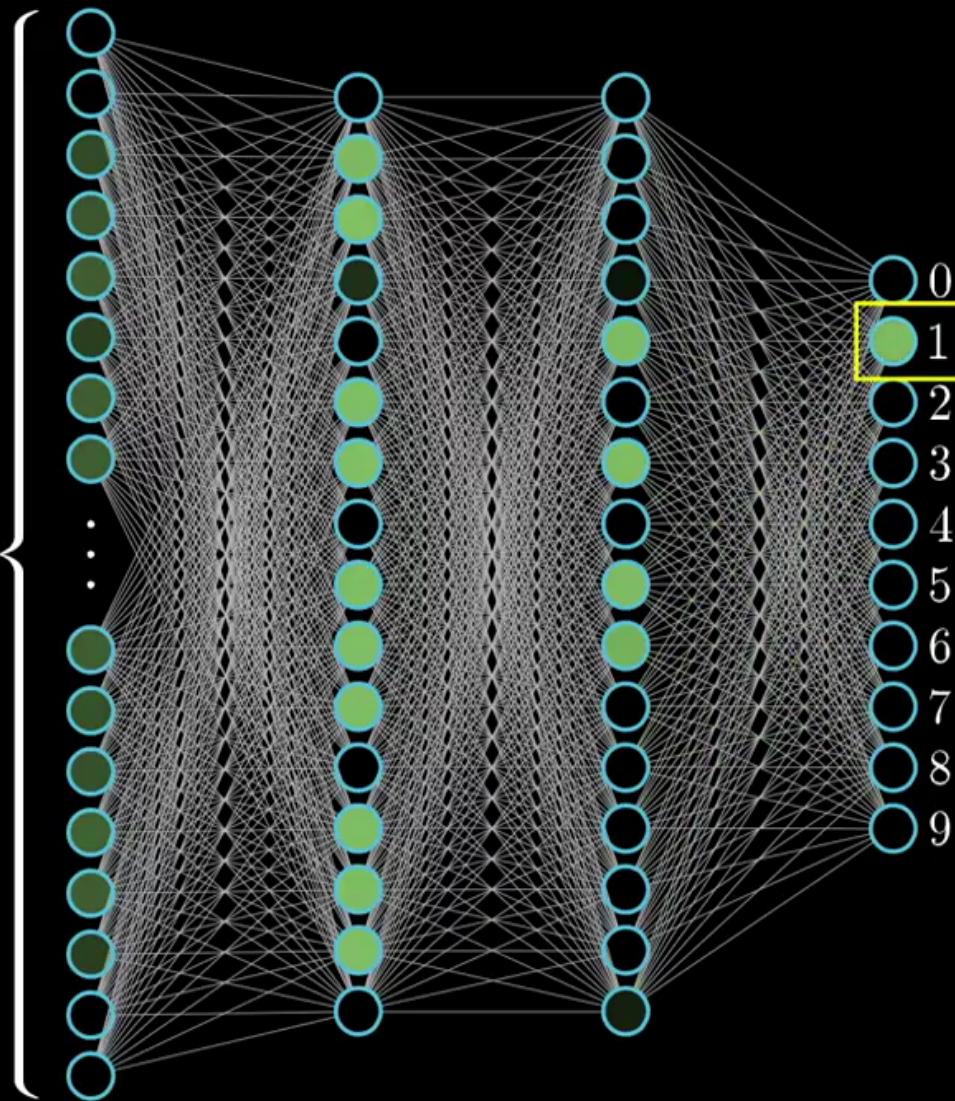






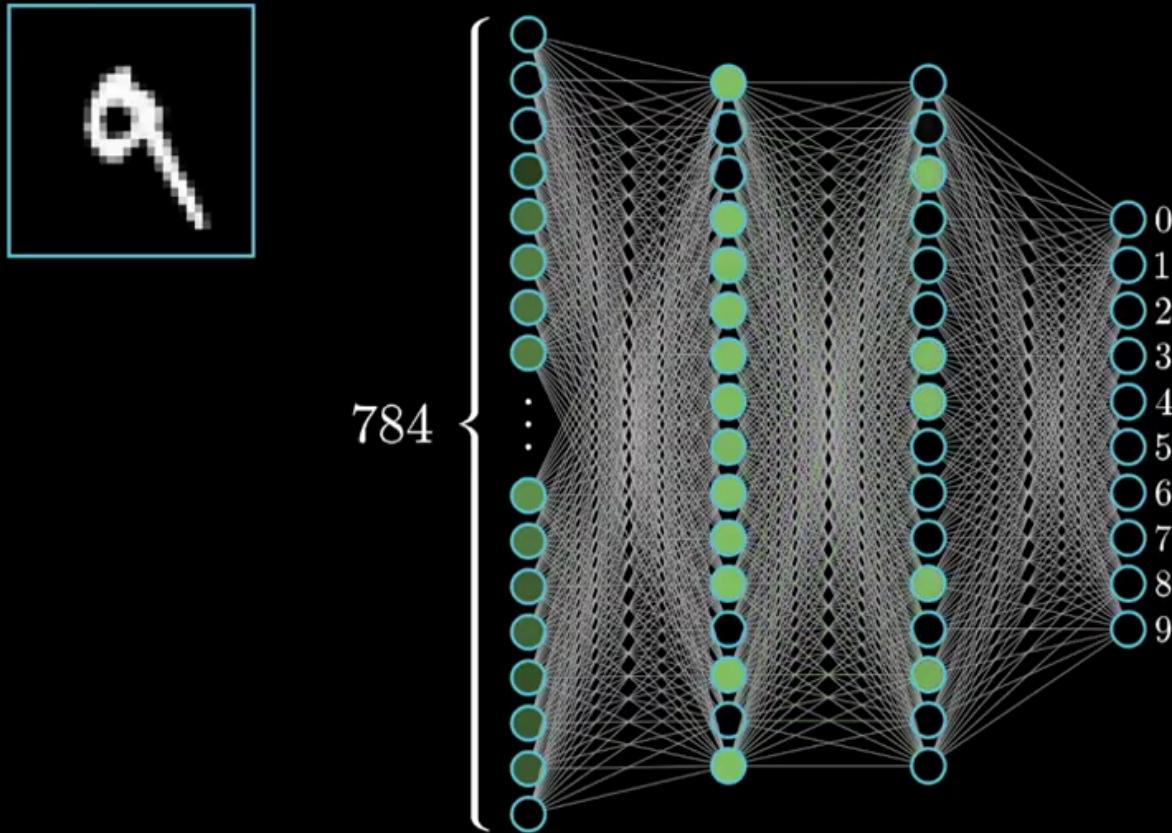


784



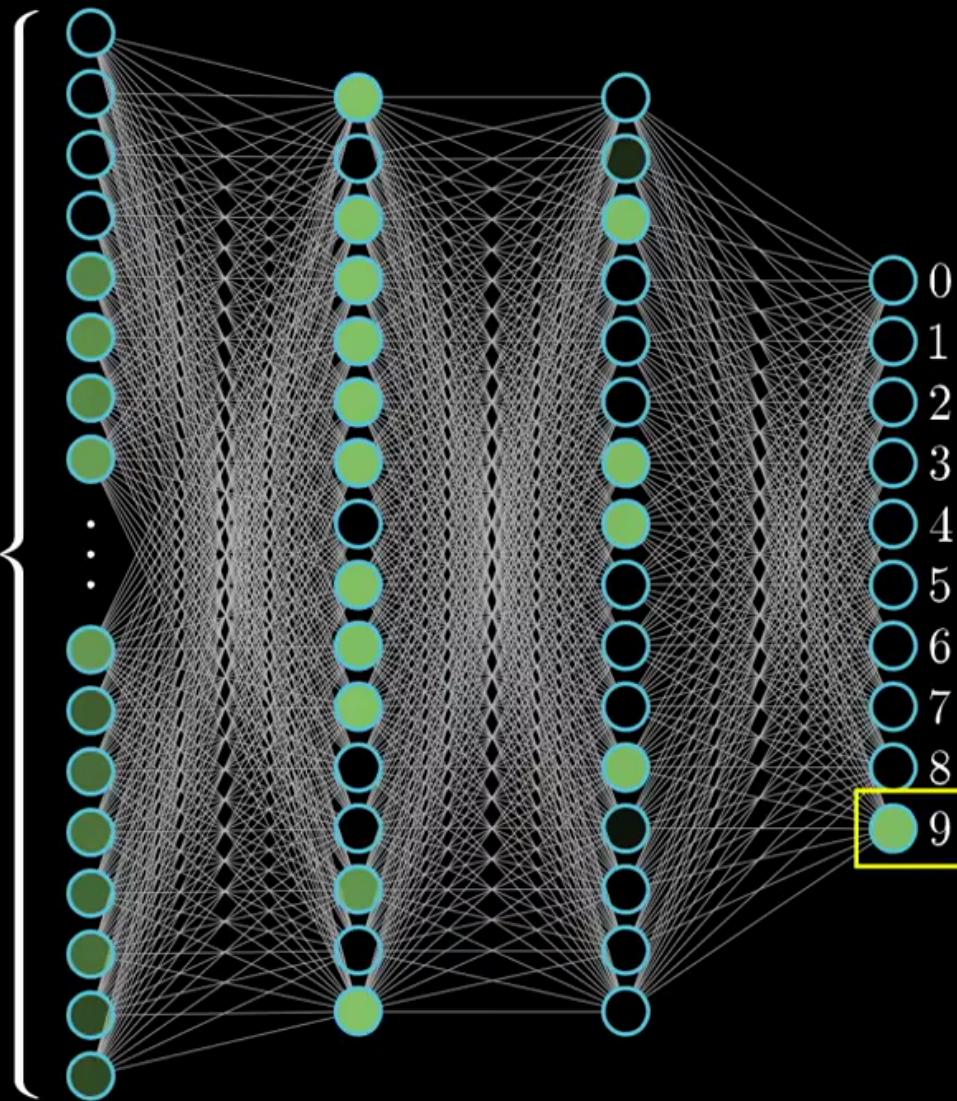
Plain vanilla

(aka “multilayer perceptron”)





784



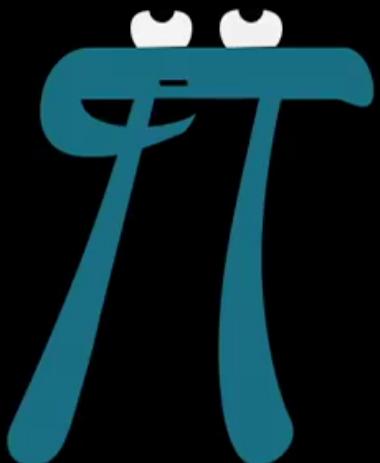


Neural network



What are
the neurons?

How are
they connected?

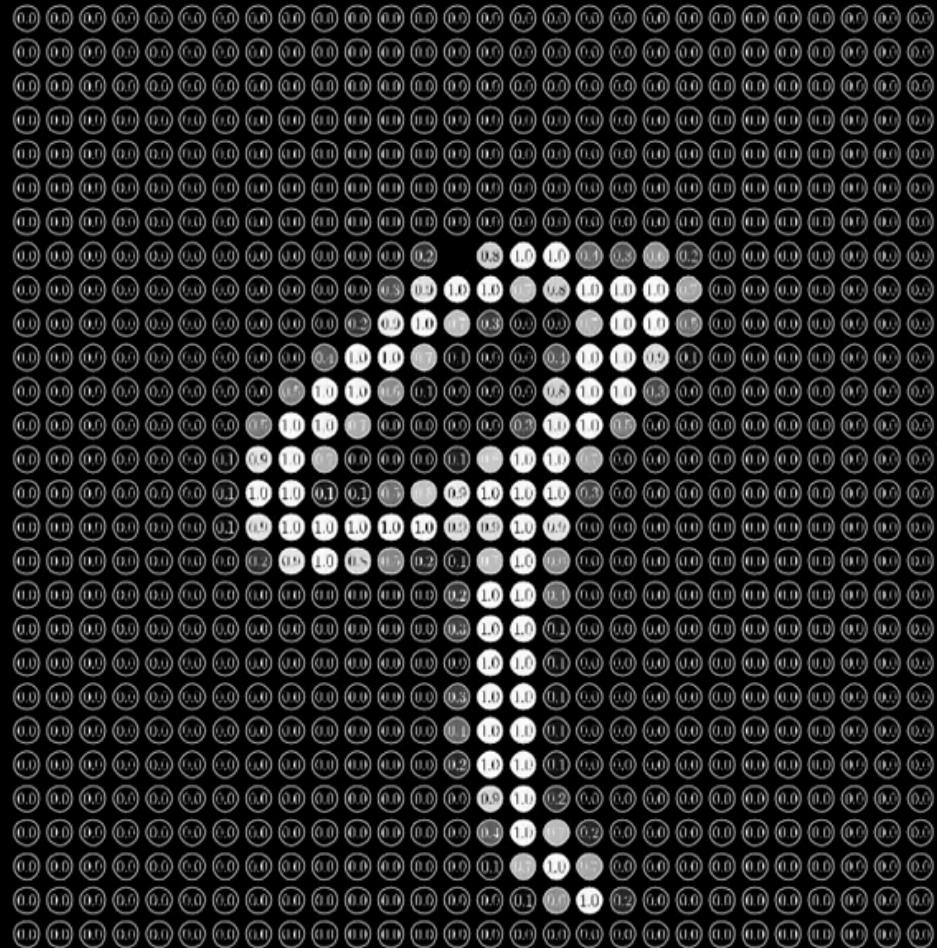


0.1

Neuron → Thing that holds a number

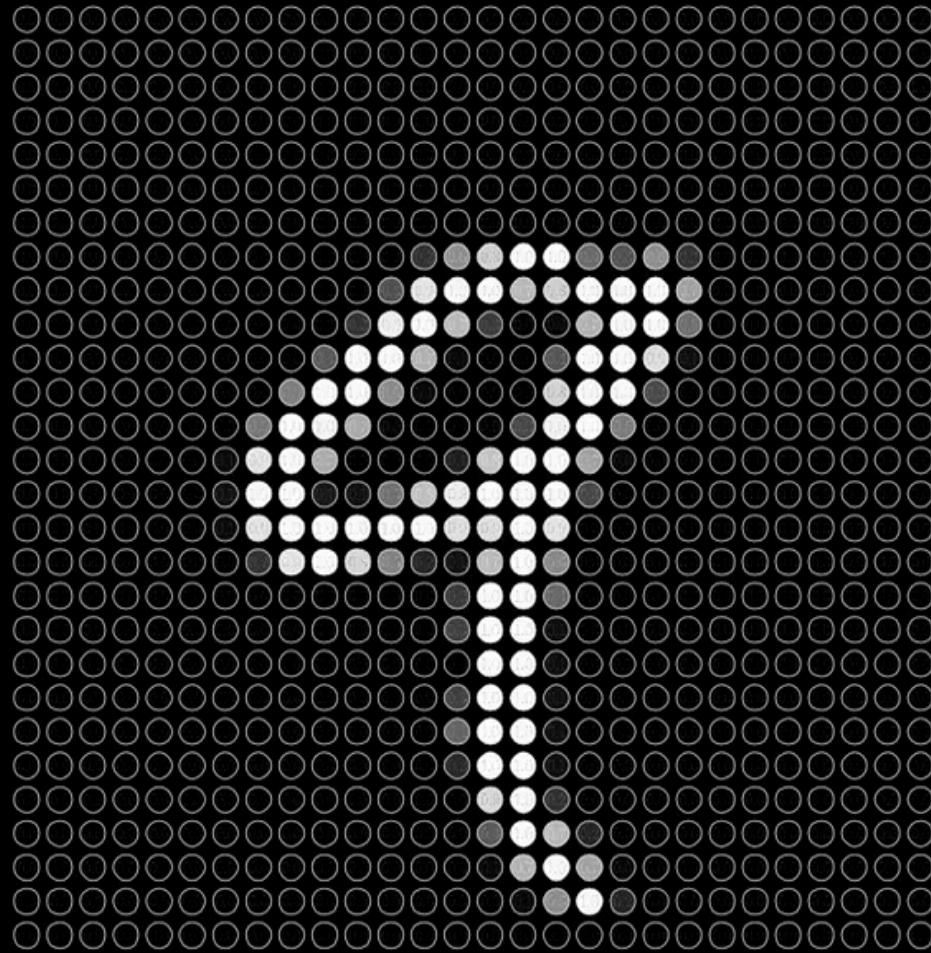
28

28

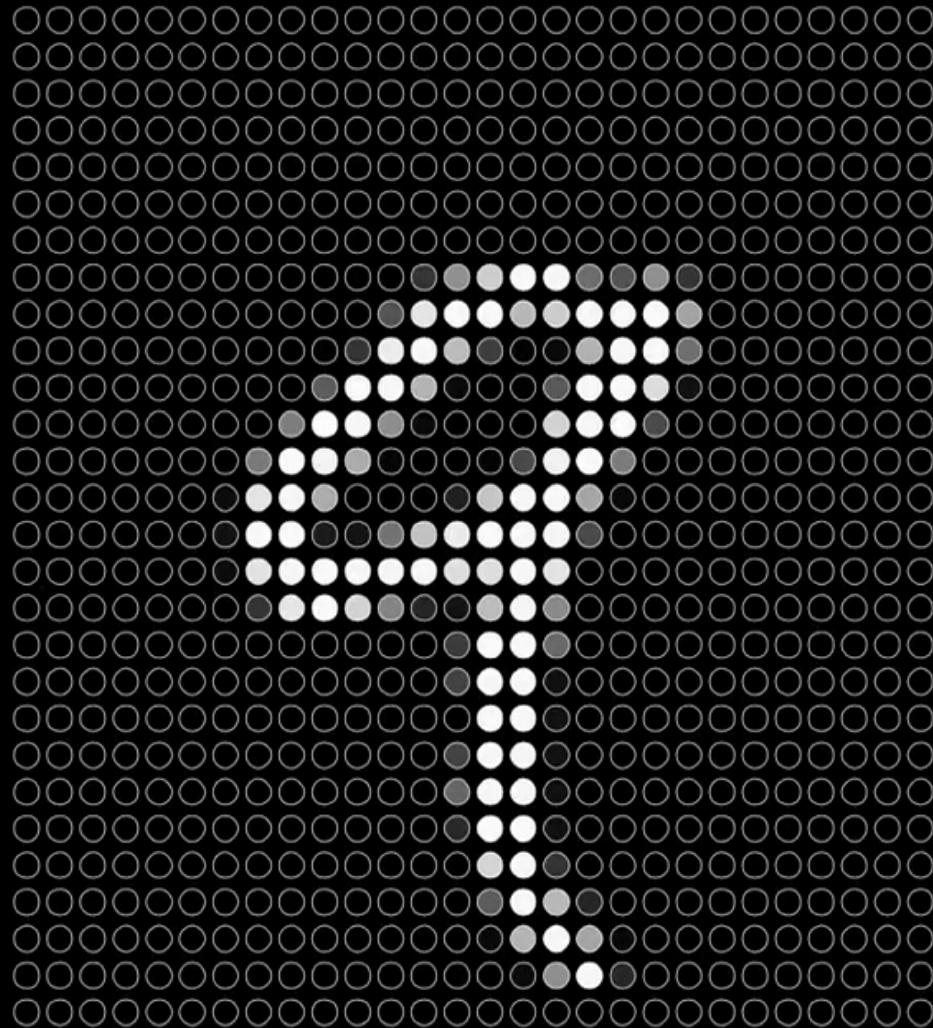


$$28 \times 28 = 784$$

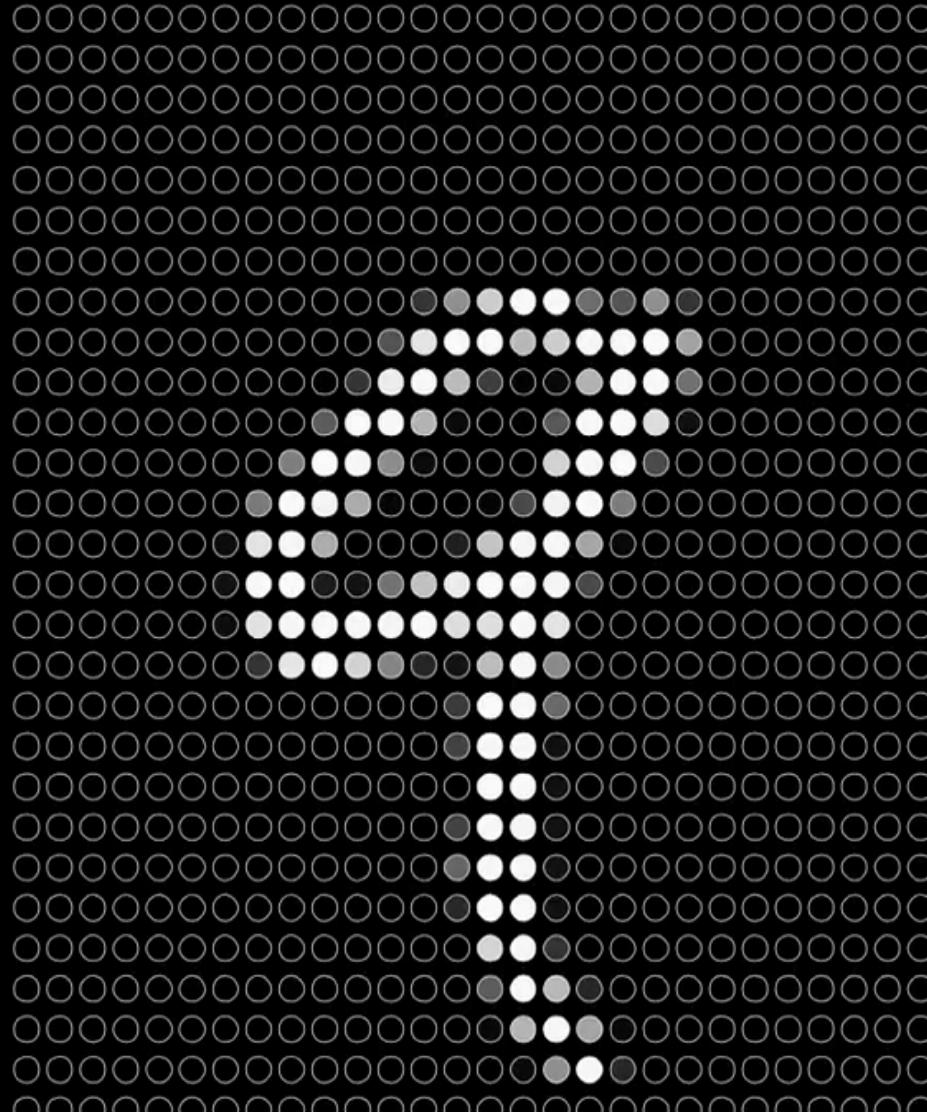
0.58



784



784



784

The image shows a 10x10 grid of small circles. The majority of the circles are white. In the bottom two columns, there is a repeating pattern of three types of gray circles: white, light gray, and dark gray. This pattern repeats every three columns across the bottom two rows.

784

oooooooooooooooooooo

A decorative horizontal bar at the bottom of the page, composed of a series of circles in different sizes and shades of gray, creating a textured pattern.

○○○○○○○●●●○○○○●●●○○○○○○○○

A decorative horizontal bar at the bottom of the page, composed of a series of circles in different sizes and shades of gray, creating a textured and organic look.

oooooooooooooooooooo●●●oooooooooooooooo

oooooooooooooooooooo●●oooooooooooooooo

784

A horizontal sequence of 20 circular icons. The icons are arranged in two rows: the top row contains 10 icons, and the bottom row contains 10 icons. The icons are colored in various shades of gray and white, suggesting a grayscale color palette. They are evenly spaced along a horizontal axis.

784

A decorative horizontal bar at the bottom of the page, composed of a series of circles in different sizes and shades of gray, creating a repeating pattern.

A decorative horizontal bar at the bottom of the slide, composed of a series of circles in different sizes and shades of gray, creating a pattern of alternating light and dark segments.

oooooooooooo

784

○●●●○○○○○○○○

○○○○○○○●●●○○○○●●●○○○○○○○○○○

○○○○○○○●●●○○●●●●●●●○○○○○○○○○○○

○○○○○○○●●●●●●●●●○○

784

oooooo

oooooooo●●●○○○●●●●○○○○○○○○○○

oooooooo●●○○●●●●●●○○○○○○○○○○○○

oooooo

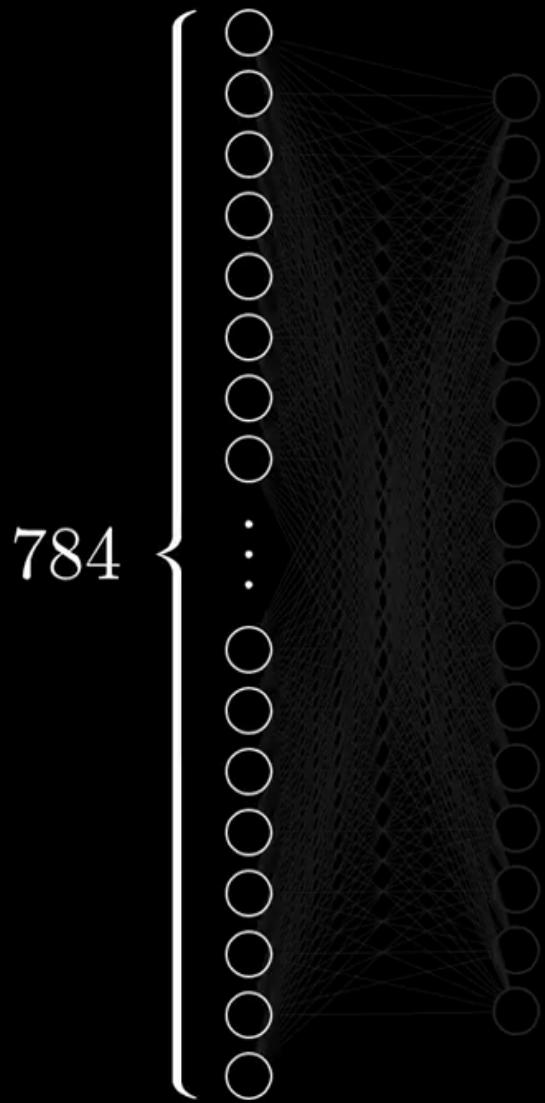
784

000 000000●●●0000●●●0000000000 0000000●●●0000●●●0000000000 000

784

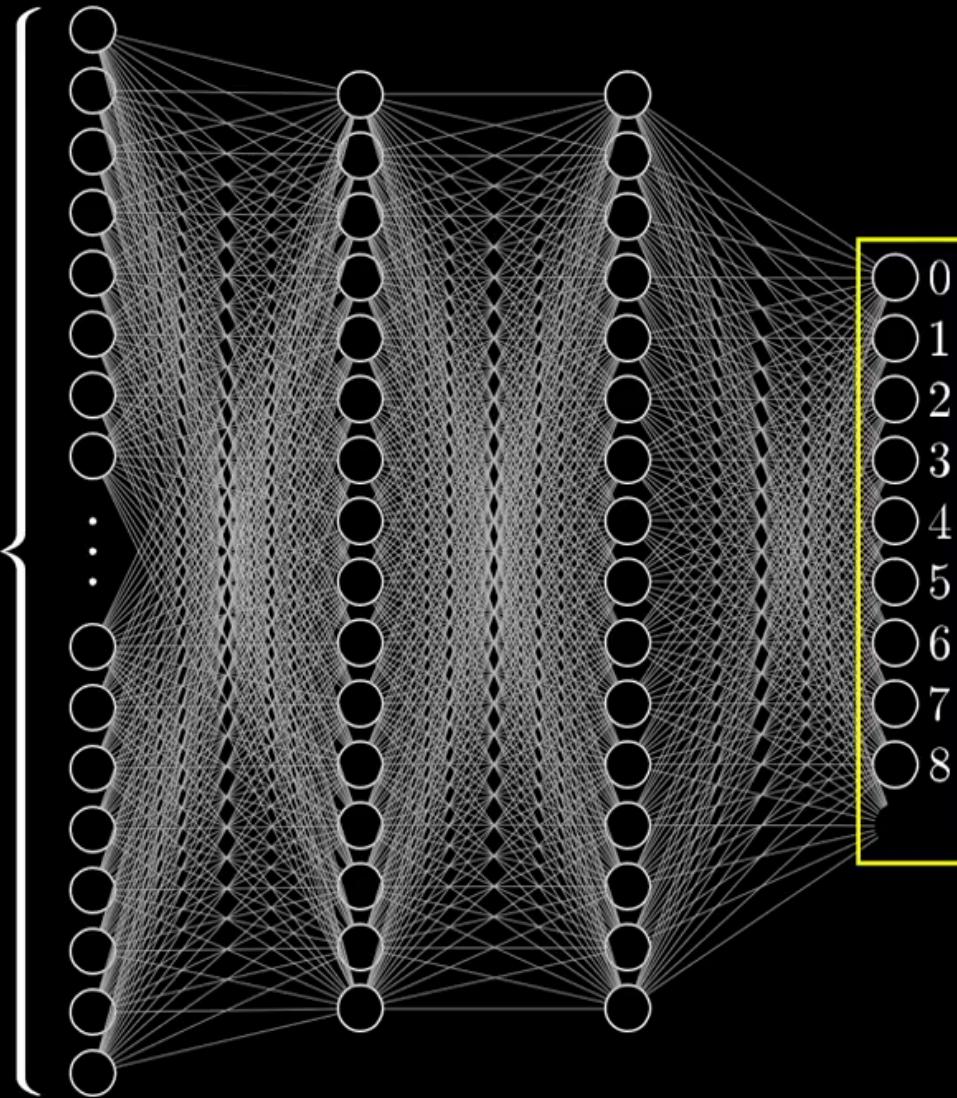


○○○○○○○○○ ··· ○○○○○○○○○
784





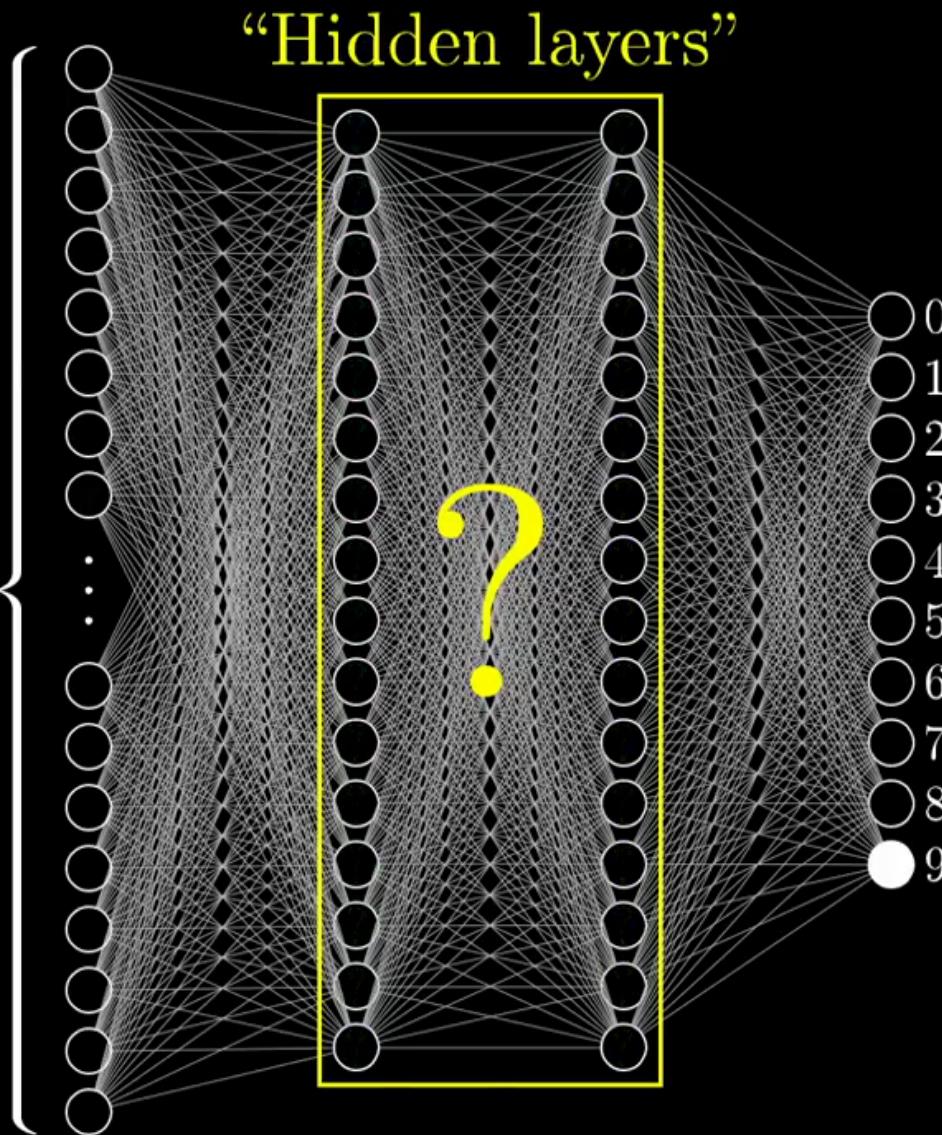
784



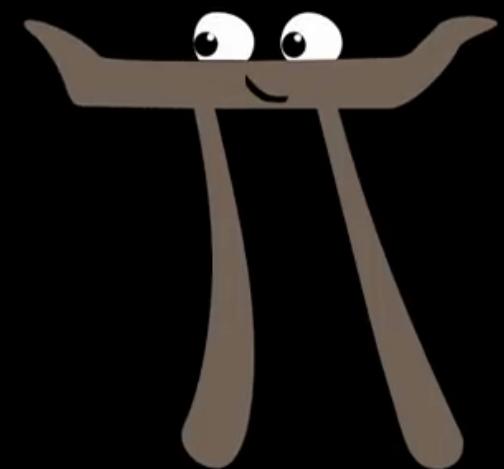
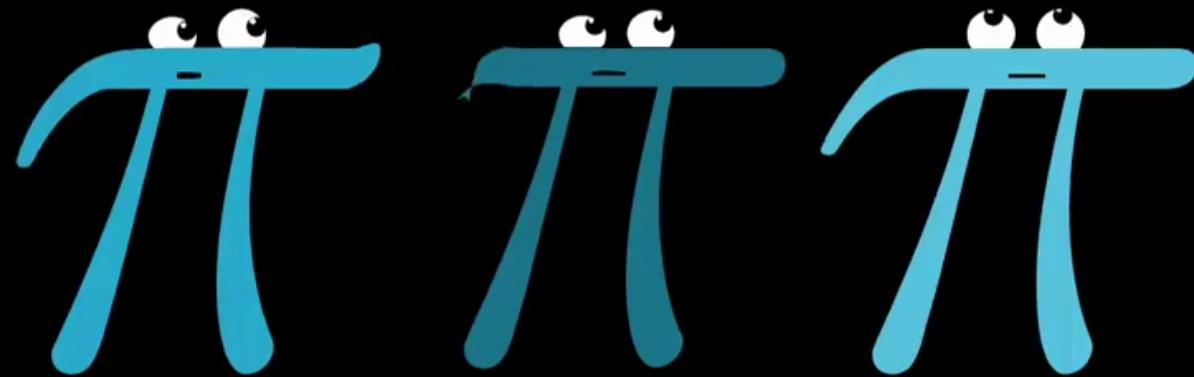
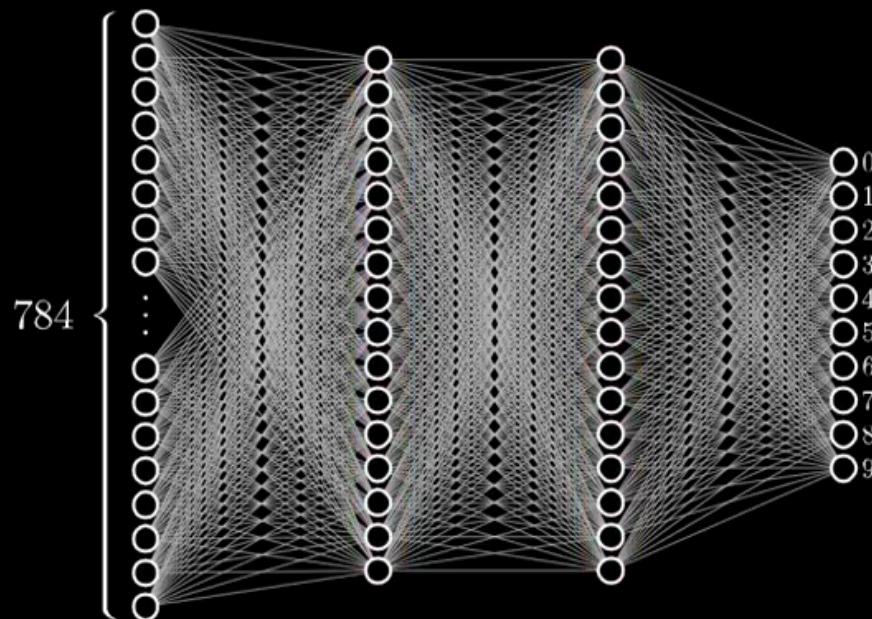
0.97 9



784



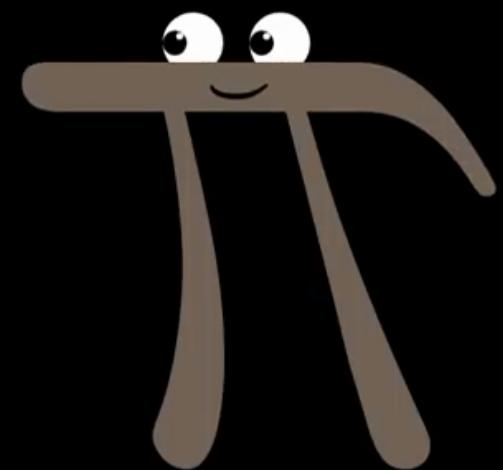
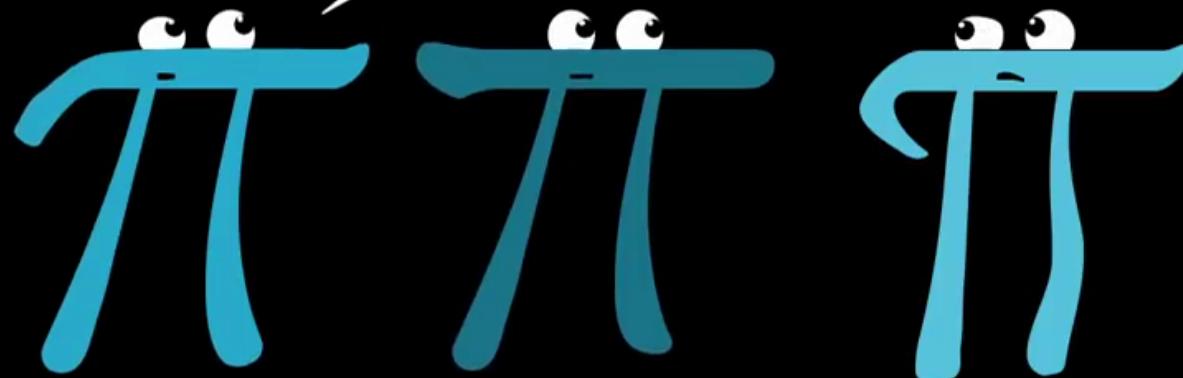
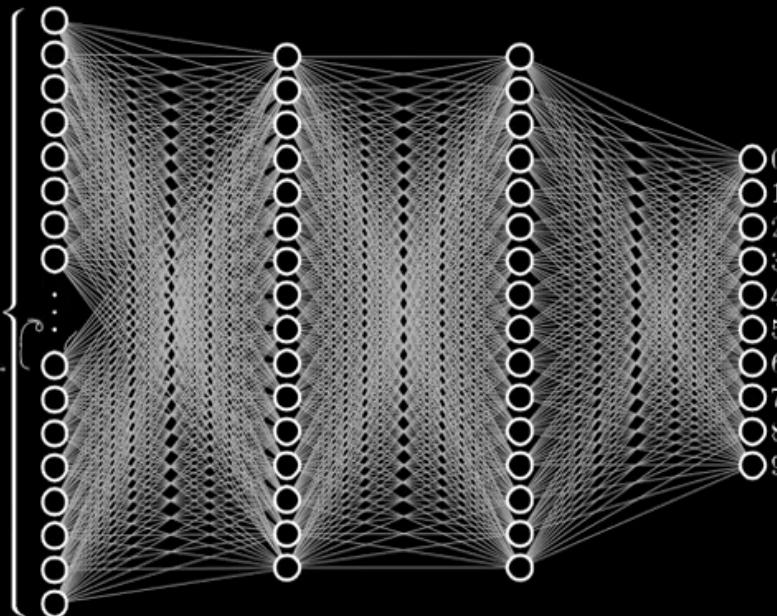
16 neurons each



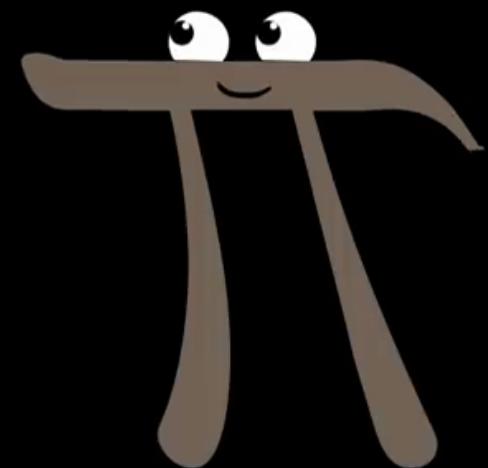
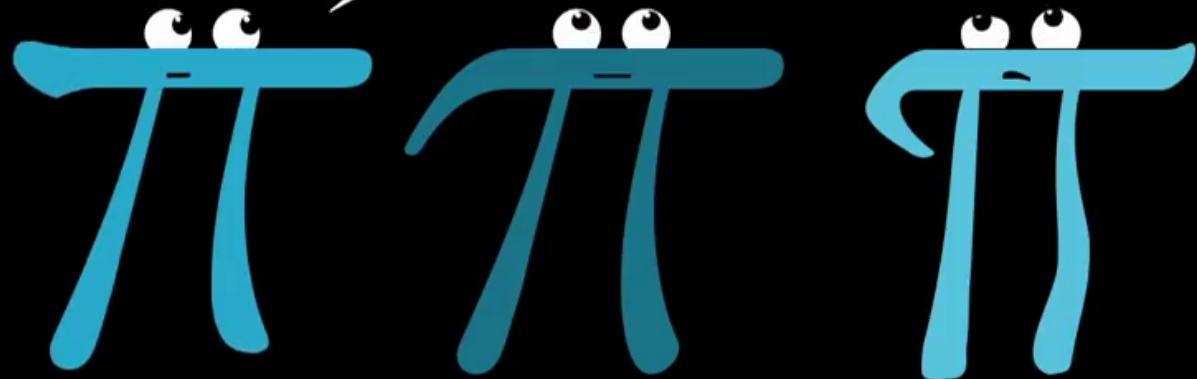
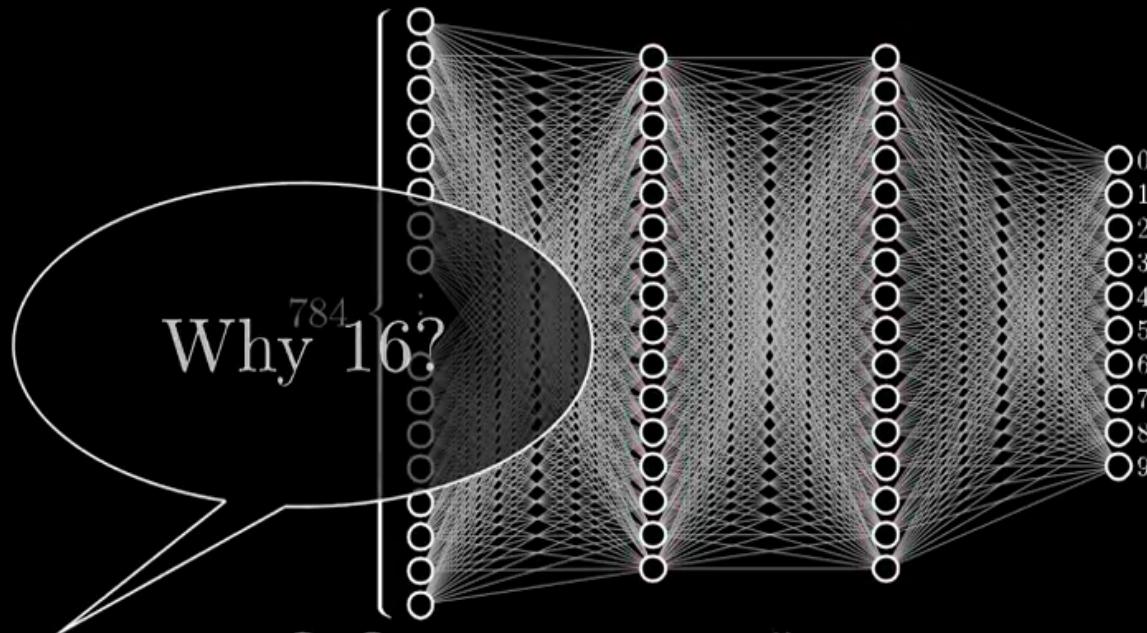
16 neurons each

Why 2
layers?

Why 784

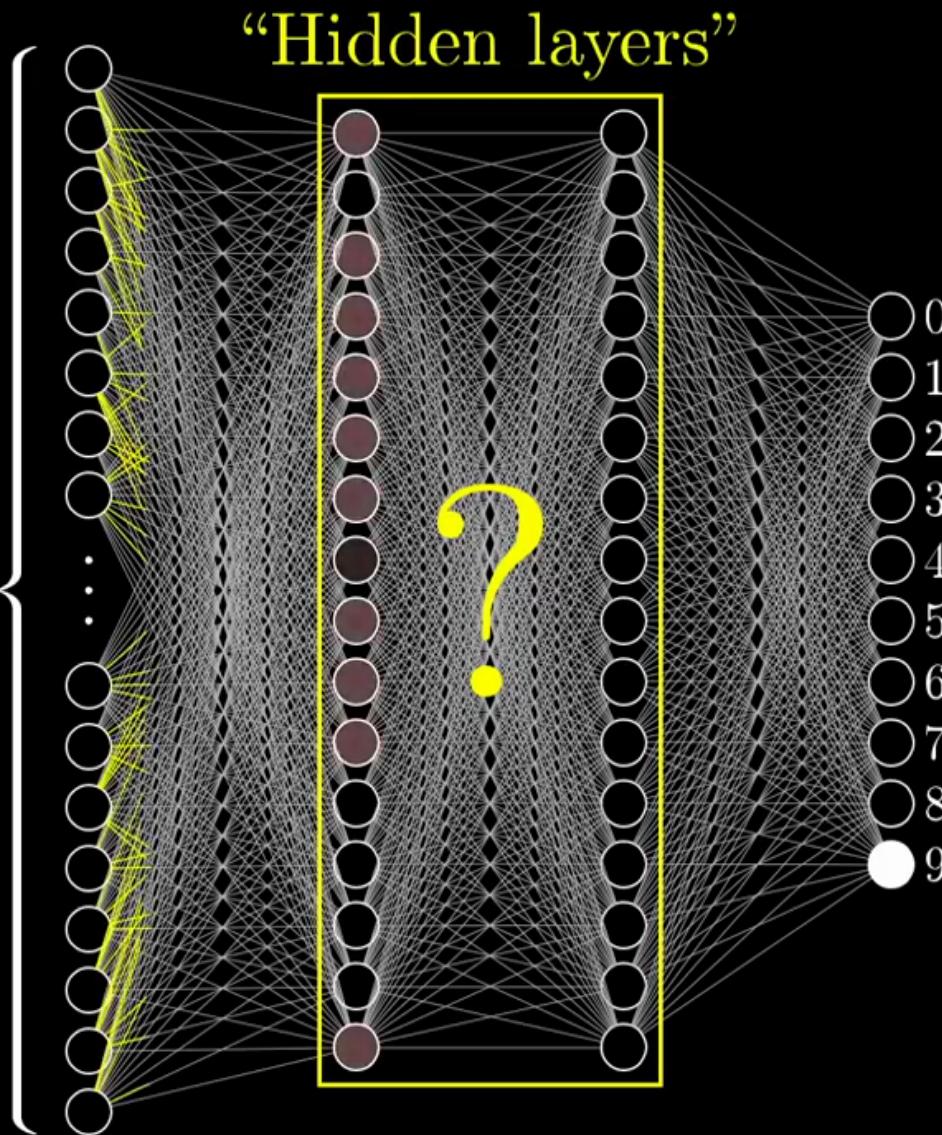


16 neurons each



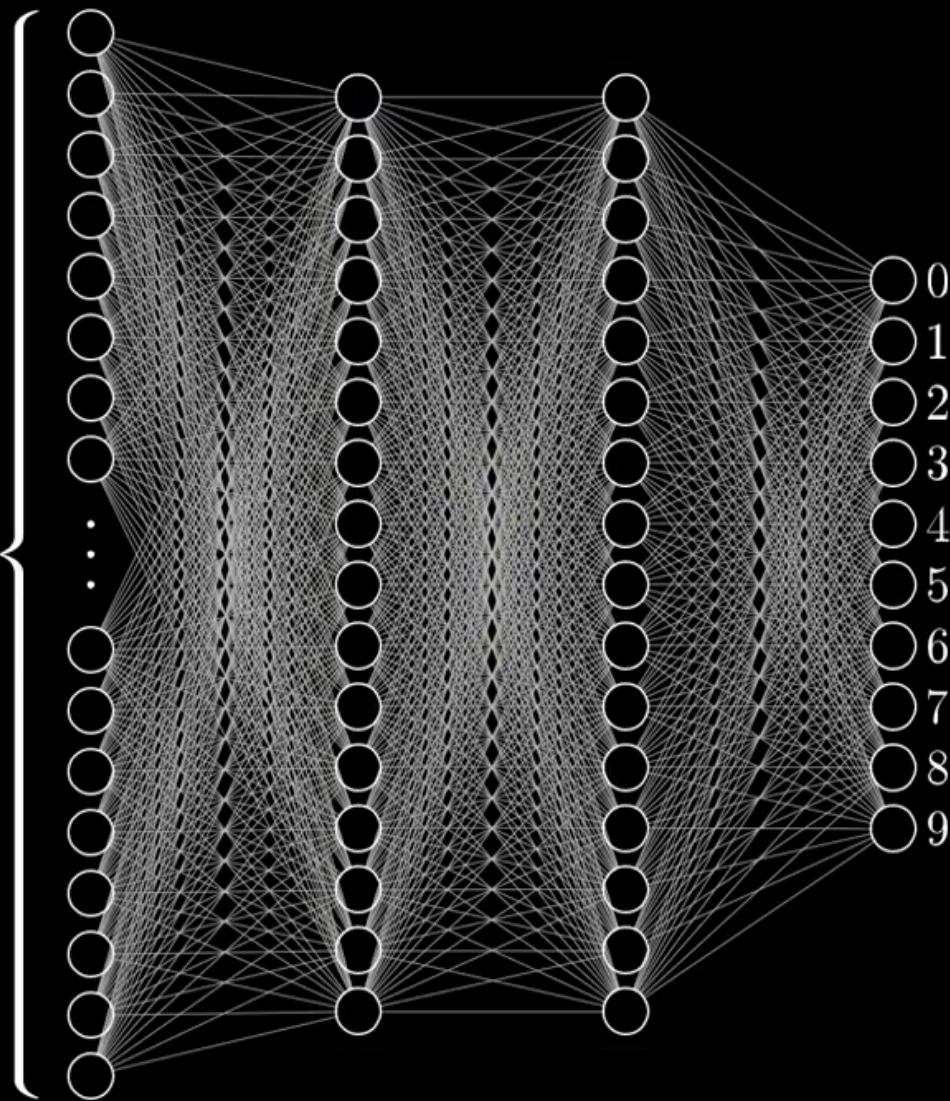


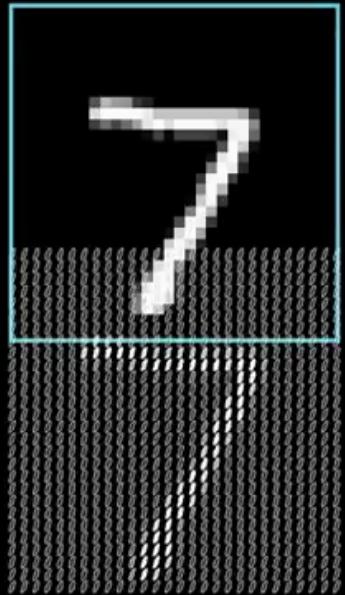
784



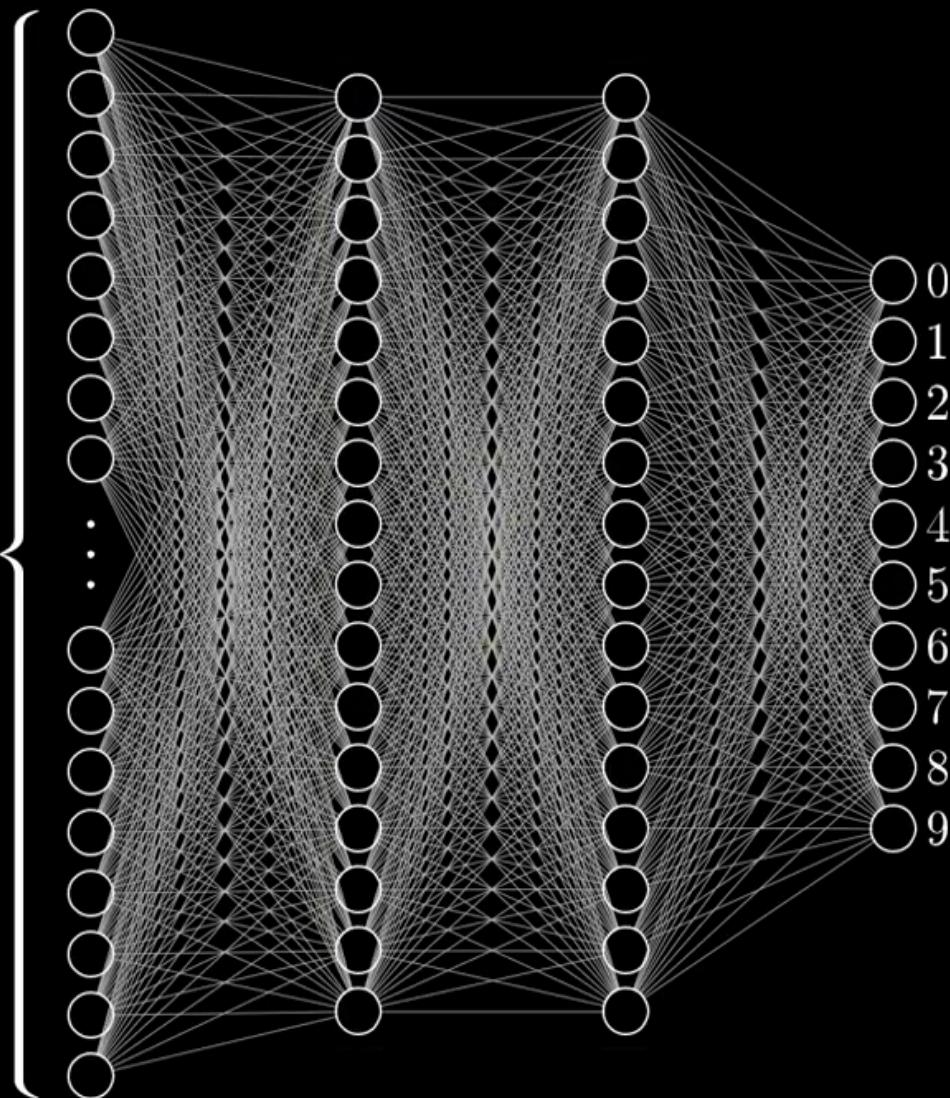


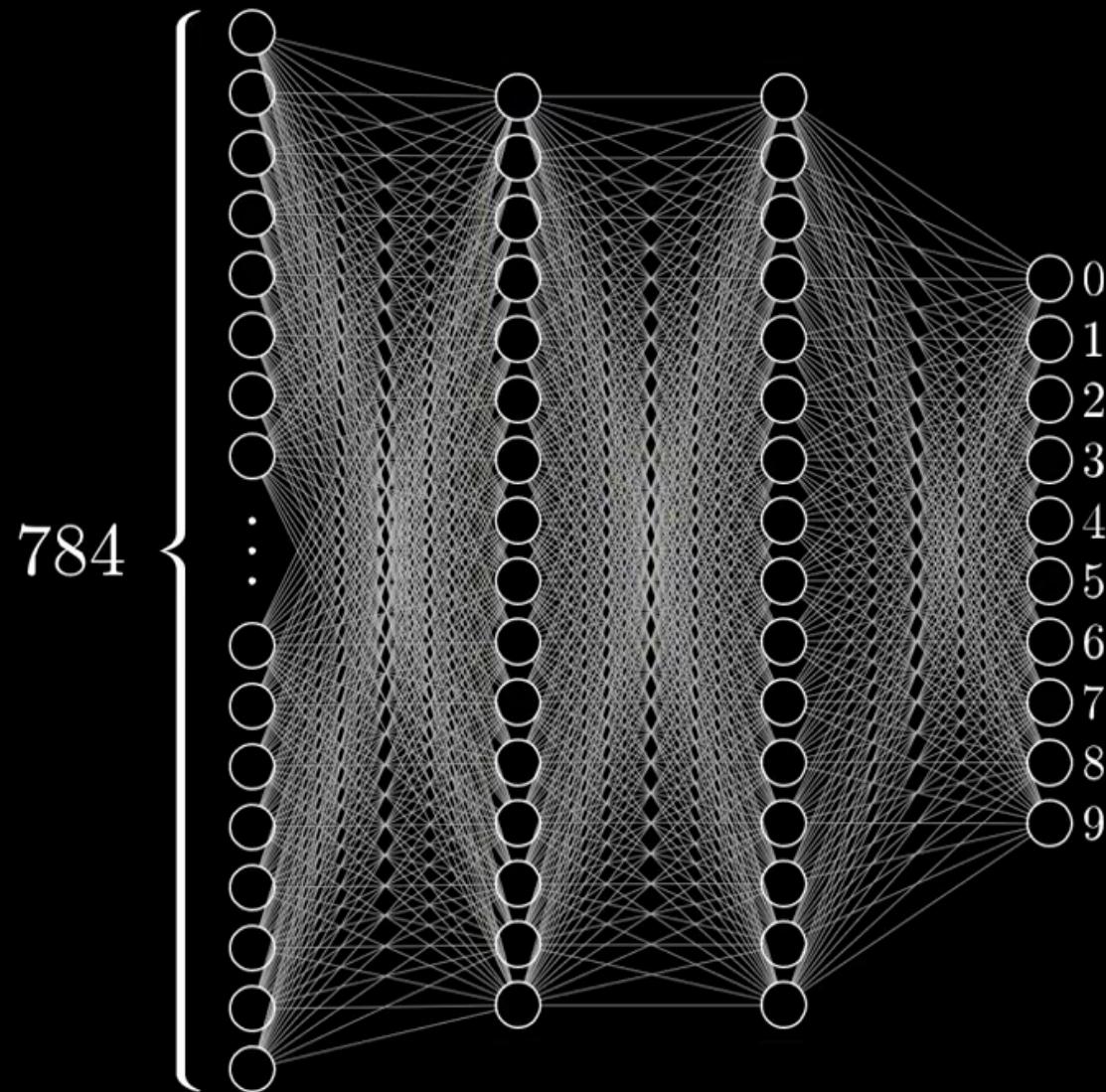
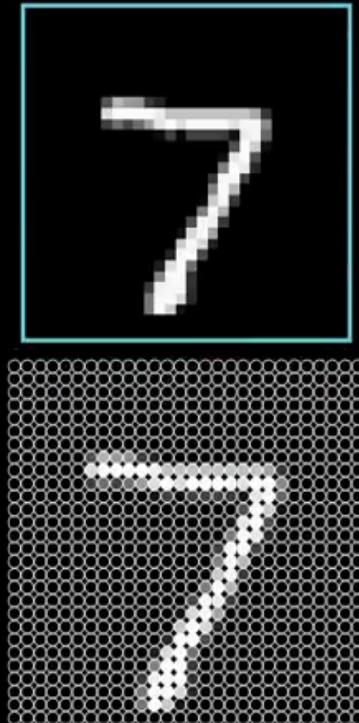
784

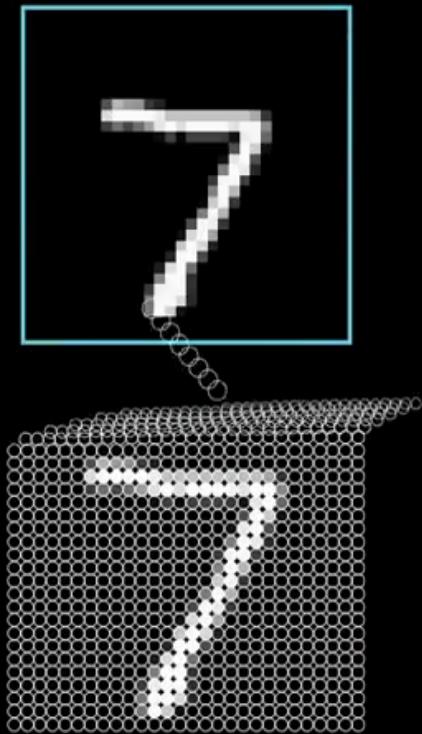




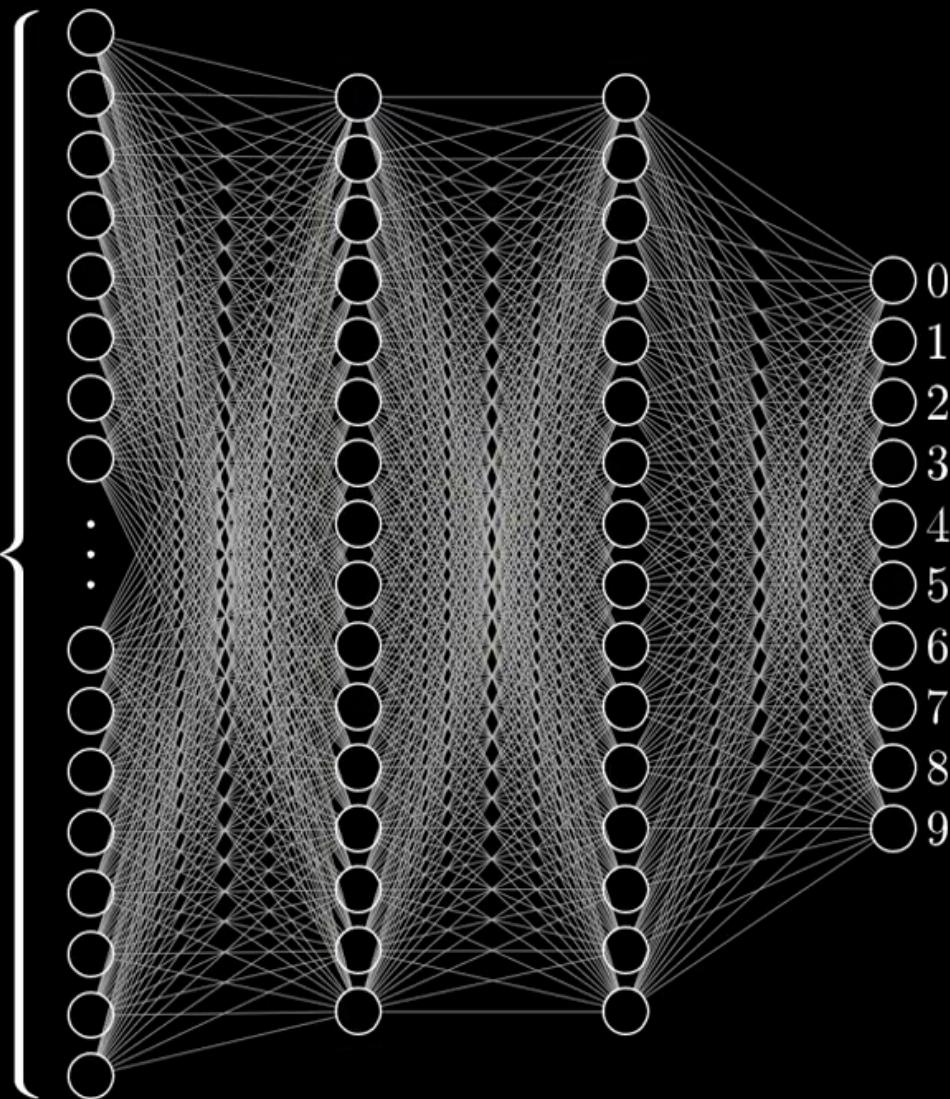
784





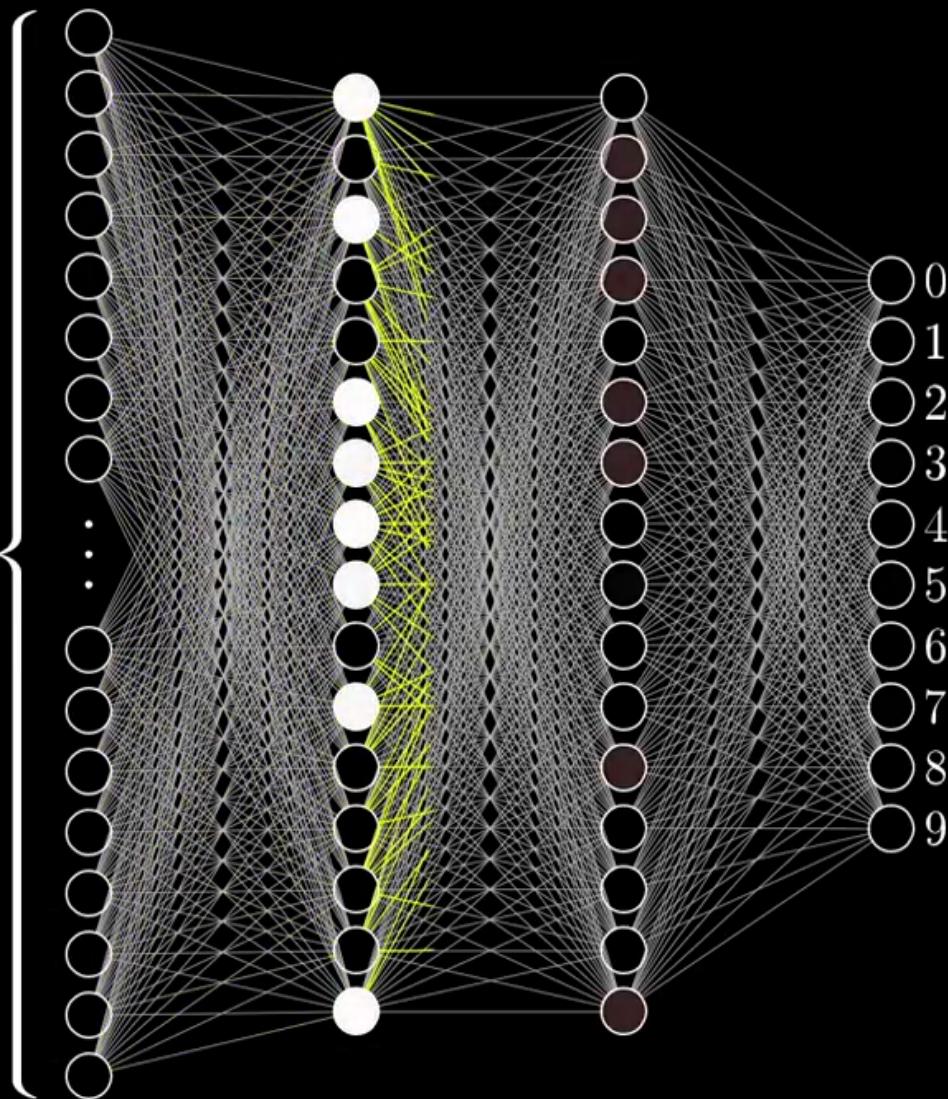


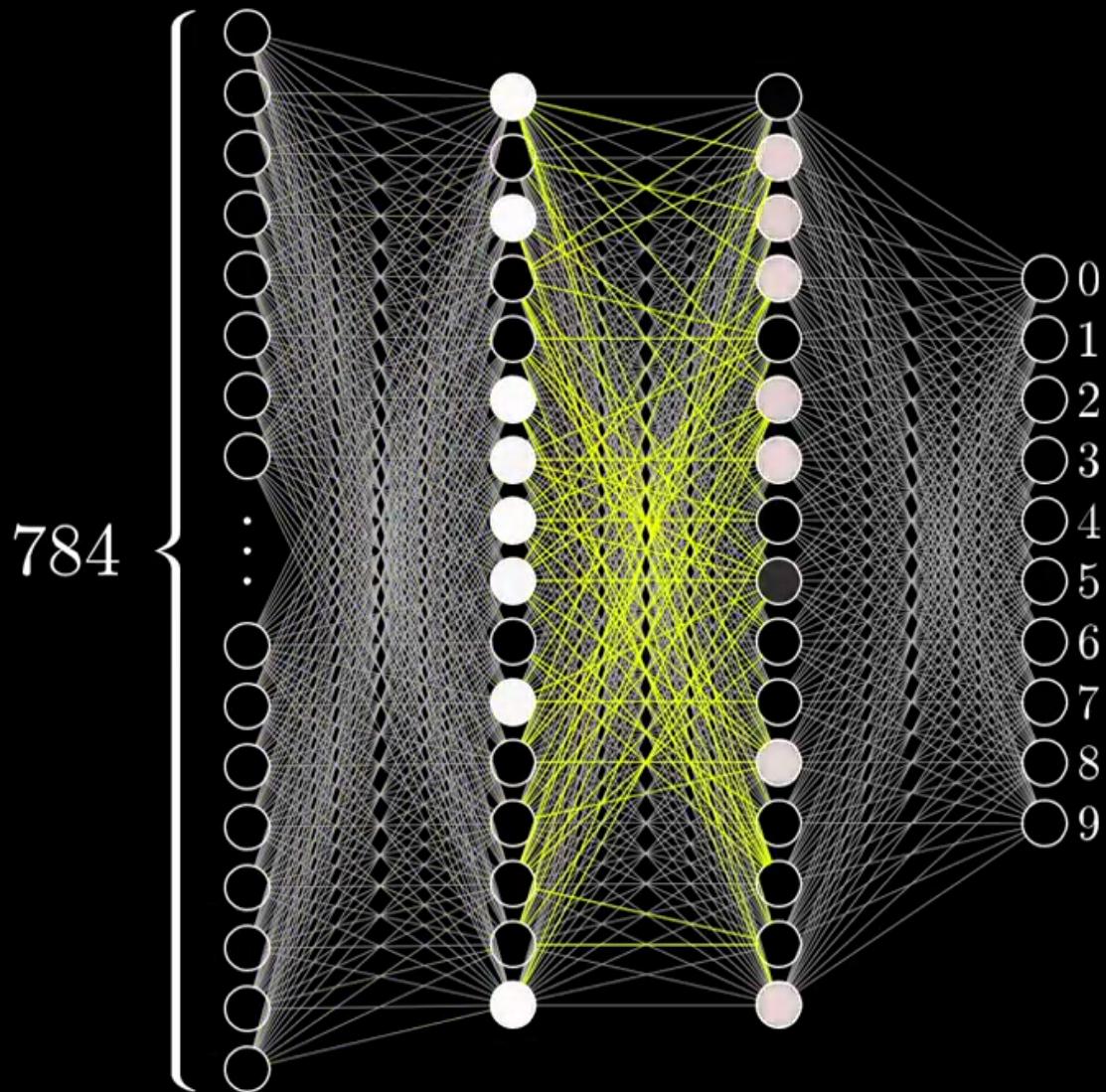
784

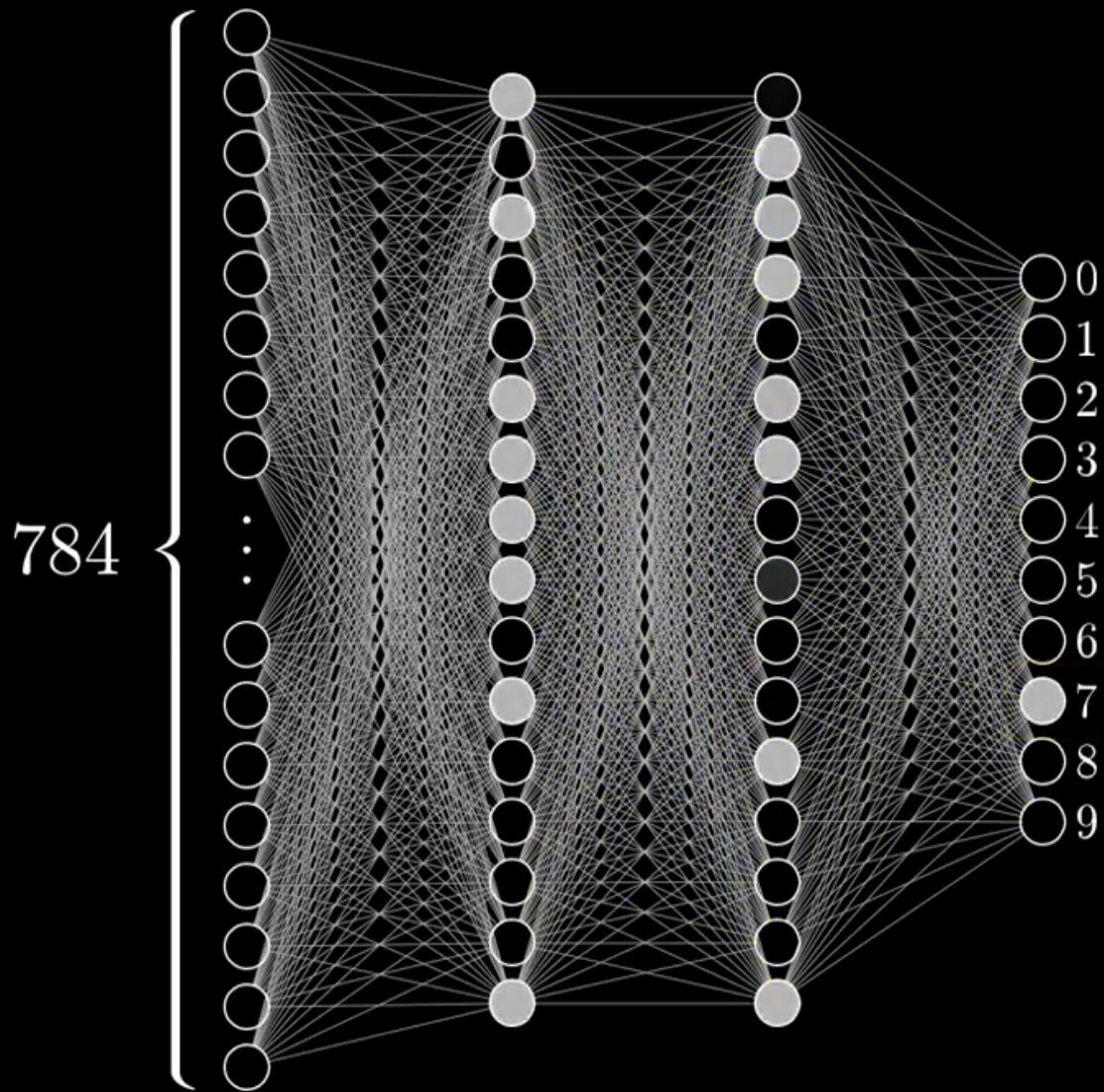


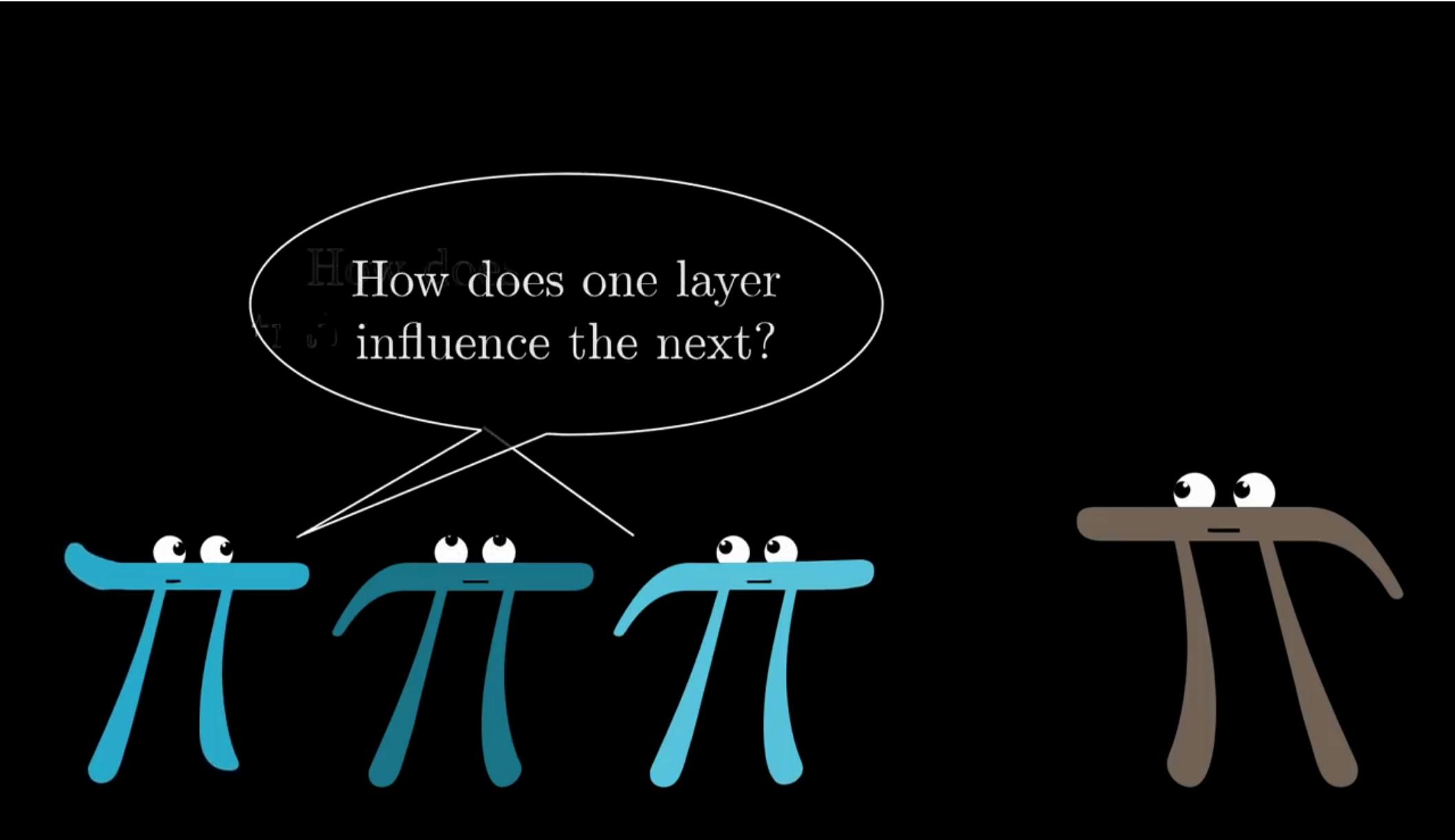


784

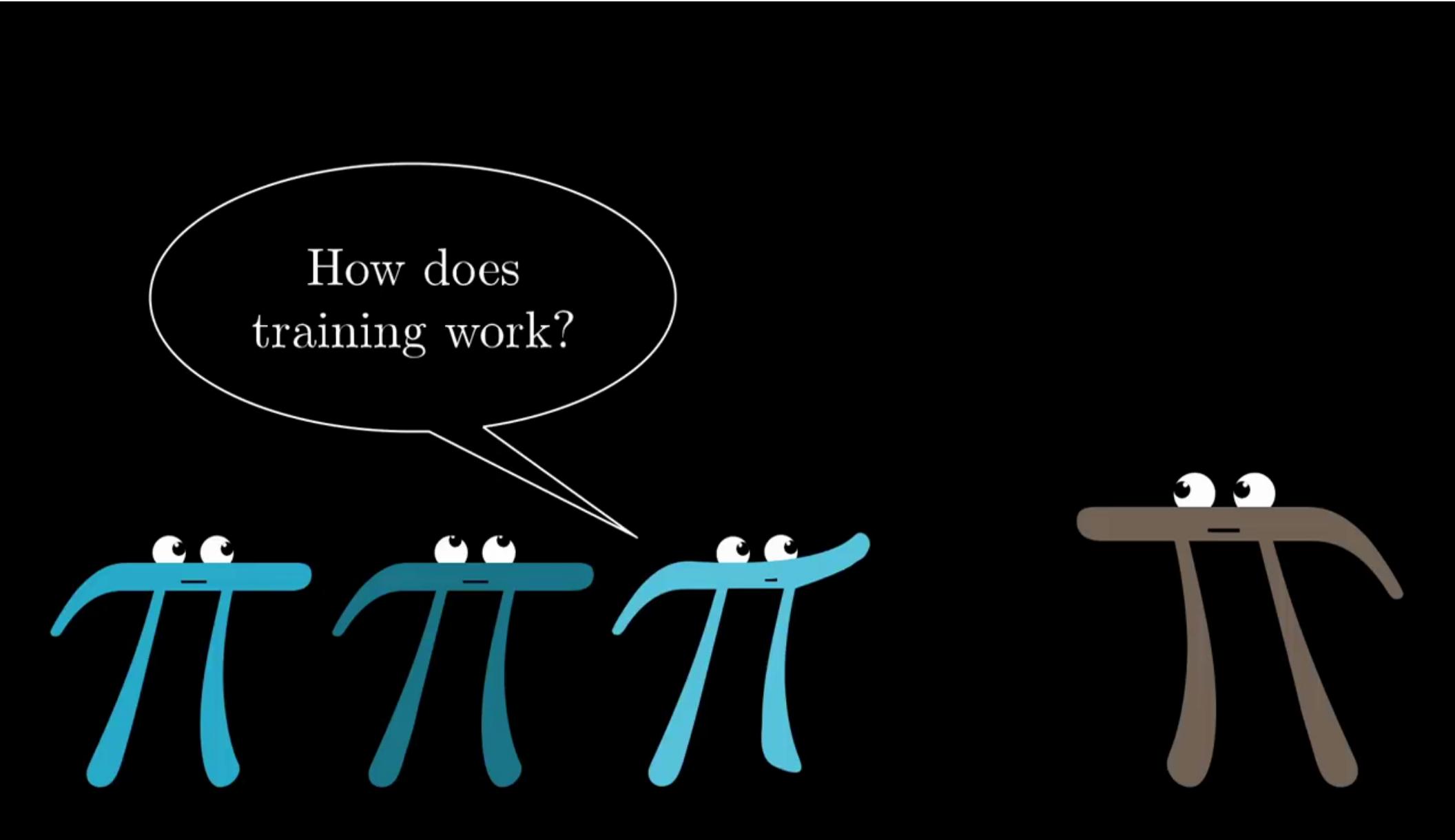




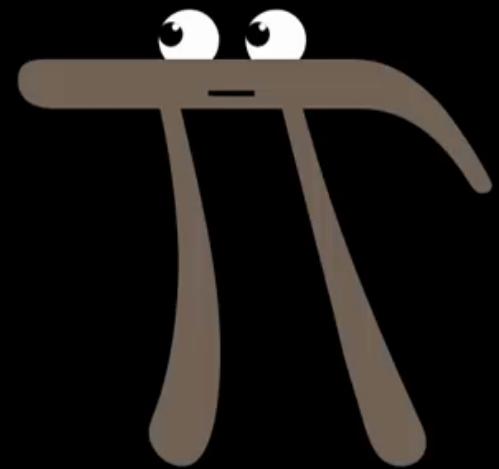
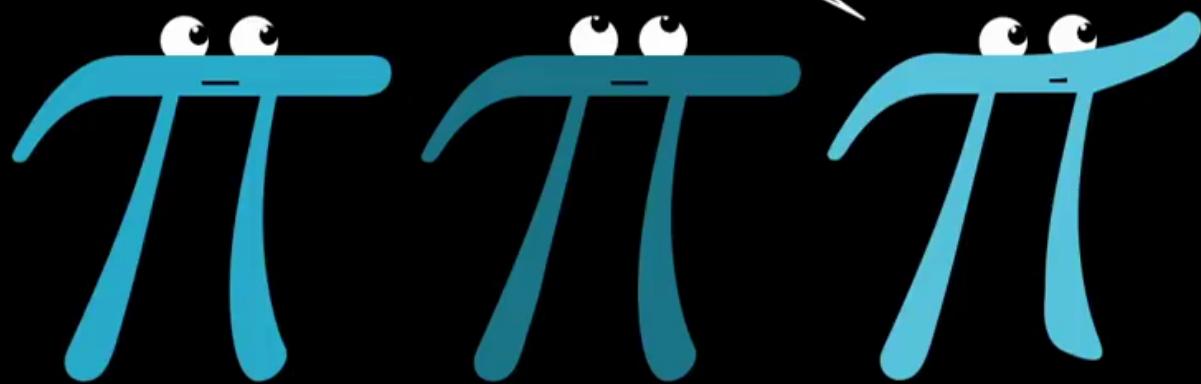




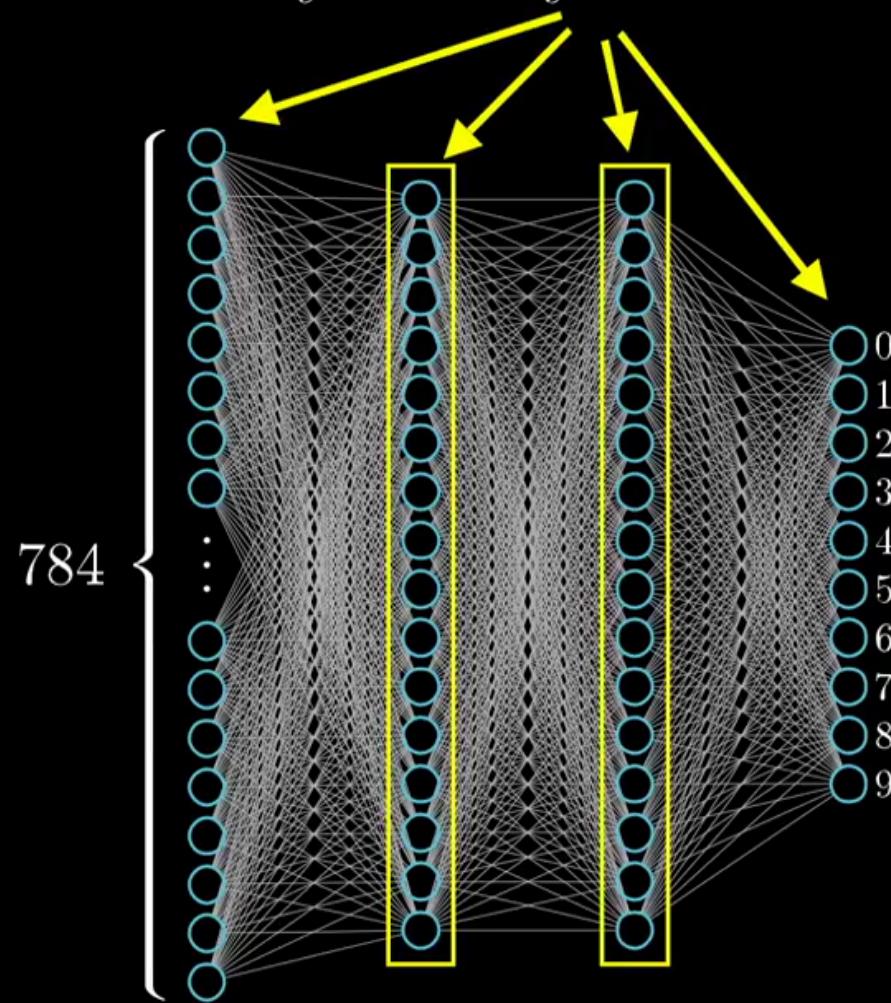
How does one layer
influence the next?



How does
training work?



Why the layers?



$$q = \textcolor{yellow}{o} + \textcolor{red}{l}$$

$$g = \textcolor{yellow}{o} + \textcolor{green}{l}$$

$$4 = \textcolor{red}{l} + \textcolor{teal}{f} + \textcolor{magenta}{r}$$

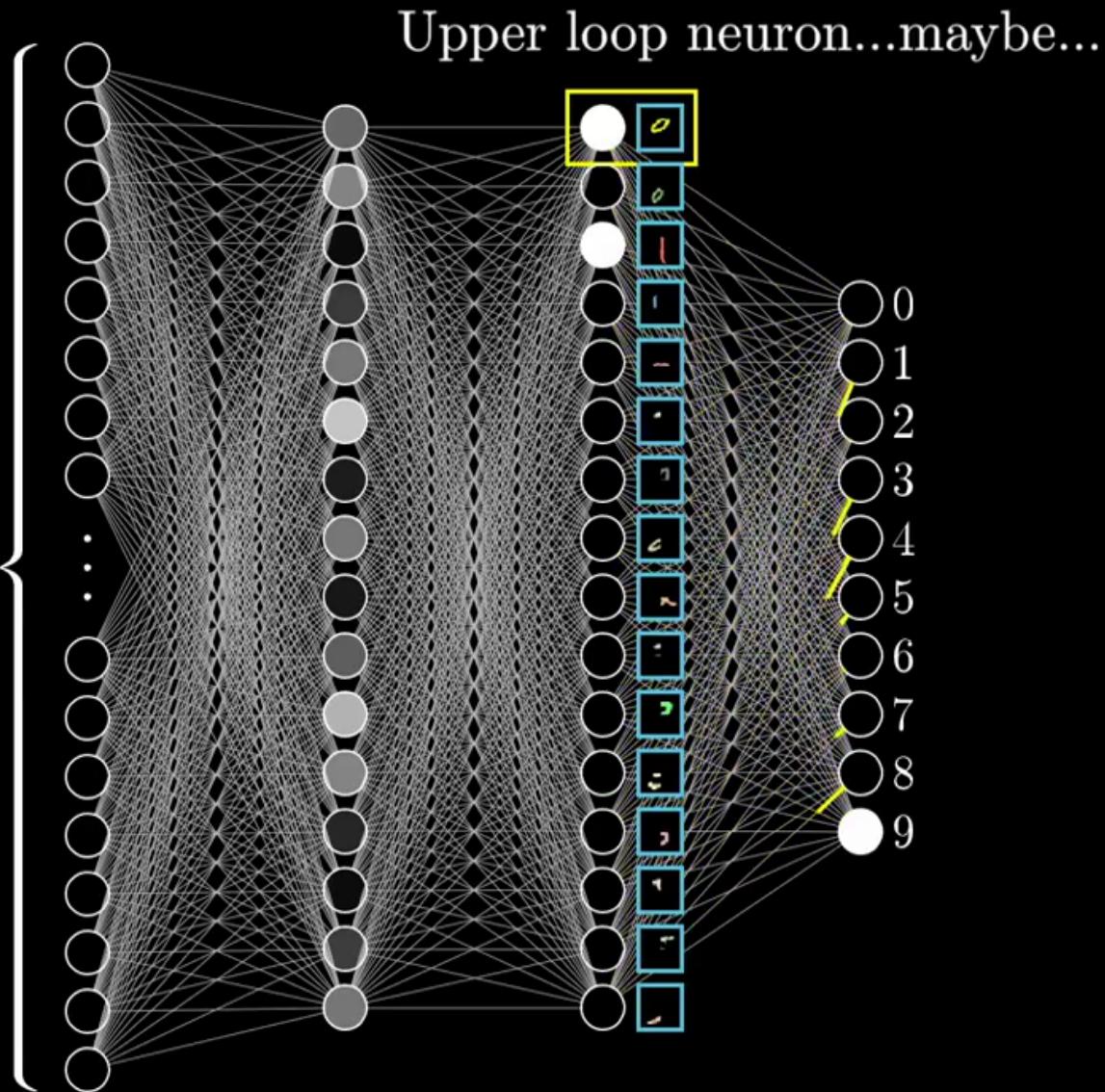
$$9 = \text{yellow circle} + \text{red vertical stroke}$$

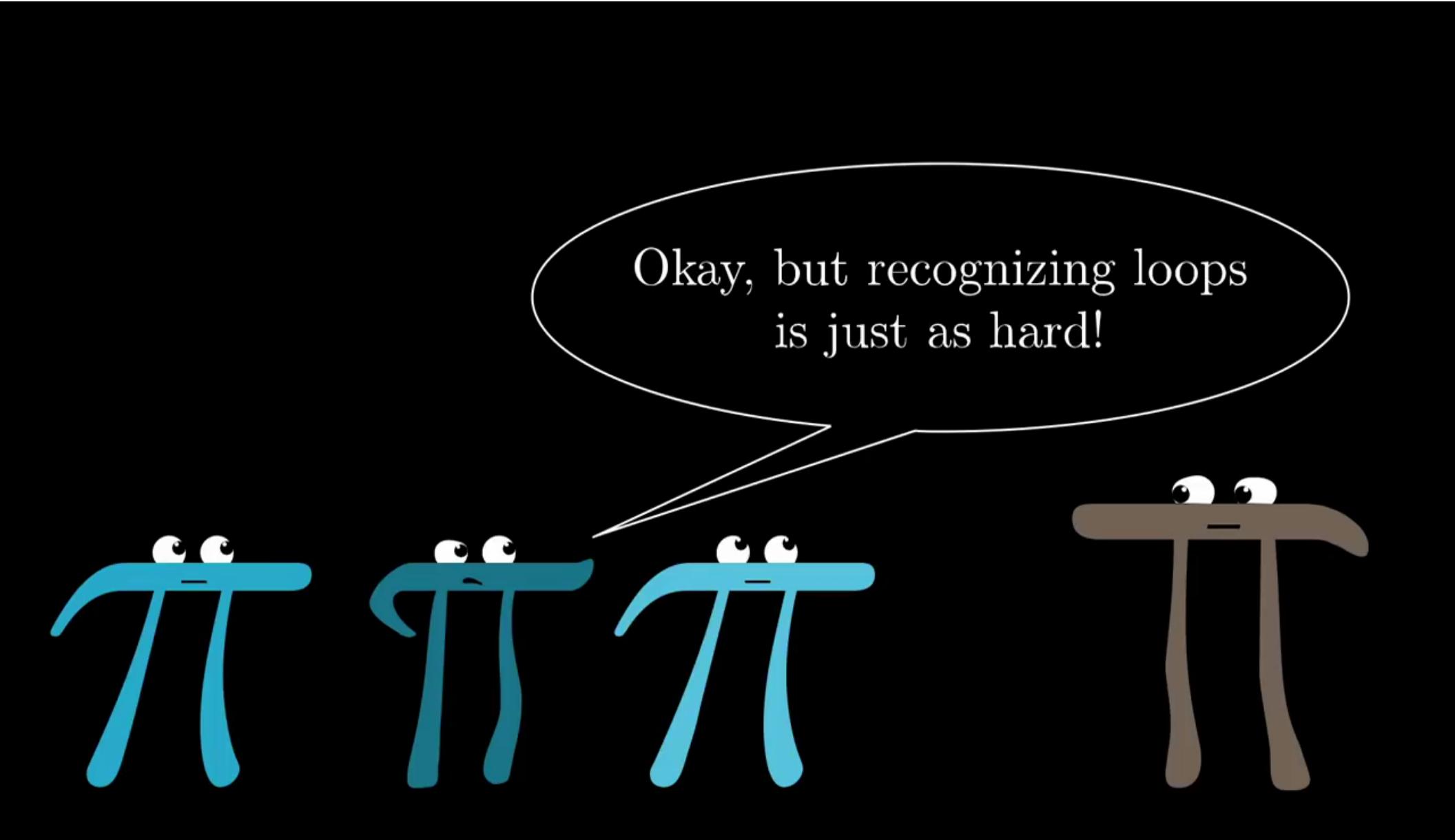
$$8 = \text{green circle} + \text{green circle}$$

$$4 = \text{dark red vertical stroke} + \text{blue vertical stroke} + \text{pink horizontal stroke}$$

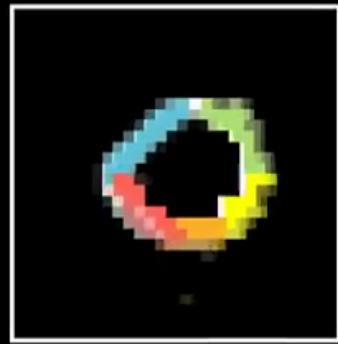
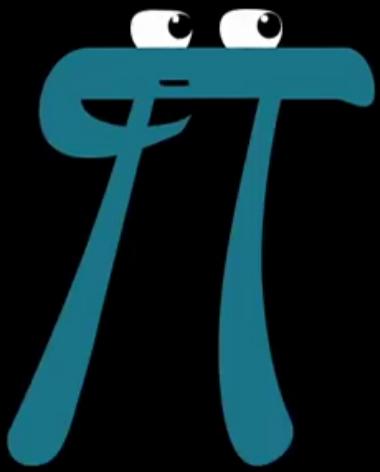


784

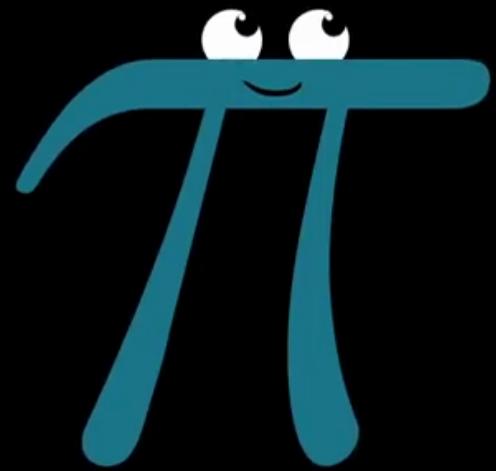




Okay, but recognizing loops
is just as hard!

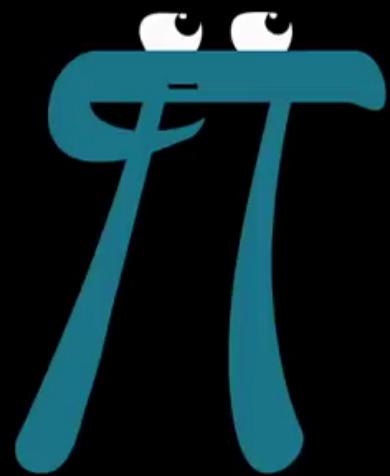


$$\text{○} = \text{✓} + \text{✗} + \text{✖} + \text{-} + \text{✗}$$

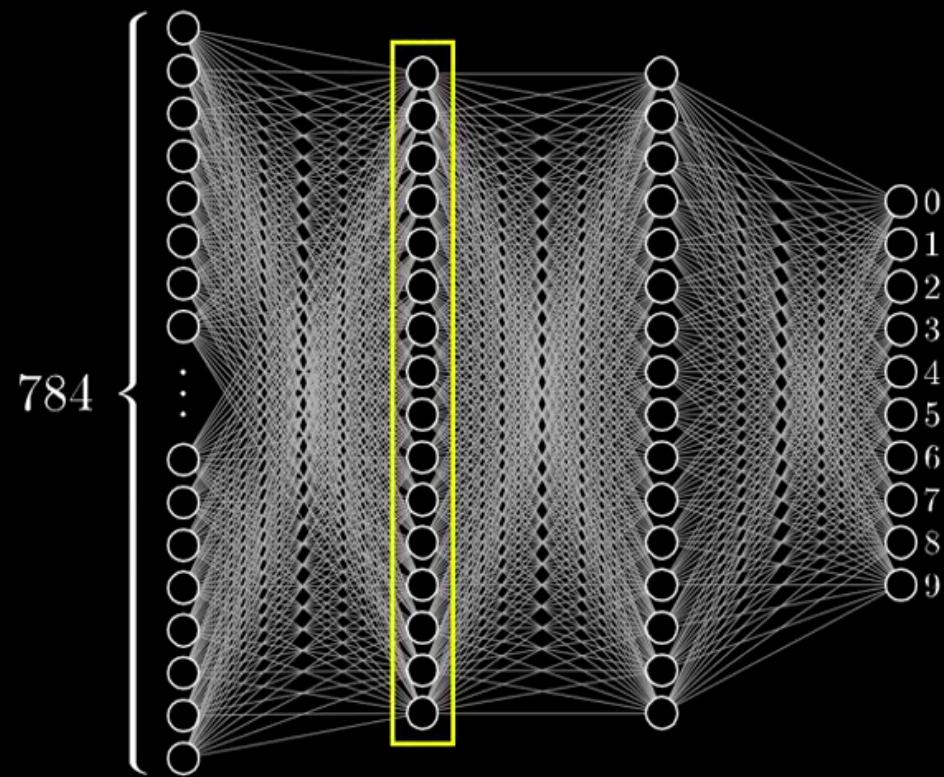


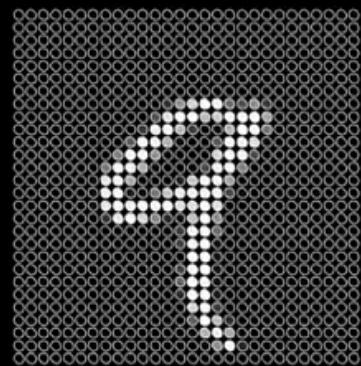
$$\text{○} = \text{○}_1 + \text{○}_2 + \text{○}_3 + \text{○}_4 + \text{○}_5$$

$$\text{I} = \text{I}_1 + \text{I}_2 + \text{I}_3$$

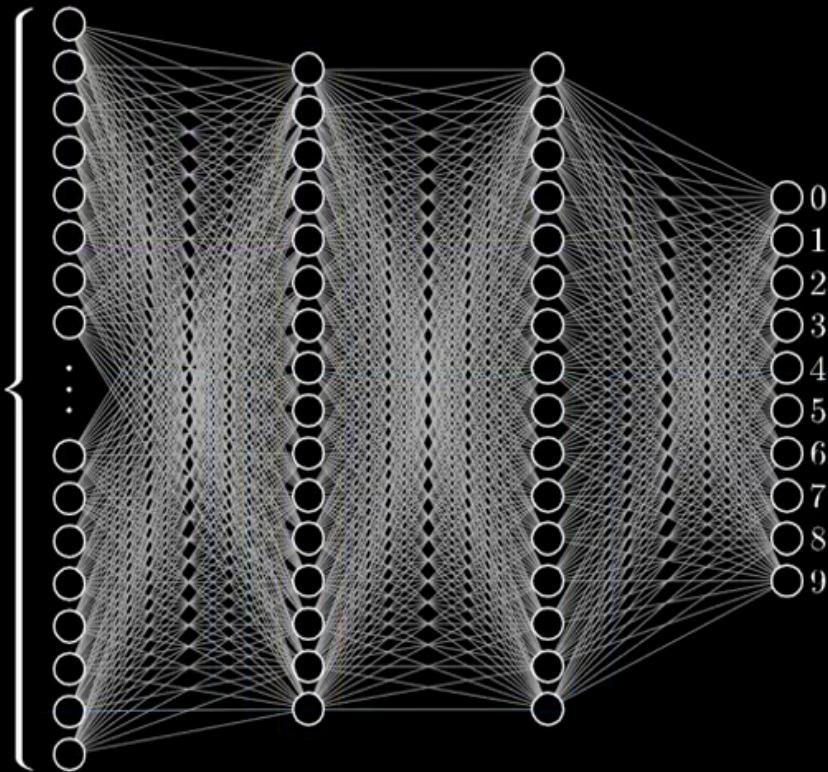


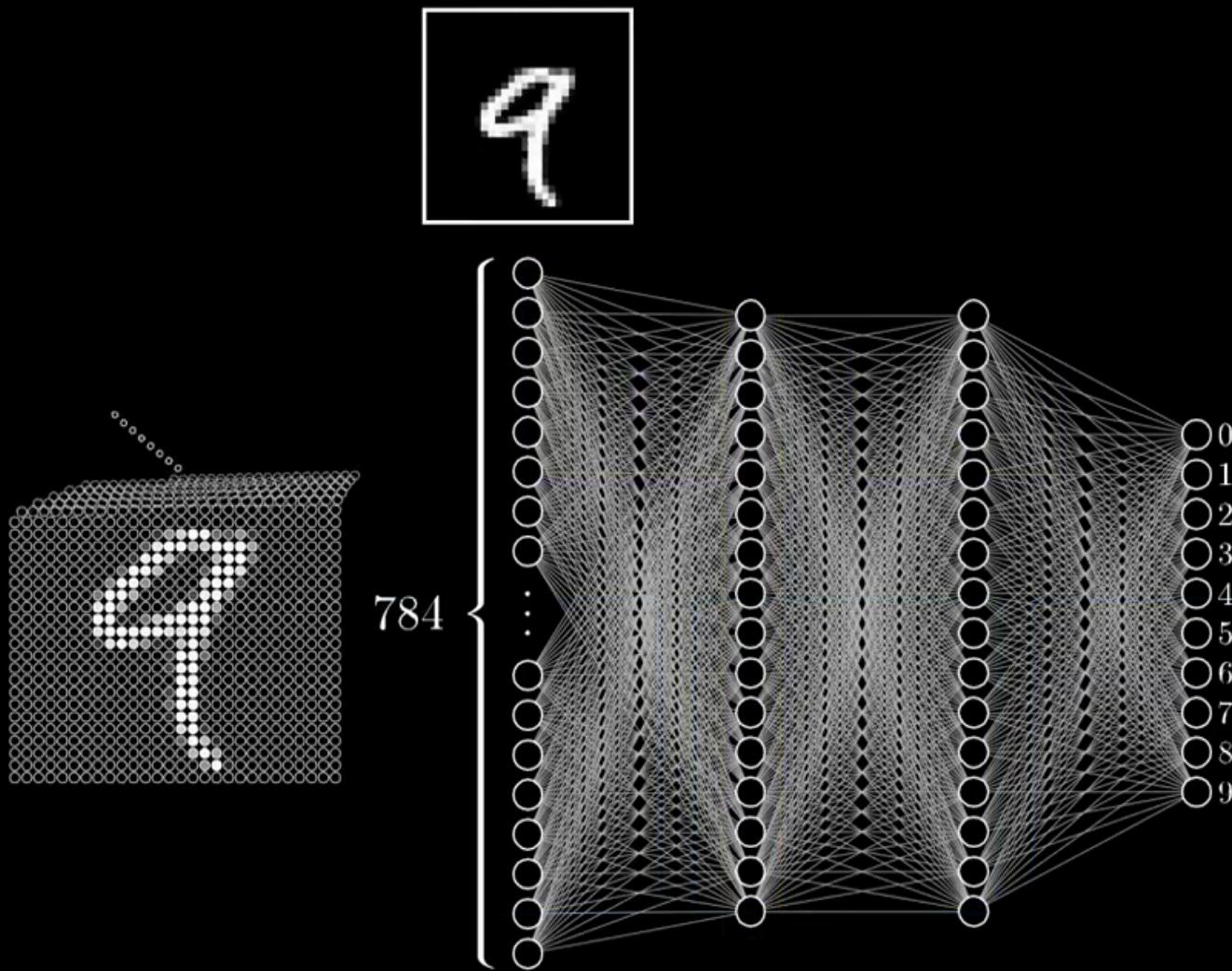
“Little edge” layer?

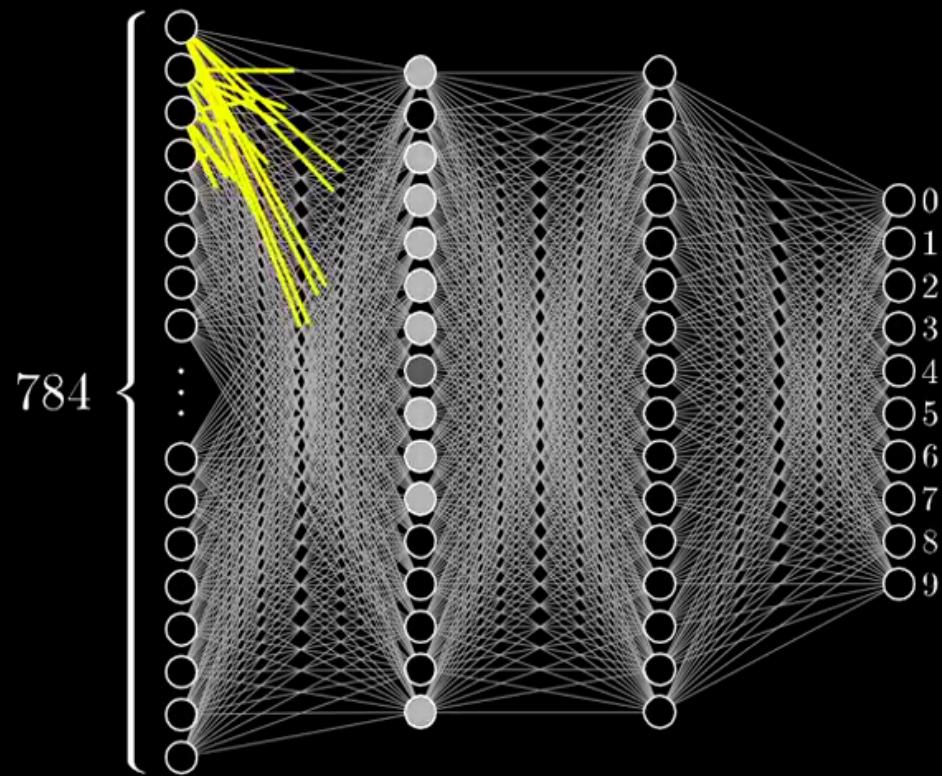


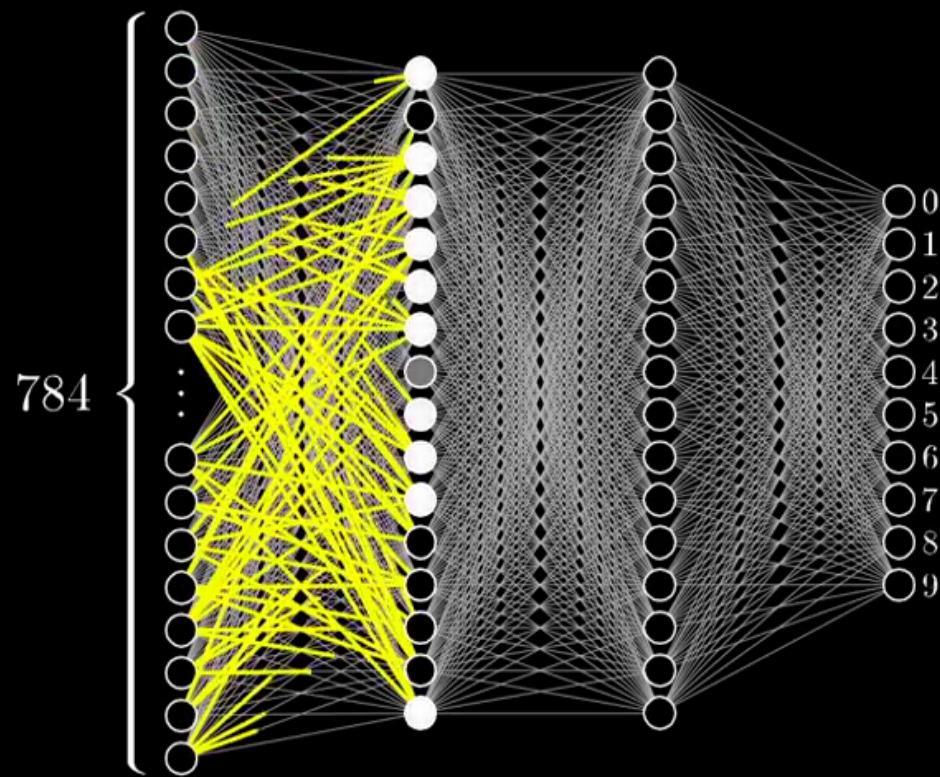


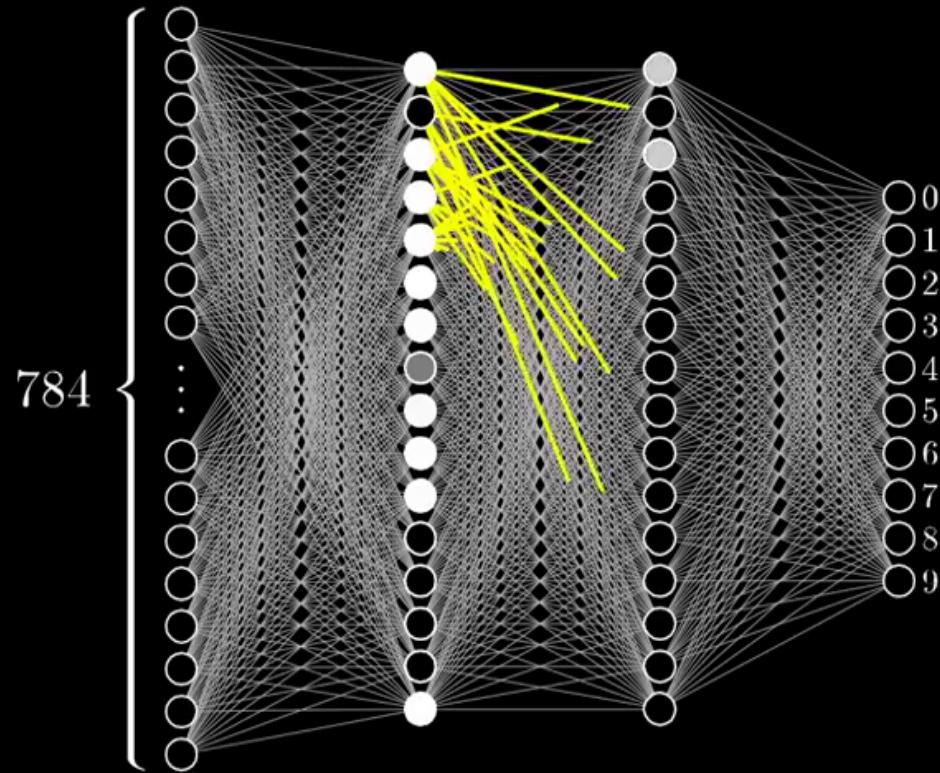
784

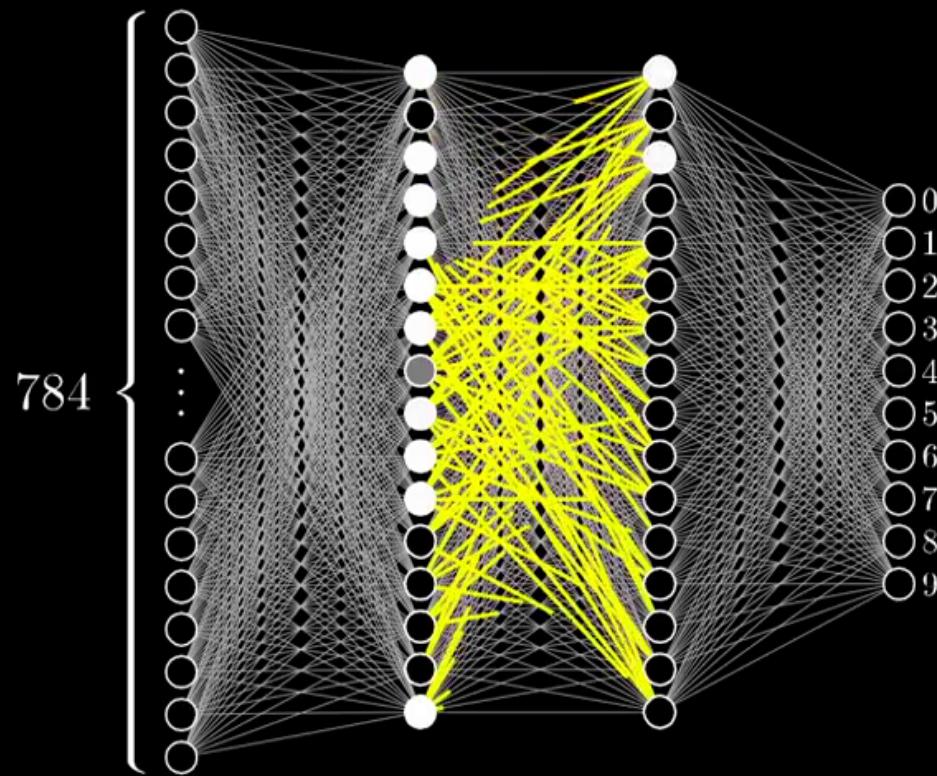








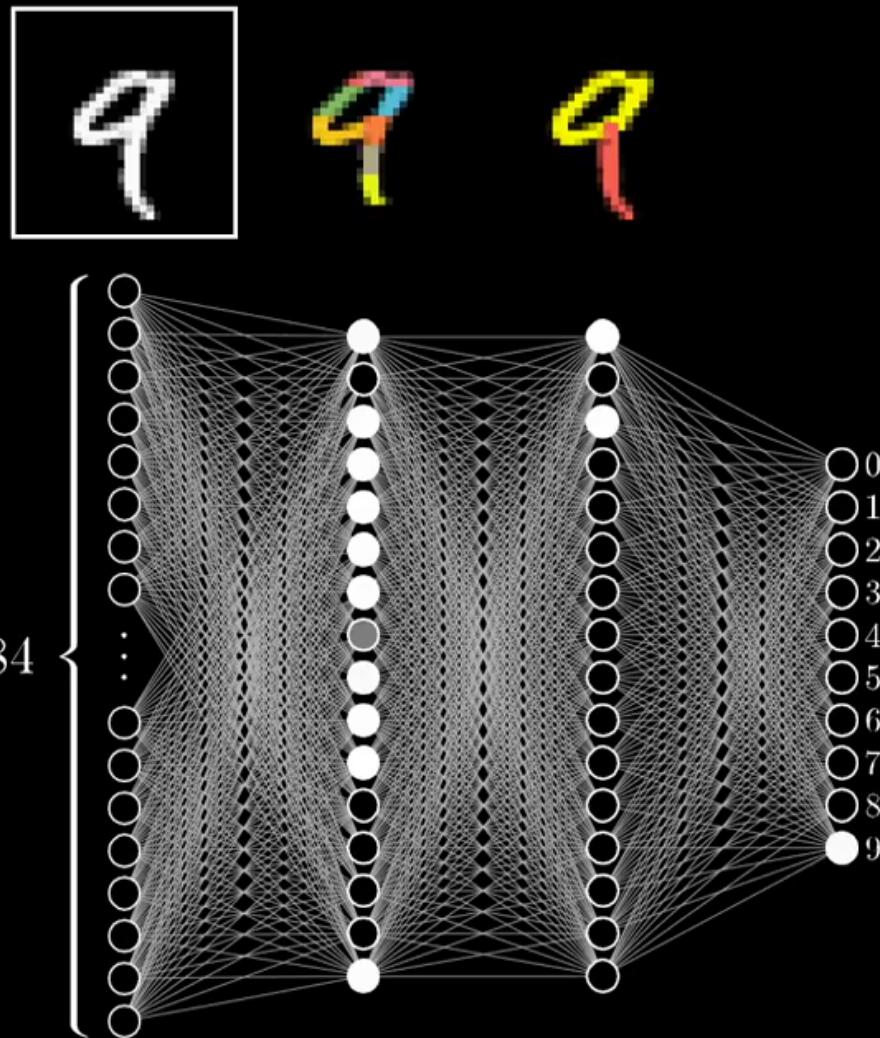




We'll get back
to this



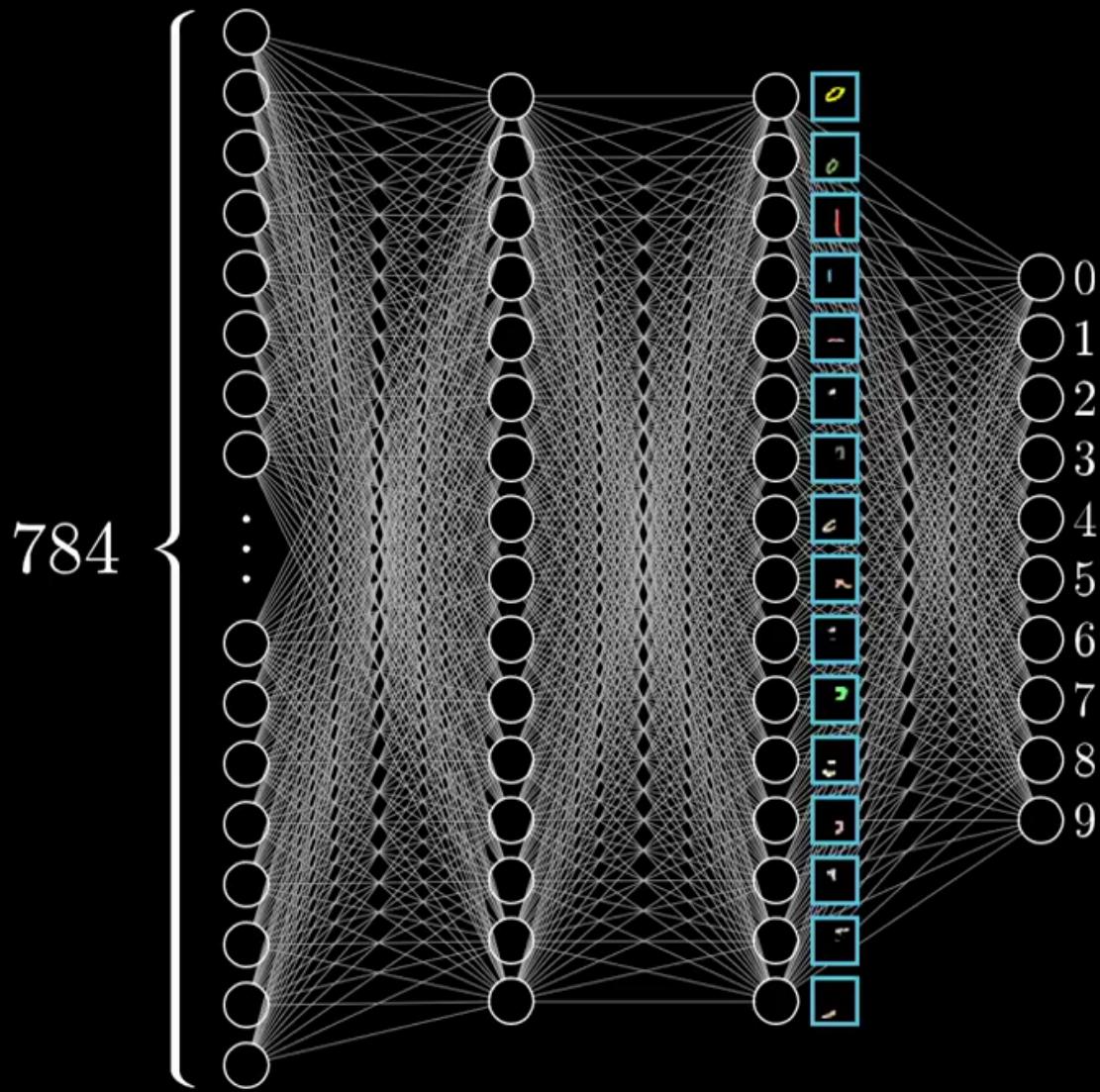
Does the network
actually do this?



$$9 = \text{yellow circle} + \text{red vertical stroke}$$

$$8 = \text{yellow circle} + \text{green circle}$$

$$4 = \text{red vertical stroke} + \text{blue vertical stroke} + \text{pink horizontal stroke}$$





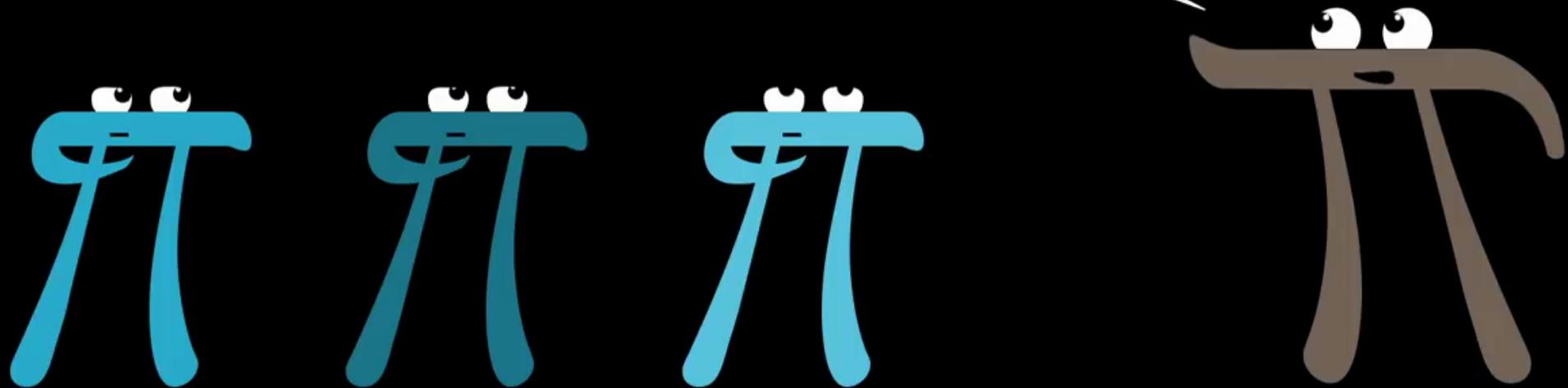




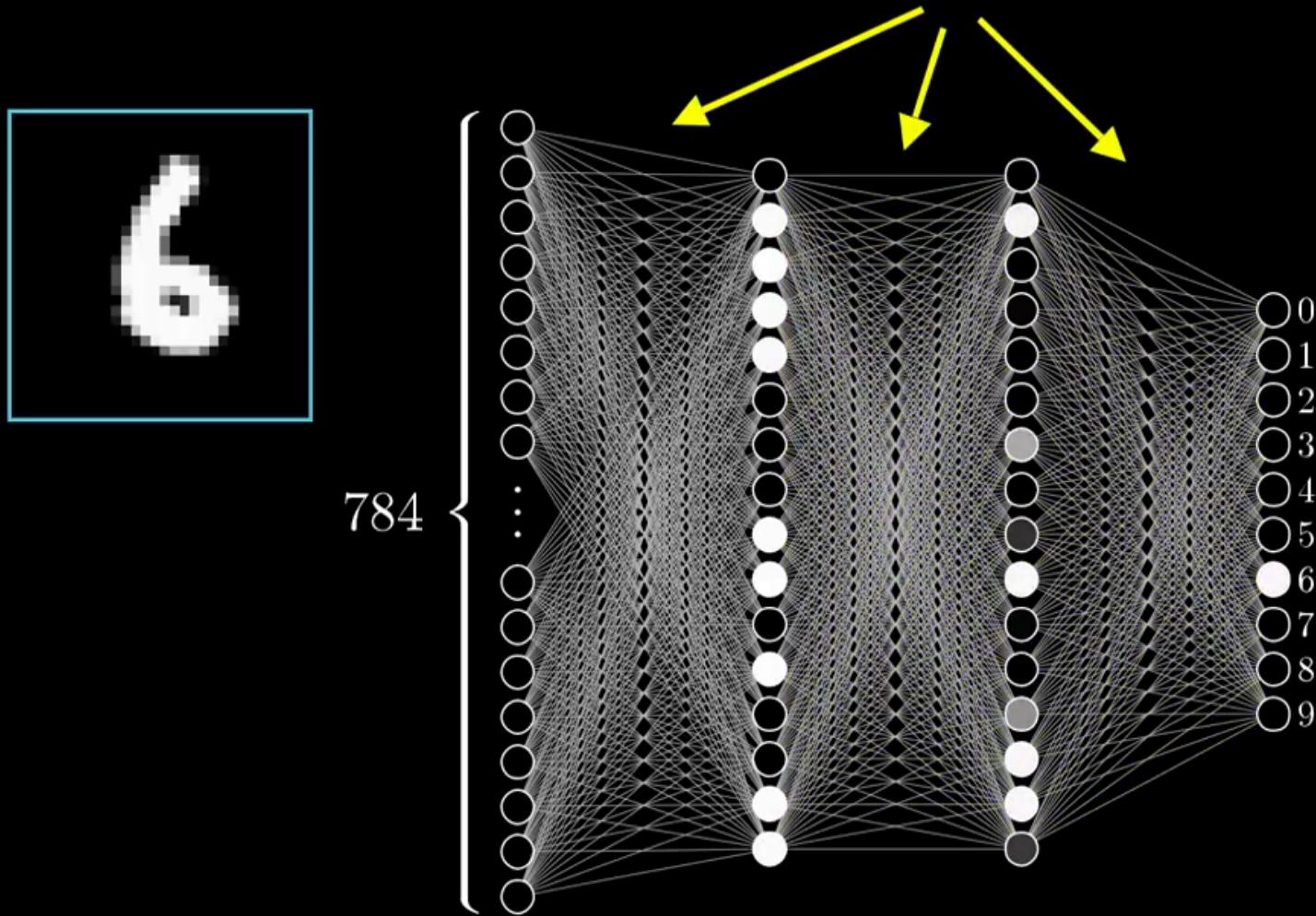
Raw audio



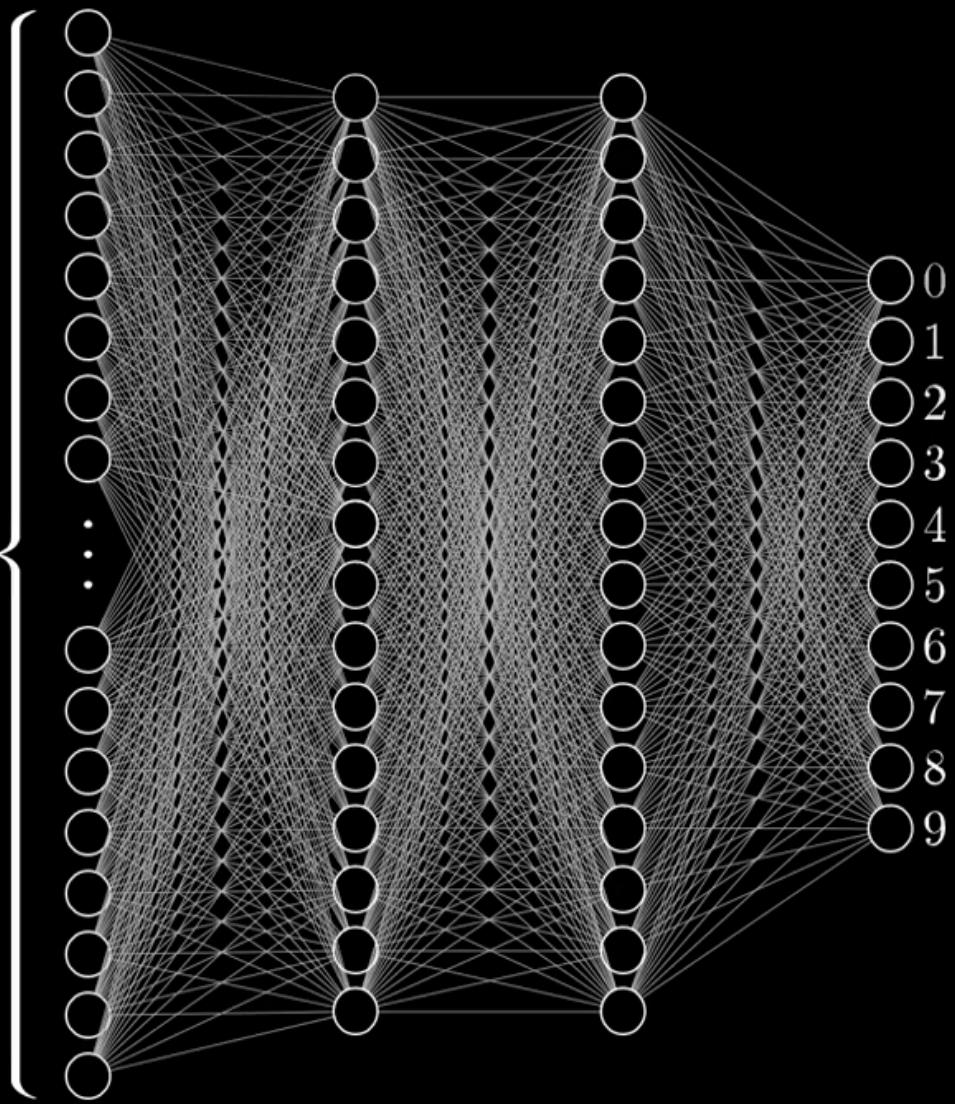
recognition



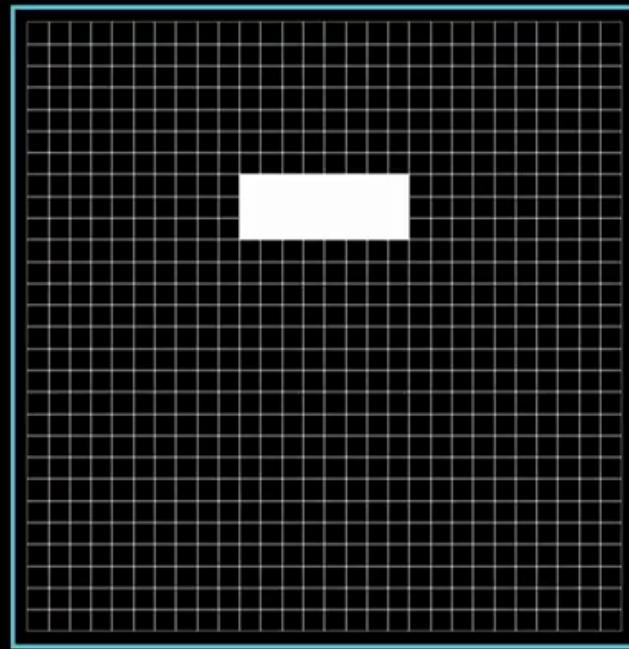
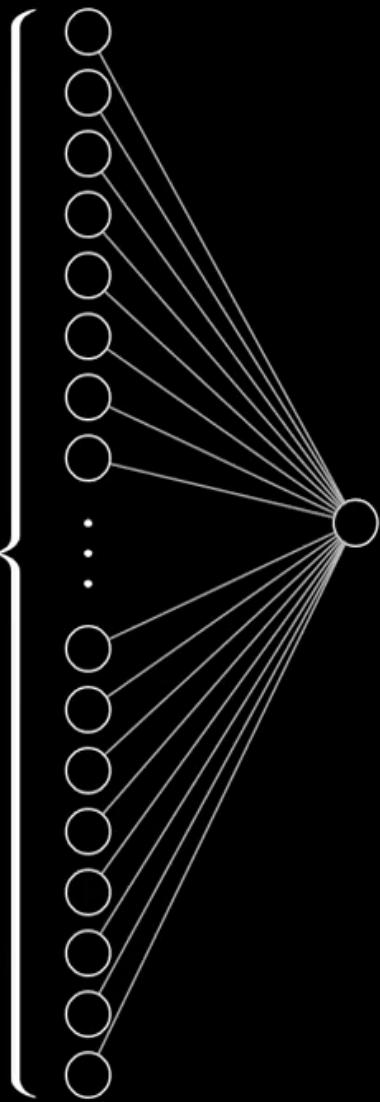
What are these connections actually doing?



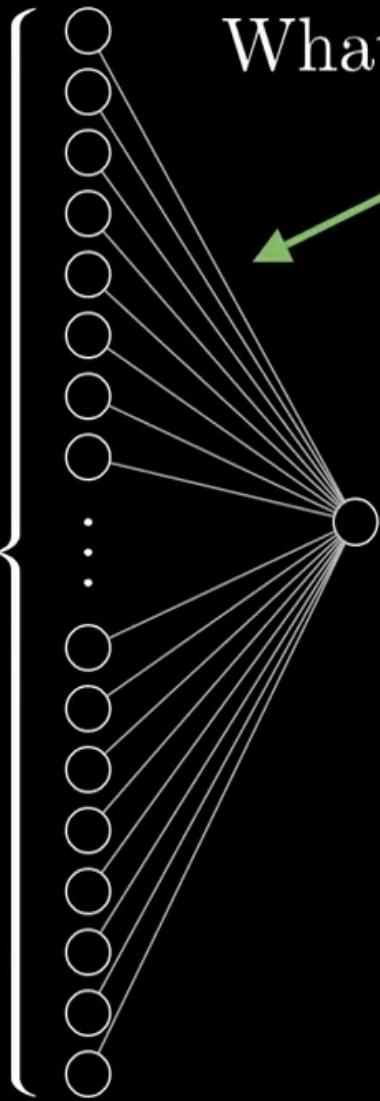
784



784



784



What parameters should exist?

$$p_1: 0.00$$

$$p_2: 0.00$$

$$p_3: 0.00$$

$$p_4: 0.00$$

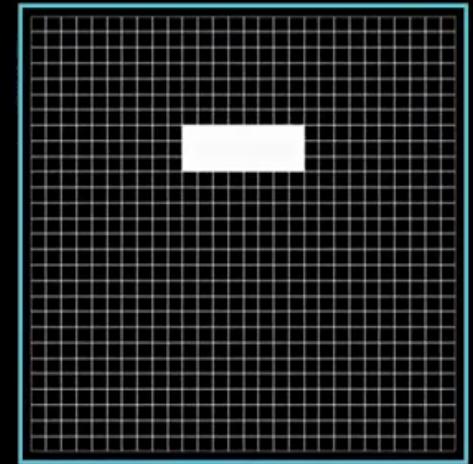
$$p_5: 0.00$$

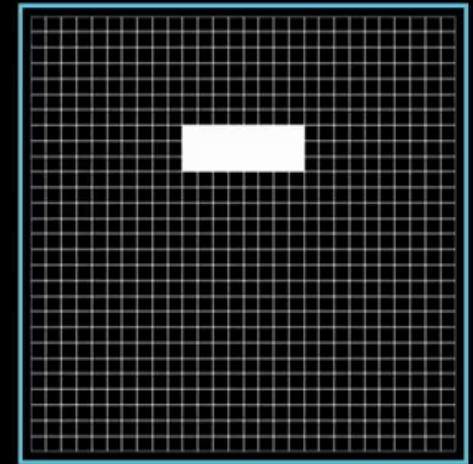
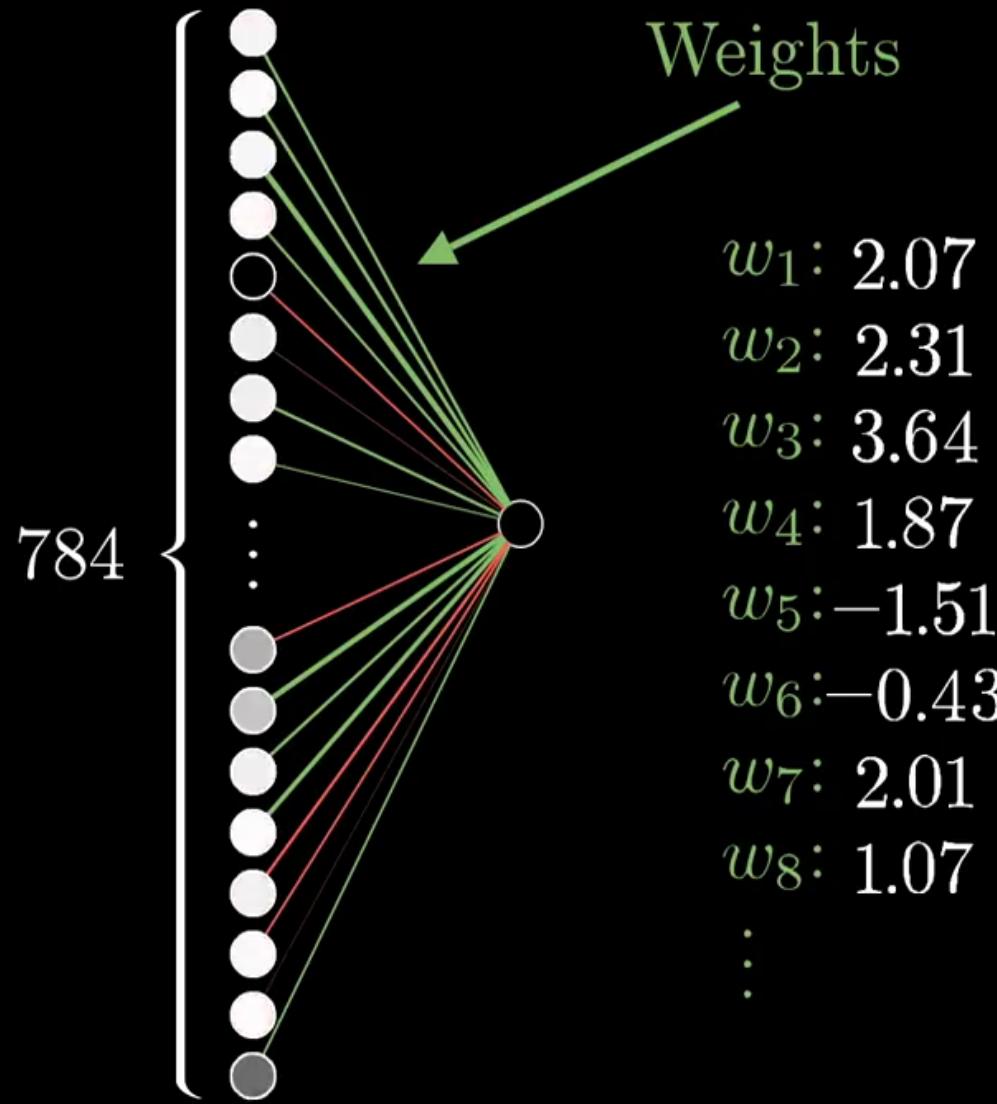
$$p_6: 0.00$$

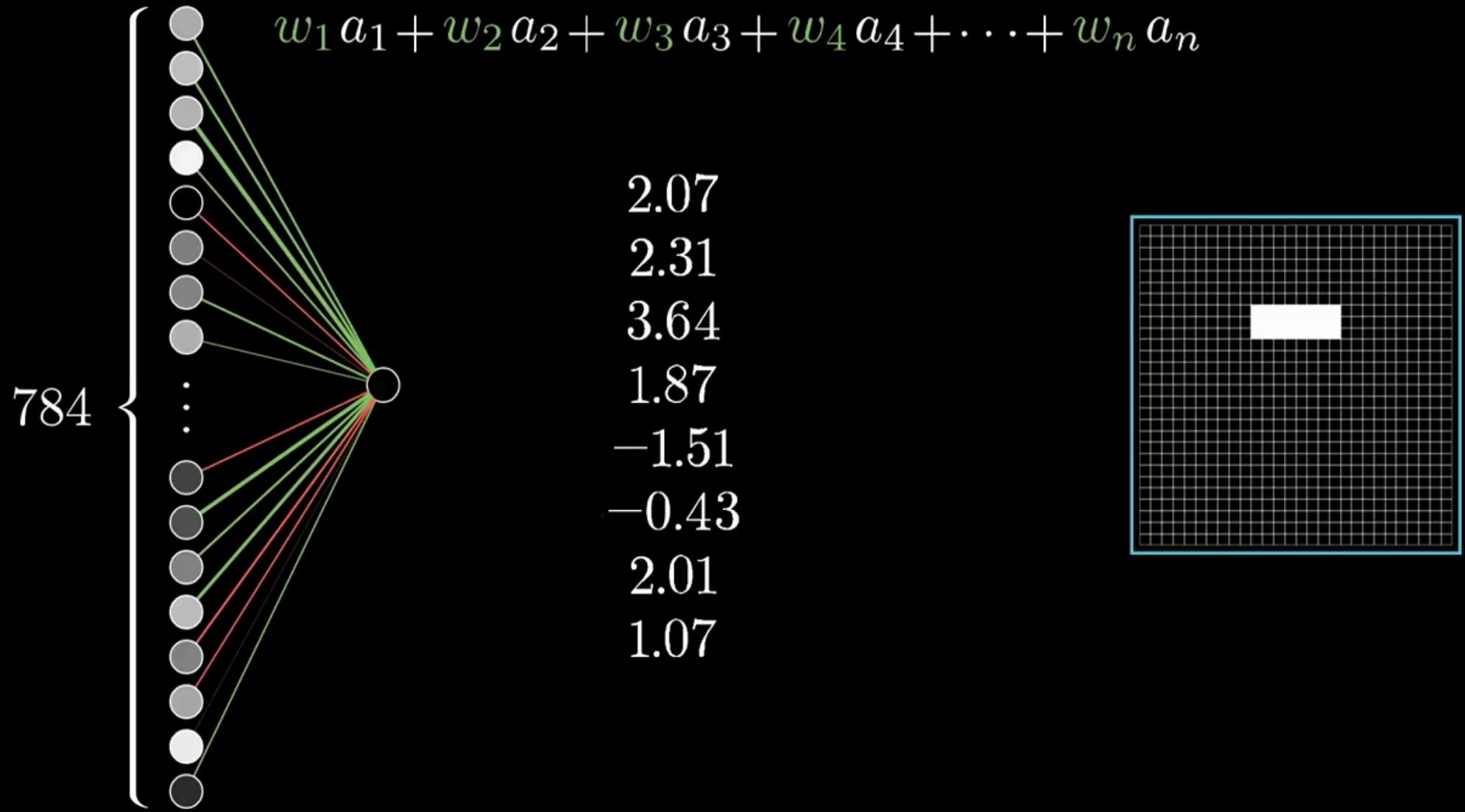
$$p_7: 0.00$$

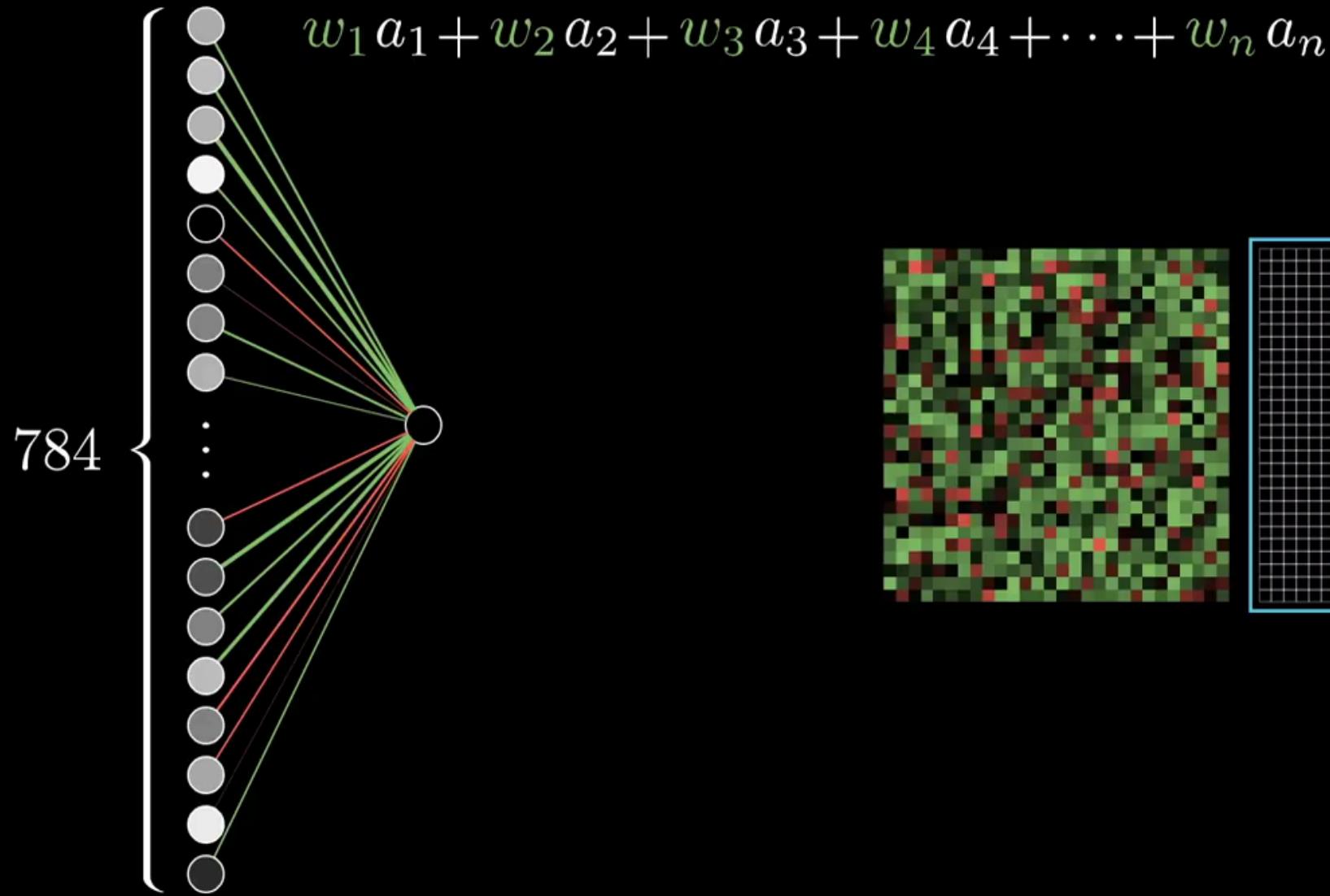
$$p_8: 0.00$$

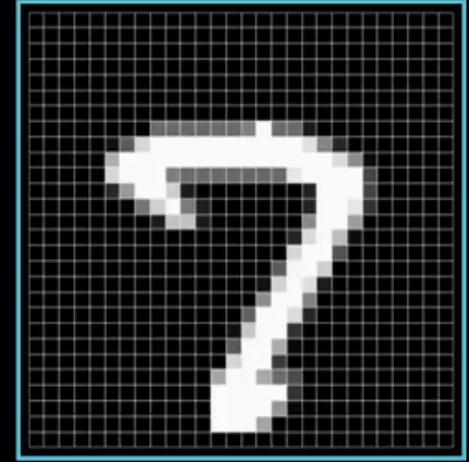
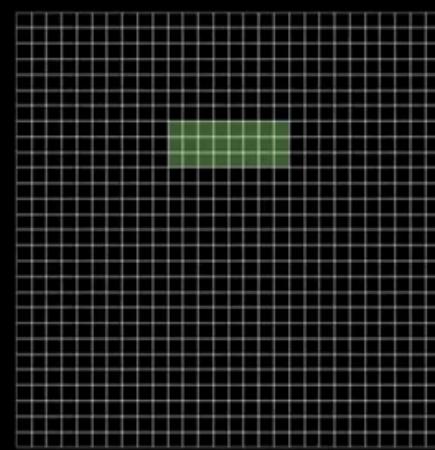
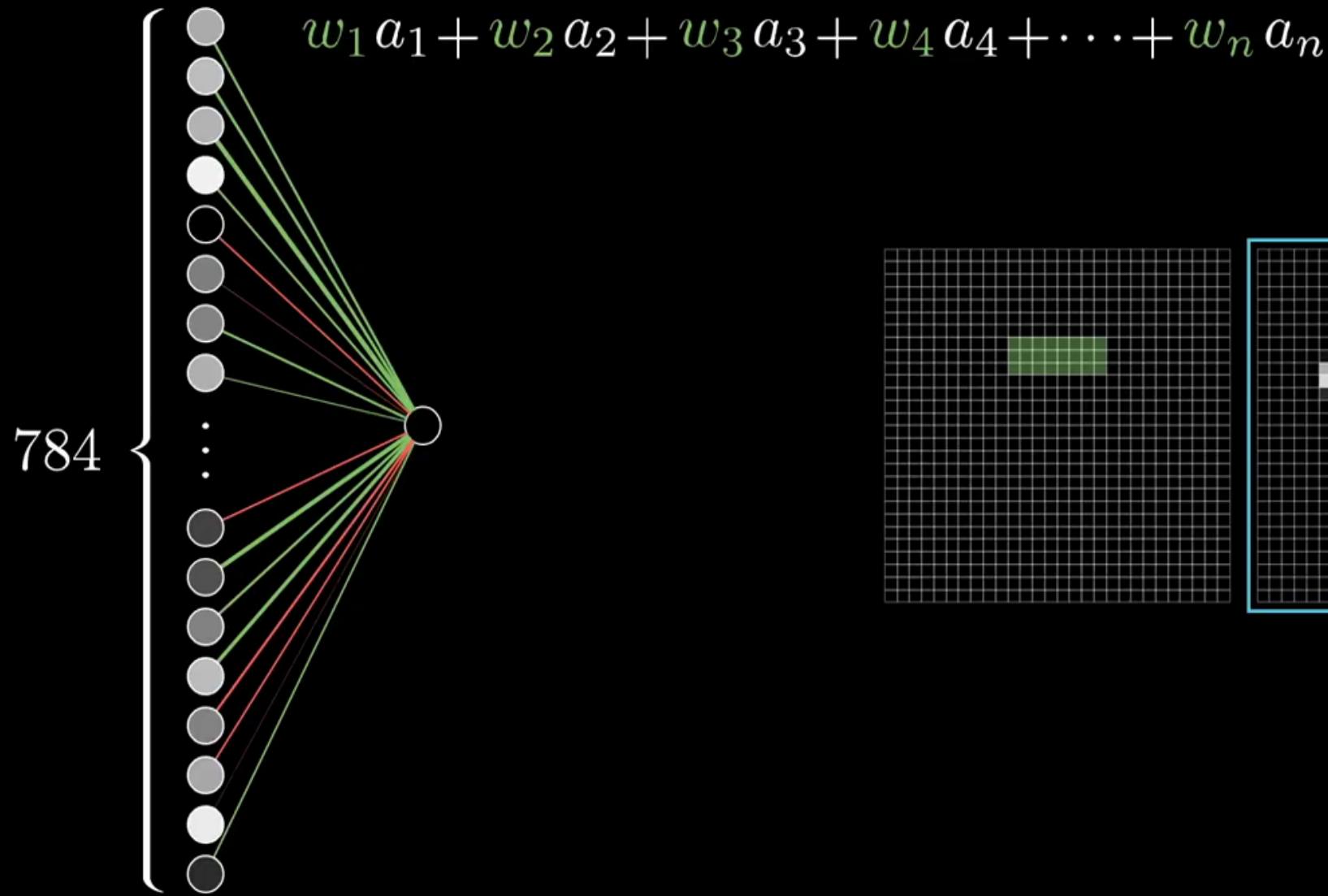
⋮

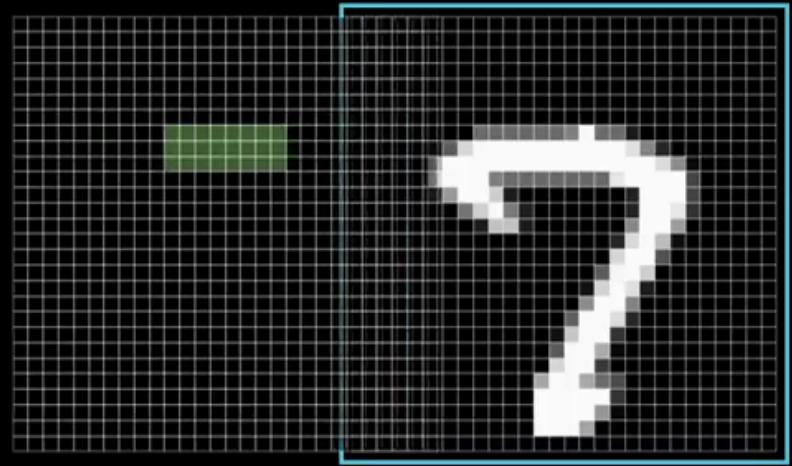
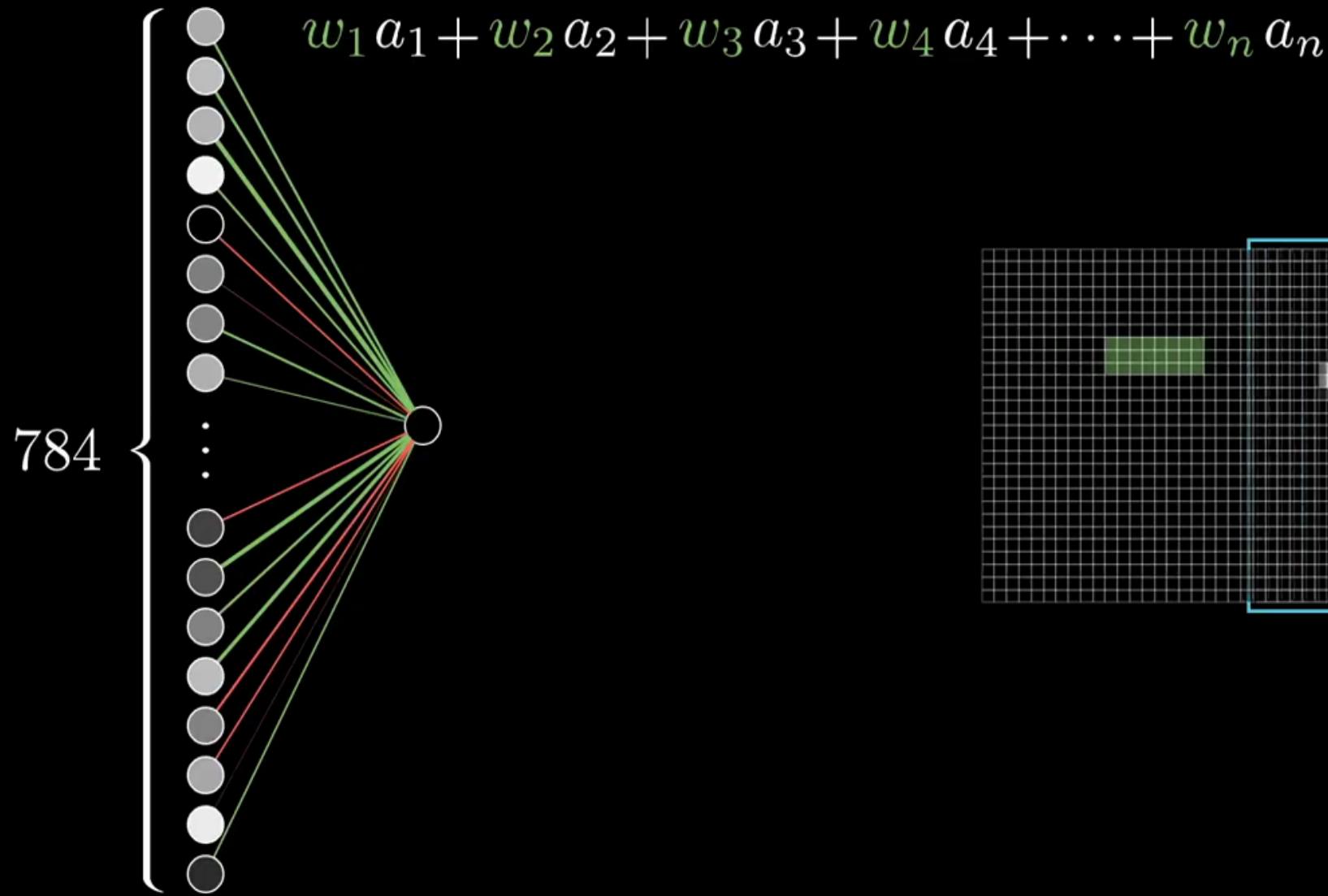


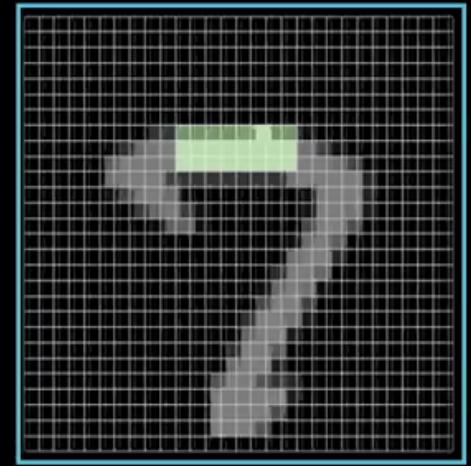
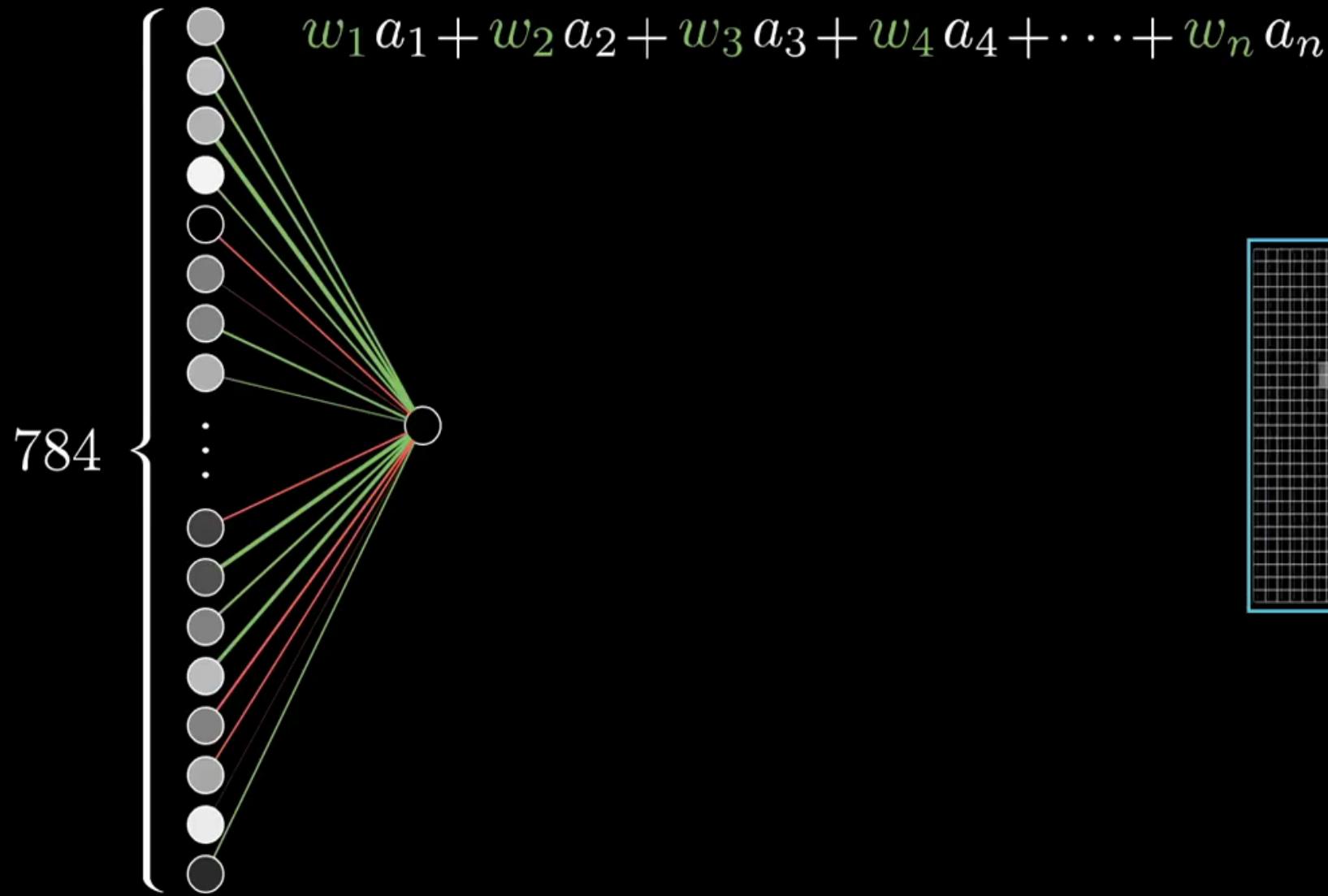


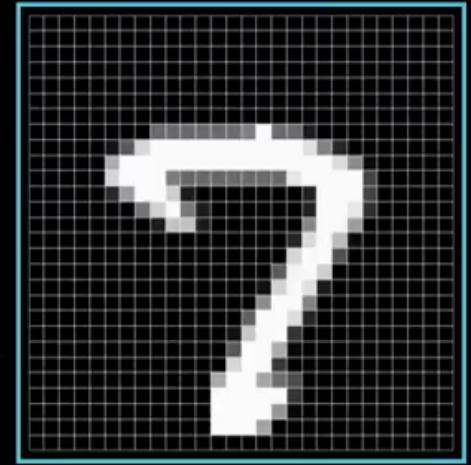
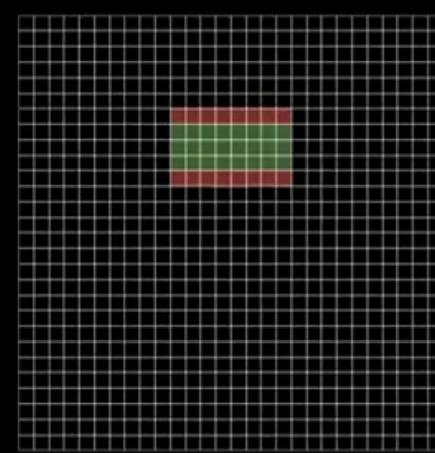
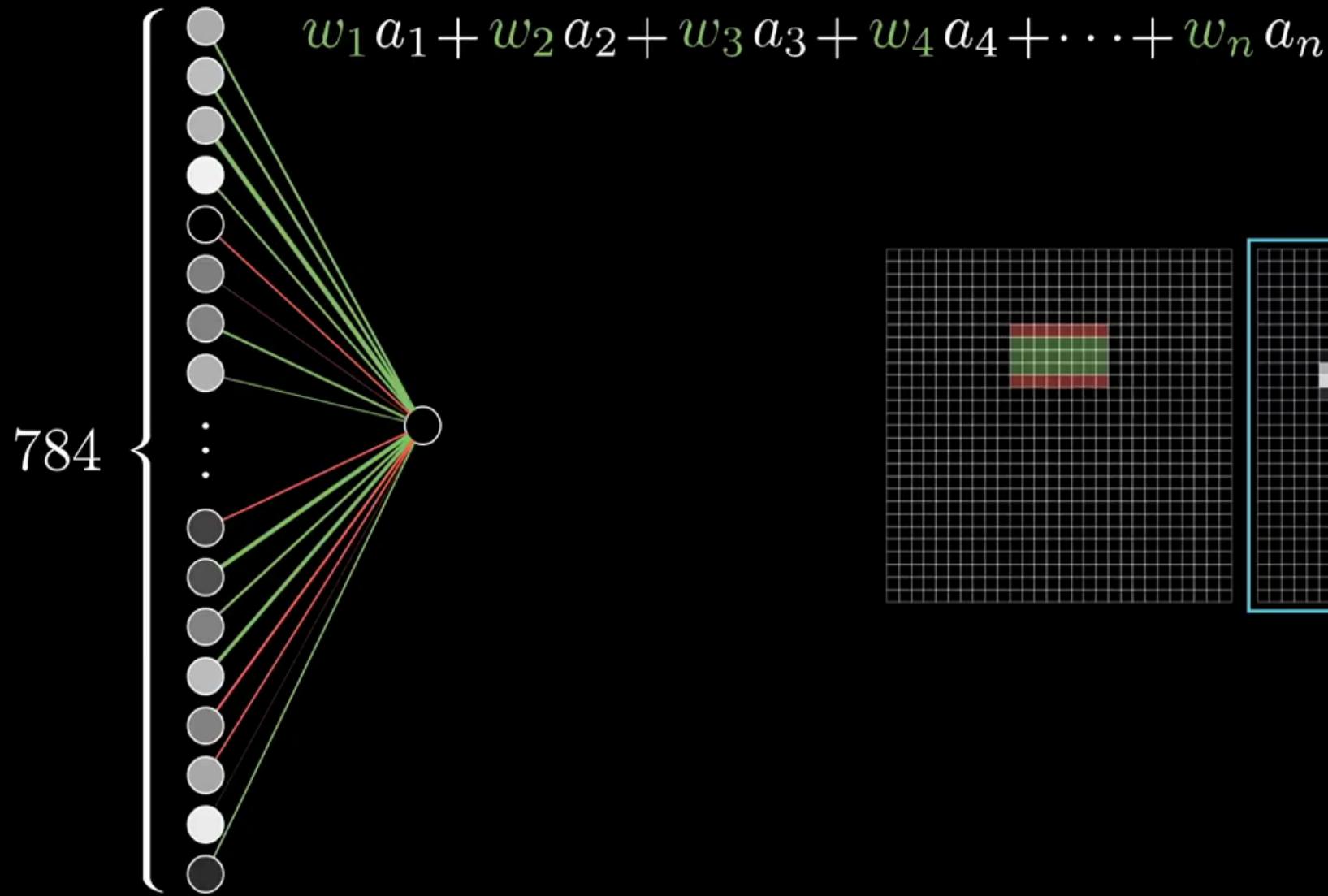


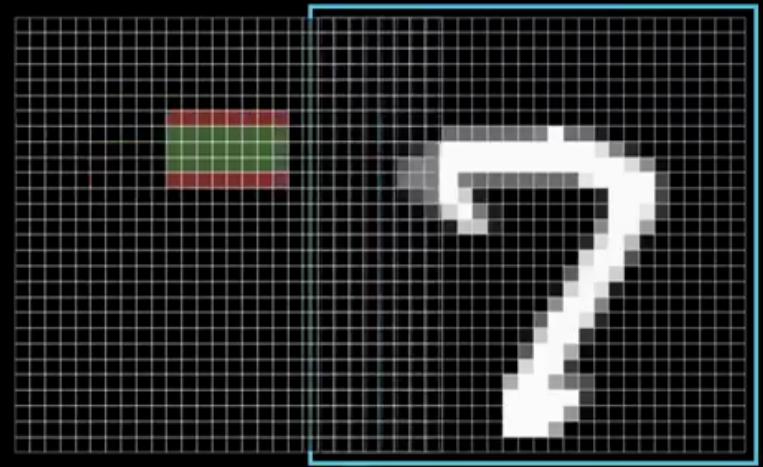
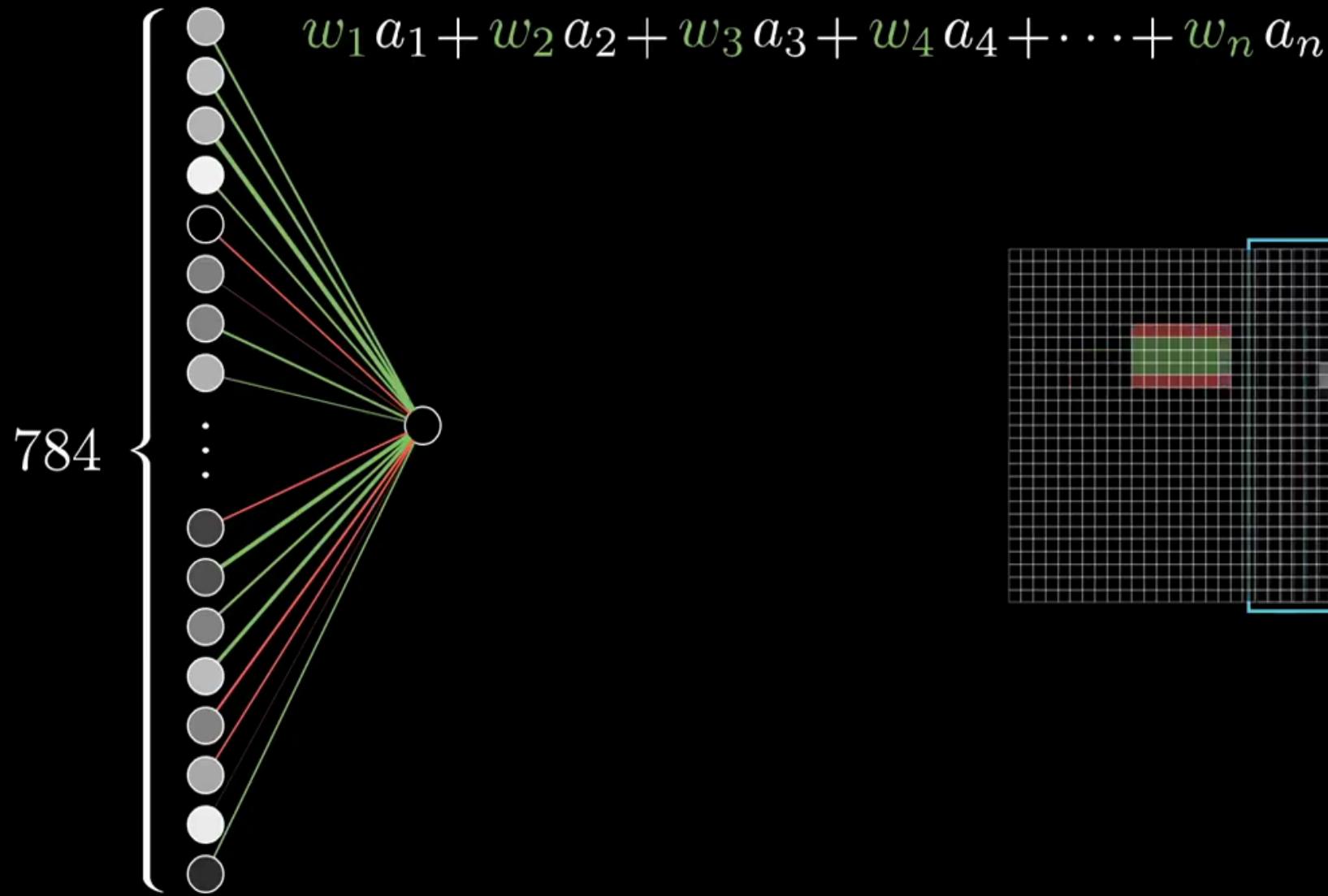


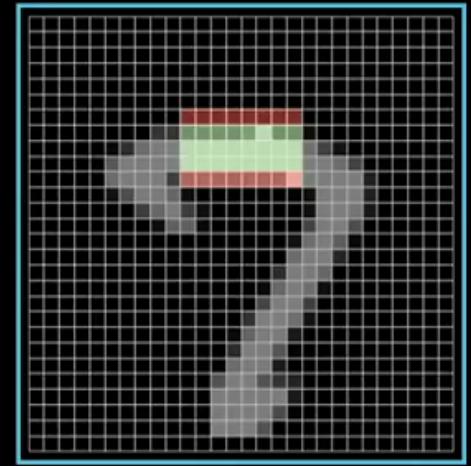
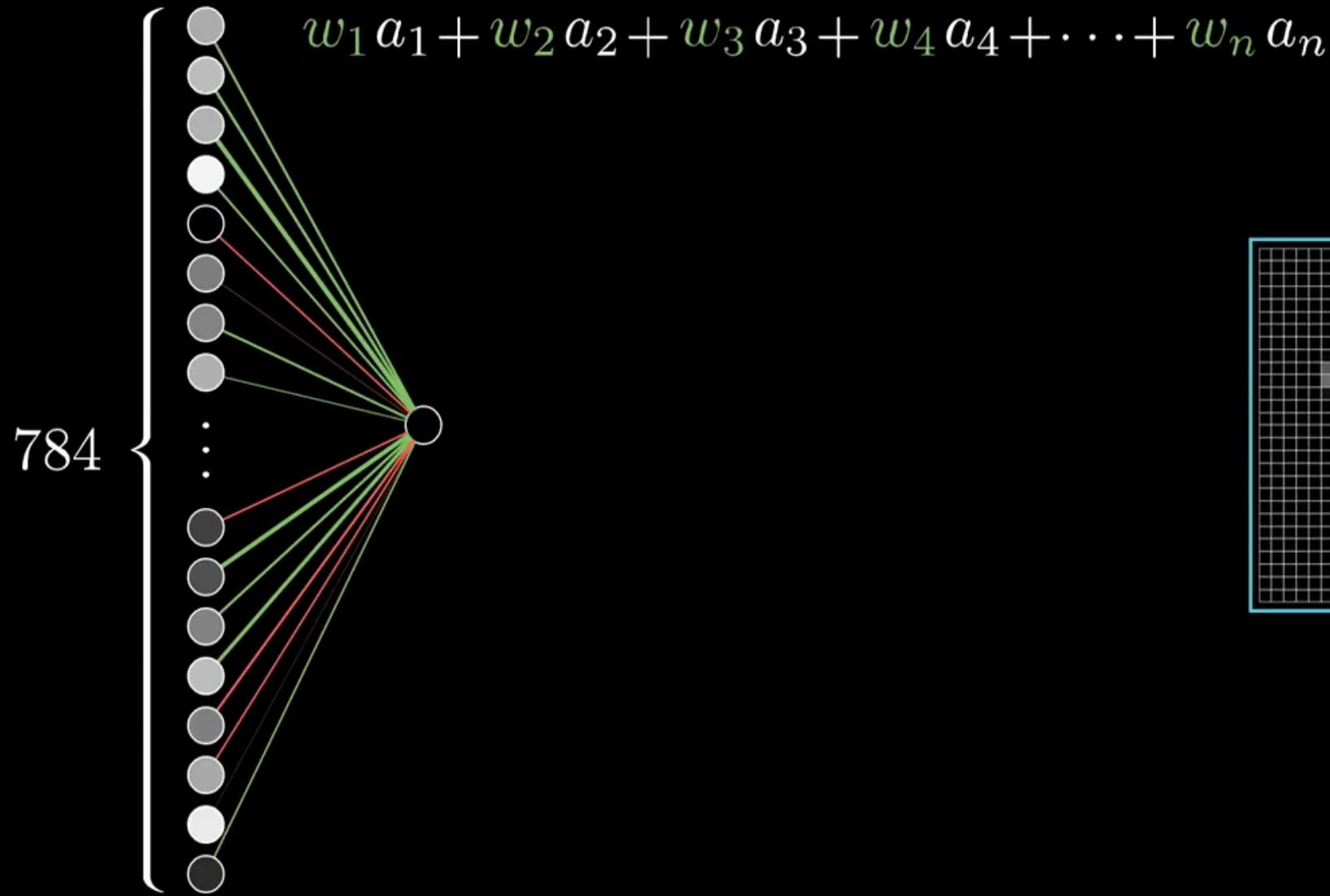




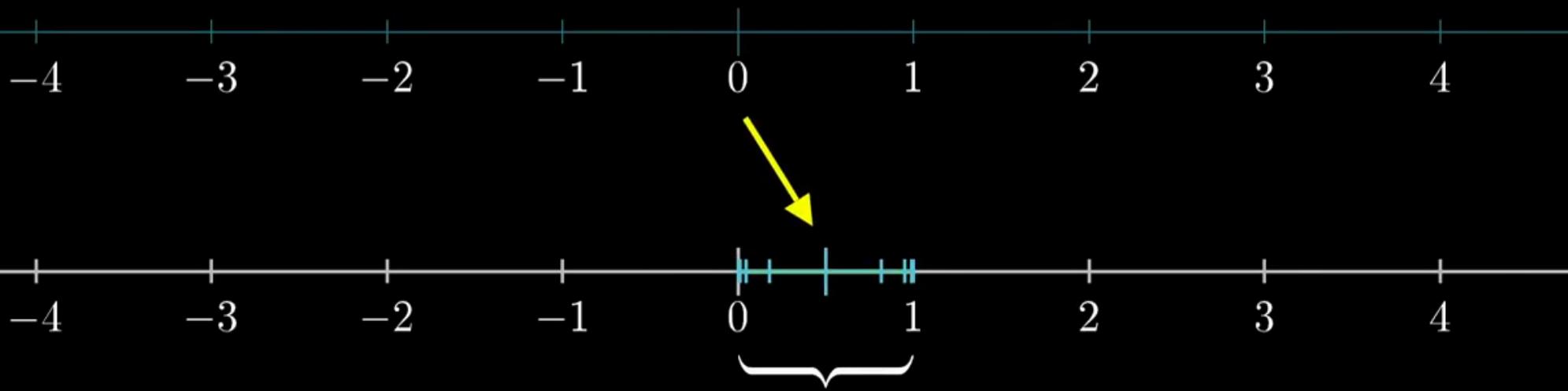








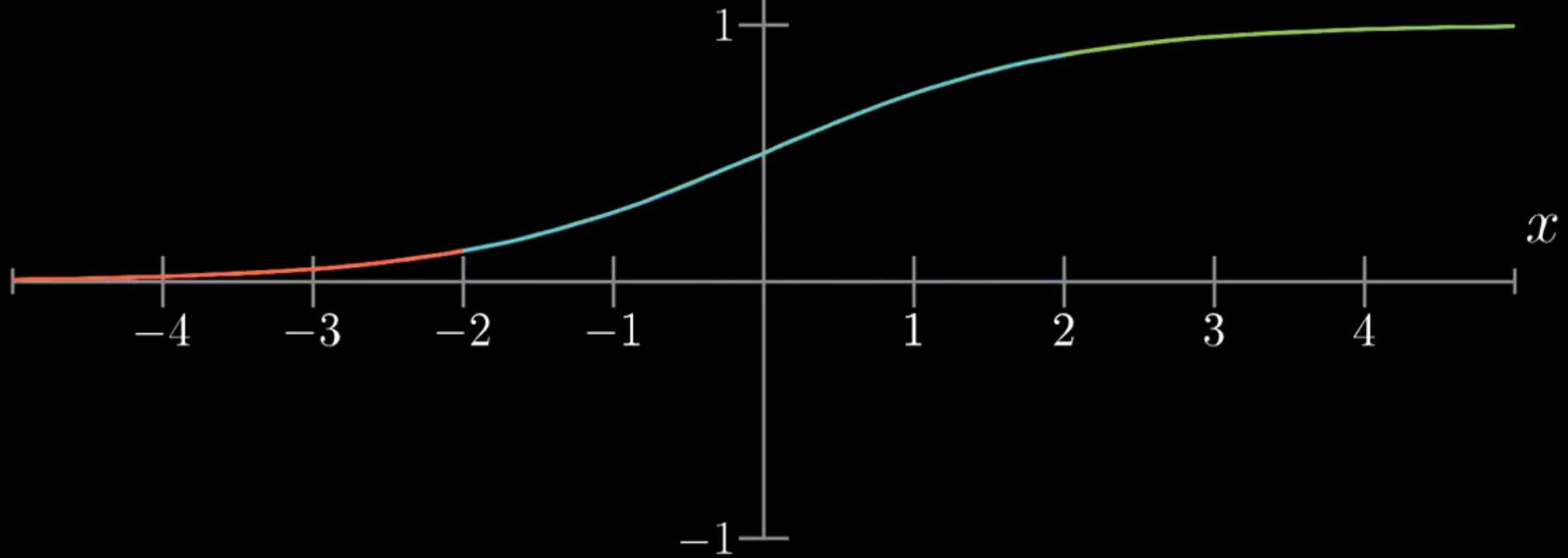
$$w_1a_1 + w_2a_2 + w_3a_3 + w_4a_4 + \cdots + w_na_n$$



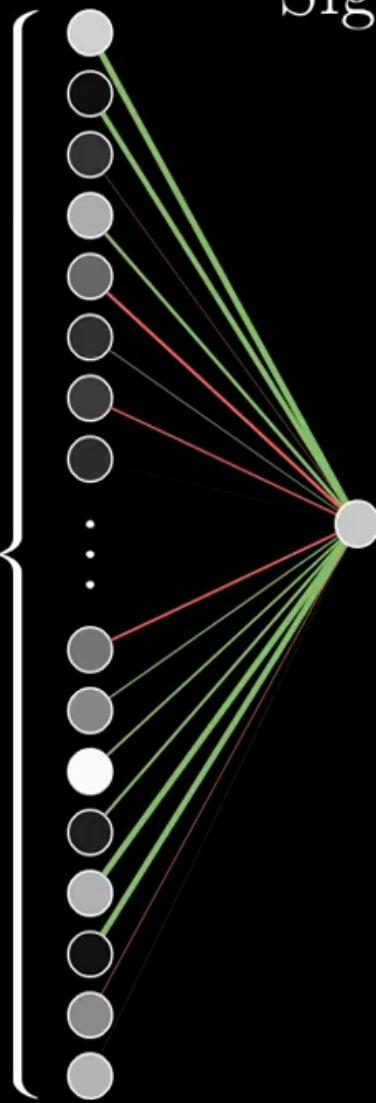
Activations should be in this range

Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



784



Sigmoid

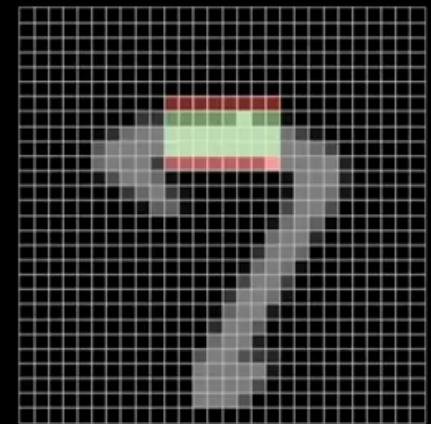


How positive is this?

$$\sigma(w_1a_1 + w_2a_2 + \textcolor{green}{w_3}a_3 + \cdots + w_na_n \boxed{-10})$$

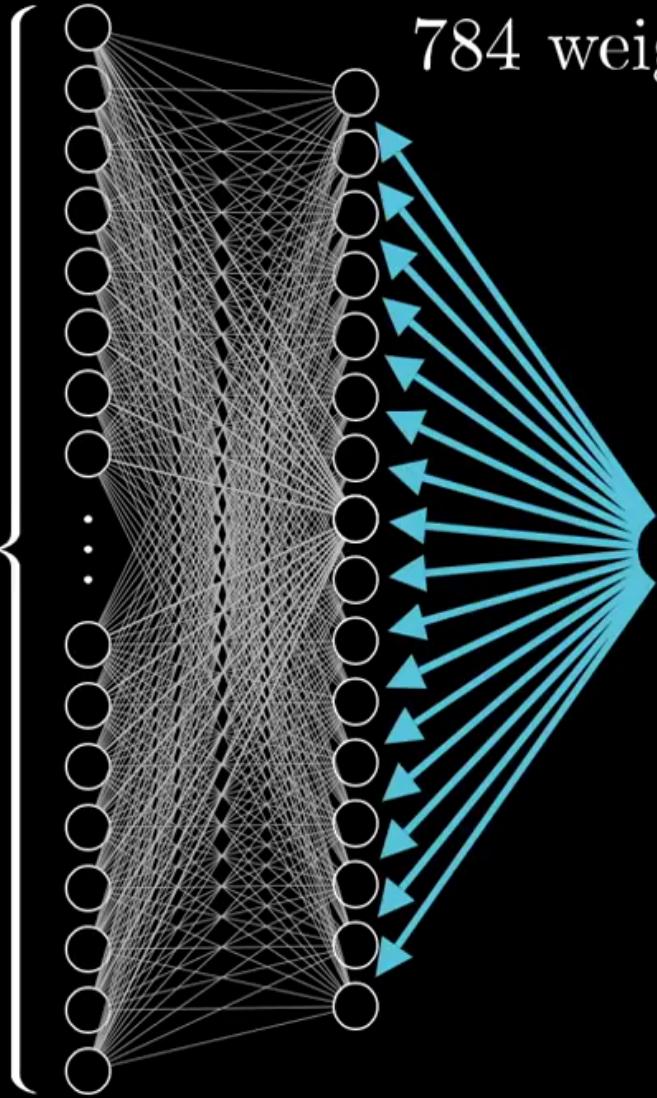
“bias”

Only activate meaningfully
when weighted sum > 10



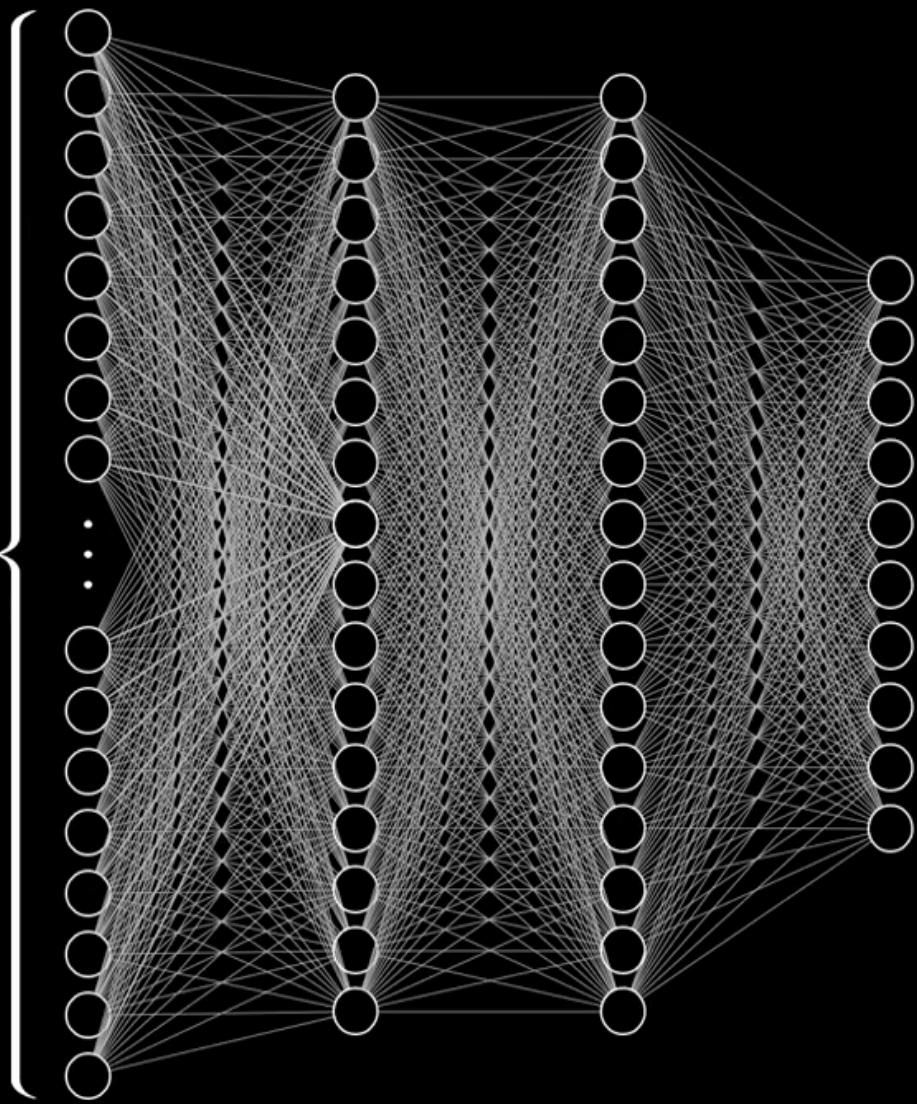
784 weights per neuron

784



One bias for each

784

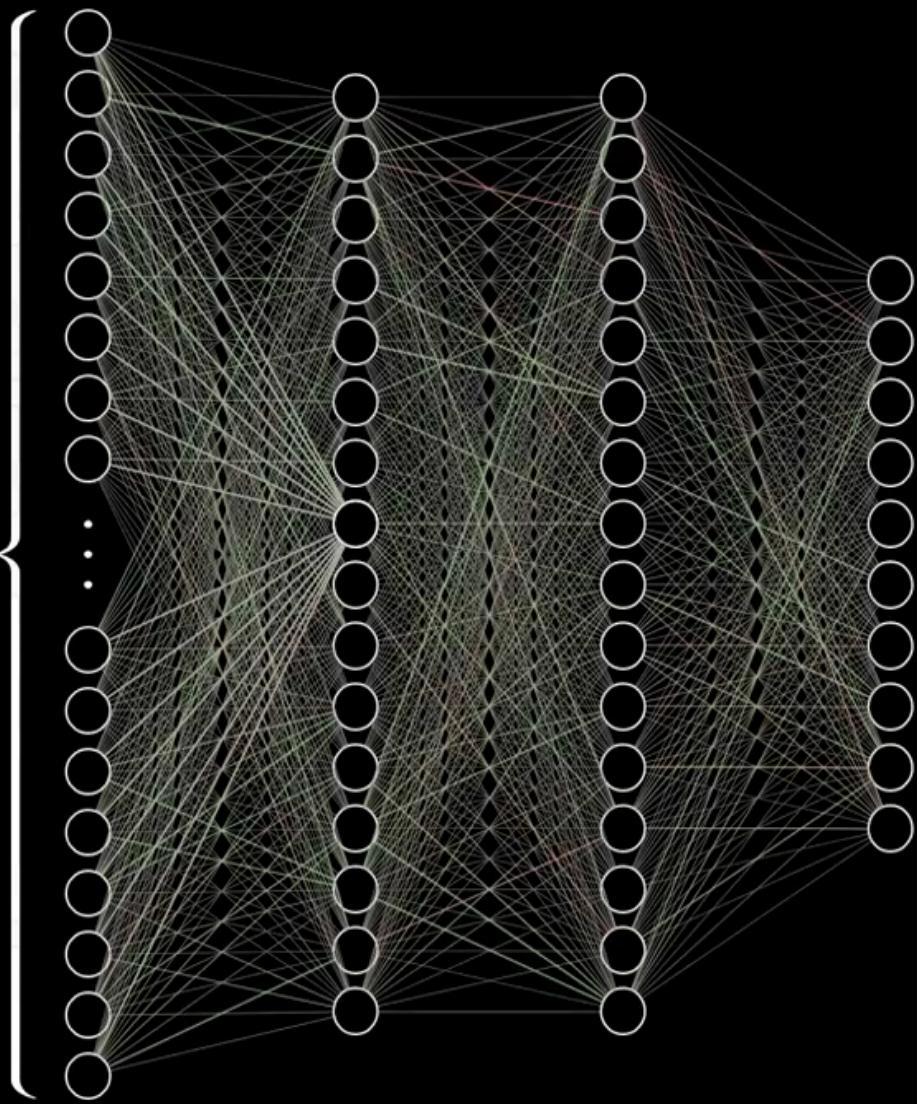


$784 \times 16 + 16 \times 16 + 16 \times 10$
weights

16 + 16 + 10
biases

13,002

784

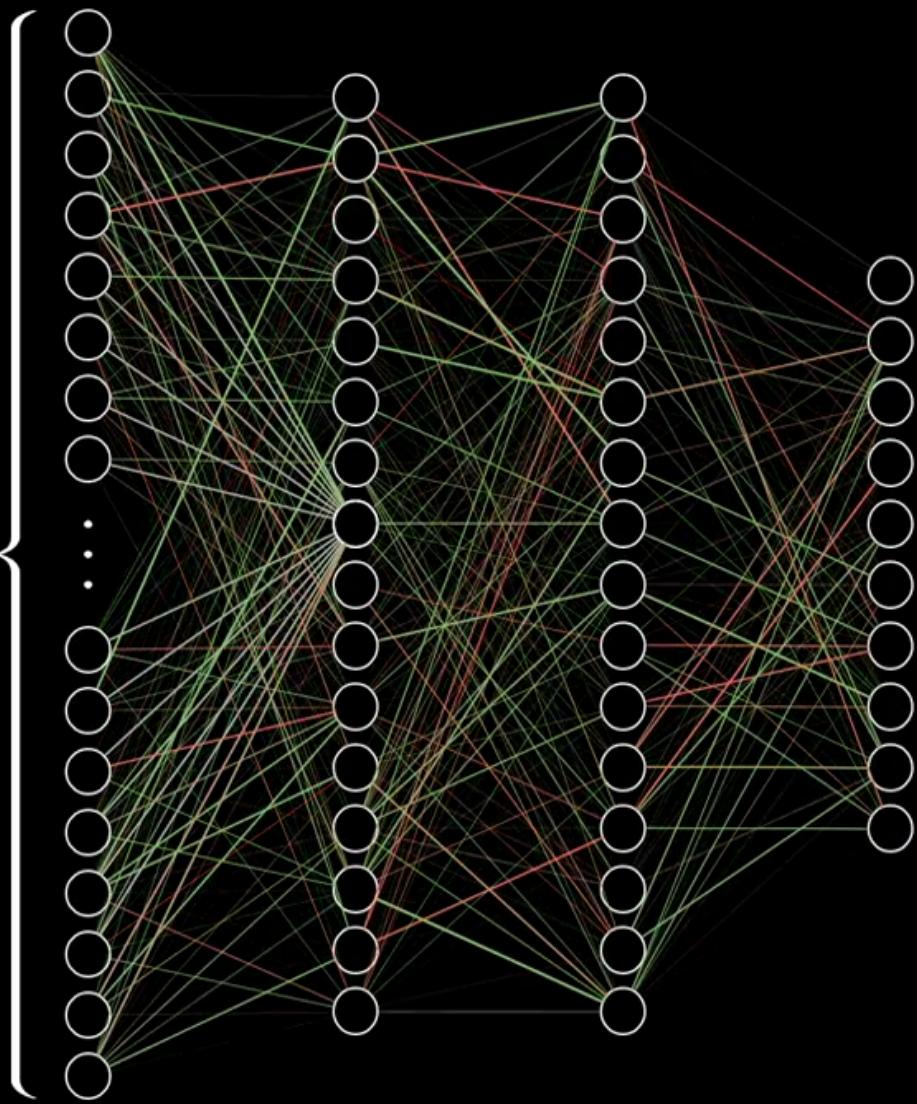


$784 \times 16 + 16 \times 16 + 16 \times 10$
weights

16 + 16 + 10
biases

13,002

784

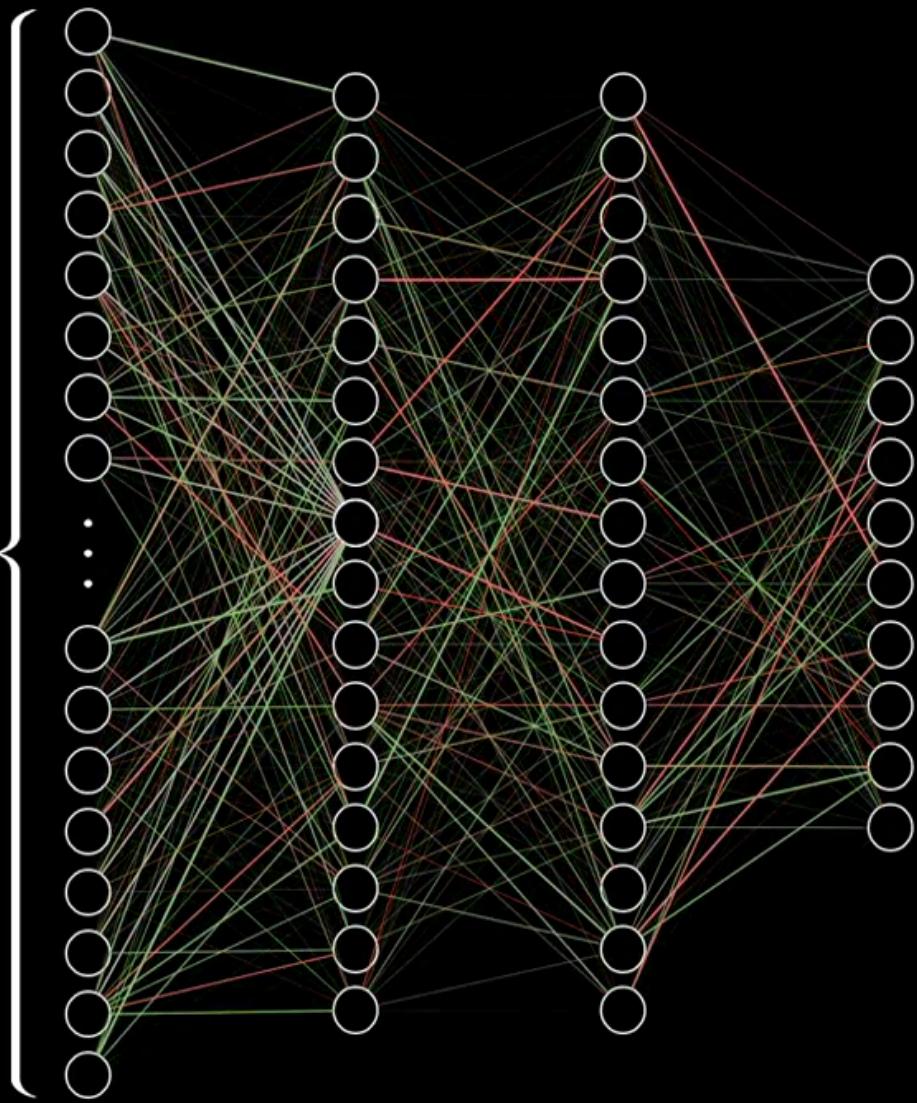


$784 \times 16 + 16 \times 16 + 16 \times 10$
weights

16 + 16 + 10
biases

13,002

784



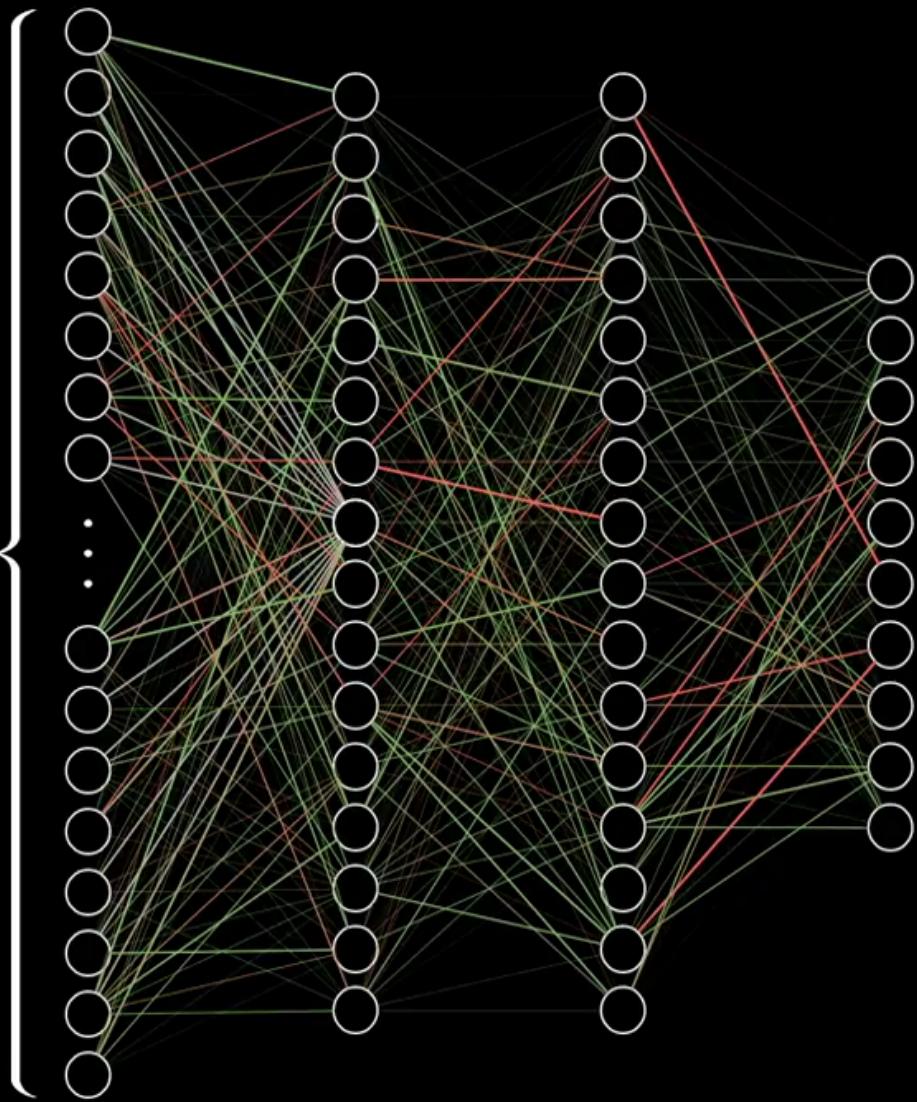
$784 \times 16 + 16 \times 16 + 16 \times 10$
weights

16 + 16 + 10
biases

13,002

Learning → Finding the right
weights and biases

784

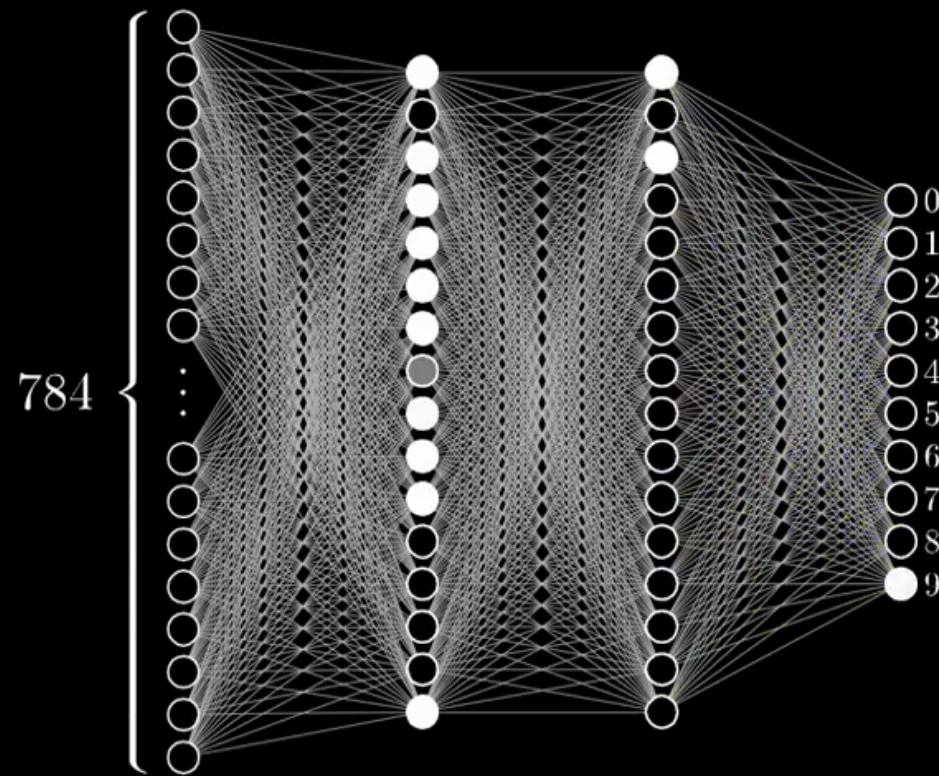


$784 \times 16 + 16 \times 16 + 16 \times 10$
weights

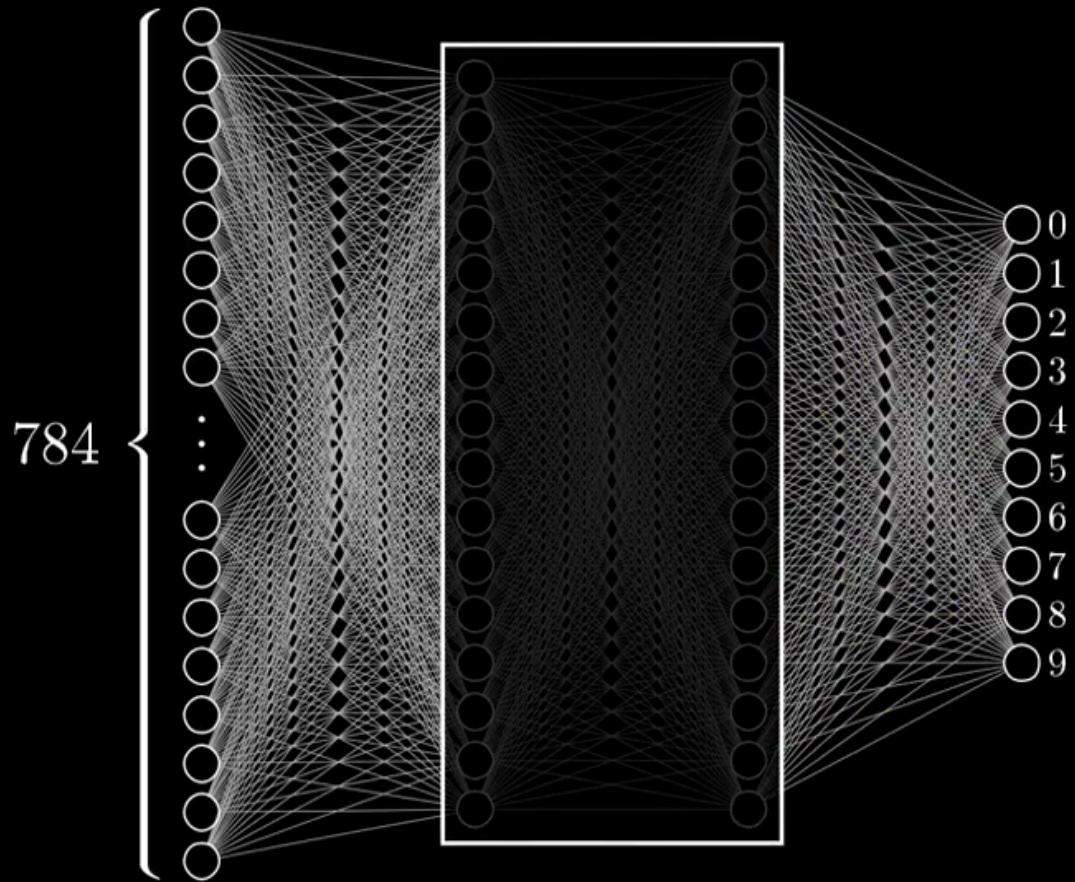
16 + 16 + 10
biases

13,002

Learning \rightarrow Finding the right
weights and biases



...rather than treating this as a black box

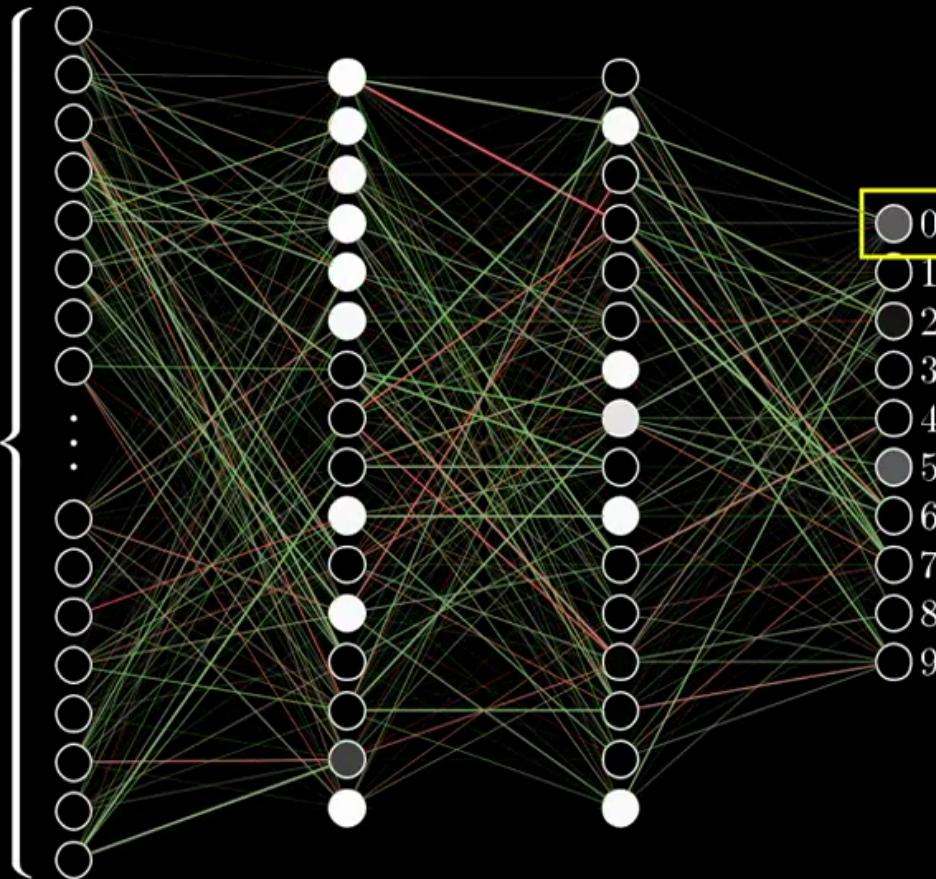


What weights are used here?

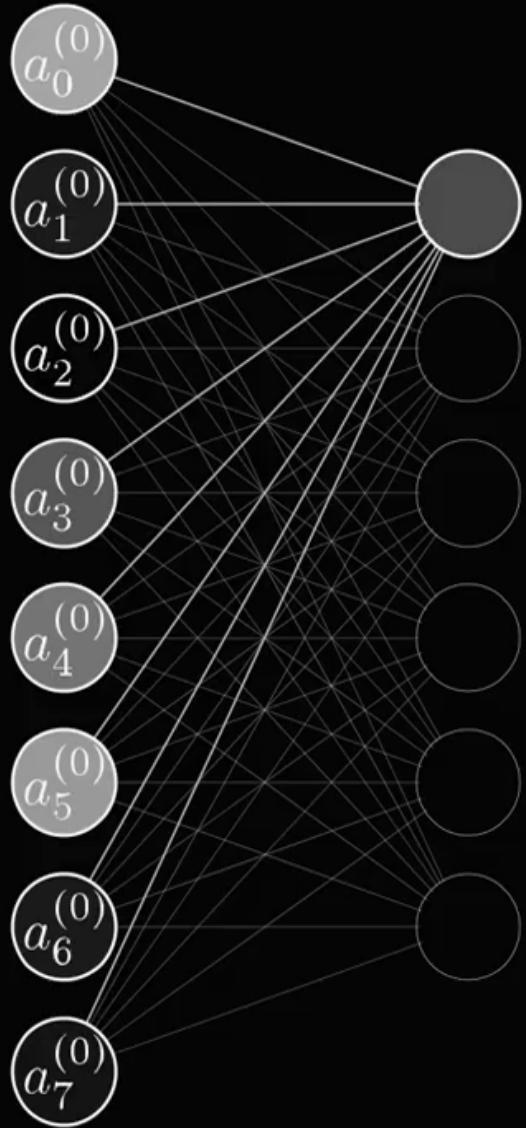
What are they doing?



784



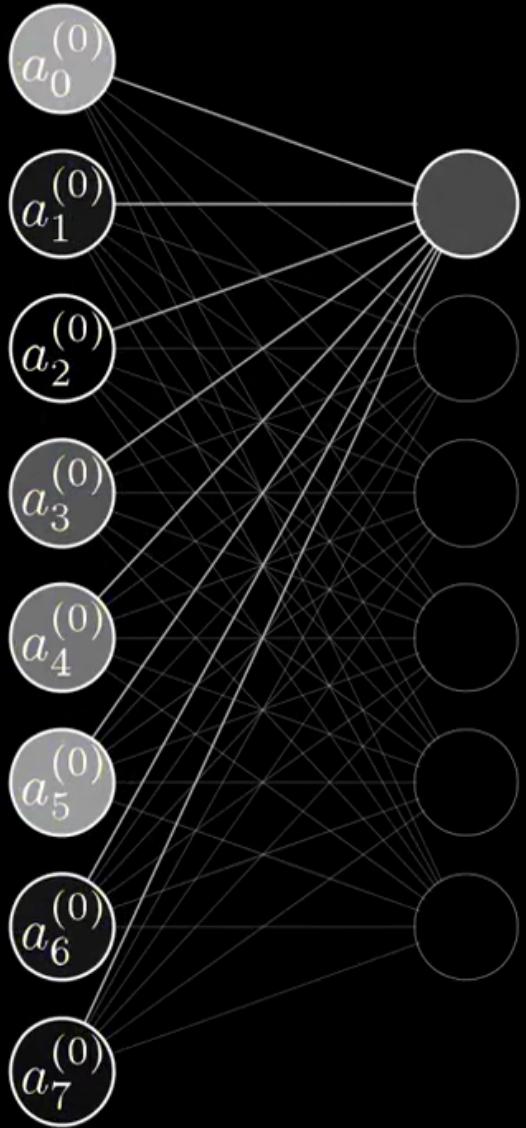
Wrong!



Sigmoid

$$a_0^{(1)} = \sigma \left(w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \cdots + w_{0,n} a_n^{(0)} + b_0 \right)$$

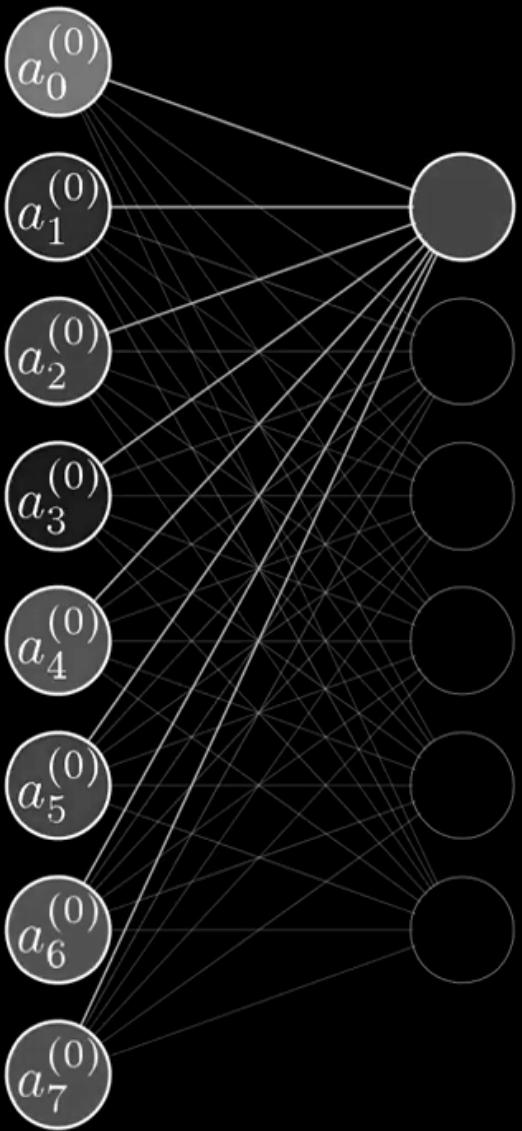
Bias



Sigmoid

$$a_0^{(1)} = \sigma(w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \dots + w_{0,n} a_n^{(0)} + b_0)$$

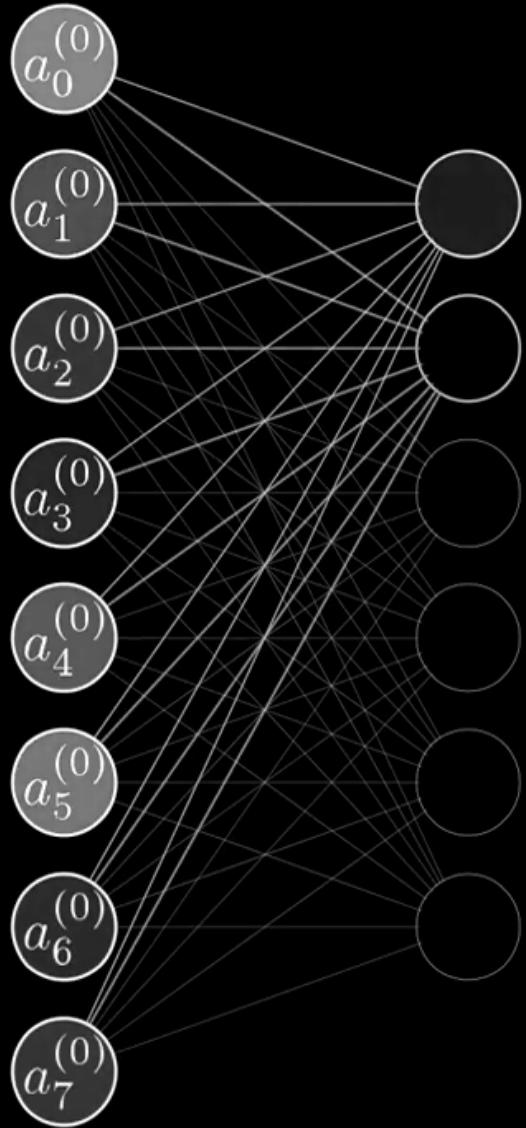
Bias



Sigmoid

$$a_0^{(1)} = \sigma \left(\underbrace{w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \cdots + w_{0,n} a_n^{(0)}}_{\text{Bias}} + b_0 \right)$$

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$



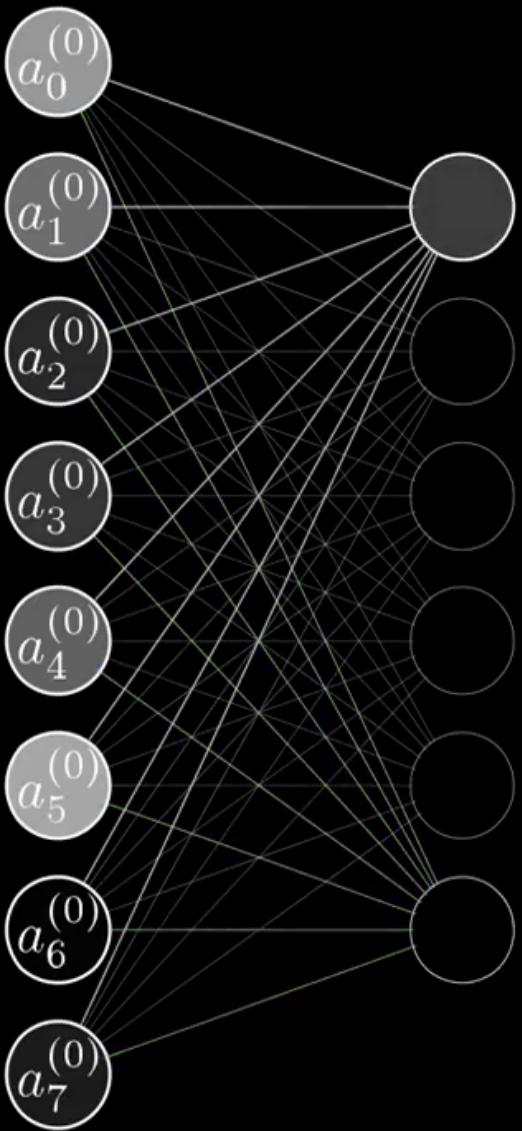
Sigmoid

$$a_0^{(1)} = \sigma \left(\underbrace{w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \cdots + w_{0,n} a_n^{(0)}}_{\text{Sum}} + b_0 \right)$$

The equation shows the calculation of the output $a_0^{(1)}$ using the sigmoid function σ . The sum of the weighted inputs $w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \cdots + w_{0,n} a_n^{(0)}$ is highlighted with a green bracket. A yellow arrow points from this bracket to the word "Bias" below it, indicating that the bias term b_0 is added to the sum.

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$

This matrix multiplication illustrates the forward pass of the neural network. The weight matrix has dimensions $(k+1) \times n$ (including the bias column) and the input vector has dimension n . The result is a vector of size $k+1$, where each element is marked with a question mark, representing the uncomputed output values.

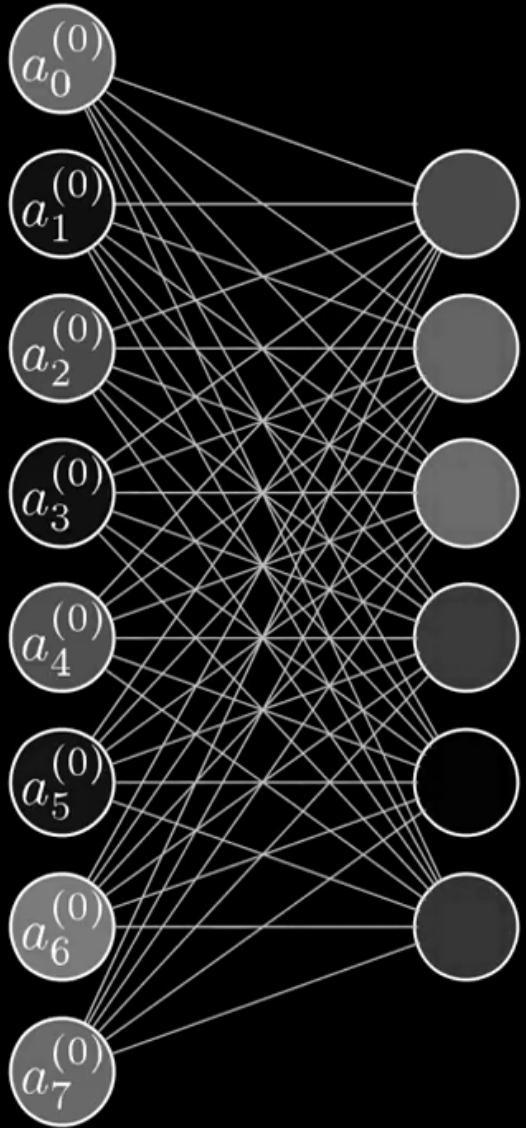


Sigmoid

$$a_0^{(1)} = \sigma \left(w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \dots + w_{0,n} a_n^{(0)} + b_0 \right)$$

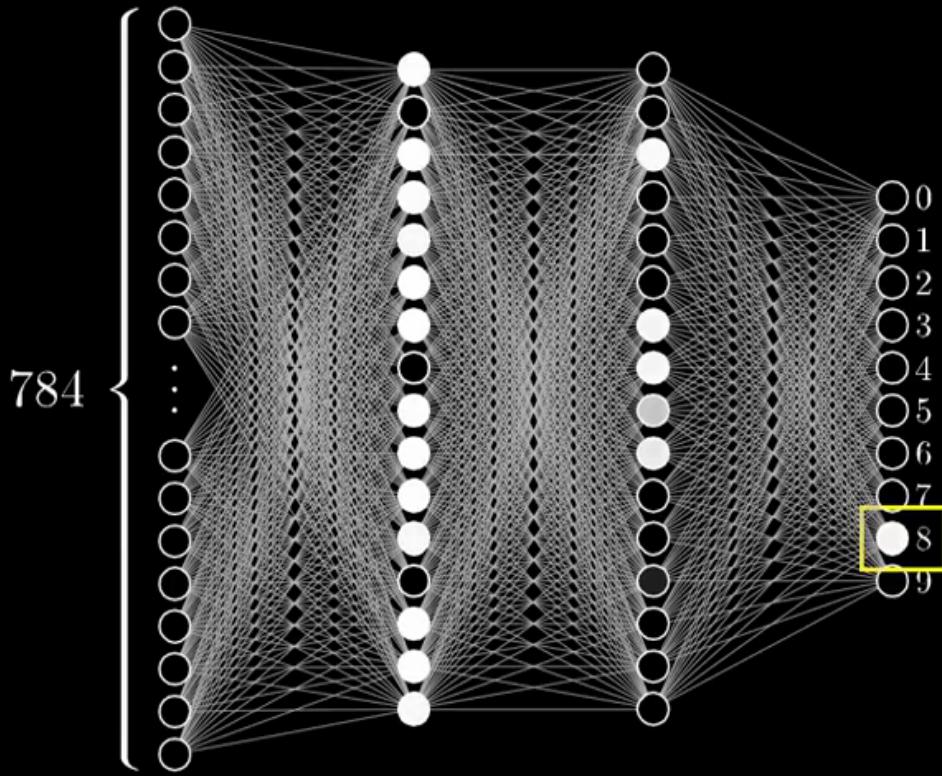
Bias

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$



$$\sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

$$\sigma(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b})$$

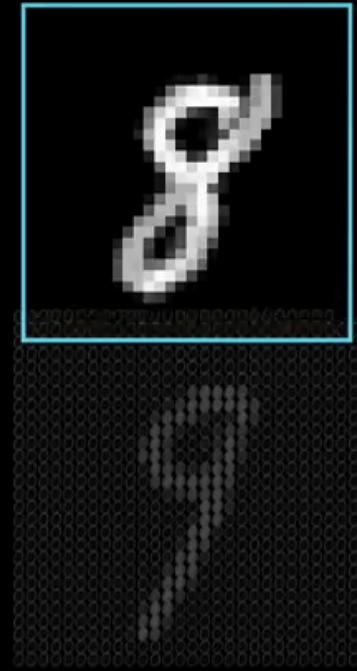


Neuron

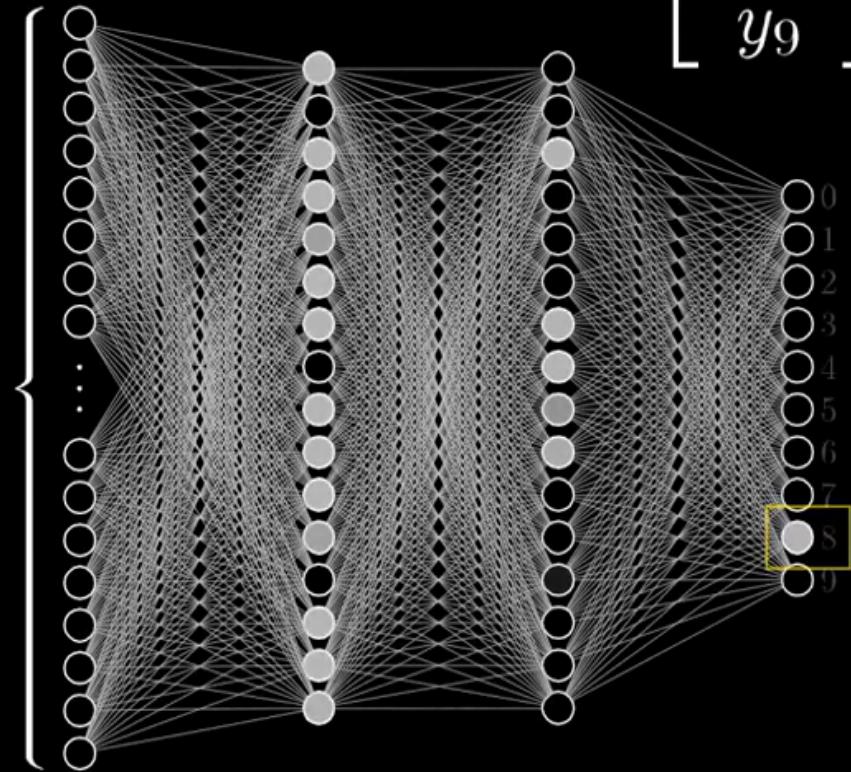


Function

~~Thing that holds
a number~~



$$f(a_0, \dots, a_{783}) = \begin{bmatrix} y_0 \\ \vdots \\ y_9 \end{bmatrix}$$



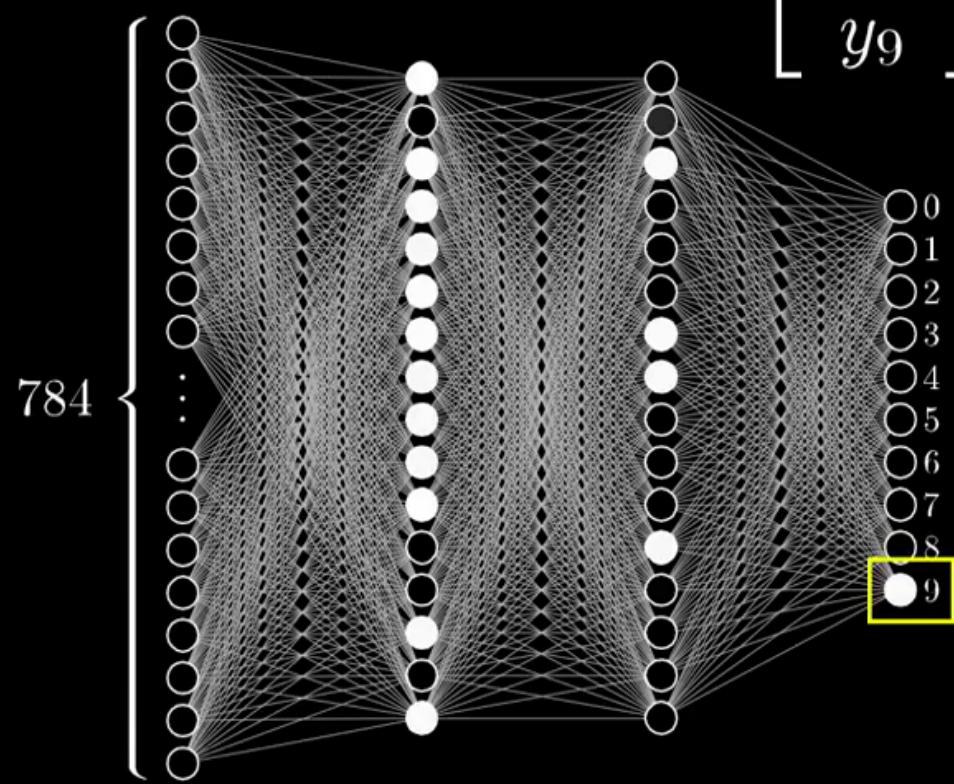
Network
↓
Function

~~Thing that holds
a number~~

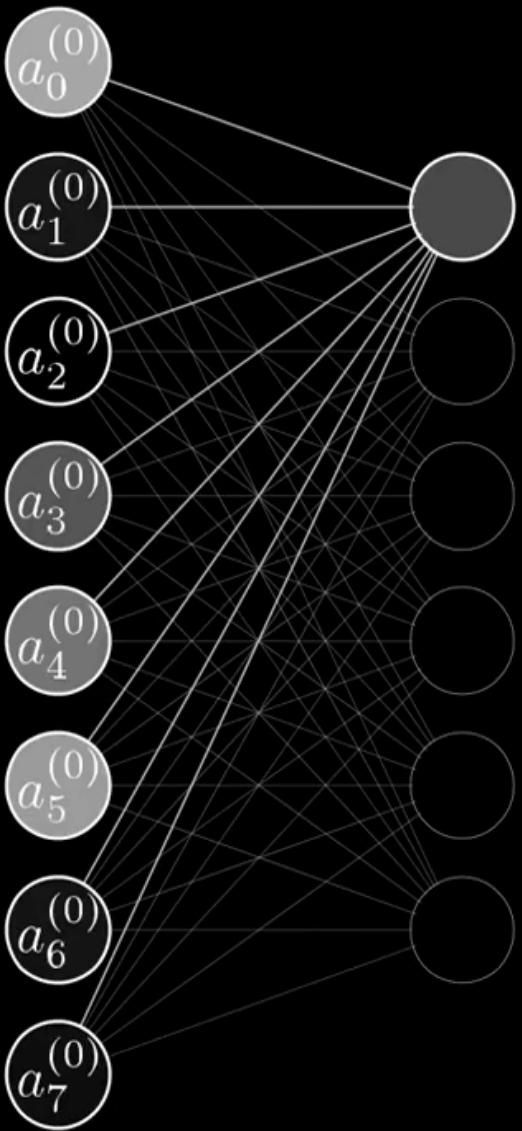


$$f(a_0, \dots, a_{783}) = \begin{bmatrix} y_0 \\ \vdots \\ y_9 \end{bmatrix}$$

Network
↓
Function



~~Thing that holds
a number~~

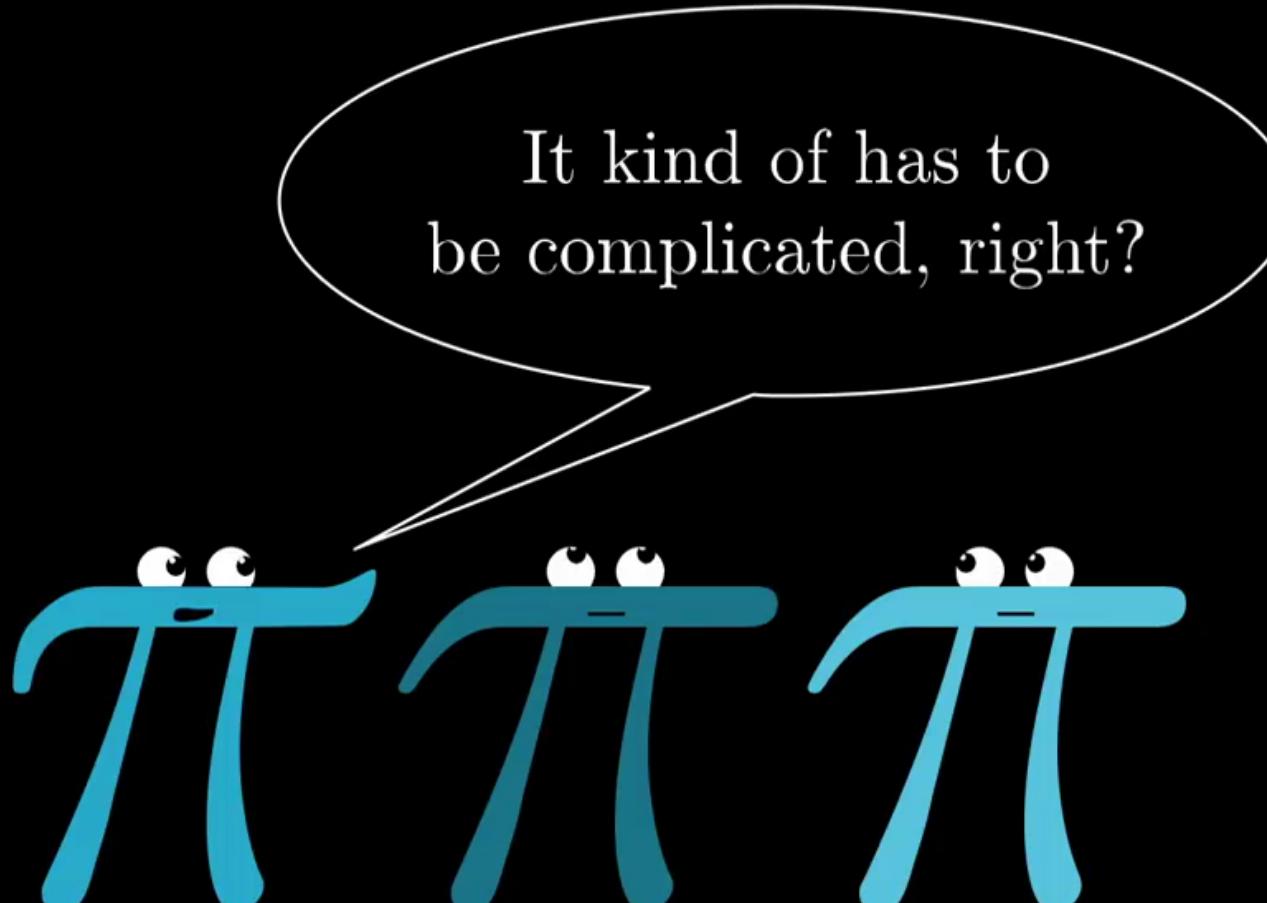


Sigmoid

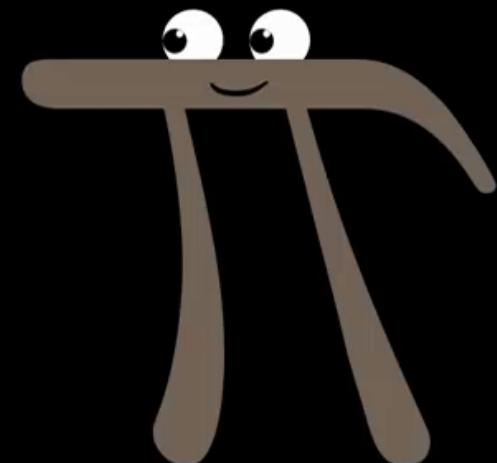
$$a_0^{(1)} = \sigma \left(w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \cdots + w_{0,n} a_n^{(0)} + b_0 \right)$$

Bias

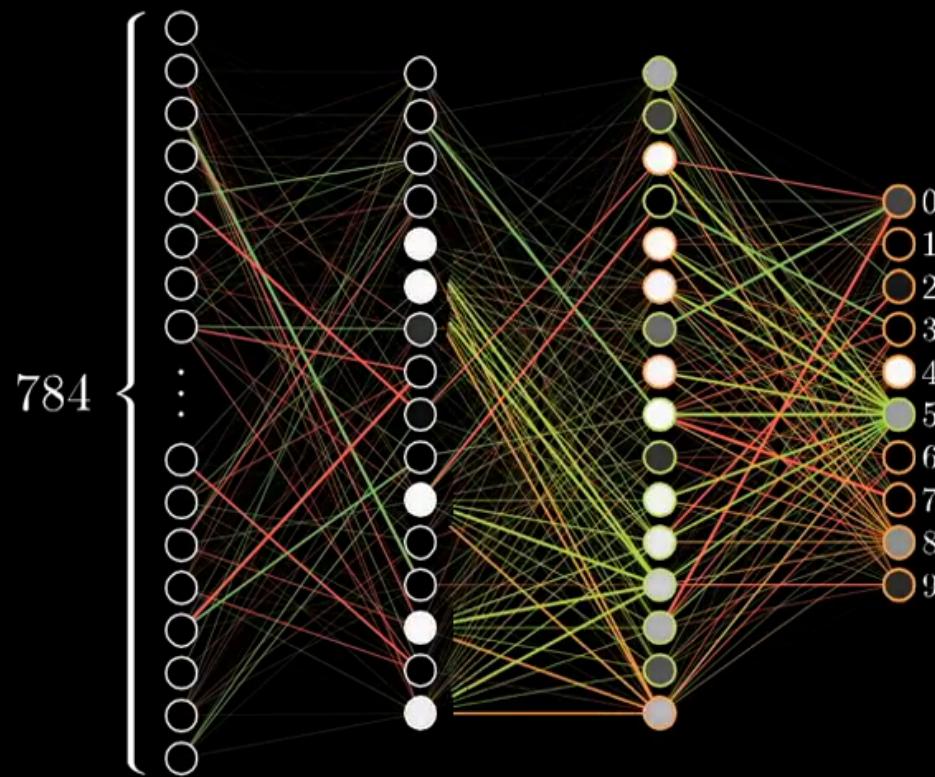
$$\sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,0} & w_{n,1} & \dots & w_{n,k} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$



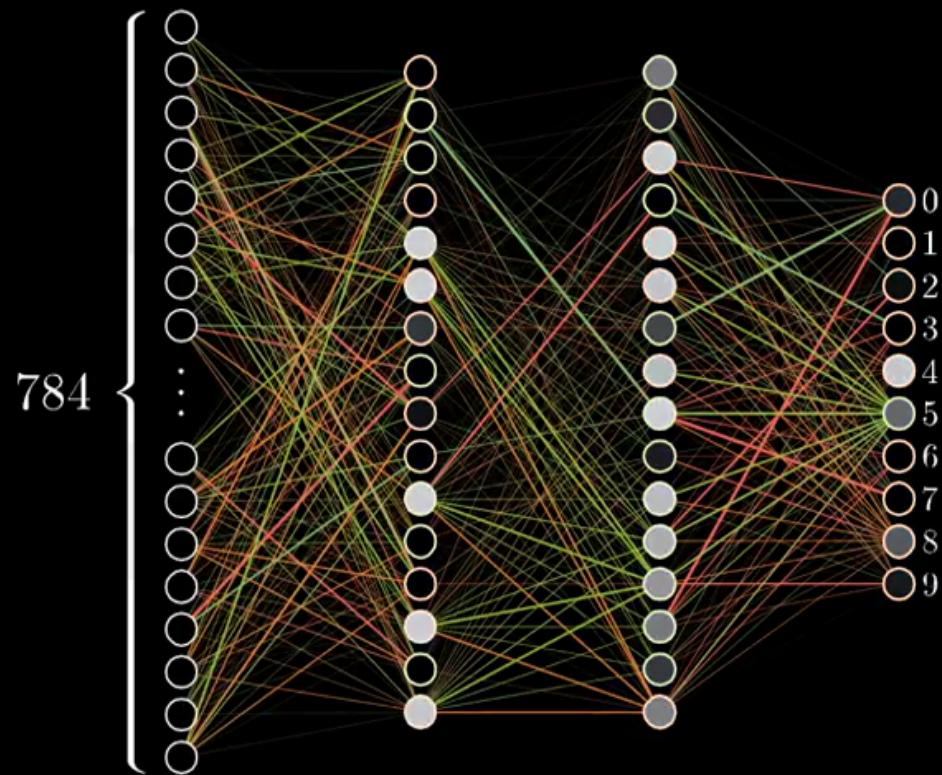
It kind of has to
be complicated, right?



Training in
progress. . .

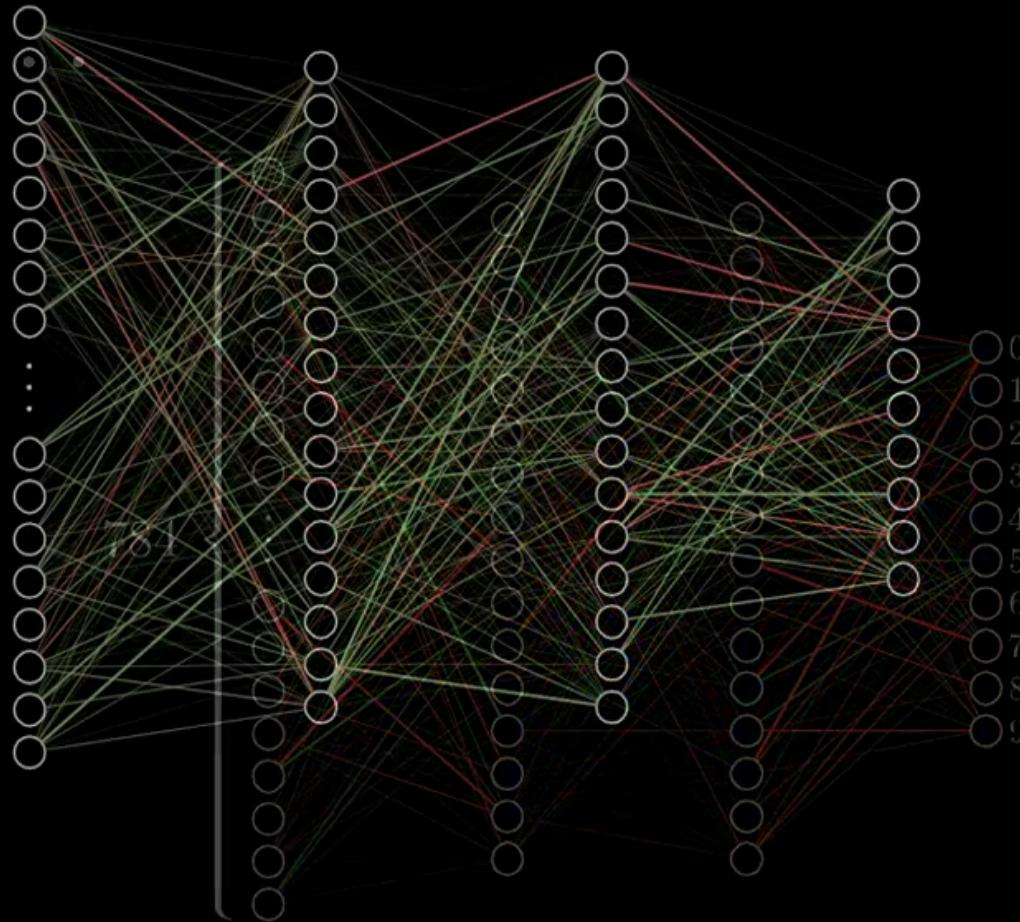


Training in
progress. . .



Training in
progress.

q → 9



A

Basics of Image & Convolution

Image can be seen as a 2D function can be plotted in 3D.
Value = $I(x, y)$

v_1 (1,1)	v_2 (1,2)	v_3 (1,3)	v_4 (1,4)
v_5 (2,1)	v_6 (2,2)	v_7 (2,3)	v_8 (2,4)
v_9 (3,1)	v_{10} (3,2)	v_{11} (3,3)	v_{12} (3,4)
v_{13} (4,1)	v_{14} (4,2)	v_{15} (4,3)	v_{16} (4,4)

$$\therefore v_6 = I(2, 2)$$

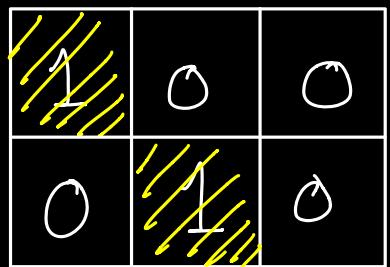
$$[I] \quad v_{15} = I(4, 3)$$

a) These (v_i)'s can be 0/1

\Rightarrow Binary Image

Off/ON

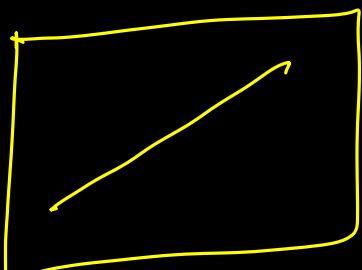
Often called as BLACK/WHITE image.



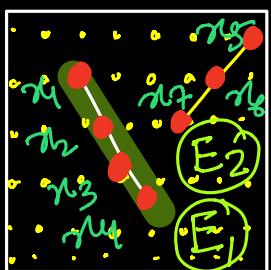
b) These (v_i 's) can be a value b/w 0 - 255

Basically 8 bit-value per pixel

$00000000 \Rightarrow 0$ (Black) } acquiring the full
 00000001 gray level spectrum
 \vdots
 $11111111 \Rightarrow 255$ (White) }
 \Rightarrow Such images will
 \Rightarrow be called as gray level images.



④ What are things that are important in an image.



* do all pixels are useful/important

* Only few of them that have edges

Edges \Rightarrow Changes \Rightarrow gradient/
high derivative

* Do pixels on edges have same information?

x_1, x_2, x_3, x_4 all are equally important?

$\{x_1, x_2, x_3, x_4\}$ Share edge E_1 $\{x_5, x_6, x_7\}$ Share edge E_2

Do they share information within their set/group?

Do they share information across the set/group?

\Rightarrow Each of (x_i) have one thing common
"They have high derivative"

But in which direction (x) or (y) .

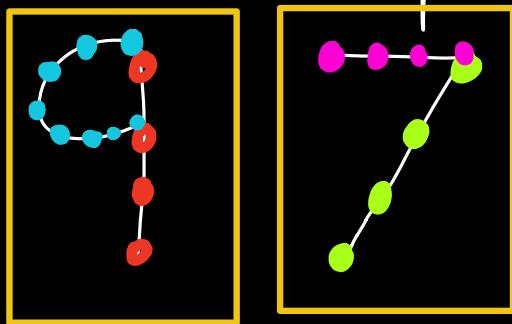
∴ Along with the magnitude of the gradient
pixel will also have a direction.

$I(x, y)$, $\nabla_x I(x, y)$, $\nabla_y I(x, y)$



Similar to \leftrightarrow & Can we have
directional derivatives, may be like

- * Can we design a filter/function that can compute these directional derivatives



$D(\leftrightarrow), D(\uparrow), D(\downarrow), D(\nwarrow)$

Yes
using Convolution.

One can observe that this directional information is important for learning the discriminative features

(Response is saved in a separate signal) (Do you belong to a horizontal line?)

These filters are used to ask specific questions to each pixel.

$$D(\leftrightarrow)[x,y] \Rightarrow R(x,y)$$

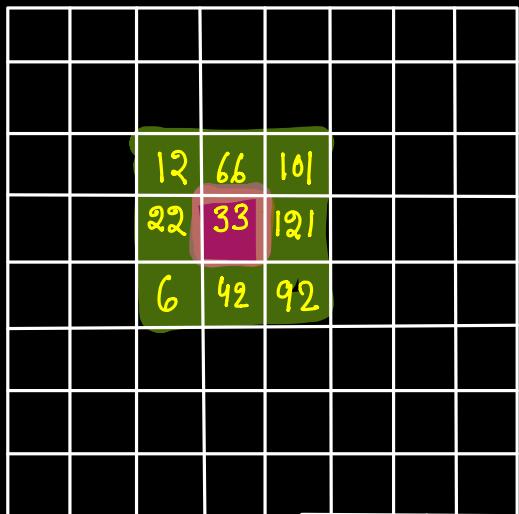
B CONVOLUTION

$$\underbrace{I(x,y)}_{\text{Signal}} \circledast \underbrace{F(x,y)}_{\text{Pixel level operator}} \Rightarrow \overline{\text{Response}}$$

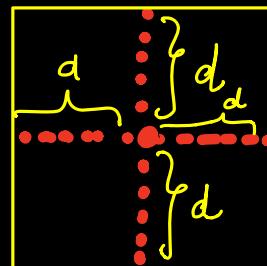
Signal Convolved using $F(x,y)$

$F(x,y) \Rightarrow$ Pixel level operator, (Computed w/ a neighbourhood)

Let $I(x, y) = m \times n$ & $F(x, y) = (K \times K)$
where $(K = 2d + 1)$



$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

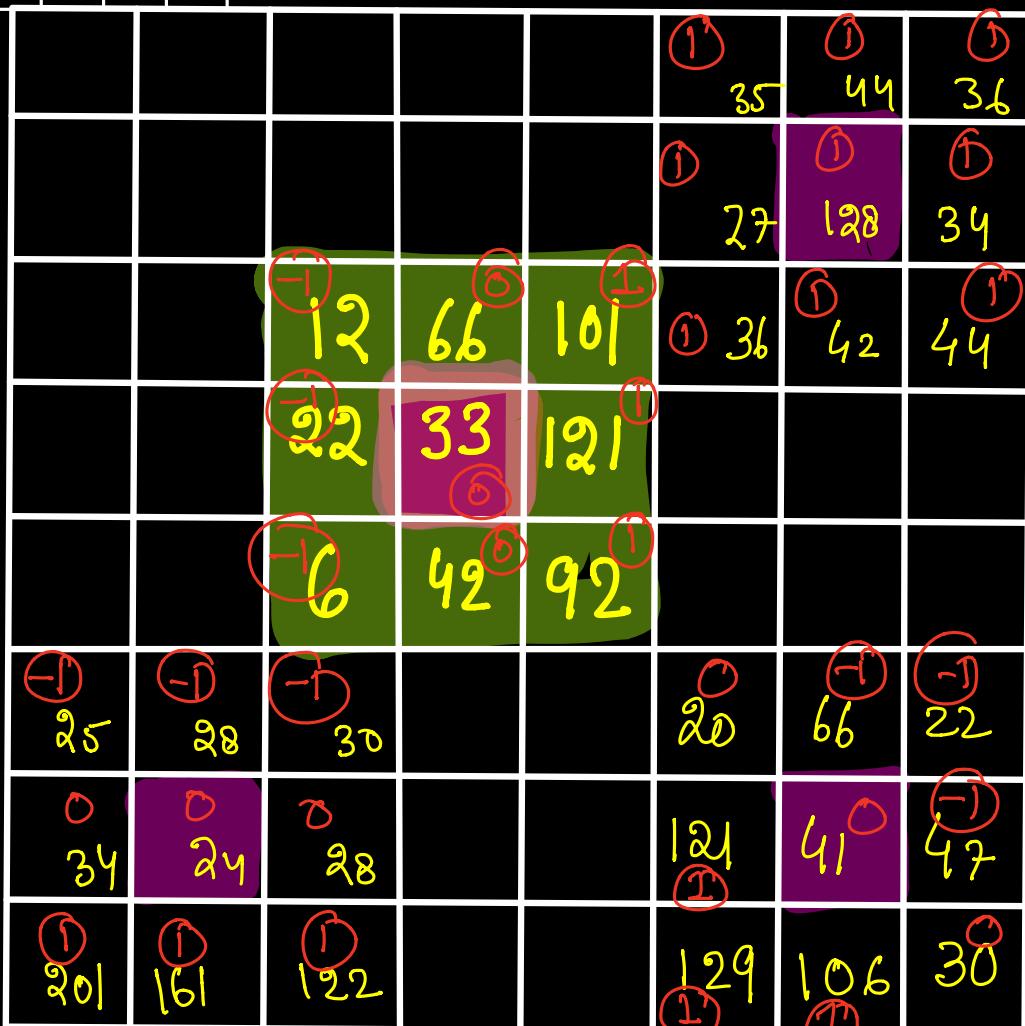


* What is your (x) gradient w.r.t (3×3) neighbourhood.
Do you have a pattern like this.

-12 +101
-22 +121
-6 +92
 \Rightarrow Some high value

implies that
Yes it has the pattern.

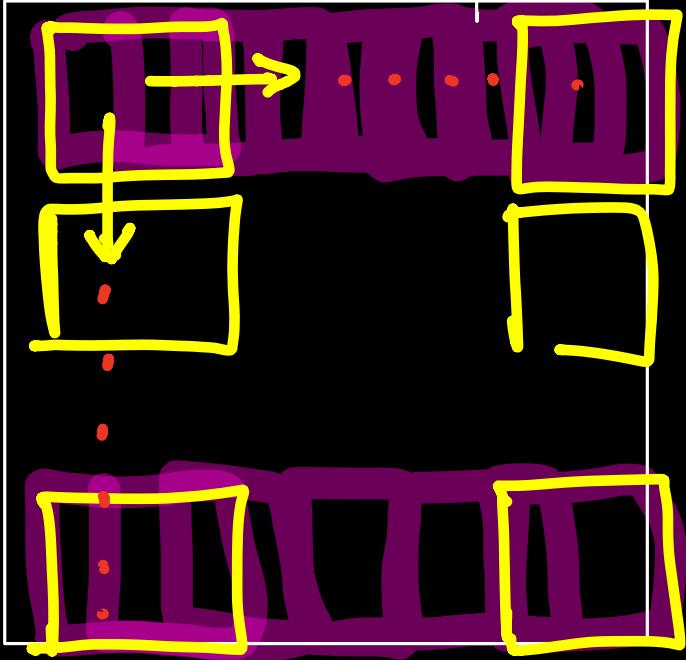
Hence one can see very useful questions can be embedded in these filters/Kernels.



$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

Full Convolution operation is like (Sliding Window)



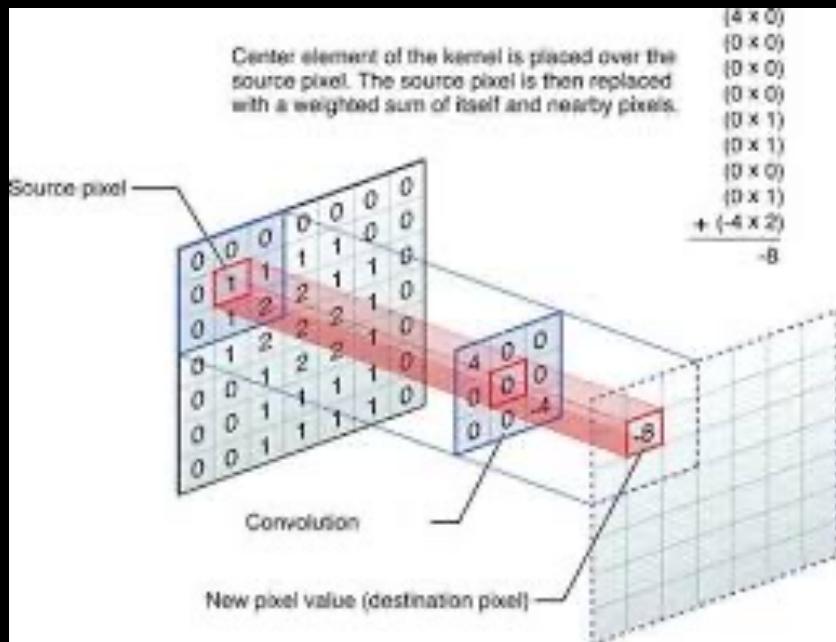
if image is 128×128
& filter is (3×3)
 $O/P \Rightarrow (126 \times 126)$
will be the o/p
top & bottom Row &
left & right Column do
not have full neighbourhood.

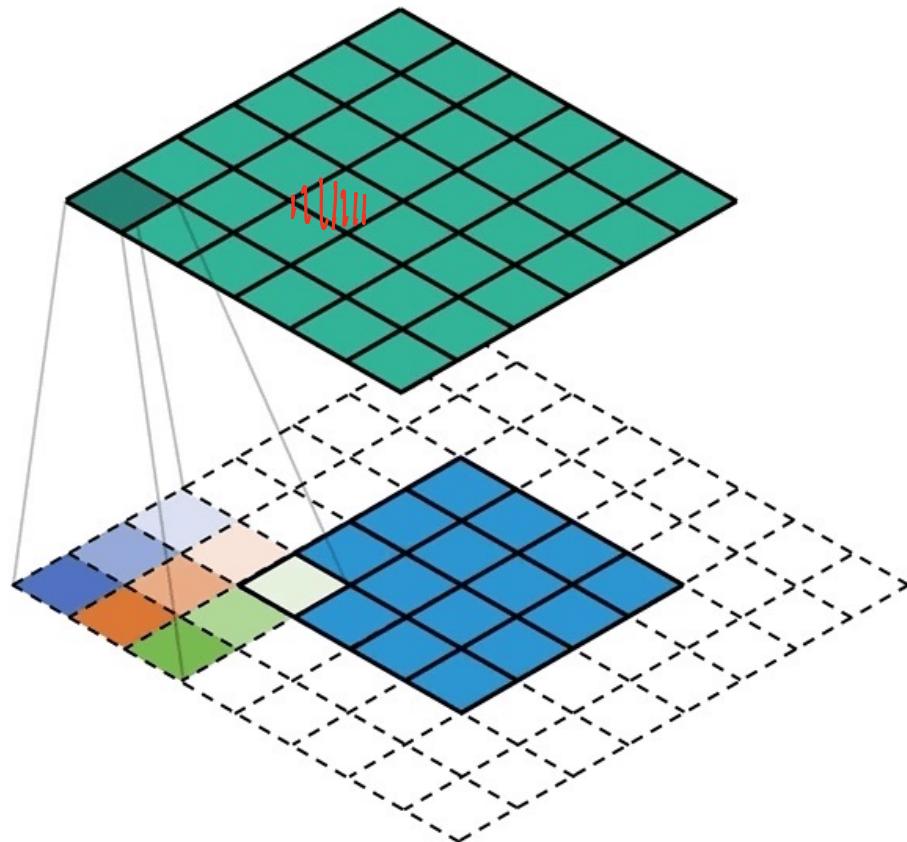
There are several ways to manage Boundary Conditions

- Ignore
- Zero padding

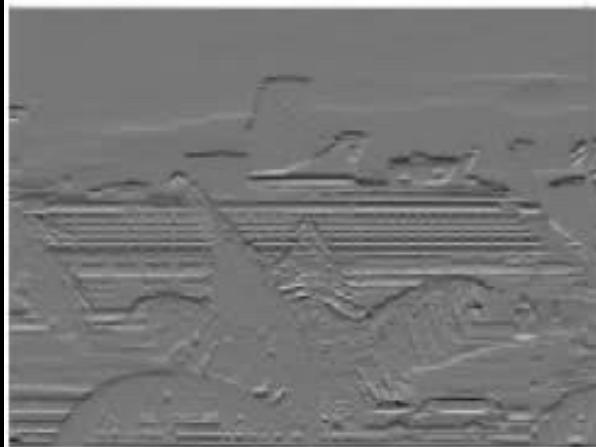
Hence

$$I(m, n) \otimes F(k \times k) \Rightarrow R(m, n)$$

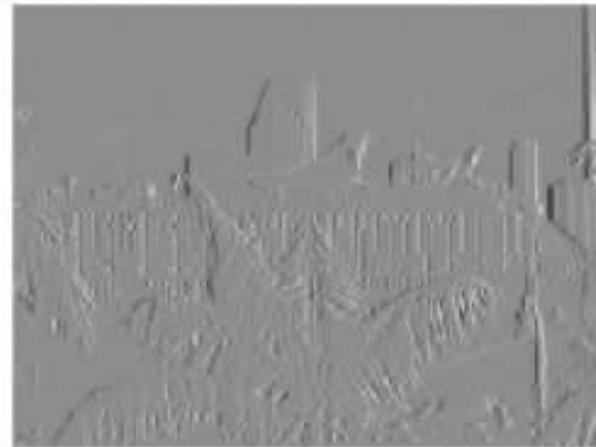




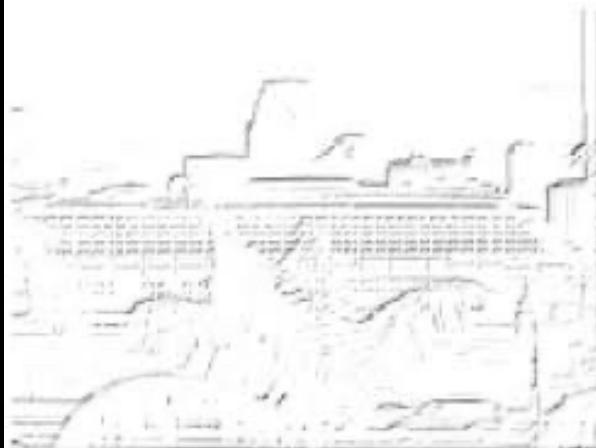
Sobel H



Sobel V

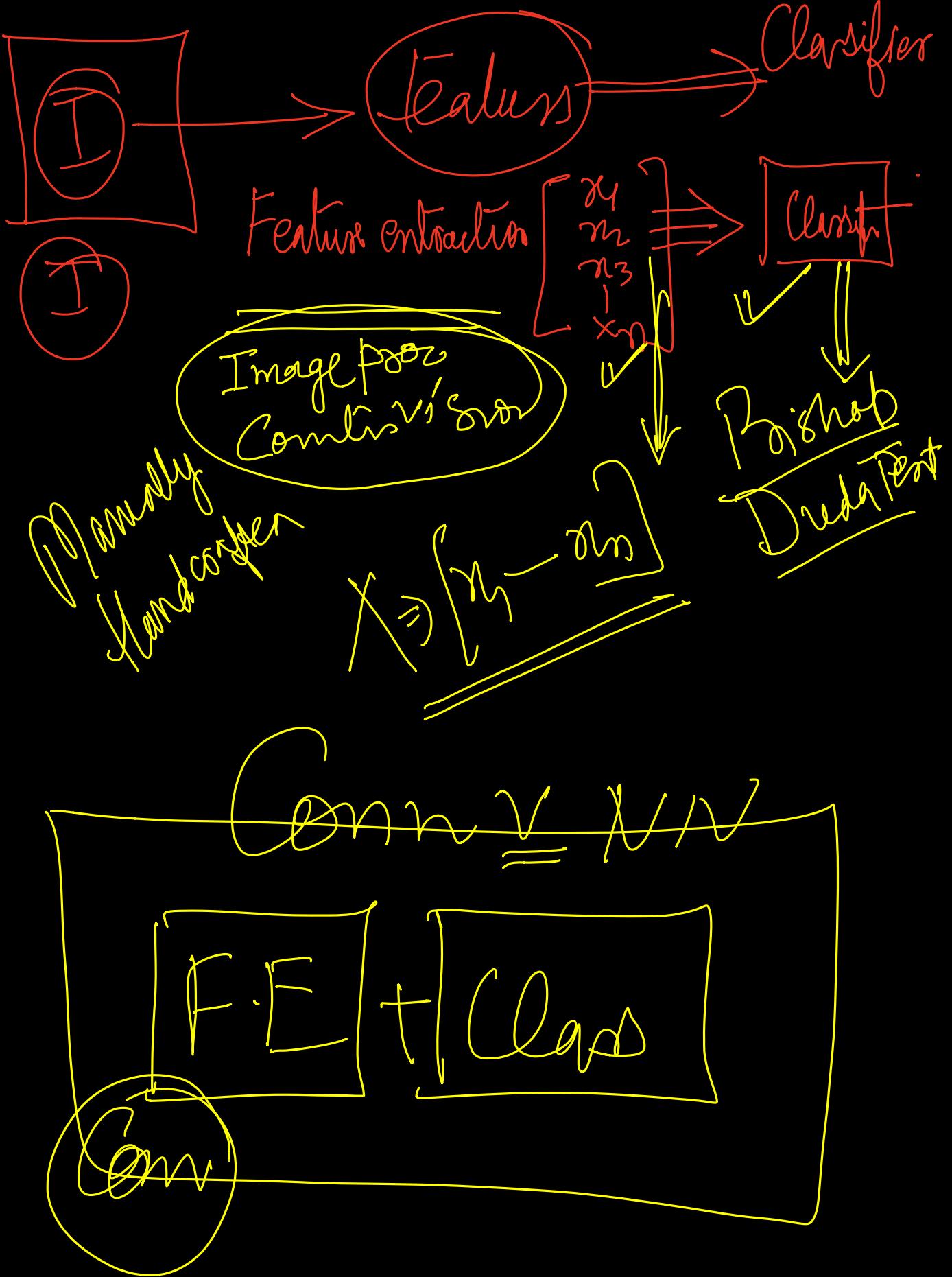


Sobel H&V



Laplacian



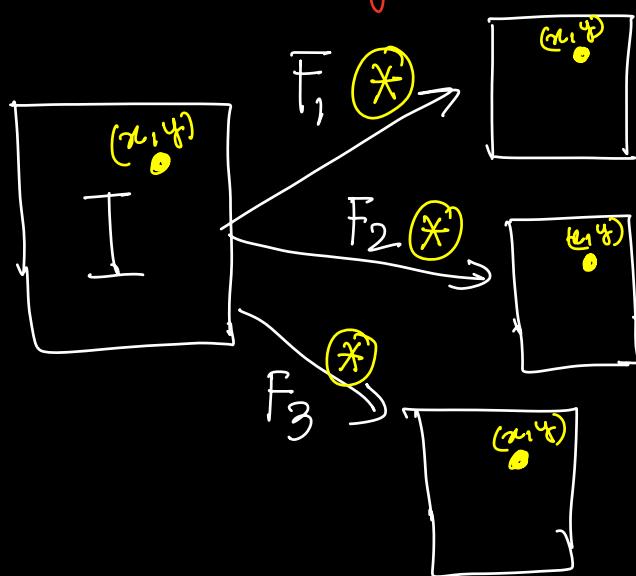


CONVOLUTIONAL NEURAL NETWORK

This lecture is adapted from "How Convolutional Neural network works by Brandon Rohrer".

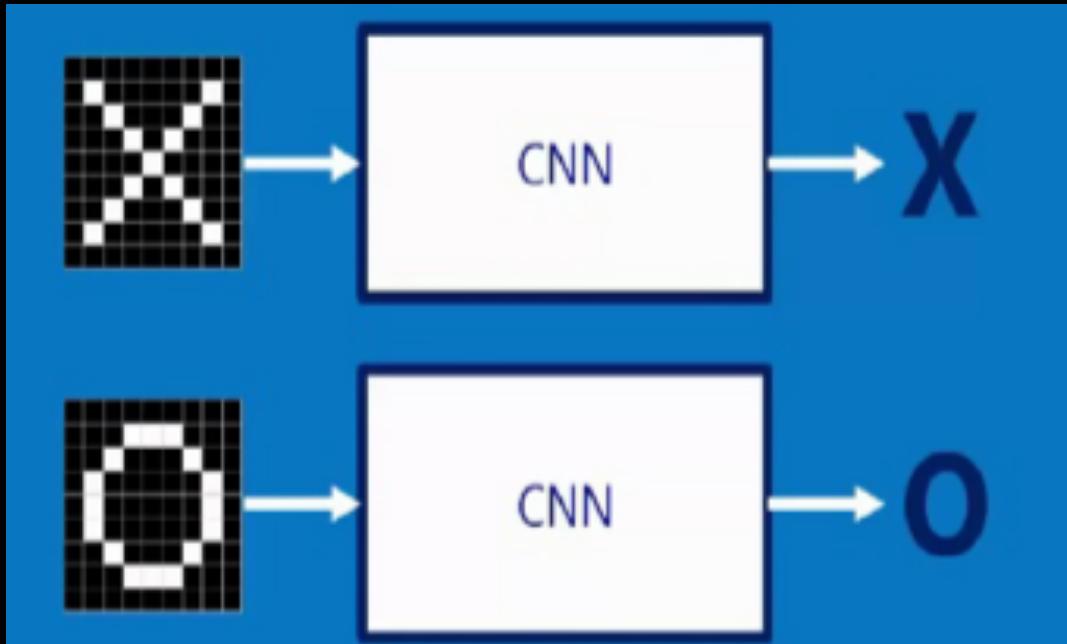
(available on YouTube)

- ⊗ Visual interview of an image $I(m, n)$.
- ⊗ Visually relevant questions, that can be useful for obtaining discriminative feature (Response), need to be asked. (They are encoded)
 - What is your directional gradient within a filter $F(K \times K)$
 - Do you belong to a line like (/) or (\) (|) ?
 - Do you have a pattern like (+) or (X) ... ?
 - Do you belong to red blob / or similar

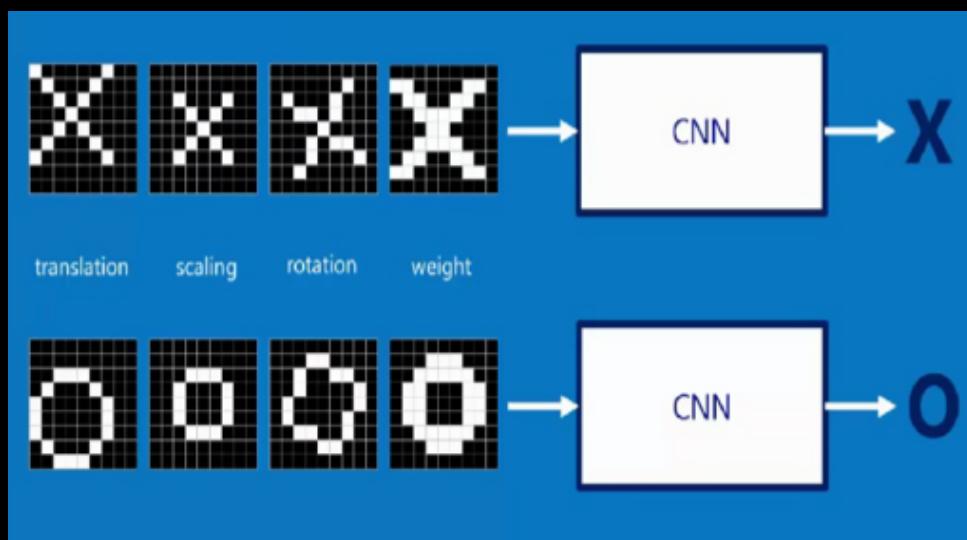


$I(x, y)$ is the pixel under consideration
↓
 $F_1(x, y), F_2(x, y), F_3(x, y) \Rightarrow$ They are its responses for the specifically designed question wst the objective fn.

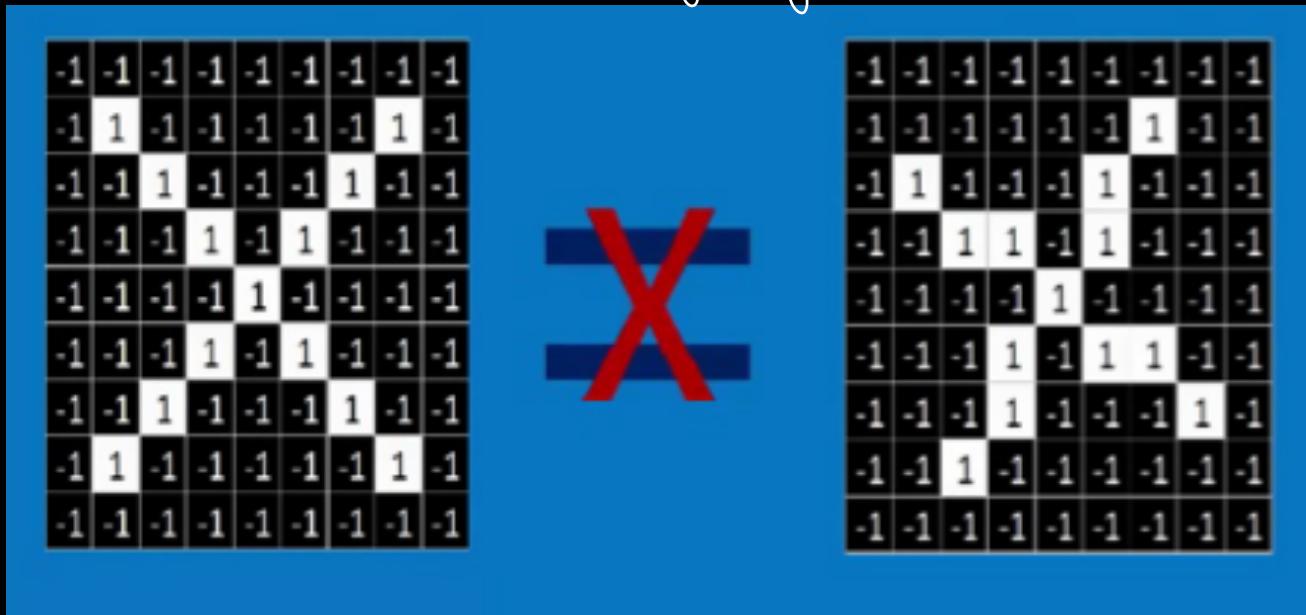
Let us talk about a problem



Design a CNN network that
Can classify/differentiate b/w \textcircled{X} &
 $\textcircled{0} \Rightarrow$ Image classifier.



These variations really makes
this problem very difficult to solve.



Q1 Both are same or different

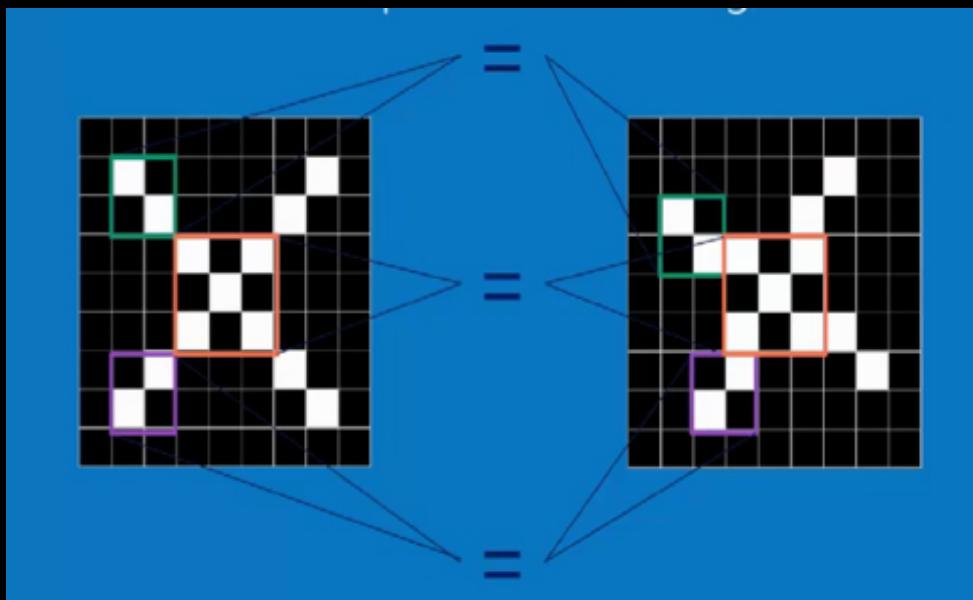
Q2 Both of them are \times or 0

\Rightarrow Q1 & Q2 Both are
2 Completely different problems .

Q1 \Rightarrow Metric learning
Q2 \Rightarrow Classification

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1	
-1	X	X	-1	-1	X	X	-1	-1	
-1	-1	X	1	-1	1	-1	-1	-1	
-1	-1	-1	-1	1	-1	-1	-1	-1	
-1	-1	-1	1	-1	1	X	-1	-1	
-1	-1	X	X	-1	-1	X	X	-1	
-1	X	X	-1	-1	-1	-1	X	-1	
-1	-1	-1	-1	-1	-1	-1	-1	-1	

One can see
that at so many
places they are
different.
 $16/81$ (Mis-match)
High degree of
dis-similarity.



Partwise
they are
exactmatch.

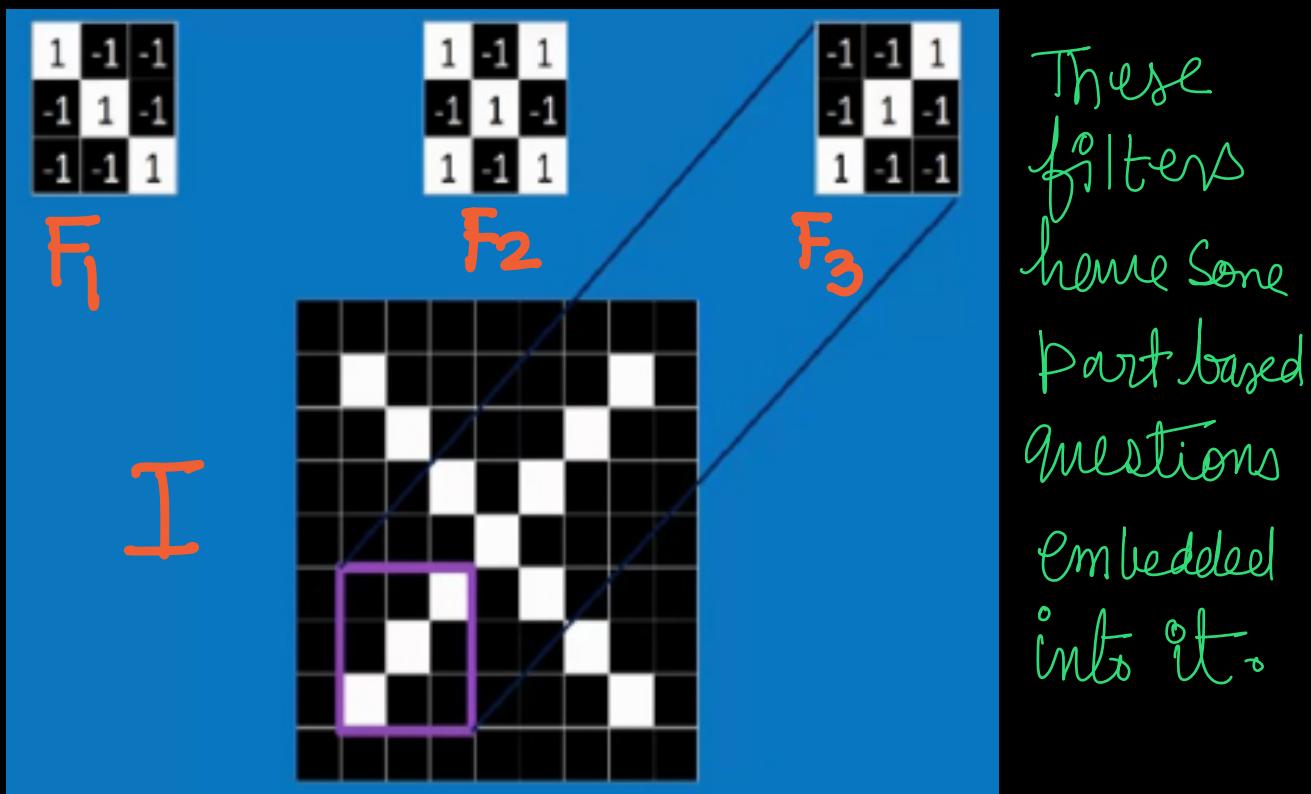


CNN can help you to see the image in parts.

1	-1	-1	1	-1	1	-1	-1	1
-1	1	-1	-1	1	-1	-1	1	-1
-1	-1	1	1	-1	1	1	-1	-1

These can be useful questions that need to be asked if we need to detect an

* Right now we assume some oracle is providing us that
⇒ But similar to FCN, they are also learned via Backpropagation.

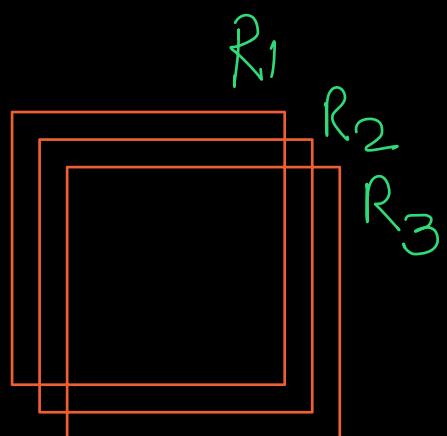


$$I \circledast F_1 \Rightarrow R_1$$

$$I \circledast F_2 \Rightarrow R_2$$

$$I \circledast F_3 \Rightarrow R_3$$

Convolve image using these filters to get the response map.



(Response Map)
or extracted features.

Basic Convolution operation.

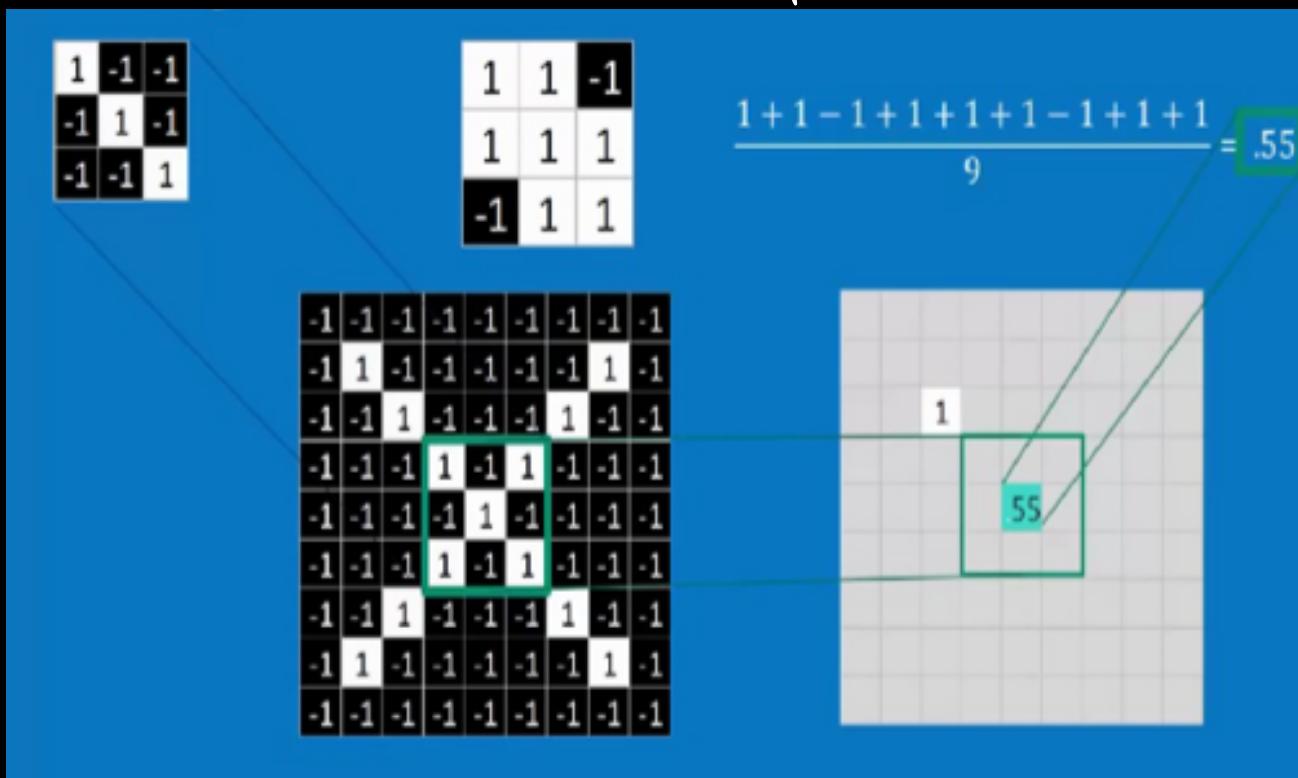


Diagram illustrating a convolution operation:

Input matrix:

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Kernel:

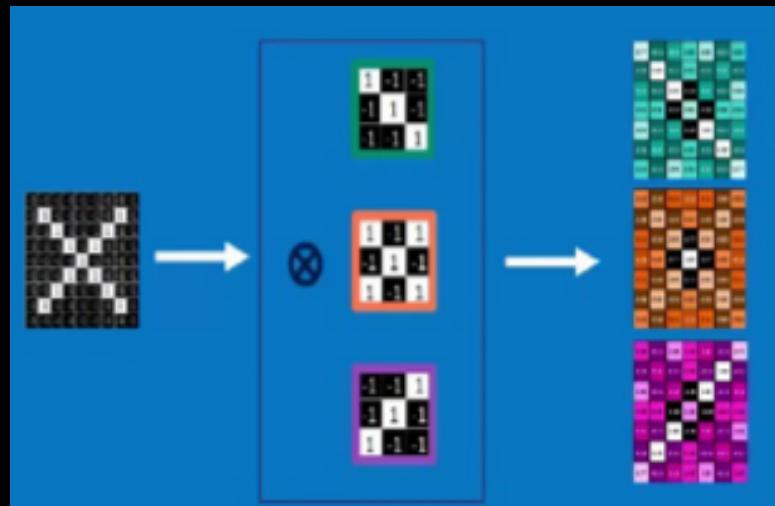
$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

Result:

$$= \begin{bmatrix} 0.77 & -0.11 & 0.11 & 0.33 & 0.55 & -0.11 & 0.33 \\ -0.11 & 1.00 & -0.11 & 0.33 & -0.11 & 0.11 & -0.11 \\ 0.11 & -0.11 & 1.00 & -0.33 & 0.11 & -0.11 & 0.55 \\ 0.33 & 0.33 & -0.33 & 0.55 & -0.33 & 0.33 & 0.33 \\ 0.55 & -0.11 & 0.11 & -0.33 & 1.00 & -0.11 & 0.11 \\ -0.11 & 0.11 & -0.11 & 0.33 & -0.11 & 1.00 & -0.11 \\ 0.33 & -0.11 & 0.55 & 0.33 & 0.11 & -0.11 & 0.77 \end{bmatrix}$$

One can understand the response obtained.

All these filters placed in one layer \Rightarrow CNN layer.



④ One filter \Rightarrow generates one Response map.

⑤ One CNN layer may have $N = 128, 256$ such filters of size may be (3×3) & (5×5)
 \Rightarrow # parameters = $N \times 3 \times 3$ OR $N \times 5 \times 5$
OR $(N \times K \times K)$.

Question:

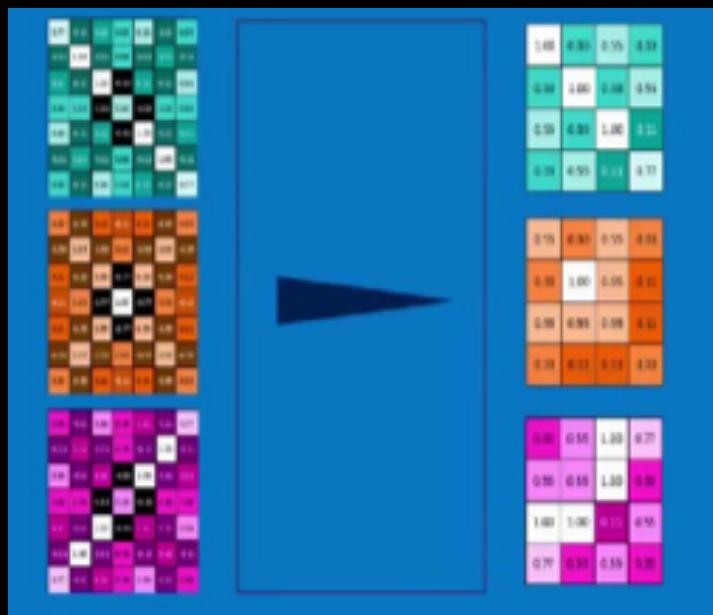
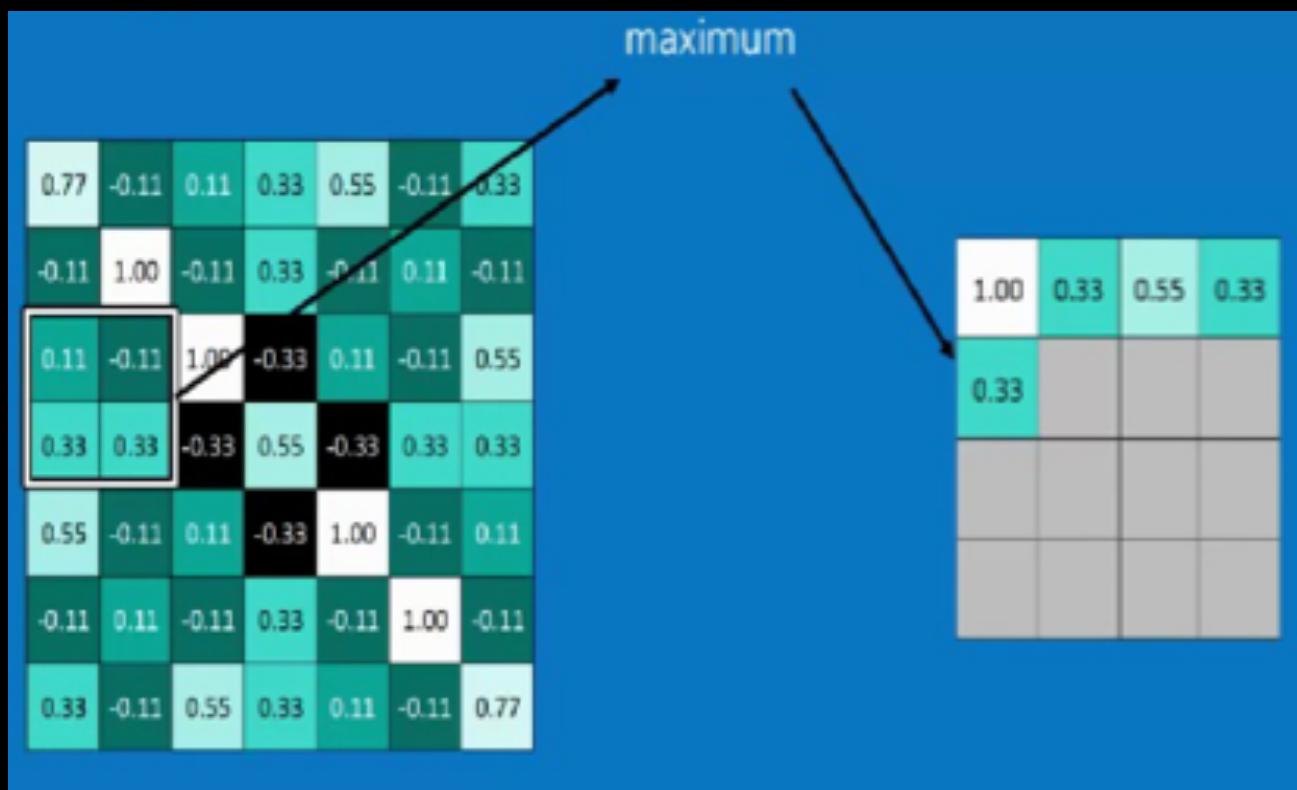
Given an input image of size 28×28 , if we apply a CNN layer with $(64) (3 \times 3)$ filters then what is the O/P dimensions.

These response maps are activated using Sigmoid, tanh or ReLU. (-Normalizing)
- Adding non-linear fn.



B

Pooling : Feature size reduction



⇒ This is
basically
the working of
2x2 pooling
over response map.

- ④ Pooling is important in order to manage the feature size.
- ④ Aggregating the features so as one can move from local to a global features.
- ④ Also provide some amount of robustness against Rotation/translation.
- ④ 2×2 pooling is common and will reduce the h & w by ②.
- ④ Several times if images are very big we need to pool after each Conv, but when feature becomes small \Rightarrow less frequent Pooling.

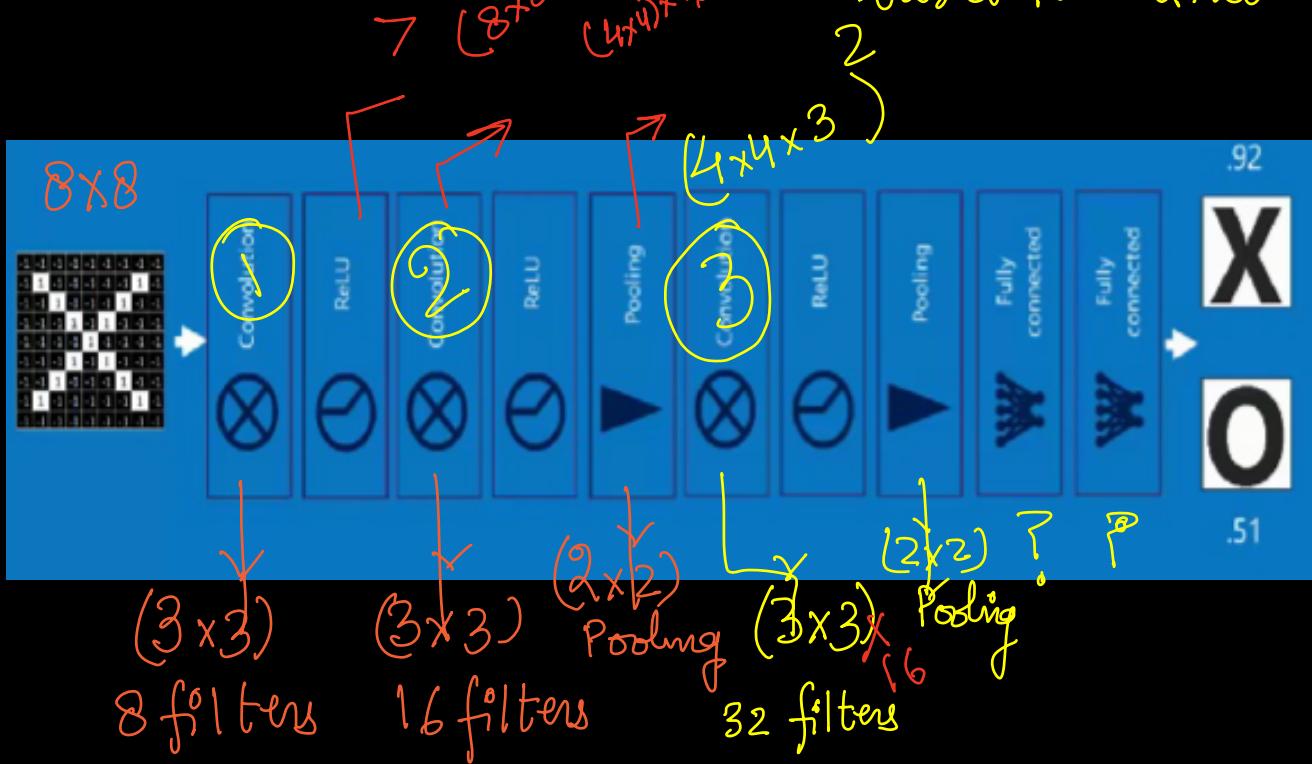
C

Deep Stacking

⇒ This will give

us our first

CNN based Neural Net.



⊗ CNN exploits spatial relationship
that got disturbed by FCN &
cannot be utilized in 2D.

CNN can ask

2D Questions

FCN was asking

1D Questions

*) Pooling allows aggregation by feature reduction (Max-Pool)

local \Rightarrow Global estimates

\hookrightarrow At some point (feature size)

global estimates contains the global image statistics like:

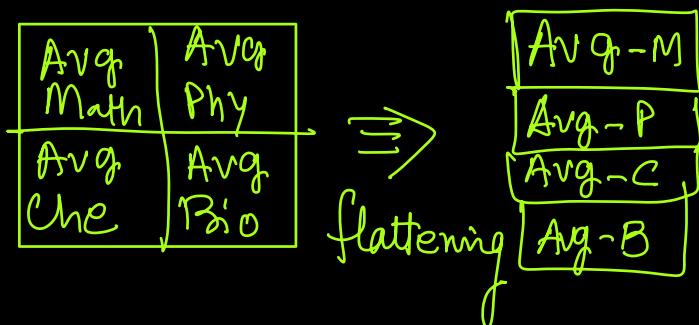
average of (Color, edges,

They will not have

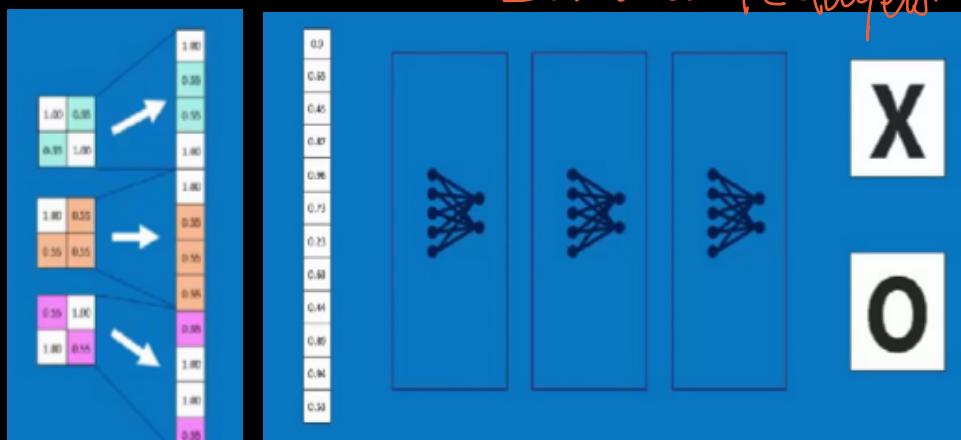
Spatially important relationships

Variation of certain type, Average

occurrence of a pattern)



Flatten & apply several FC layers.



Hyper-parameters

a) Convolutional layers

- A.F → Stride in
x & y
- # of filter
- Size of filter

b) Pooling

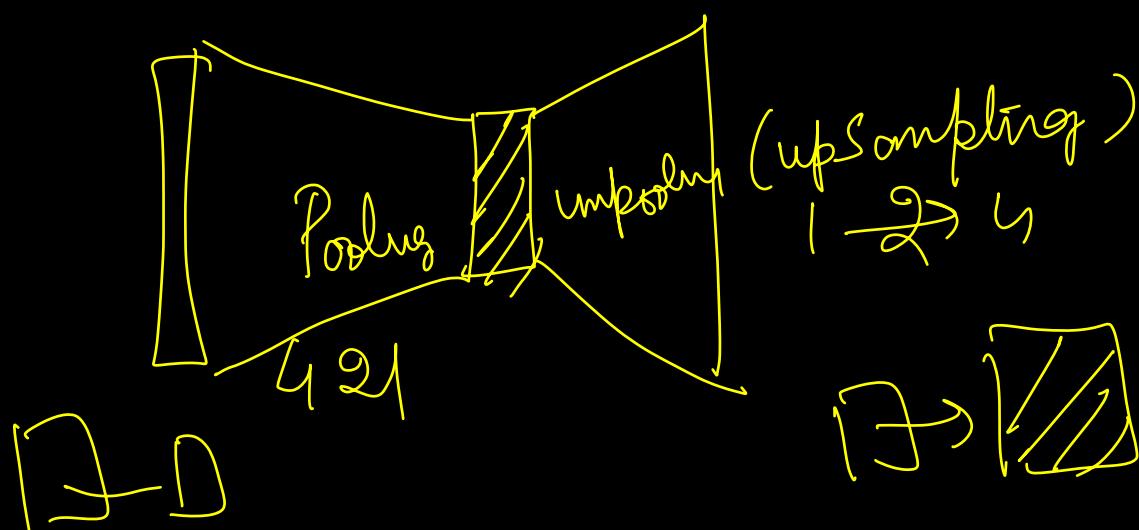
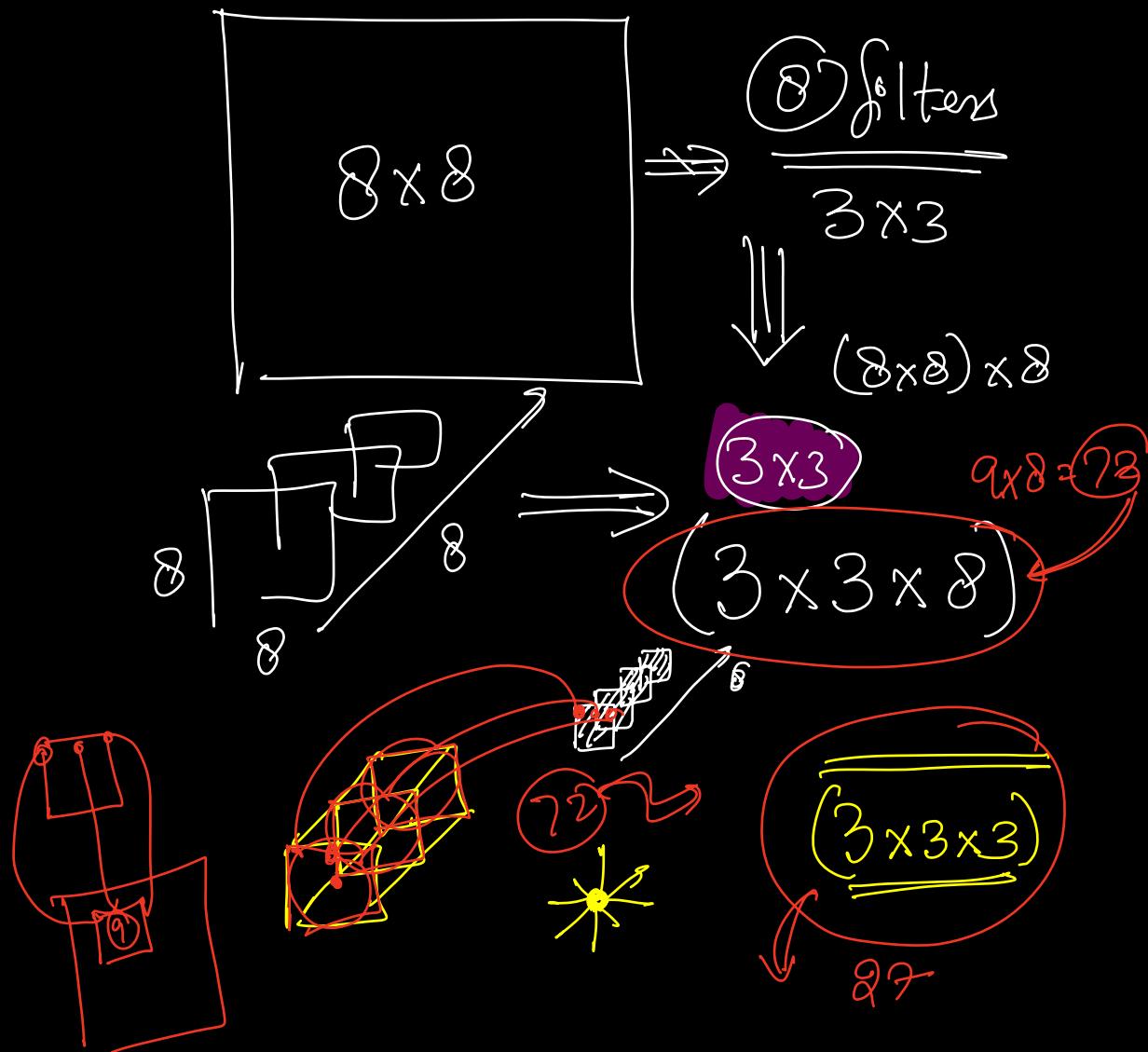
- Window Size

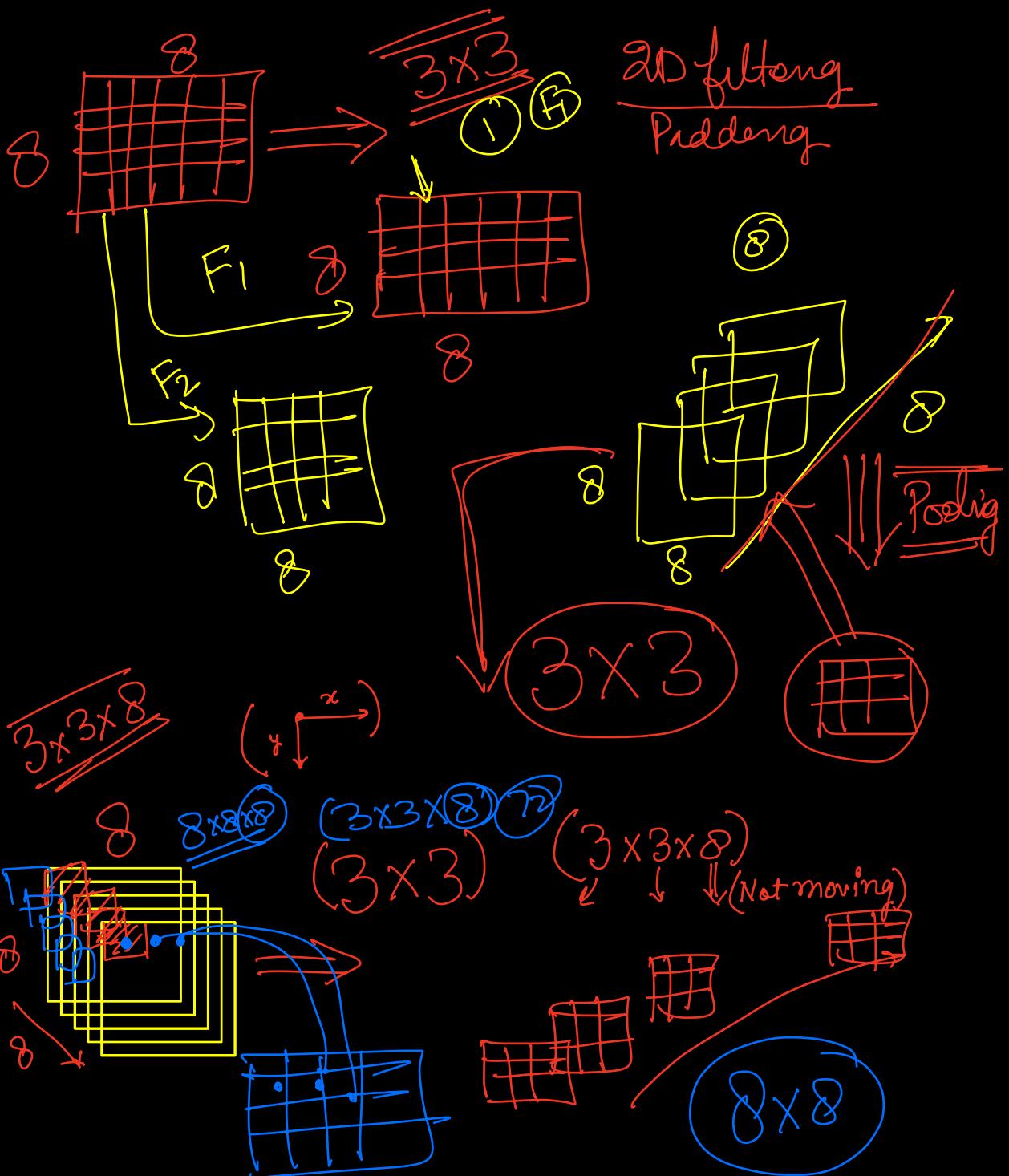
- Window Stride in
x & y

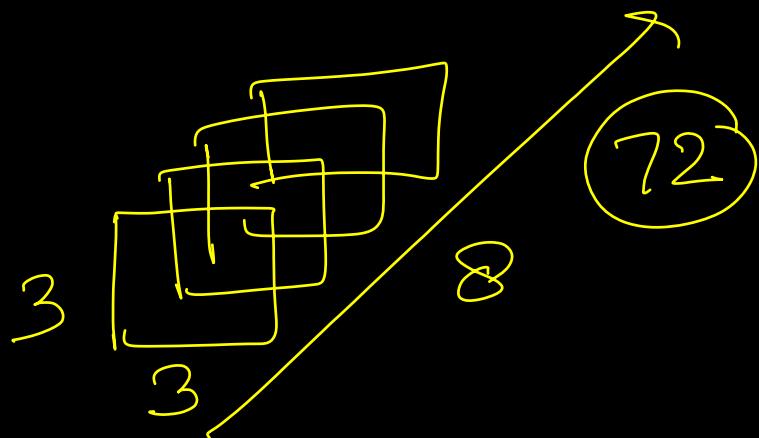
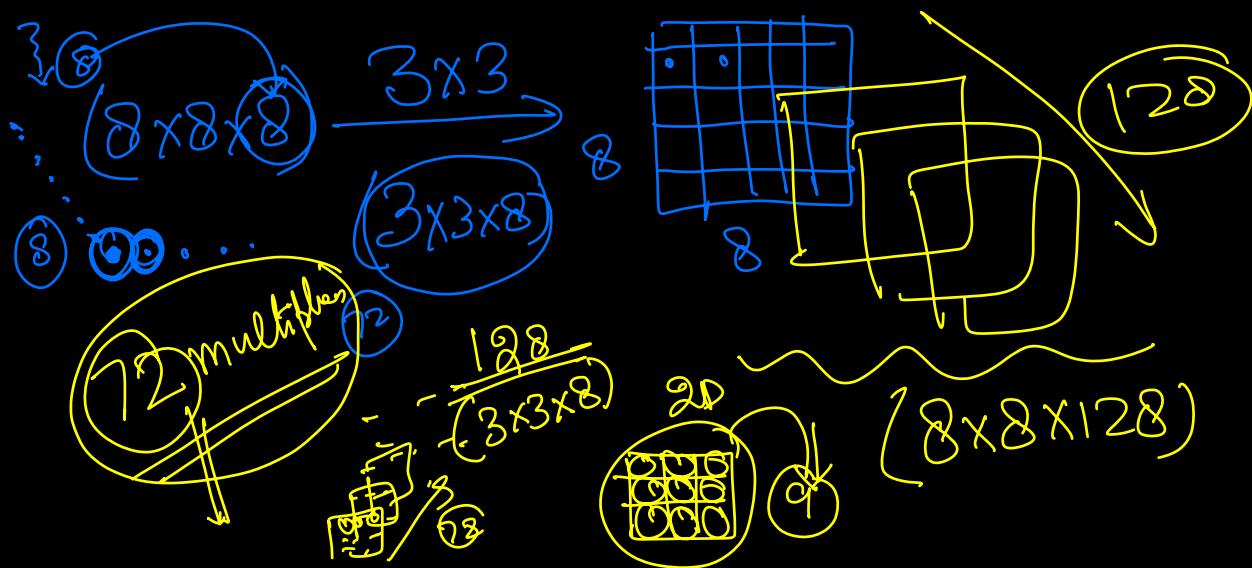
c) Full Connected

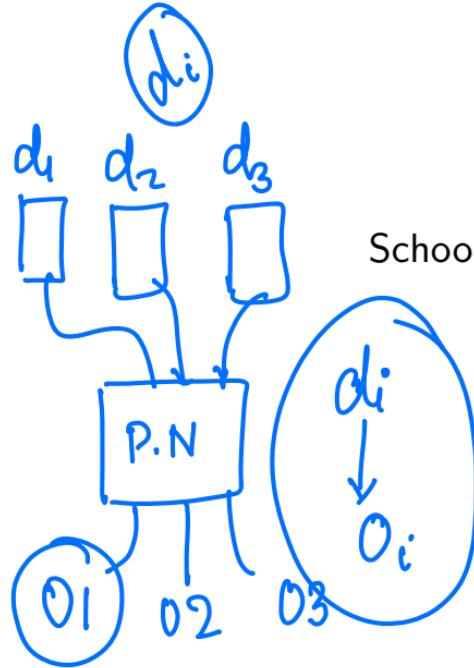
- # of layers

- # of Neurons in each layer.









Recurrent Neural Networks

No Temporal Relationship

Aditya Nigam, Assistant Professor

School of Computing and Electrical Engineering (SCEE)

Indian Institute of Technology, Mandi

<http://faculty.iitmandi.ac.in/~aditya/>

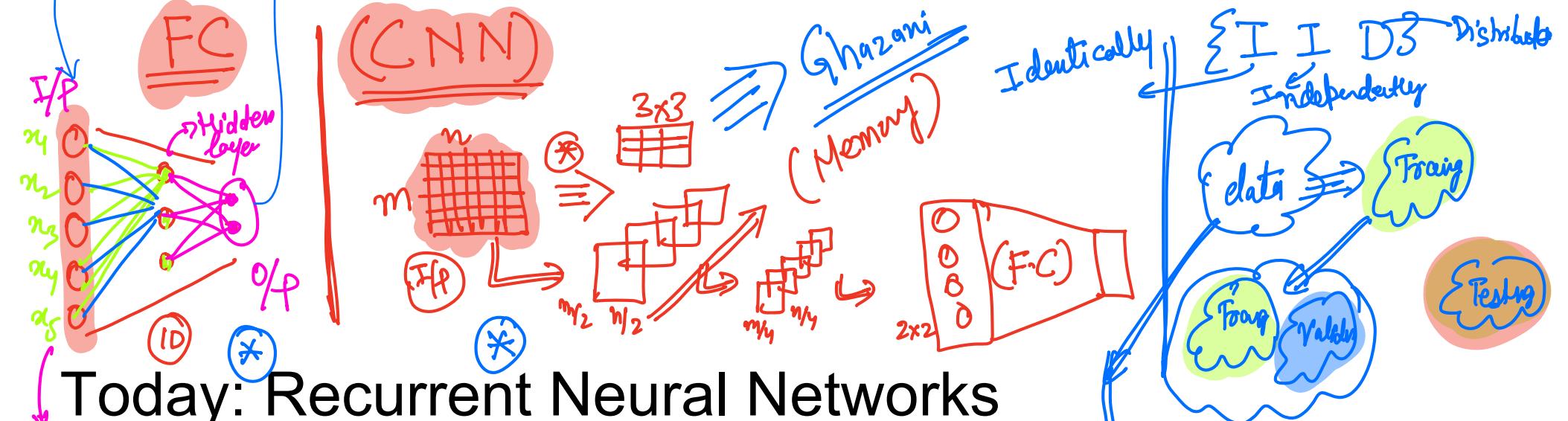
Email : aditya@iitmandi.ac.in



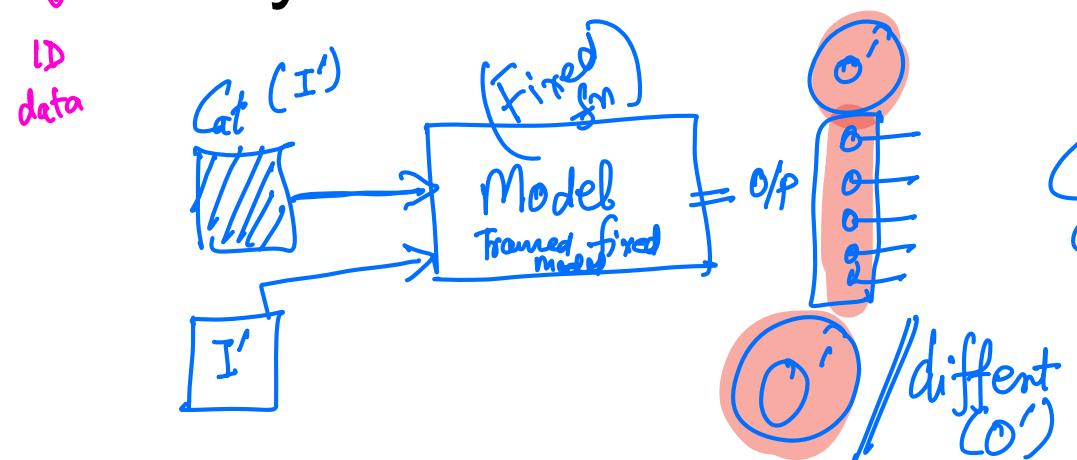
Presentation at IWADL@IITMandi

(*Slides : CS231n by Justin)

<https://www.youtube.com/watch?v=6niqTuYFZLQ&t=291s>

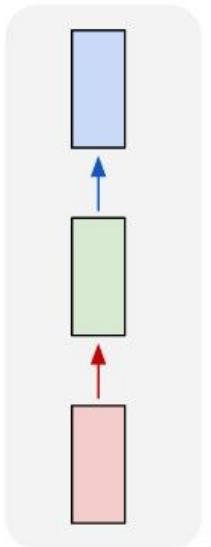


Today: Recurrent Neural Networks



“Vanilla” Neural Network

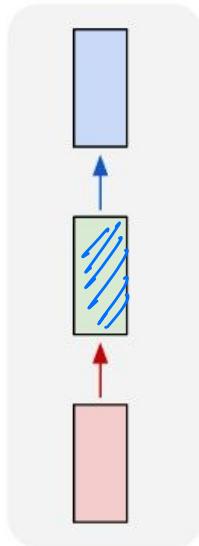
one to one



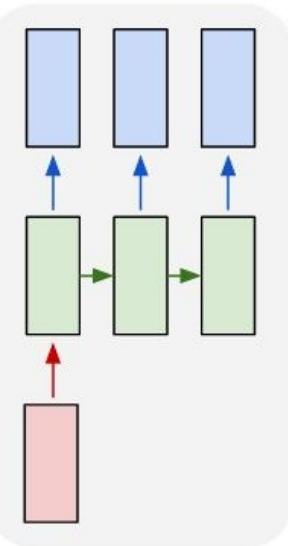
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences

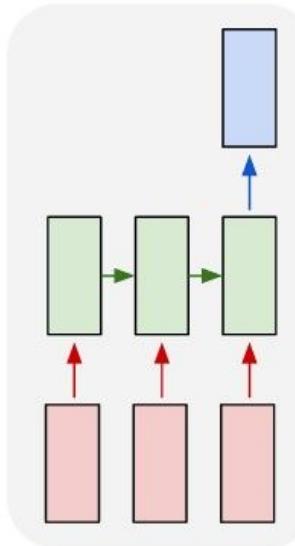
one to one



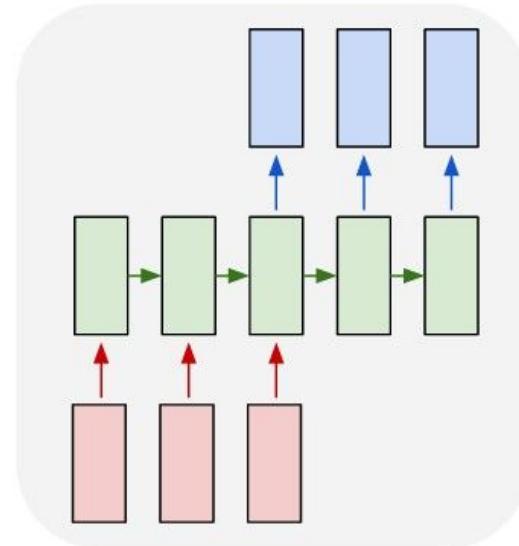
one to many



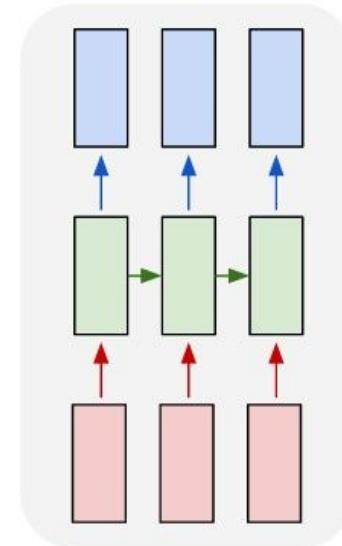
many to one



many to many



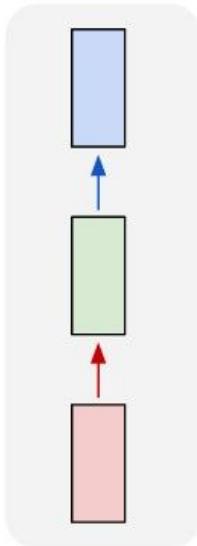
many to many



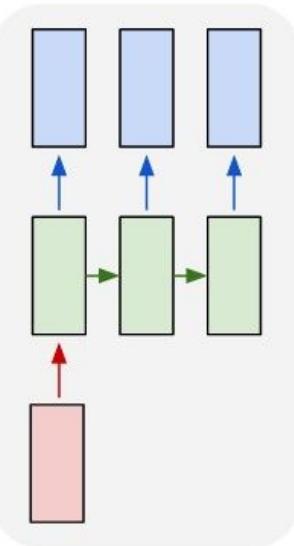
→
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences

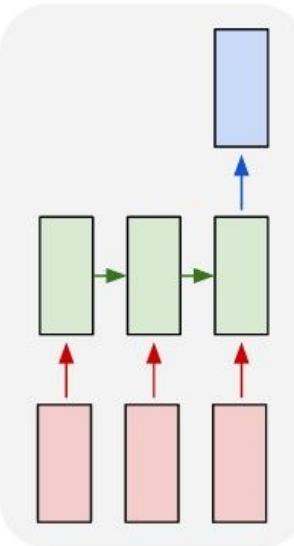
one to one



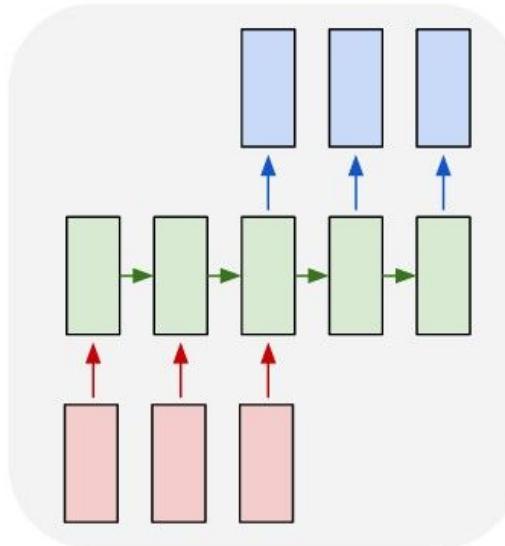
one to many



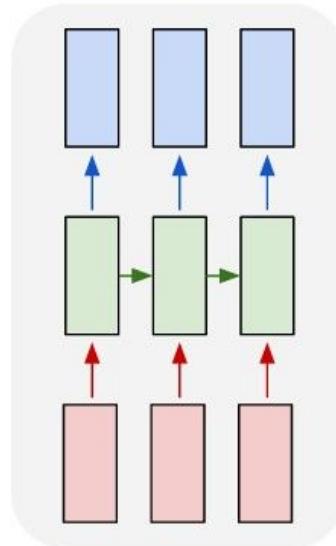
many to one



many to many



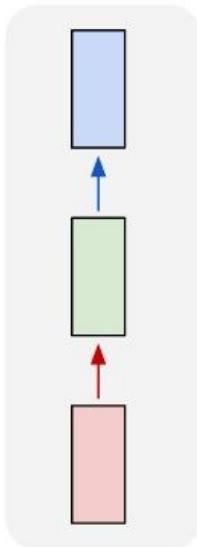
many to many



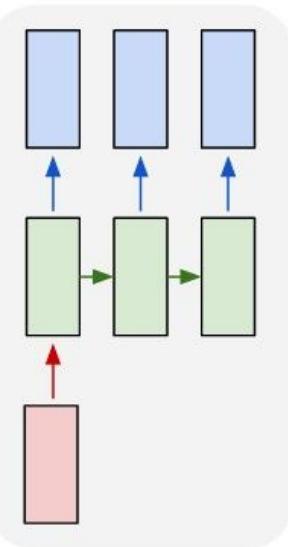
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences

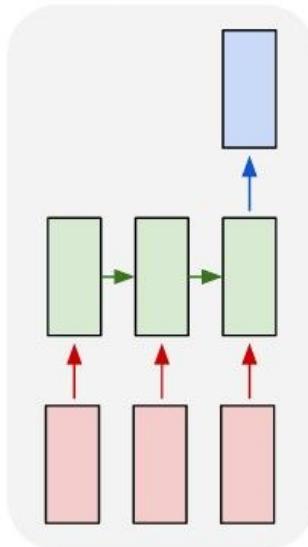
one to one



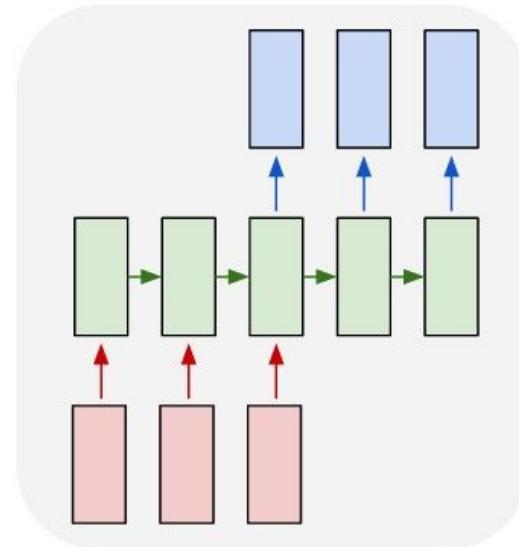
one to many



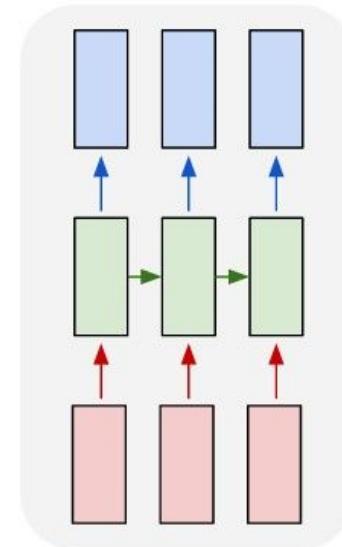
many to one



many to many



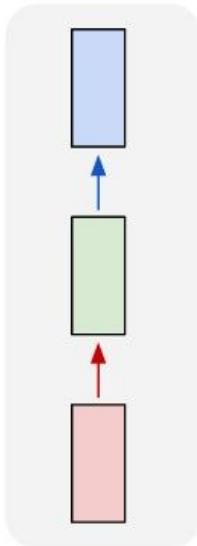
many to many



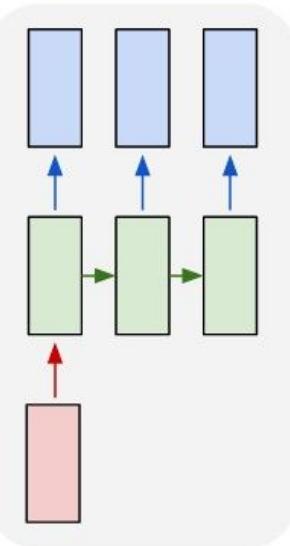
↑
e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks: Process Sequences

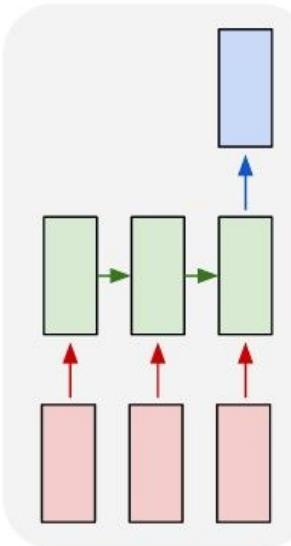
one to one



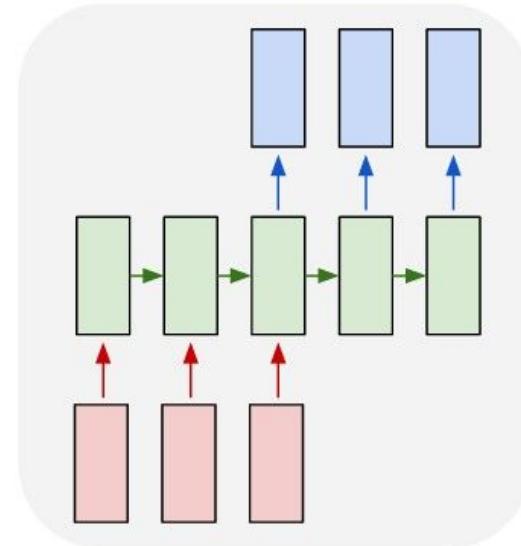
one to many



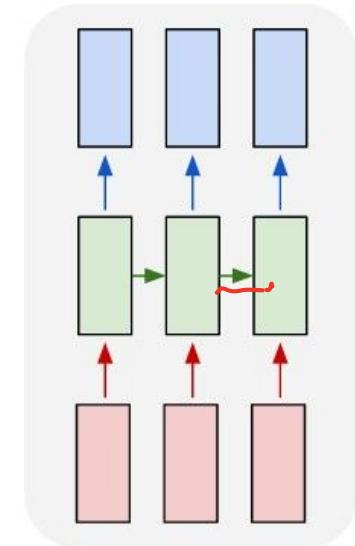
many to one



many to many



many to many



e.g. **Video classification on frame level**

Sequential Processing of Non-Sequence Data

Classify images by taking a series of “glimpses”



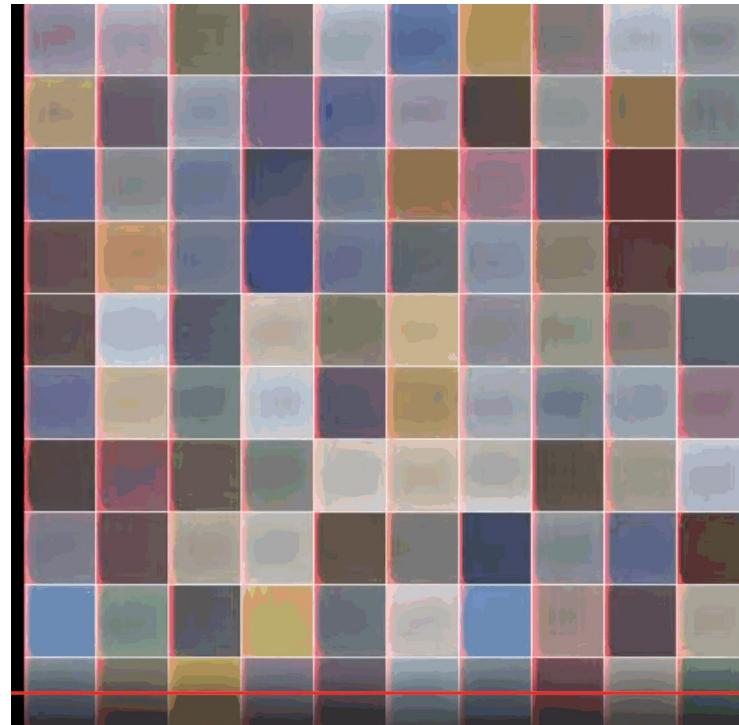
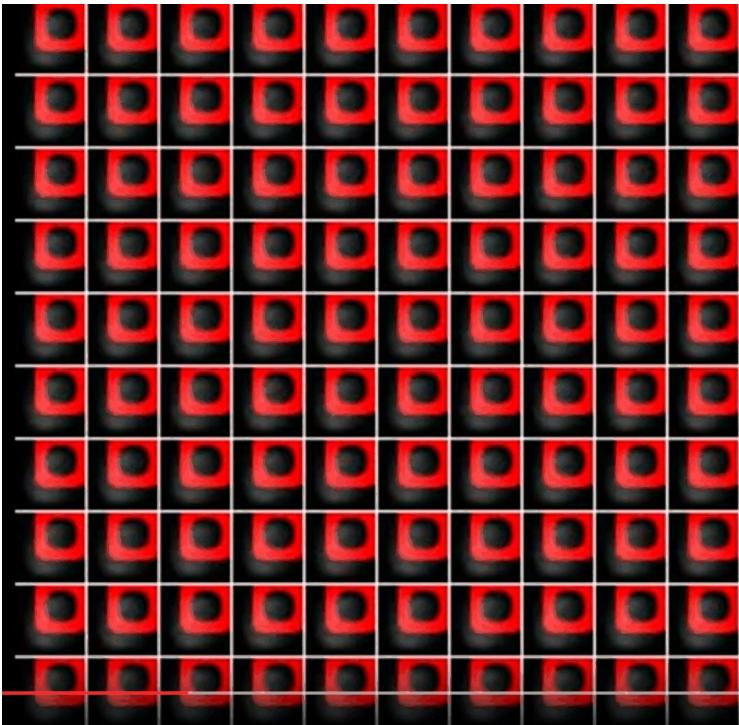
Ba, Mnih, and Kavukcuoglu, "Multiple Object Recognition with Visual Attention", ICLR 2015.

Gregor et al, "DRAW: A Recurrent Neural Network For Image Generation", ICML 2015

Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.

Sequential Processing of Non-Sequence Data

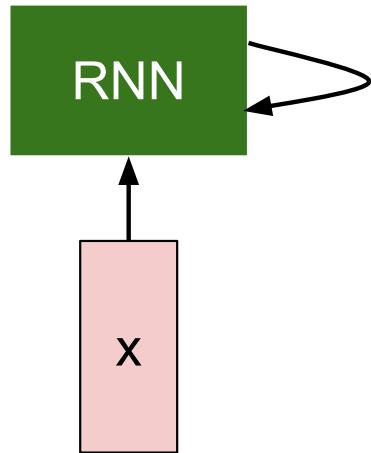
Generate images one piece at a time!



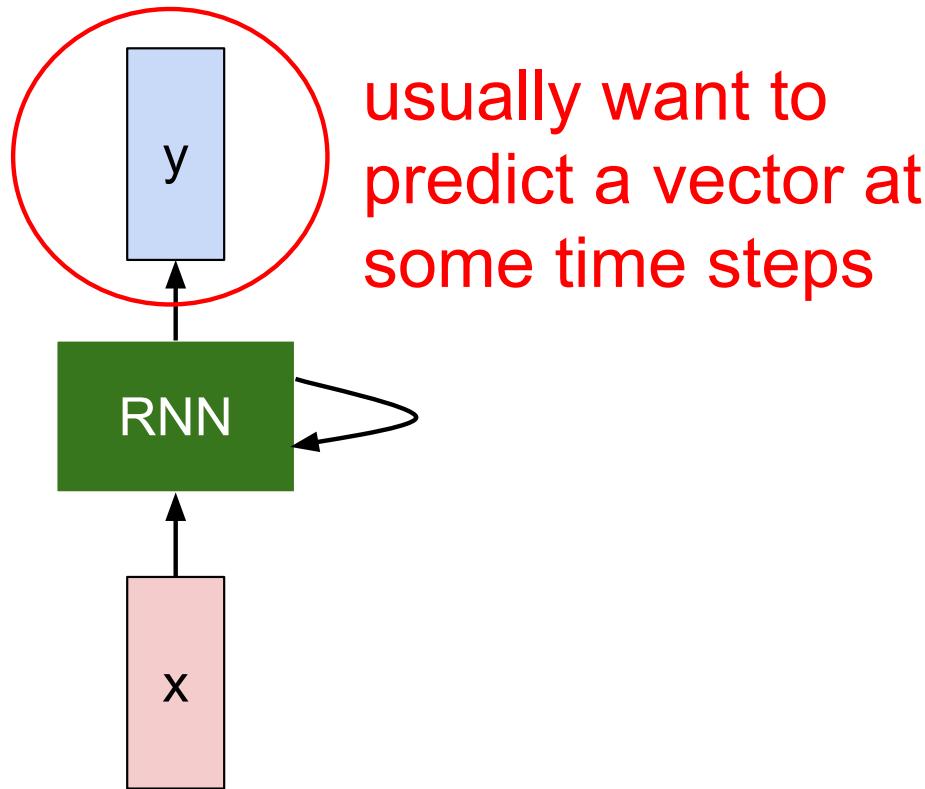
Gregor et al, "DRAW: A Recurrent Neural Network For image Generation", ICML 2015

Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.

Recurrent Neural Network



Recurrent Neural Network

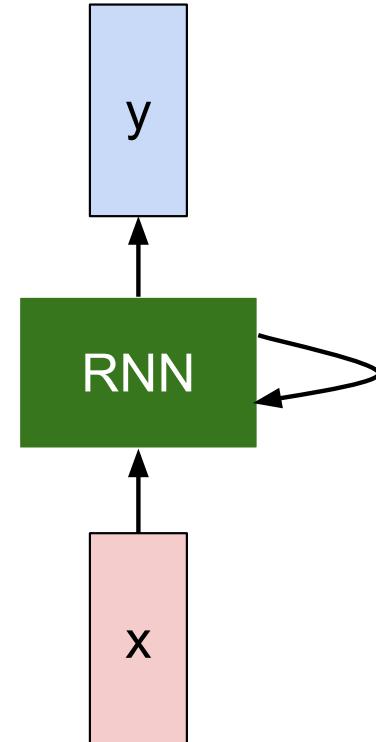


Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state old state input vector at
some function some time step
with parameters W

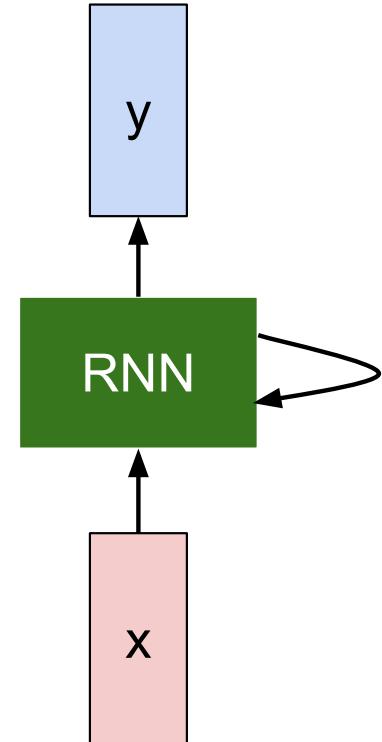


Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

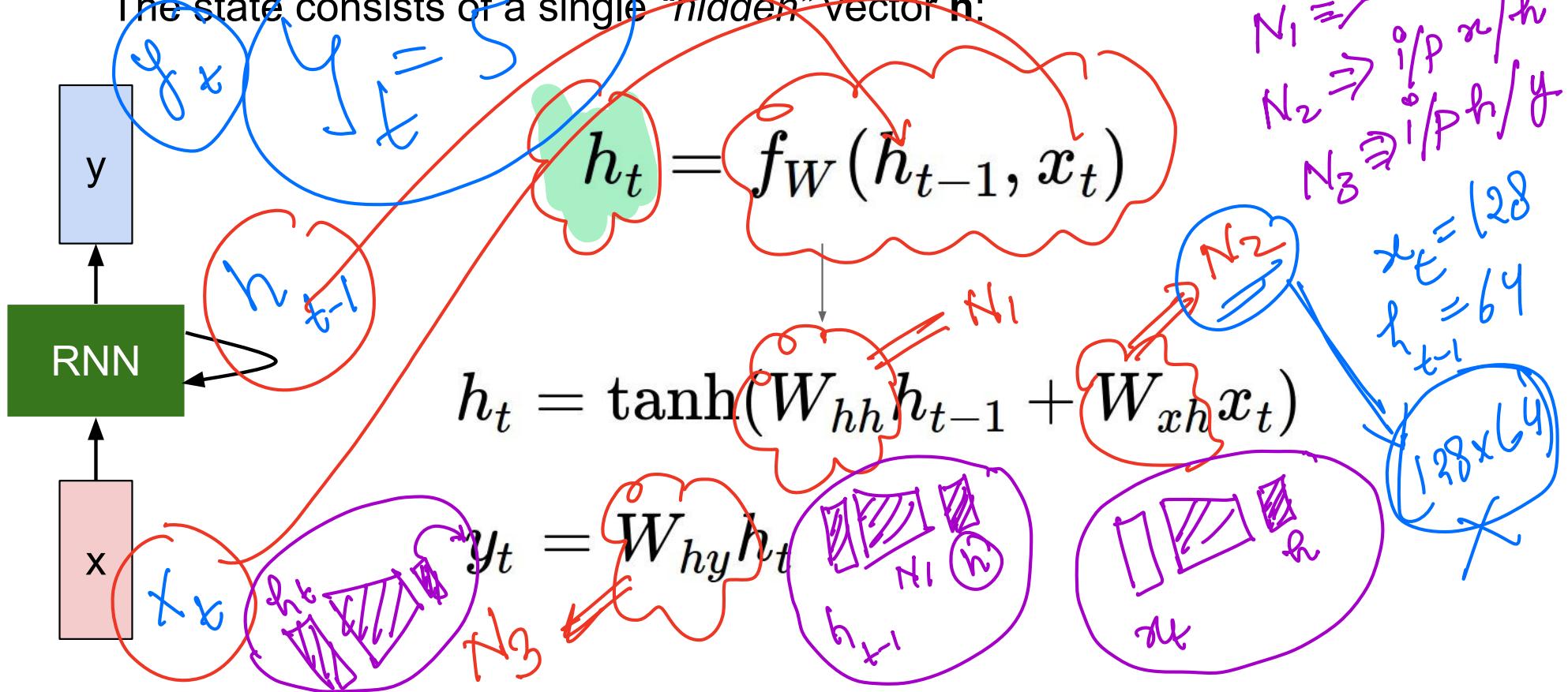
$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.

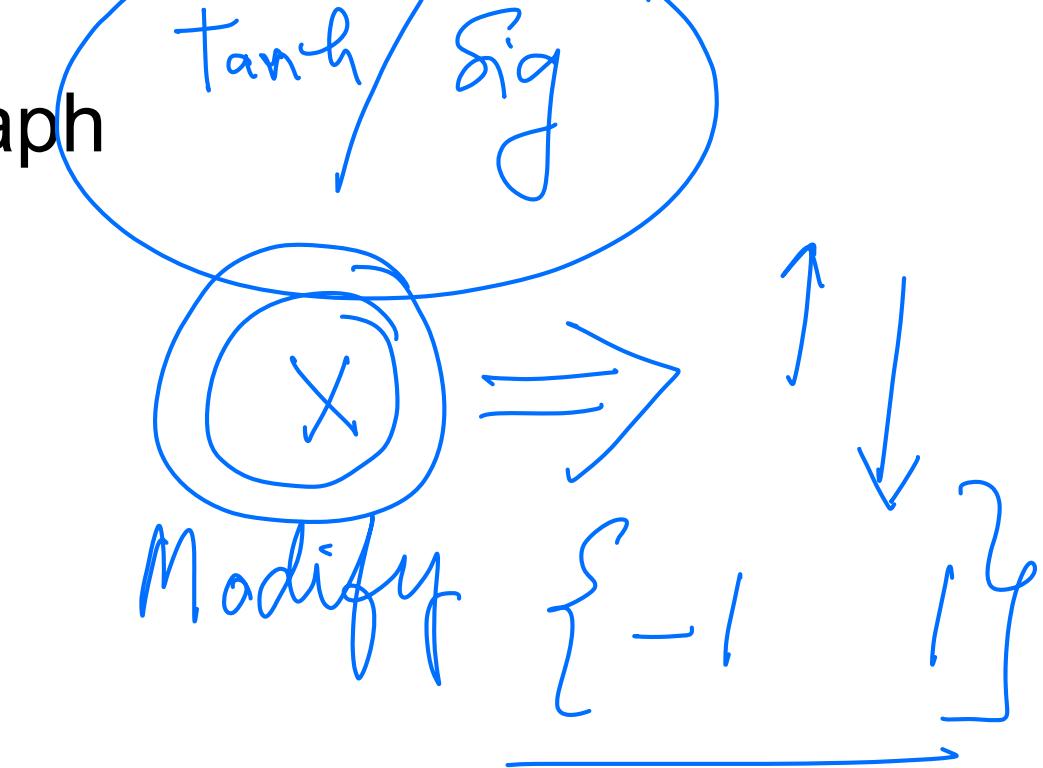
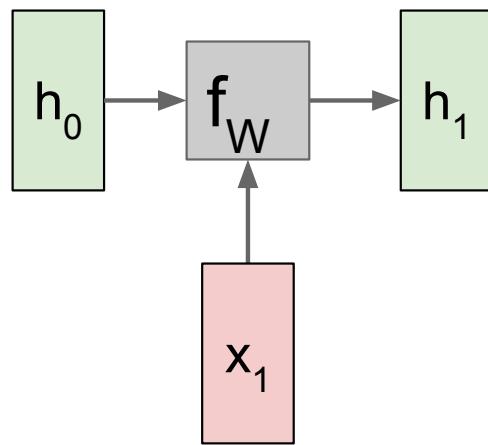


(Vanilla) Recurrent Neural Network

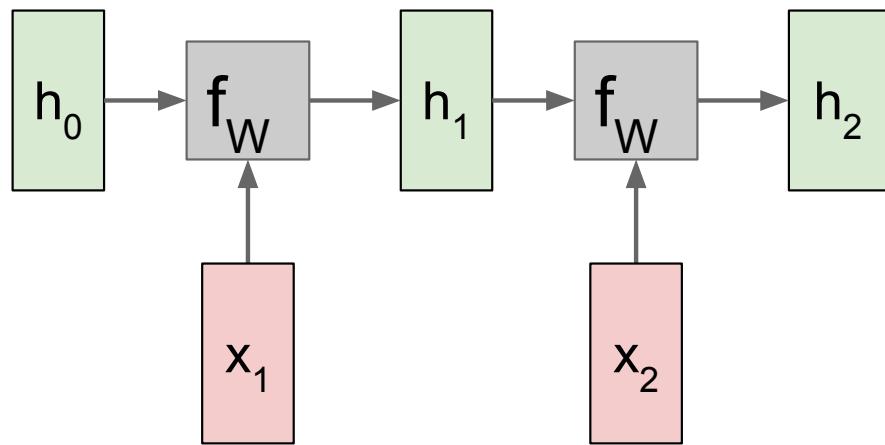
The state consists of a single “hidden” vector h :



RNN: Computational Graph

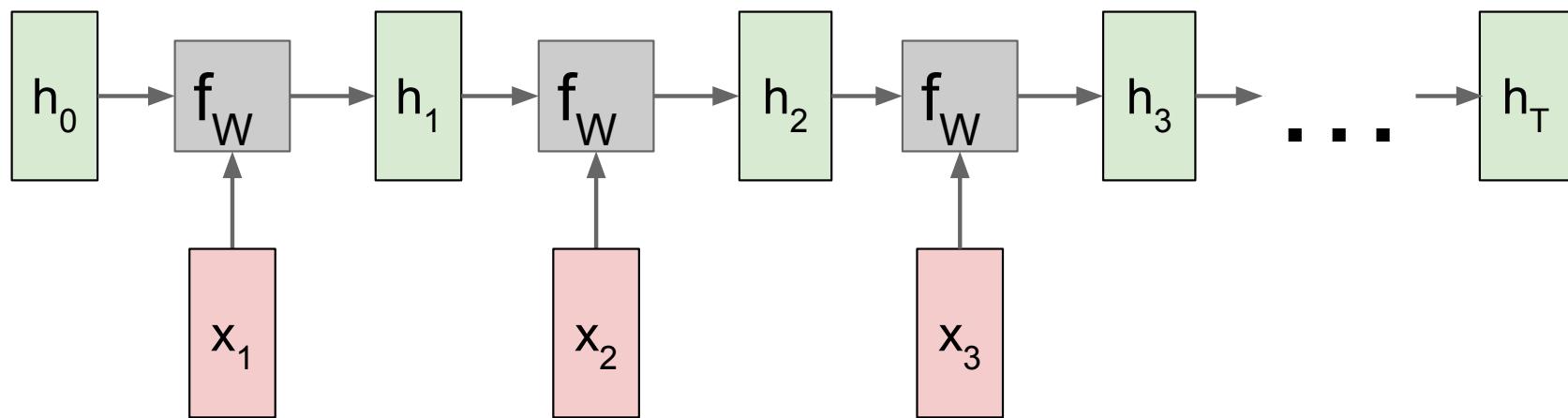


RNN: Computational Graph



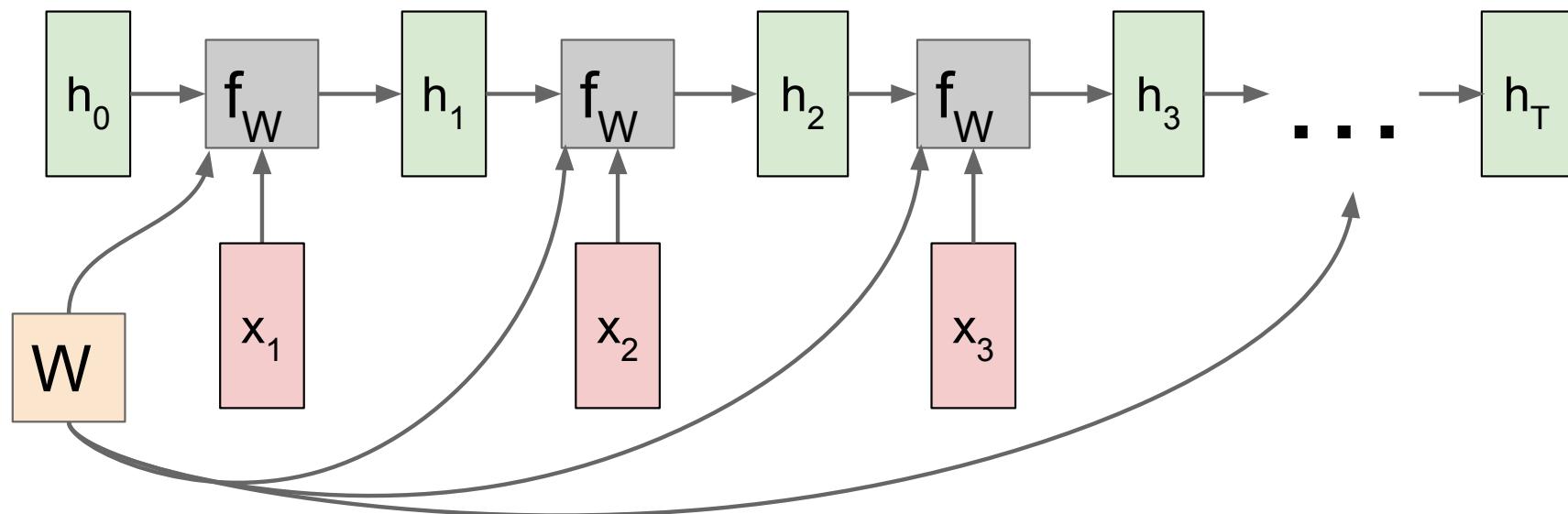
RNN: Computational Graph

UNROLLING of RNN

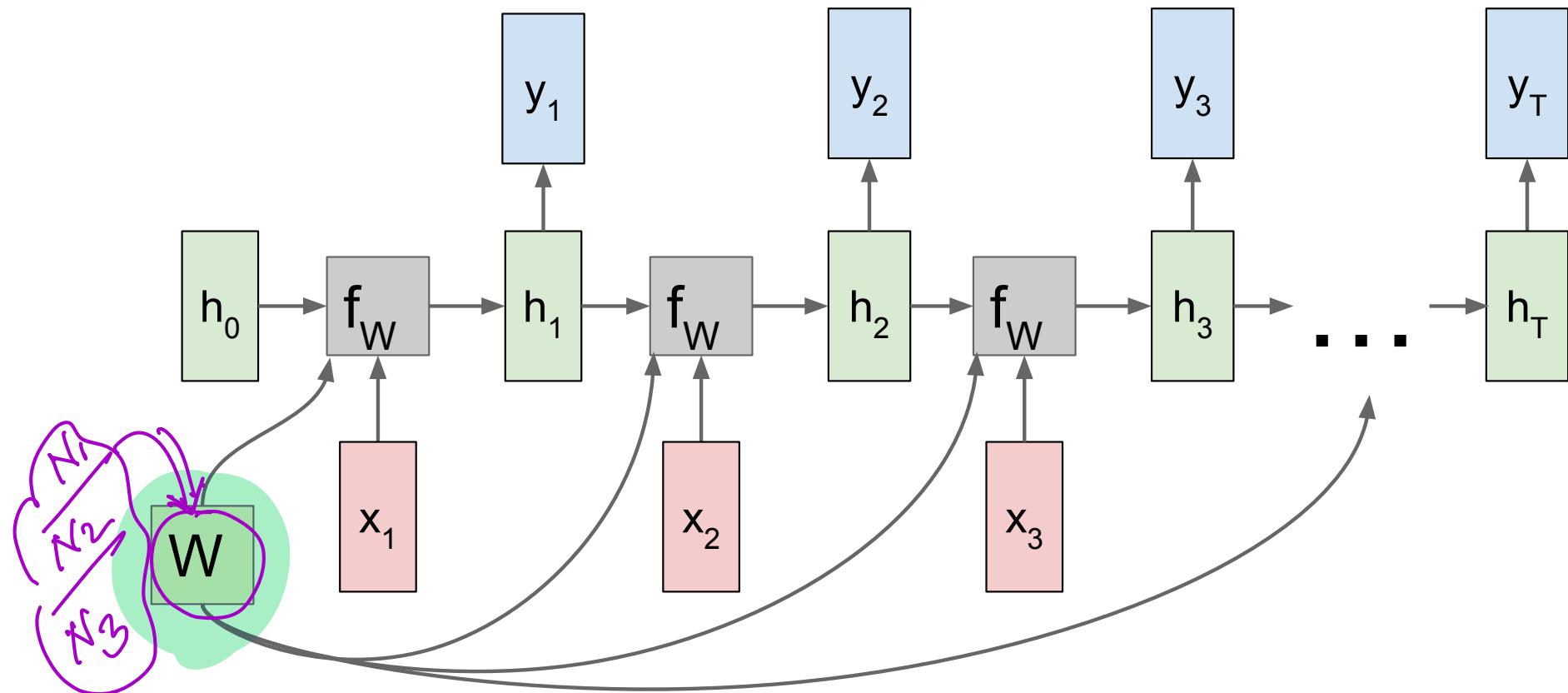


RNN: Computational Graph

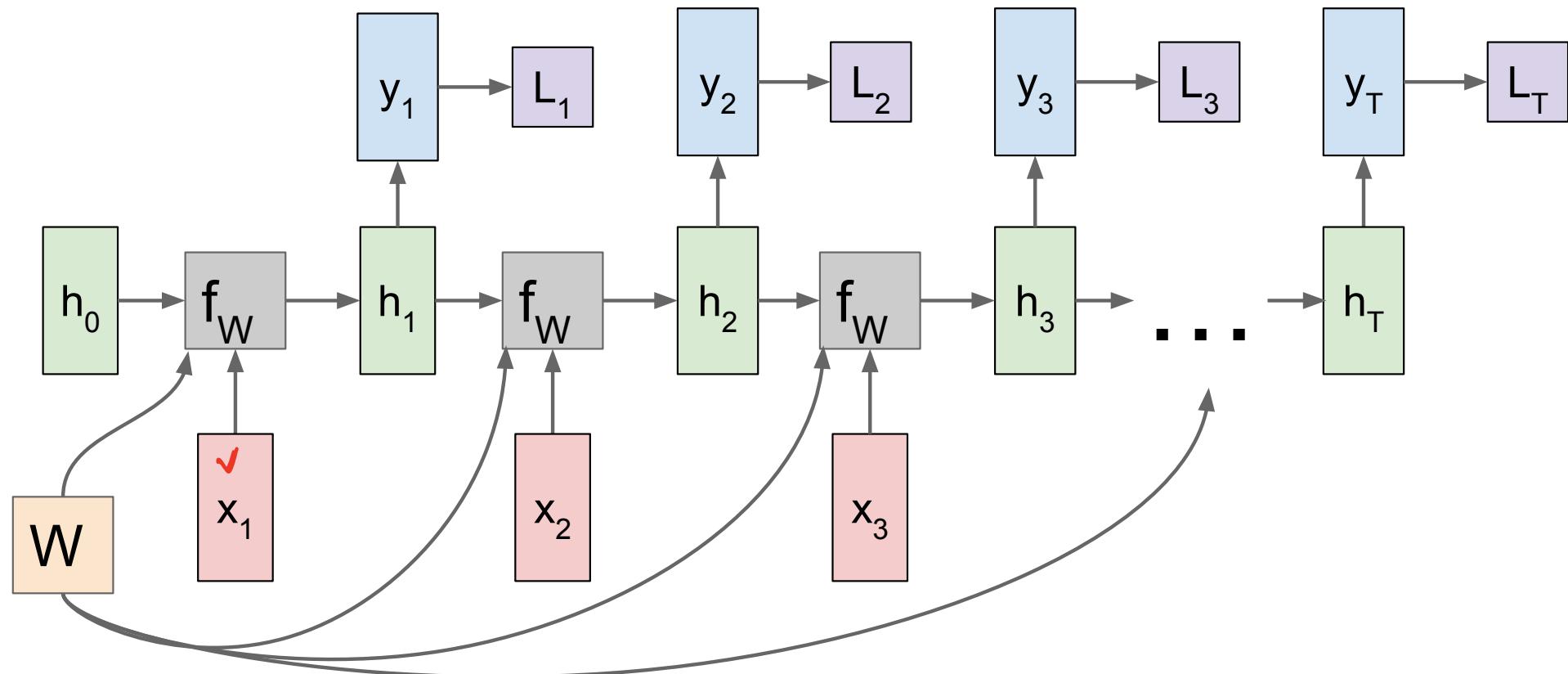
Re-use the same weight matrix at every time-step



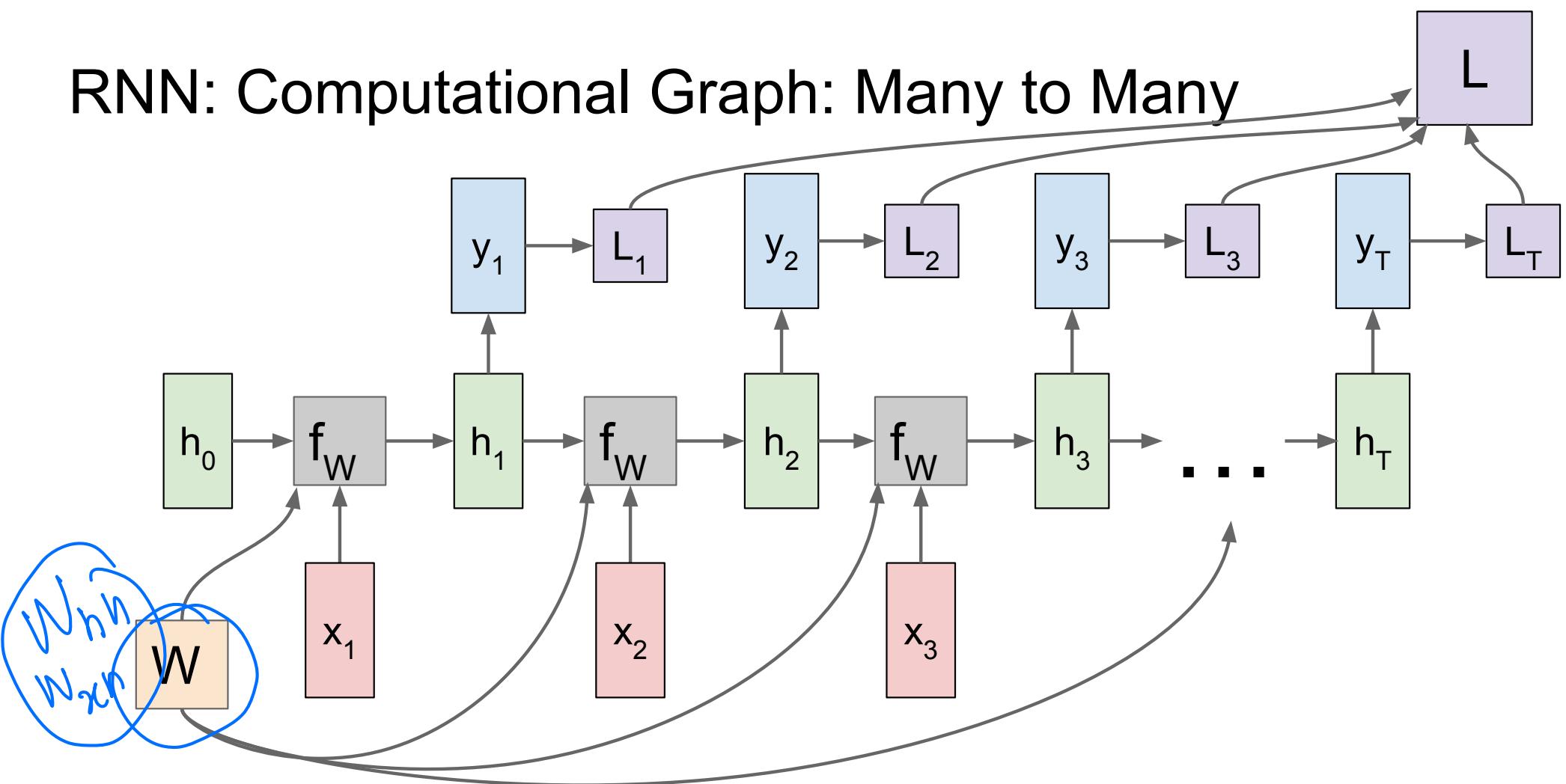
RNN: Computational Graph: Many to Many



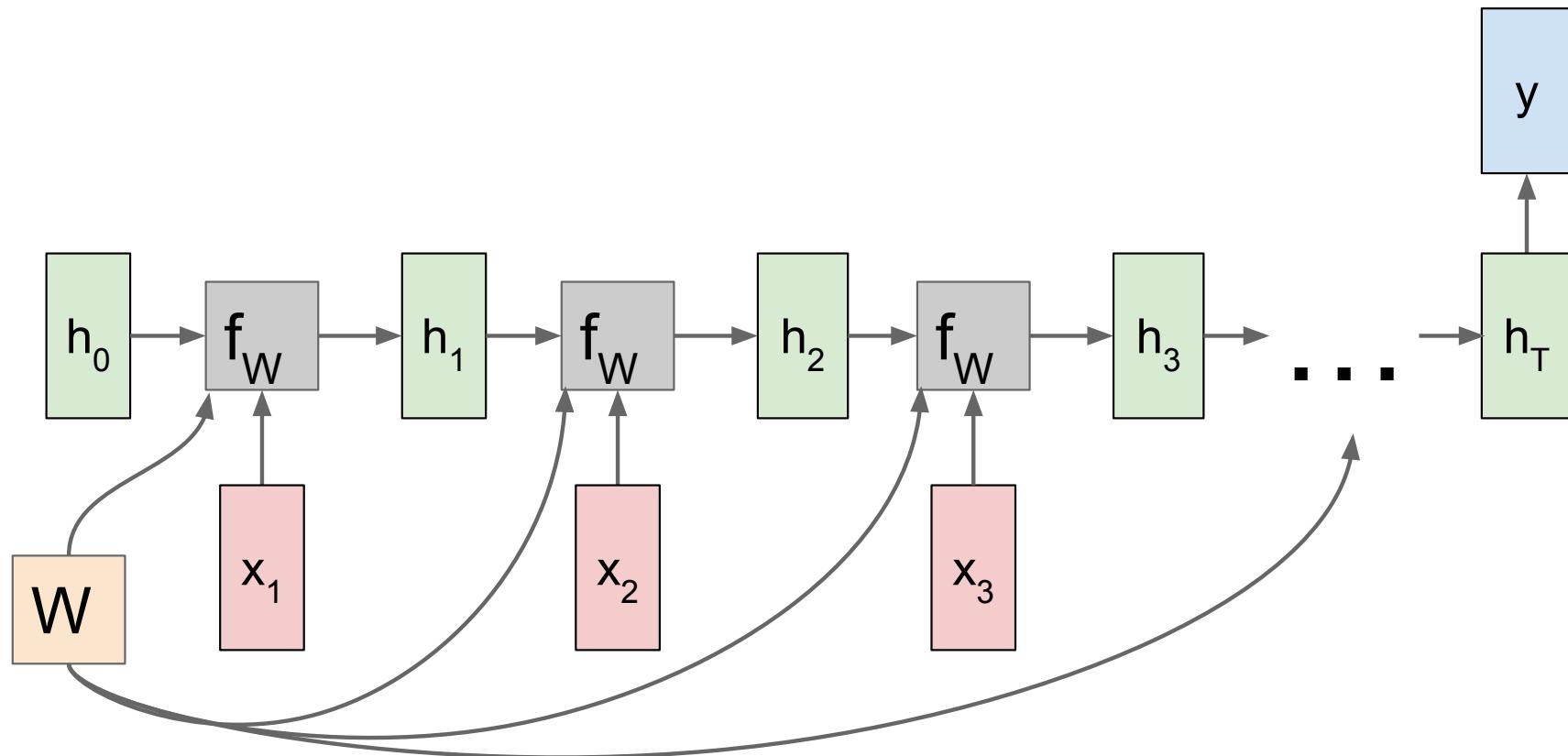
RNN: Computational Graph: Many to Many



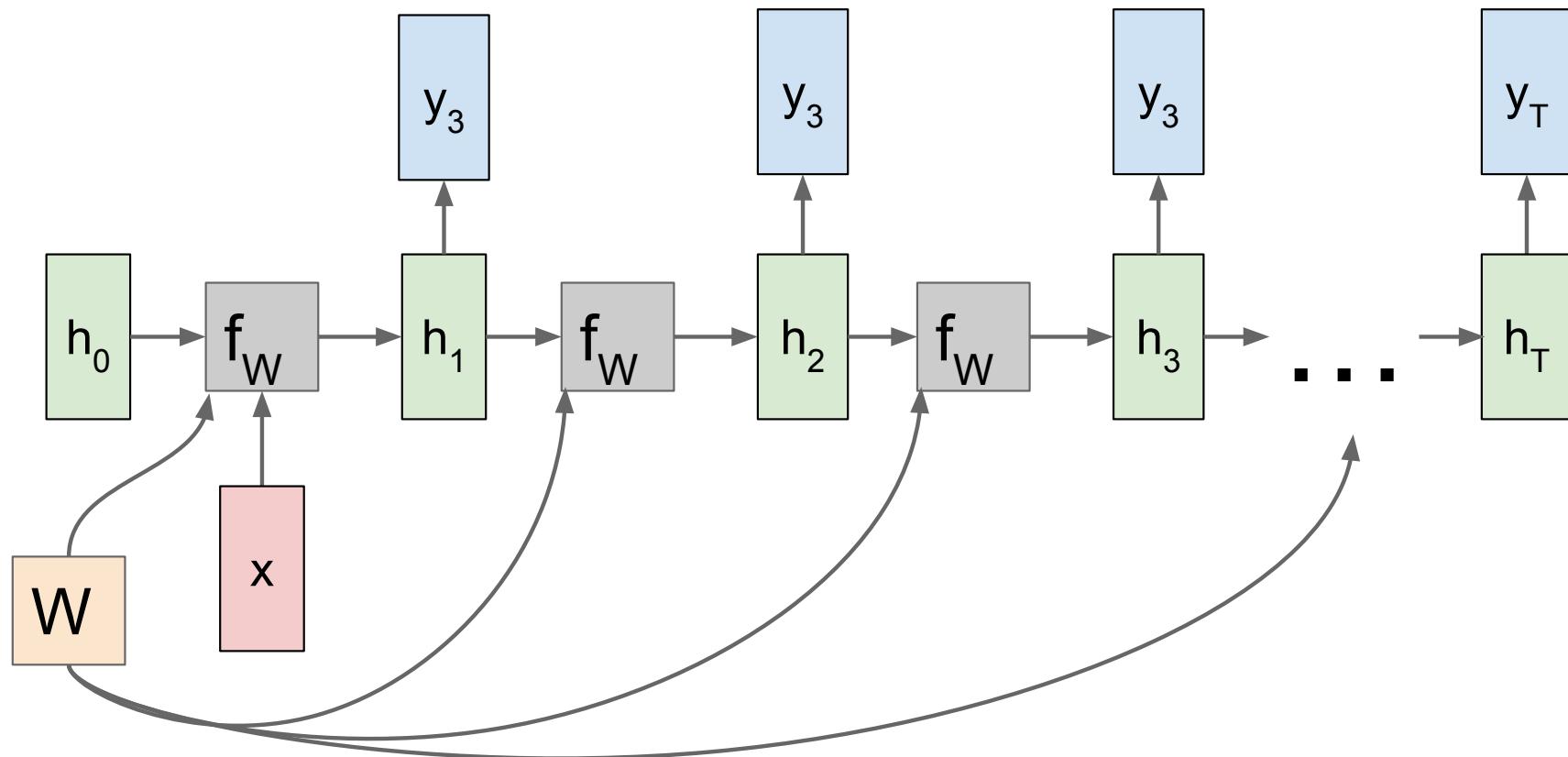
RNN: Computational Graph: Many to Many



RNN: Computational Graph: Many to One

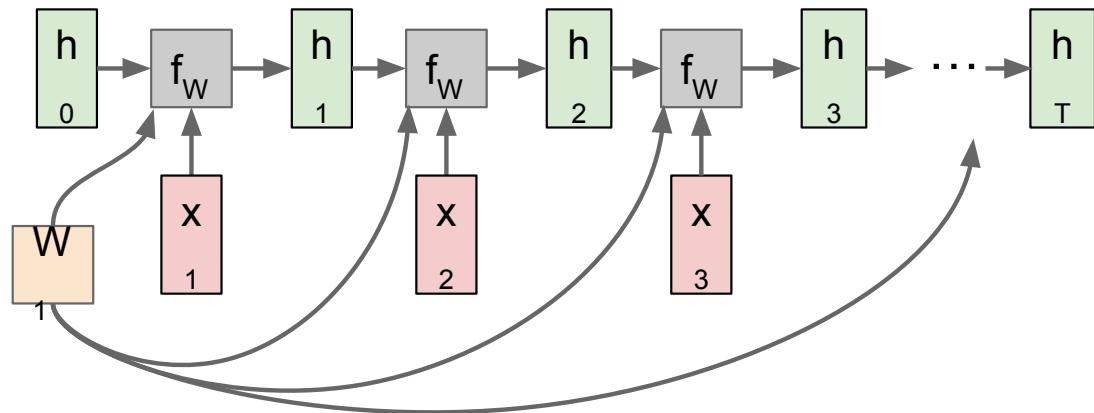


RNN: Computational Graph: One to Many



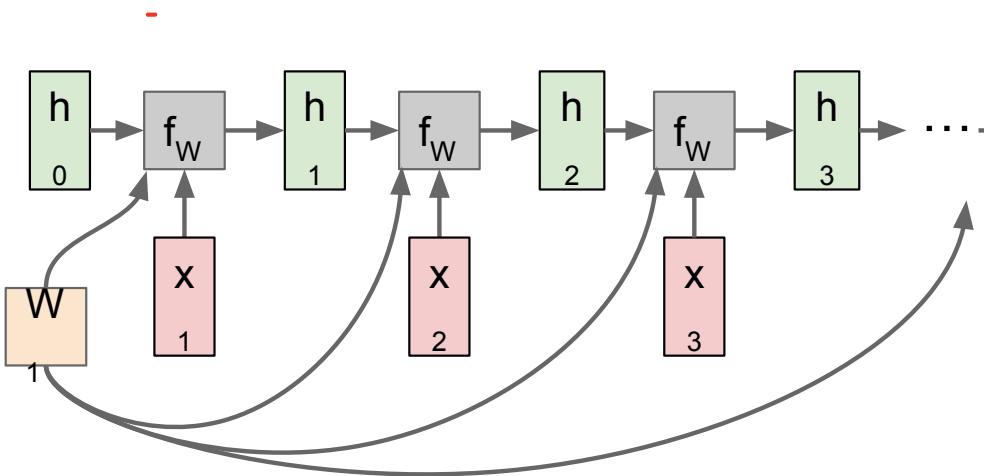
Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

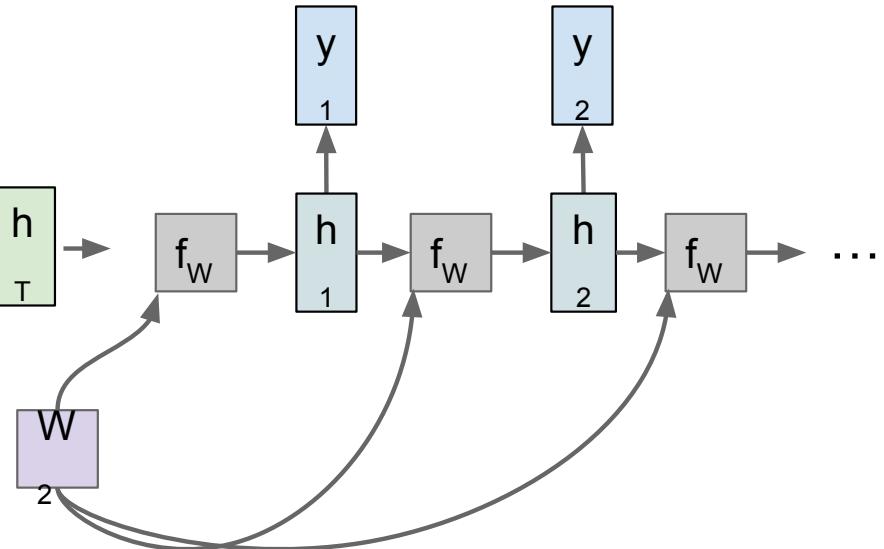


Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector



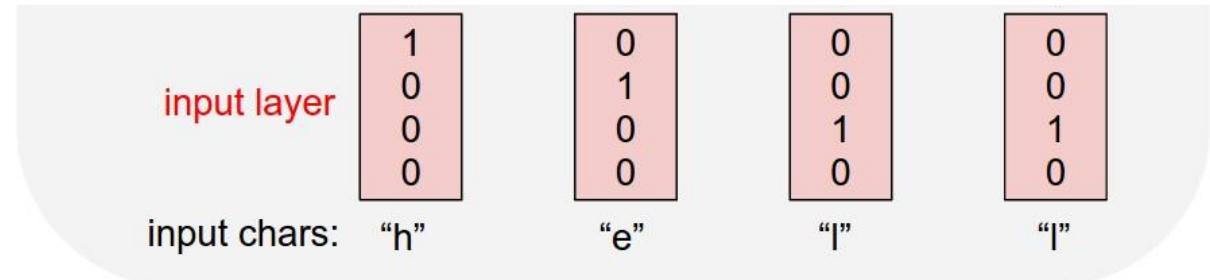
One to many: Produce output sequence from single input vector



Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

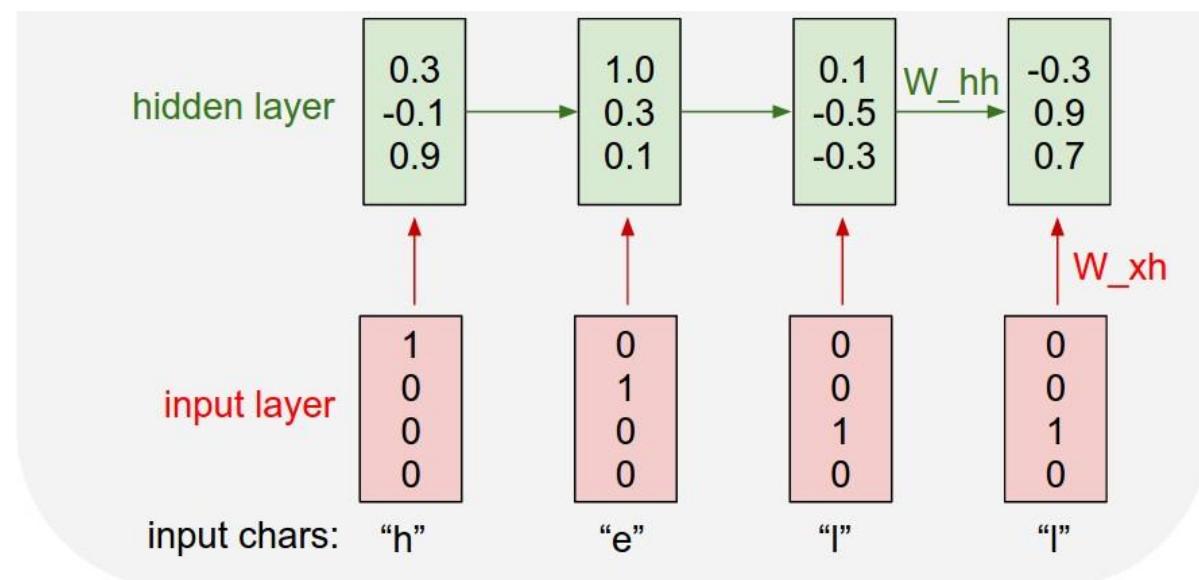


Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

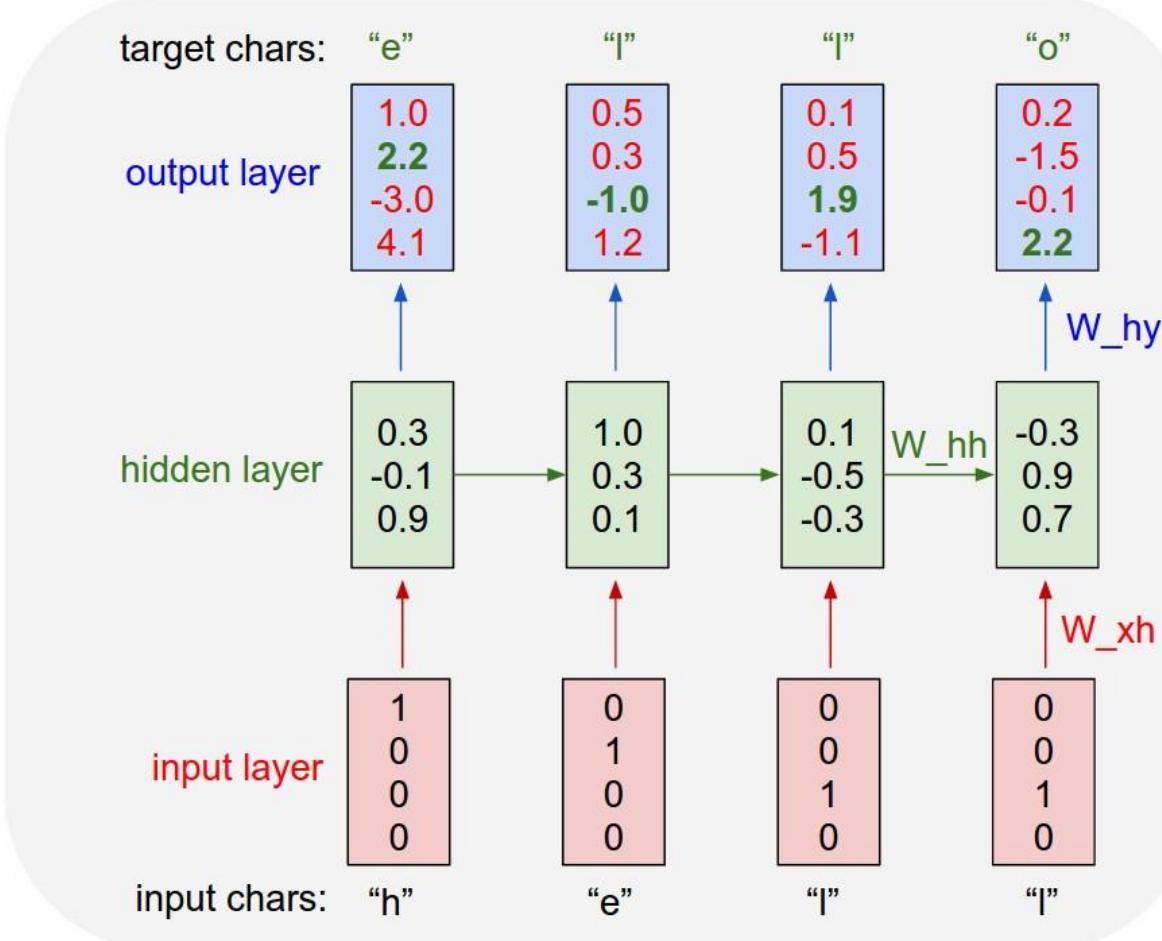
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

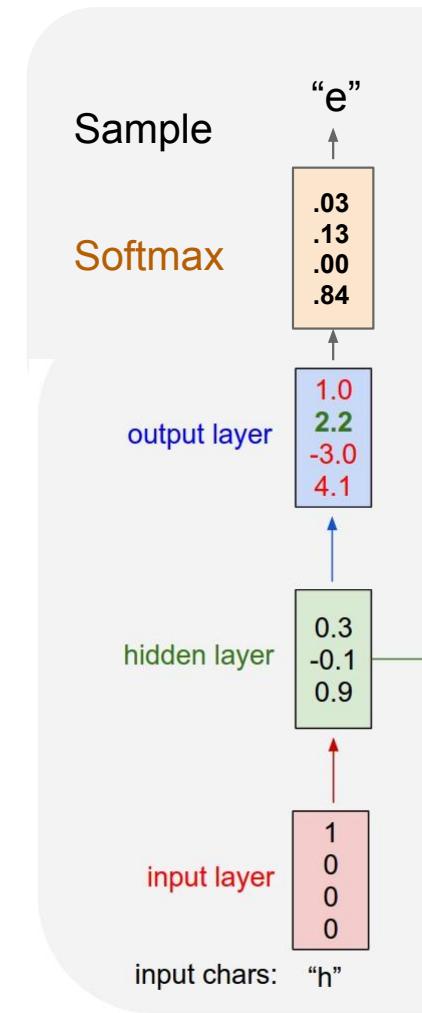
Example training
sequence:
“hello”



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

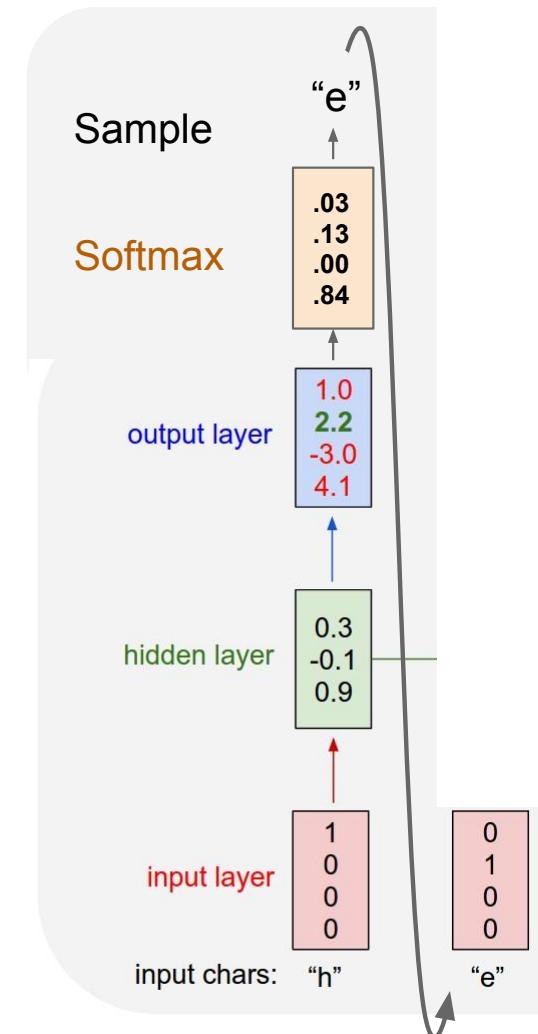
At test-time sample
characters one at a time,
feed back to model



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

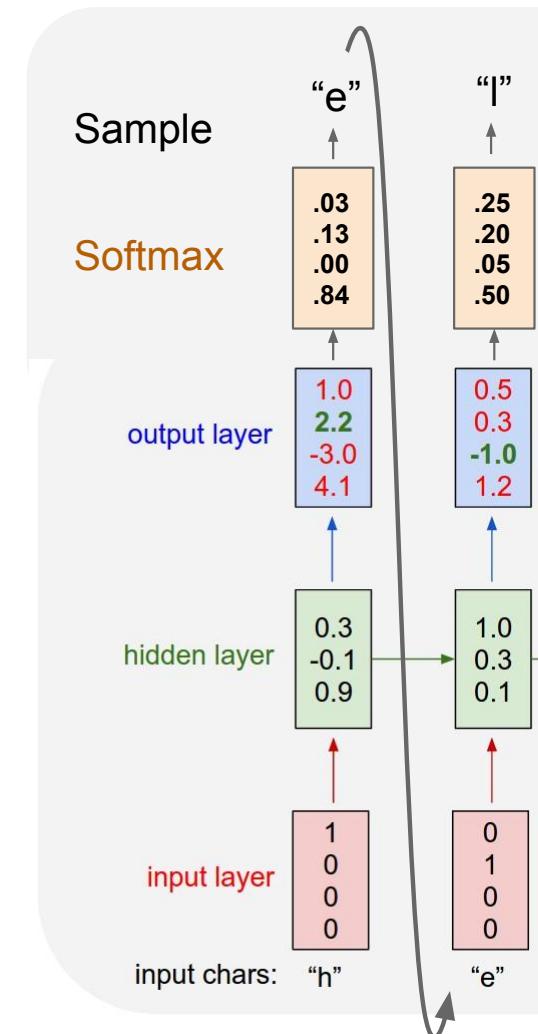
At test-time sample
characters one at a time,
feed back to model



Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

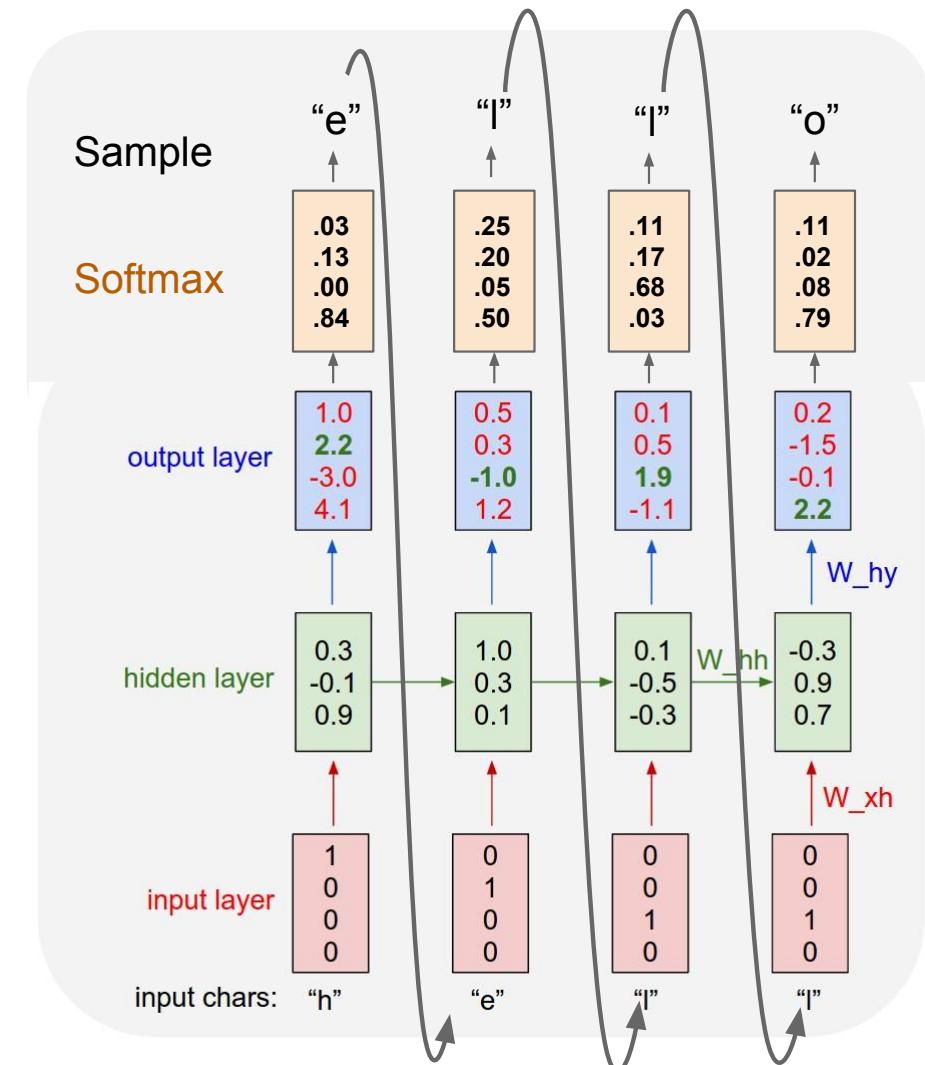
At test-time sample
characters one at a time,
feed back to model



Example: Character-level Language Model Sampling

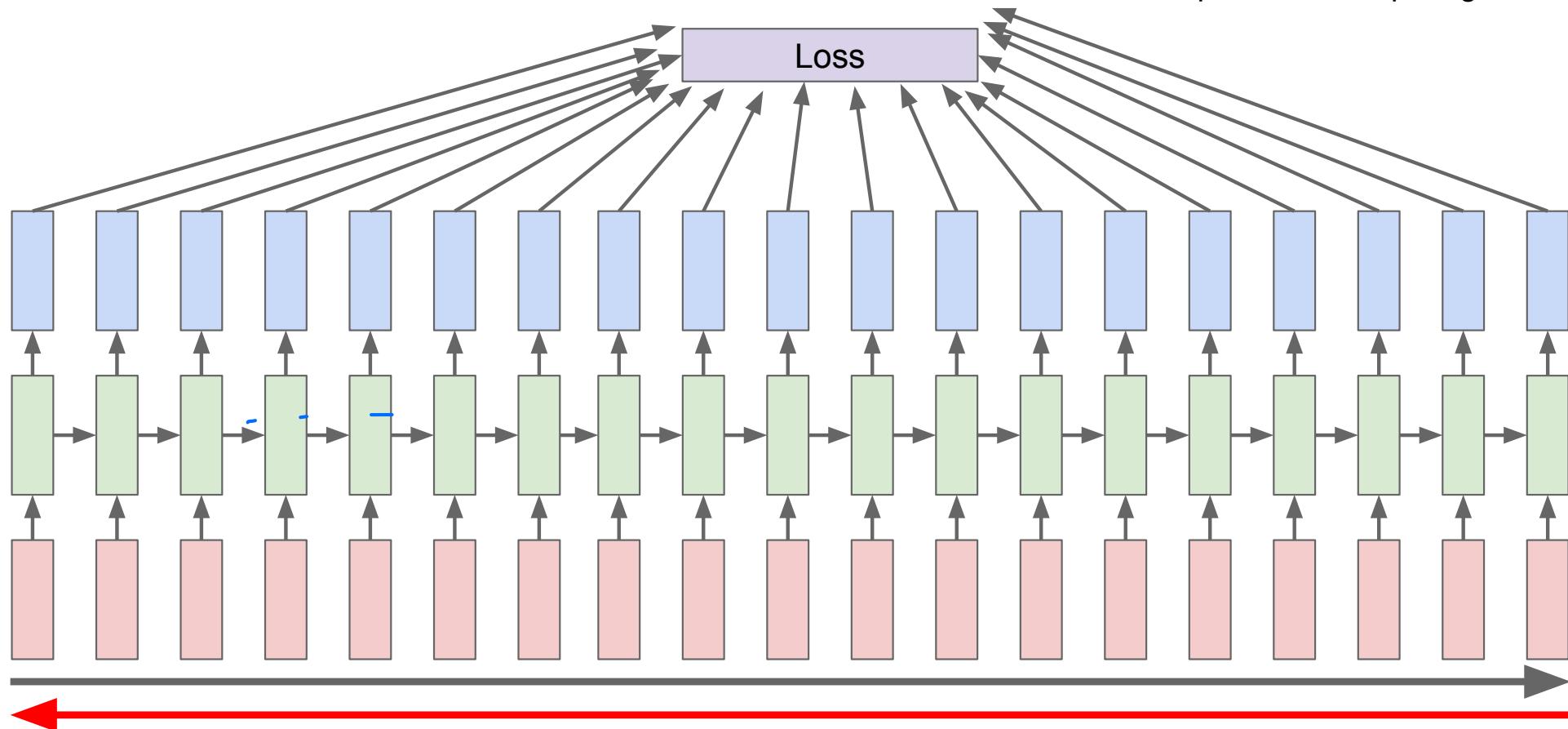
Vocabulary:
[h,e,l,o]

At test-time sample
characters one at a time,
feed back to model

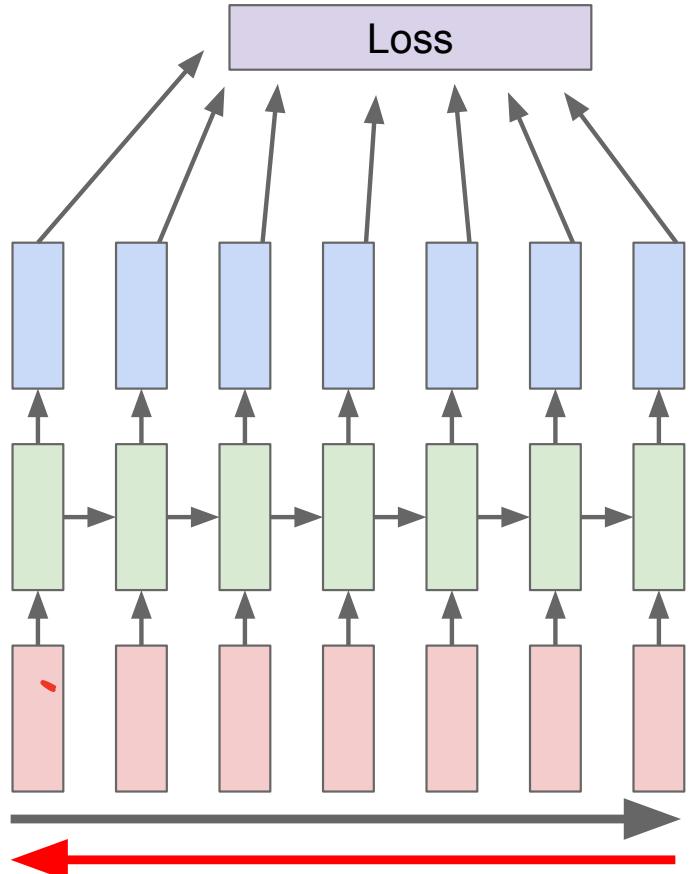


Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

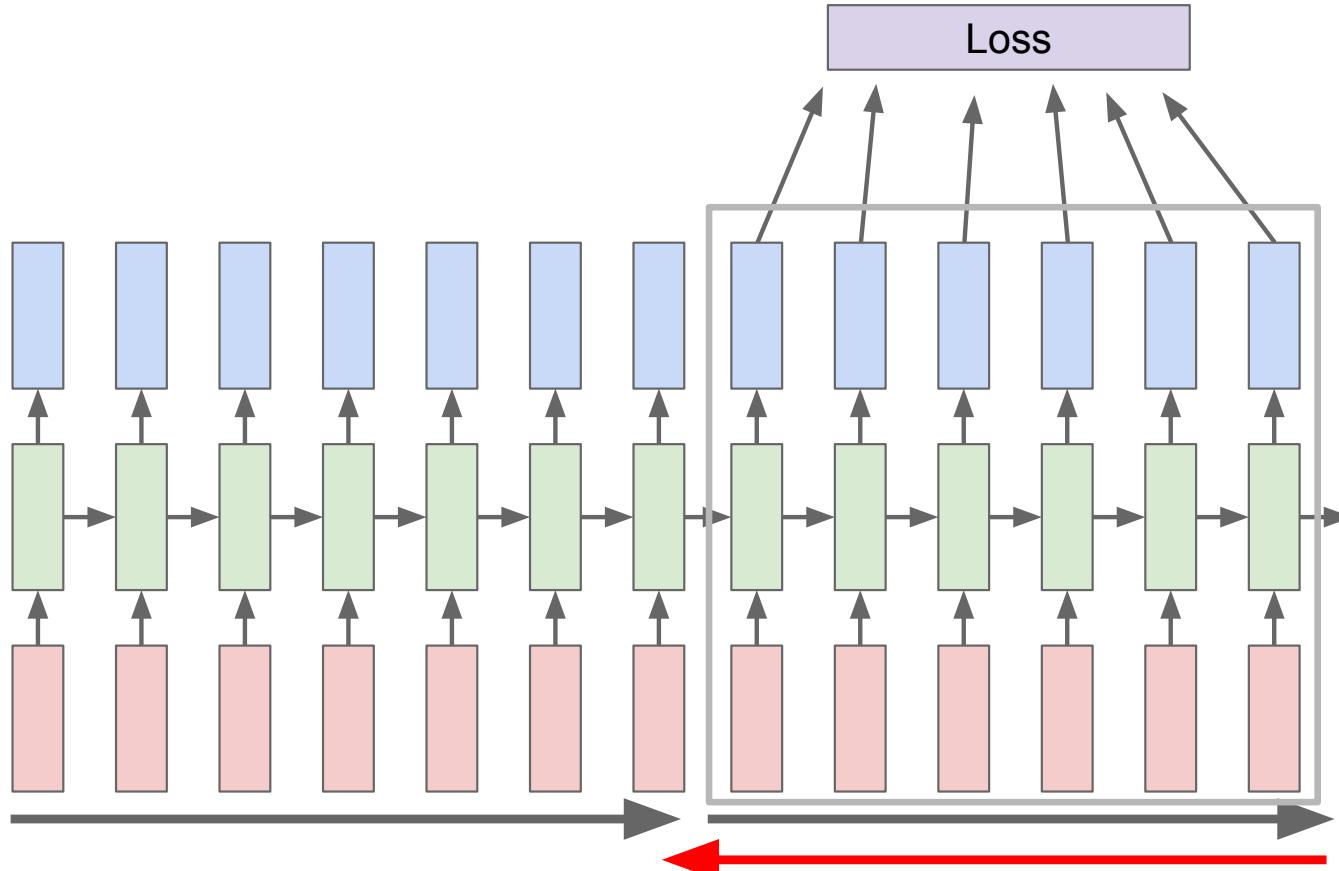


Truncated Backpropagation through time



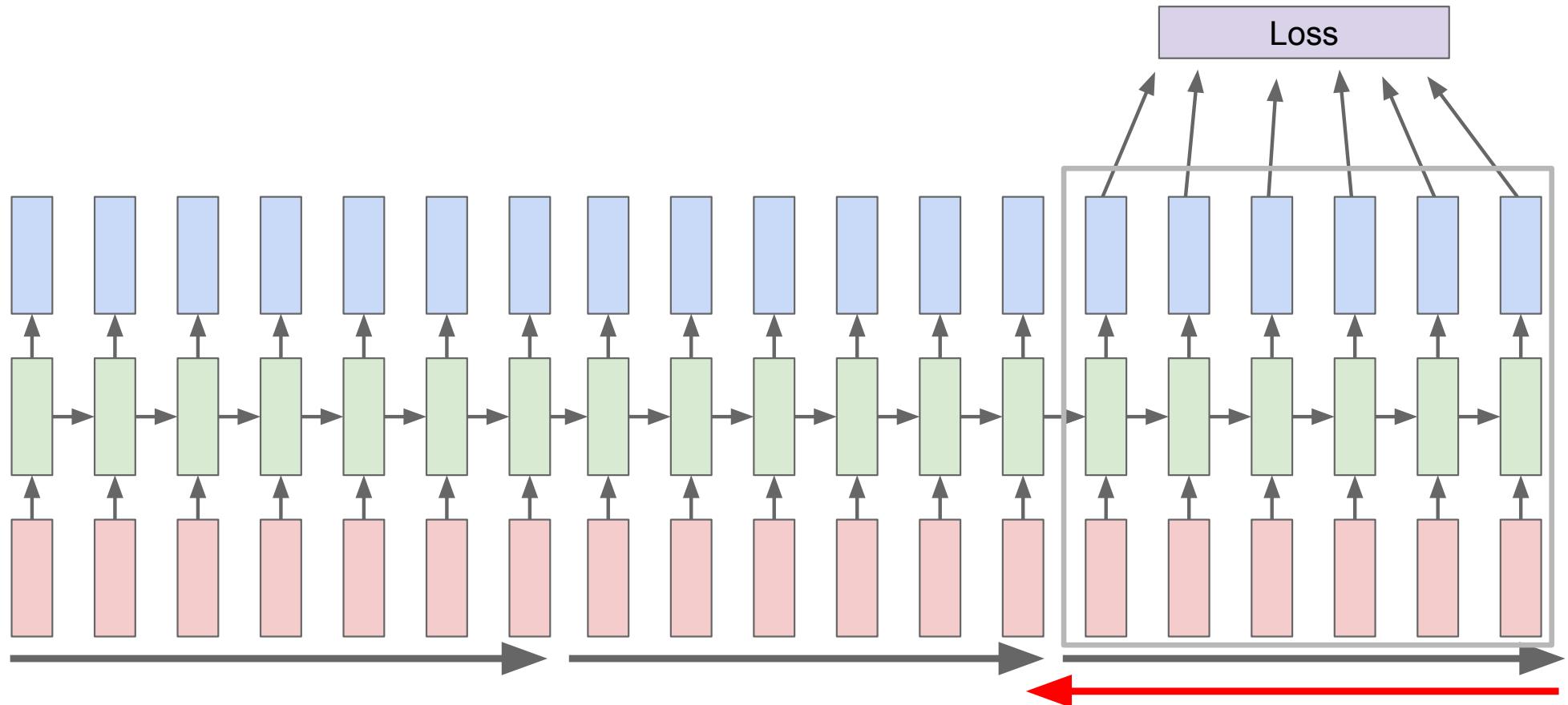
Run forward and backward
through chunks of the
sequence instead of whole
sequence

Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through time



min-char-rnn.py gist: 112 lines of Python

```
1 """
2 Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3 BSD License
4 """
5 import numpy as np
6
7 # data I/O
8 data = open('input.txt', 'r').read() # should be simple plain text file
9 chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print 'data has %d characters, %d unique.' % (data_size, vocab_size)
12 char_to_ix = {ch:i for i,ch in enumerate(chars)}
13 ix_to_char = {i:ch for i,ch in enumerate(chars)}
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 wkh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
22 whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) # hidden bias
25 by = np.zeros((vocab_size, 1)) # output bias
26
27 def lossFun(inputs, targets, hprev):
28     """
29     inputs,targets are both list of integers.
30     hprev is Hx1 array of initial hidden state
31     returns the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = {}, {}, {}, {}
34     hs[-1] = np.copy(hprev)
35     loss = 0
36     # forward pass
37     for t in xrange(len(inputs)):
38         xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
39         xs[t][inputs[t]] = 1
40         hs[t] = np.tanh(np.dot(wkh, xs[t]) + np.dot(whh, hs[t-1]) + bh) # hidden state
41         ys[t] = np.dot(why, hs[t]) + by # unnormalized log probabilities for next chars
42         ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
43         loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
44     # backward pass: compute gradients going backwards
45     dwhx, dwhh, dwhy = np.zeros_like(wkh), np.zeros_like(whh), np.zeros_like(why)
46     dbh, dy = np.zeros_like(bh), np.zeros_like(by)
47     dhnext = np.zeros_like(hs[0])
48     for t in reversed(xrange(len(inputs))):
49         dy = np.copy(ps[t])
50         dy[targets[t]] -= 1 # backprop into y
51         dwhy += np.dot(dy, hs[t].T)
52         dyb += dy
53         dh = np.dot(Why.T, dy) + dhnext # backprop into h
54         ddraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
55         dbh += ddraw
56         dwhx += np.dot(ddraw, xs[t].T)
57         dwhh += np.dot(ddraw, hs[t-1].T)
58         dhnext = np.dot(Wh.T, ddraw)
59     for dpParam in [dwhx, dwhh, dwhy, dbh, dyb]:
60         np.clip(dpParam, -5, 5, out=dpParam) # clip to mitigate exploding gradients
61     return loss, dwhx, dwhh, dwhy, dbh, dyb, hs[len(inputs)-1]
```

```
63 def sample(h, seed_ix, n):
64     """
65     sample a sequence of integers from the model
66     h is memory state, seed_ix is seed letter for first time step
67     """
68     x = np.zeros((vocab_size, 1))
69     x[seed_ix] = 1
70     ixes = []
71     for t in xrange(n):
72         h = np.tanh(np.dot(wkh, x) + np.dot(whh, h) + bh)
73         y = np.dot(why, h) + by
74         p = np.exp(y) / np.sum(np.exp(y))
75         ix = np.random.choice(range(vocab_size), p=p.ravel())
76         x = np.zeros((vocab_size, 1))
77         x[ix] = 1
78         ixes.append(ix)
79     return ixes
80
81 n, p = 0, 0
82 dwhx, dwhh, dwhy = np.zeros_like(wkh), np.zeros_like(whh), np.zeros_like(why)
83 dbh, dyb = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
84 smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
85 while True:
86     # prepare inputs (we're sweeping from left to right in steps seq_length long)
87     if p-seq_length+1 >= len(data) or n == 0:
88         hprev = np.zeros((hidden_size,1)) # reset RNN memory
89         p = 0 # go from start of data
90     inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
91     targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
92
93     # sample from the model now and then
94     if n % 100 == 0:
95         sample_ix = sample(hprev, inputs[0], 200)
96         txt = ''.join(ix_to_char[ix] for ix in sample_ix)
97         print '----\n%s \n----' % (txt, )
98
99     # forward seq_length characters through the net and fetch gradient
100    loss, dwhx, dwhh, dwhy, dbh, dyb, hprev = lossFun(inputs, targets, hprev)
101    smooth_loss = smooth_loss * 0.999 + loss * 0.001
102    if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress
103
104    # perform parameter update with Adagrad
105    for param, dparam, mem in zip([wkh, whh, why, bh, by],
106                                 [dwhx, dwhh, dwhy, dbh, dyb],
107                                 [mem, mem, mem, mem, mem]):
108        mem += dparam * dparam
109        param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
110
111    p += seq_length # move data pointer
112    n += 1 # iteration counter
```

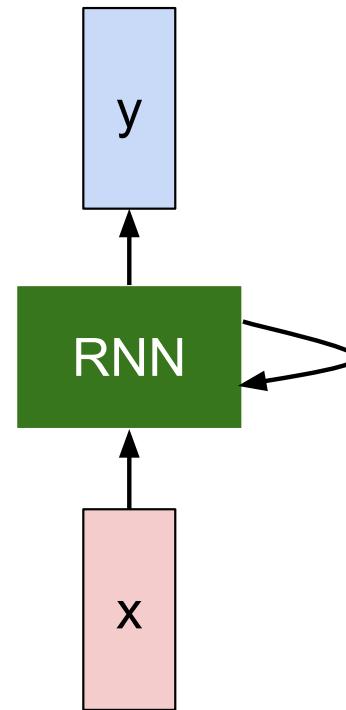
(<https://gist.github.com/karpathy/d4dee566867f8291f086>)

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserved thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.



at first:

tyntd-iafhatawiaoahrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tkldrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓ train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓ train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.

↓ train more

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

VIOLA:

Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

The Stacks Project: open source algebraic geometry textbook

The Stacks Project

[home](#) [about](#) [tags explained](#) [tag lookup](#) [browse](#) [search](#) [bibliography](#) [recent comments](#) [blog](#) [add slogans](#)

Browse chapters

Part	Chapter	online	TeX source	view pdf
Preliminaries	1. Introduction	online	tex	pdf
	2. Conventions	online	tex	pdf
	3. Set Theory	online	tex	pdf
	4. Categories	online	tex	pdf
	5. Topology	online	tex	pdf
	6. Sheaves on Spaces	online	tex	pdf
	7. Sites and Sheaves	online	tex	pdf
	8. Stacks	online	tex	pdf
	9. Fields	online	tex	pdf
	10. Commutative Algebra	online	tex	pdf

Parts

1. [Preliminaries](#)
2. [Schemes](#)
3. [Topics in Scheme Theory](#)
4. [Algebraic Spaces](#)
5. [Topics in Geometry](#)
6. [Deformation Theory](#)
7. [Algebraic Stacks](#)
8. [Miscellany](#)

Statistics

The Stacks project now consists of

- o 455910 lines of code
- o 14221 tags (56 inactive tags)
- o 2366 sections

Latex source

<http://stacks.math.columbia.edu/>

The stacks project is licensed under the [GNU Free Documentation License](#)

For $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m,n} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section ?? and the fact that any U affine, see Morphisms, Lemma ???. Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{\mathcal{M}}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)^{\text{opp}}_{fppf}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ???. It may replace S by $X_{\text{spaces,étale}}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ???. Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x,\dots,0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq p$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.

Lemma 0.3. In Situation ???. Hence we may assume $q' = 0$.

Proof. We will use the property we see that p is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

Proof. Omitted. \square

Lemma 0.1. Let \mathcal{C} be a set of the construction.

Let \mathcal{C} be a gerber covering. Let \mathcal{F} be a quasi-coherent sheaves of \mathcal{O} -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves \mathcal{F} on $X_{\text{étale}}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where \mathcal{G} defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ of \mathcal{O} -modules. \square

Lemma 0.2. This is an integer \mathcal{Z} is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over S and Y .

Proof. Let X be a nonzero scheme of X . Let X be an algebraic space. Let \mathcal{F} be a quasi-coherent sheaf of \mathcal{O}_X -modules. The following are equivalent

- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. \square

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccccc}
 S & \xrightarrow{\quad} & & & \\
 \downarrow & & & & \\
 \xi & \longrightarrow & \mathcal{O}_{X'} & \nearrow & \\
 \text{gor}_S & & \uparrow & & \\
 & = \alpha' & \longrightarrow & & X \\
 & \uparrow & & & \downarrow \\
 & = \alpha' & \longrightarrow & \text{Mor}_{\text{Sets}} & \text{d}(\mathcal{O}_{X/k}, \mathcal{G}) \\
 \text{Spec}(K_\psi) & & & &
 \end{array}$$

is a limit. Then \mathcal{G} is a finite type and assume S is a flat and \mathcal{F} and \mathcal{G} is a finite type f_* . This is of finite type diagrams, and

- the composition of \mathcal{G} is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

\square

Proof. We have see that $X = \text{Spec}(R)$ and \mathcal{F} is a finite type representable by algebraic space. The property \mathcal{F} is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U . \square

Proof. This is clear that \mathcal{G} is a finite presentation, see Lemmas ??.

A reduced above we conclude that U is an open covering of C . The functor \mathcal{F} is a “field”

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\bar{x}} \dashv (\mathcal{O}_{X_{\text{étale}}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1} \mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\bar{v}})$$

is an isomorphism of covering of \mathcal{O}_{X_ℓ} . If \mathcal{F} is the unique element of \mathcal{F} such that X is an isomorphism.

The property \mathcal{F} is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme \mathcal{O}_X -algebra with \mathcal{F} are opens of finite type over S .

If \mathcal{F} is a scheme theoretic image points. \square

If \mathcal{F} is a finite direct sum \mathcal{O}_{X_λ} is a closed immersion, see Lemma ???. This is a sequence of \mathcal{F} is a similar morphism.



This repository Search

Explore Gist Blog Help



karpathy



torvalds / linux

Watch 3,711

Star 23,054

Fork 9,141

Linux kernel source tree

520,037 commits

1 branch

420 releases

5,039 contributors



branch: master

linux / +



74
Pull requests



Pulse



Graphs

HTTPS clone URL

<https://github.com/torvalds/linux>

You can clone with [HTTPS](#),
[SSH](#), or [Subversion](#).

[Clone in Desktop](#)

[Download ZIP](#)

Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux ...

torvalds authored 9 hours ago

latest commit 4b1706927d

	Documentation	Merge git://git.kernel.org/pub/scm/linux/kernel/git/nab/target-pending	6 days ago
	arch	Merge branch 'x86-urgent-for-linus' of git://git.kernel.org/pub/scm/l...	a day ago
	block	block: discard bdi_unregister() in favour of bdi_destroy()	9 days ago
	crypto	Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6	10 days ago
	drivers	Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux	9 hours ago
	firmware	firmware/hex2fw.c: restore missing default in switch statement	2 months ago
	fs	vfs: read file_handle only once in handle_to_path	4 days ago
	include	Merge branch 'perf-urgent-for-linus' of git://git.kernel.org/pub/scm/...	a day ago
	init	init: fix regression by supporting devices with major:minor:offset fo...	a month ago
	io	bio: remove bio_endio from bio_endio function	a month ago

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffffff8) & 0x000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

Generated C code

```
/*
 * Copyright (c) 2006-2010, Intel Mobile Communications. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software Foundation,
 * Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setevid.h>
#include <asm/pgproto.h>
```

```

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/seteew.h>
#include <asm/pgproto.h>

#define REG_PG      vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)      (func)

#define SWAP_ALLOCATE(nr)      (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %esp, %0, %3" : : "r" (0)); \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
                                              pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
                (unsigned long)-1->lr_full; low;
}

```

Supervised Machine Learning:

Regression

Time Series Prediction

Time Series Data

- Time series is a sequential set of data points, measured typically over successive times
- Time series data are simply a collection of observations gathered over time
- Time series is a time oriented sequence of observations on a variable of interest
- It is clearly structured and numeric in nature
- Time series data is collected at some intervals
 - These intervals can be as large as years or as small as seconds
- Example:
 - Weekly sales – time interval is week
 - Daily temperature in Kamand – time interval is day

Time Series Data

- Time series is a sequential set of data points, measured typically over successive times
- Time series data are simply a collection of observations gathered over time
- Time series is a time oriented sequence of observations on a variable of interest
- It is clearly structured and numeric in nature
- Example:
 - Weekly sales – time interval is week
 - Daily temperature in Kamand – time interval is day
- Time series data is collected at some intervals
 - These intervals can be as large as years or as small as seconds

Date/Time	Temperature (C)/ Humidity (%)	Pressure (Pa)	Rain (Inches)	Light Intensity (lux)	Accelerations (g)	Force (N)	Moisture (%)
2017-09-06 18:44:32	23.00,56.00	617.64	0.01	3	0.52,0.31,-0.80,0.00,0.00,0.00,31.36,-159.01	0.02	81.00
2017-09-06 18:33:32	24.00,58.00	619.47	0.01	12	0.52,0.30,-0.79,0.00,0.00,0.00,31.45,-159.12	0.02	82.00
2017-09-06 18:22:39	24.00,58.00	623.37	0.00	71	0.52,0.31,-0.80,0.00,0.00,0.00,31.35,-158.88	0.02	83.00
2017-09-06 18:11:31	25.00,60.00	627.02	0.05	194	0.51,0.31,-0.80,0.00,0.00,0.00,30.80,-159.00	0.02	81.00

Time Series Data

- Time series data is given as:

$$\mathbf{X} = (x_1, x_2, \dots, x_t, \dots, x_T)$$

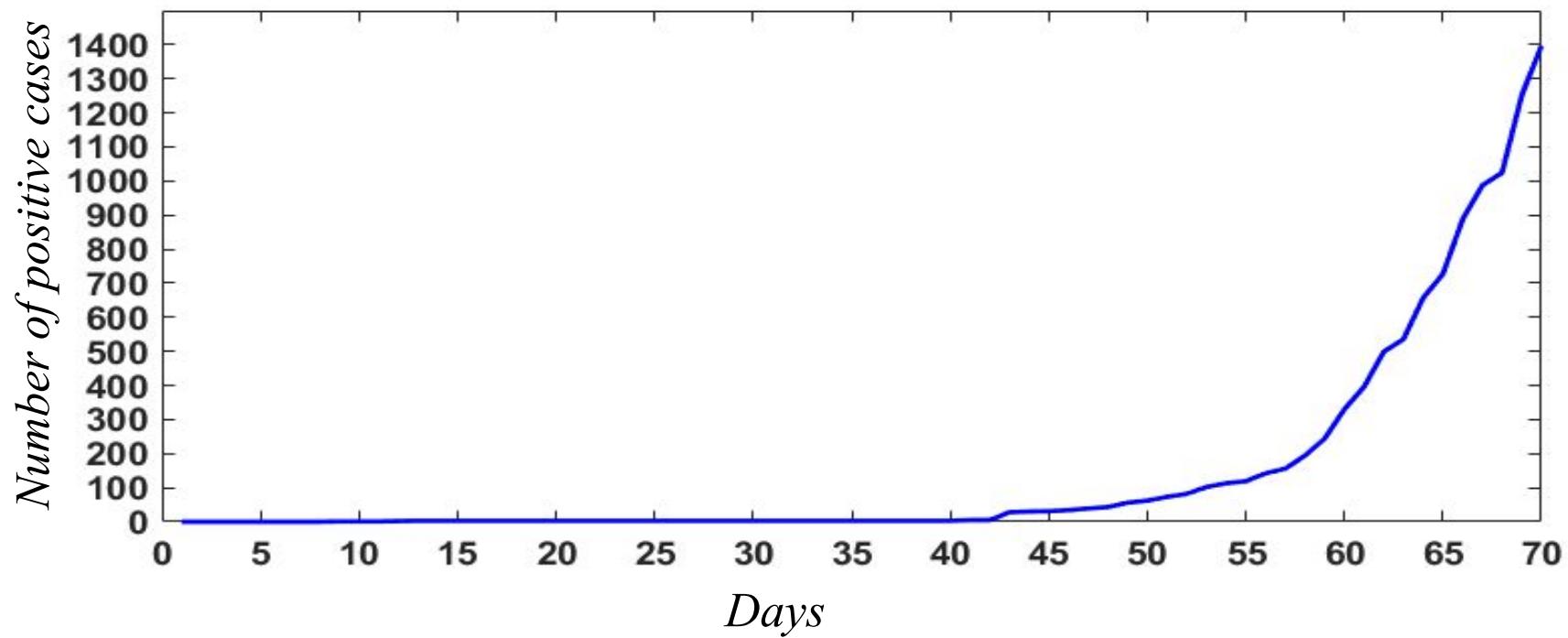
– x_t is the observation at time t

– T be the number of observations

- Scope: We consider single variable x_t

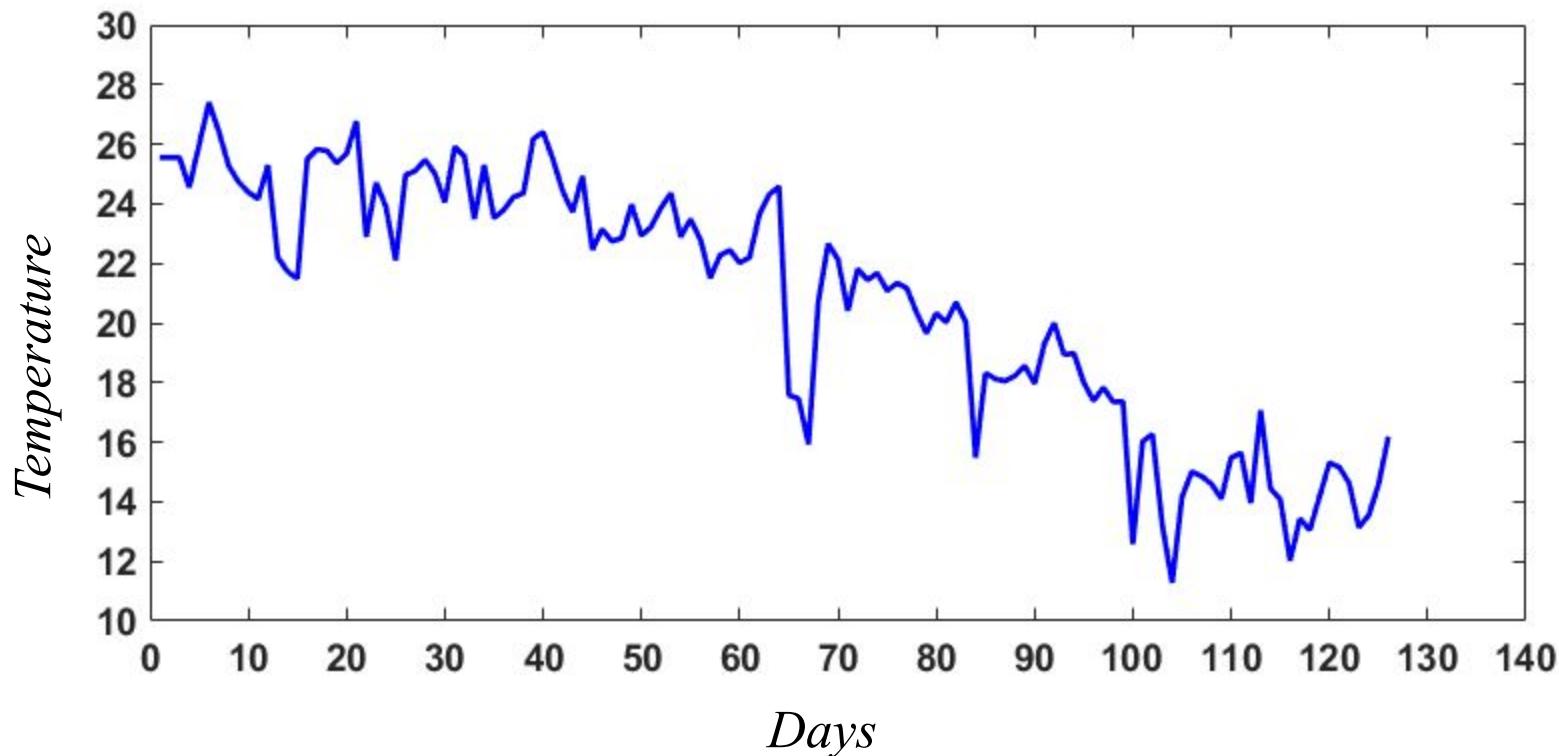
Time Series Data

- Trend: Shows how data moves over a period of time
 - COVID positive cases in India between 22 Jan 2020 to 31 March 2020



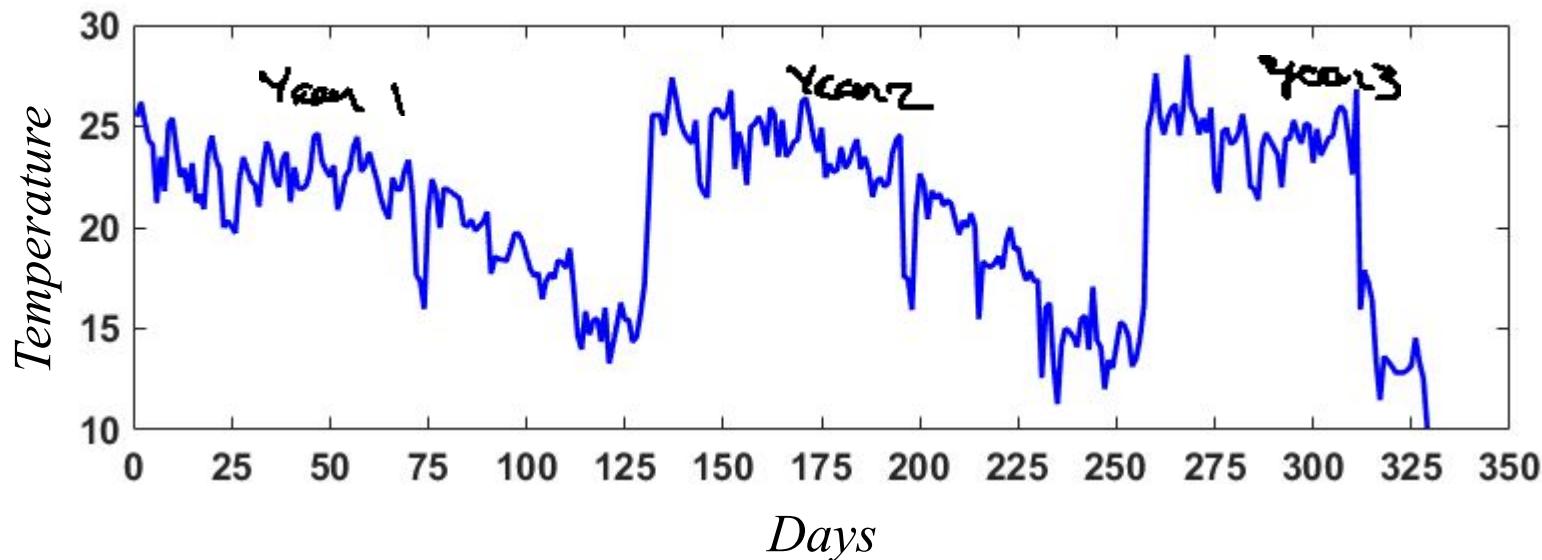
Time Series Data

- Trend: Shows how data moves over a period of time
 - Daily temperature at IIT Mandi from June-Nov 2018



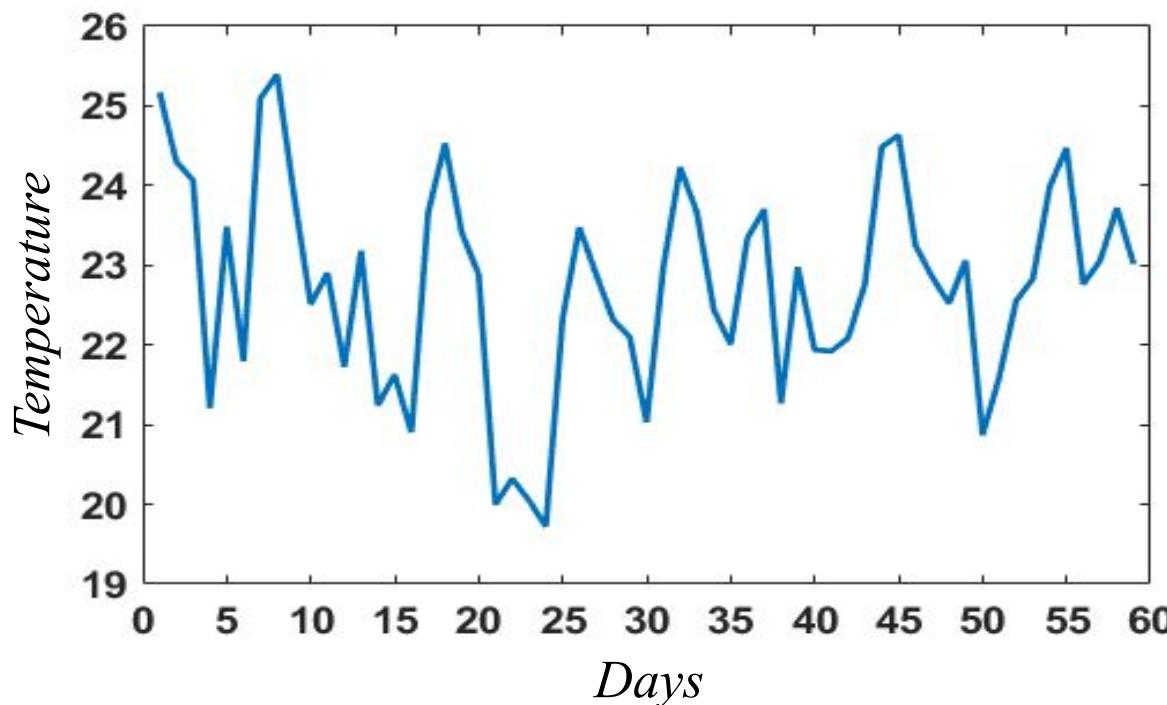
Time Series Data

- Seasonality: A type of pattern which repeats over a specific period of time
 - Daily temperature recorded in IIT Mandi for 3 years
 - Duration of recorded: July-Nov (2017-2019)



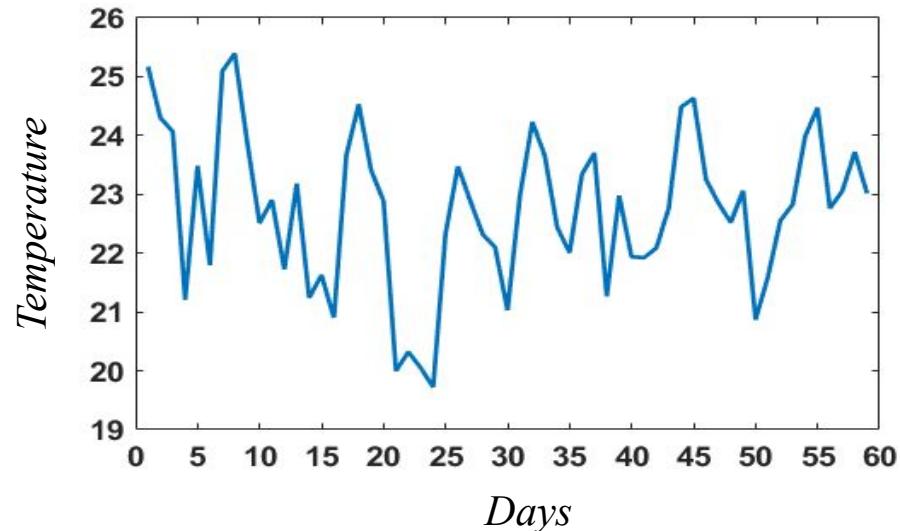
Time Series Data

- Random or error: Series does not have any trend, seasonality or cyclic component
 - Daily temperature recorded in IIT Mandi (1 July - 30 August 2019)



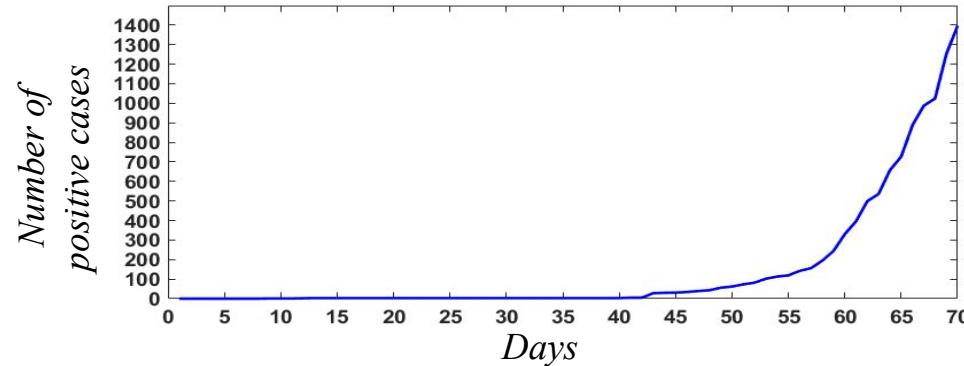
Stationary Time Series

- Stationary time series:
 - Statistical properties remain same at any given interval of time
 - Time independent kind of series
 - Mean and variance should be time independent
 - Mean and variance computed at any one part of the series should be similar to that of the mean and variance computed at another part
 - Stationary time series are easier to predict
- Example:
 - Daily temperature recorded in IIT Mandi (1 July - 30 August 2019)

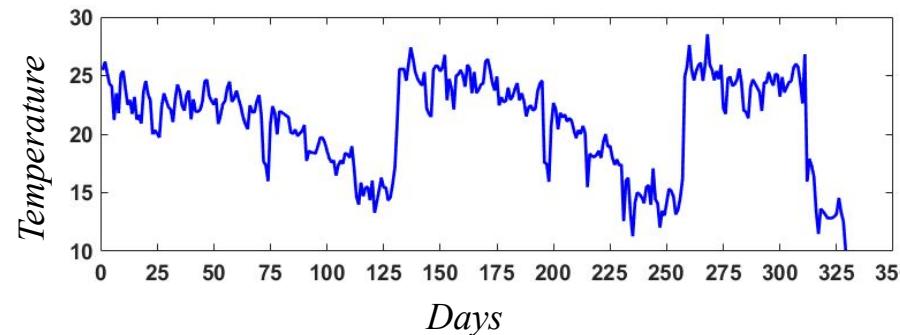


Non-stationary Time Series

- Non-stationary time series: Time series having trends or seasonality
 - Mean and variance are not time independent
- Example:
 - COVID positive cases in India between 22 Jan 2020 to 31 March 2020

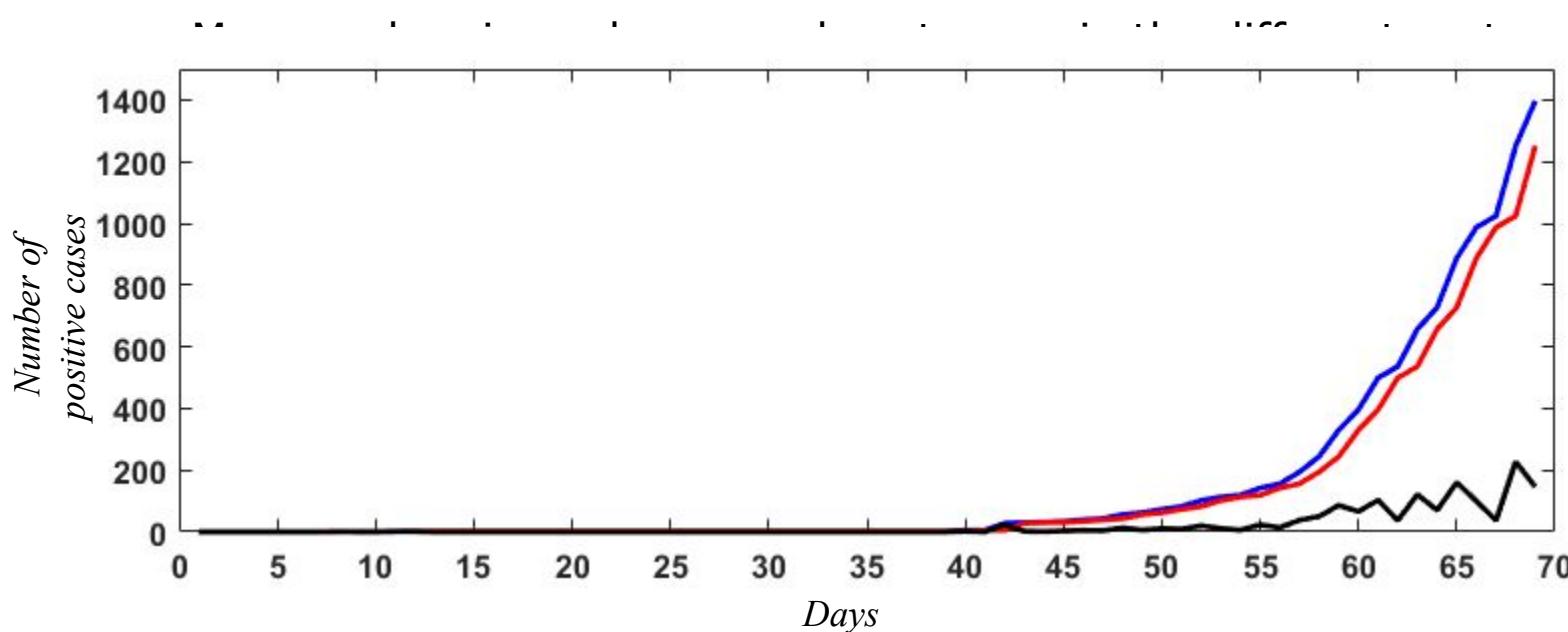


- Daily temperature recorded in IIT Mandi for 3 years



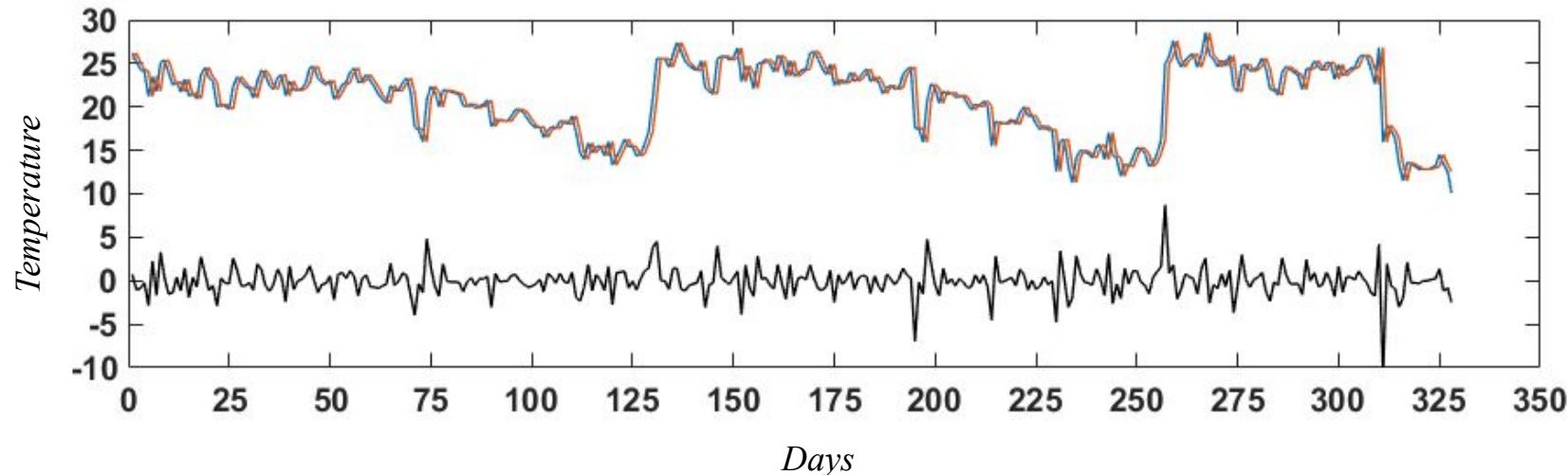
Differencing

- Non-stationary time series are made stationary by differencing
 - Difference between the original series and the lag series
 - Lag is the shift in the time series by a given number of observations
 - Lag 1: Shift by one time step
 - Lag 2: Shift by two time step
 - By differencing, non-stationary time series become more stationary



Differencing

- Non-stationary time series are made stationary by differencing
 - Difference between the original series and the lag series
 - Lag is the shift in the time series by a given number of observations
 - Lag 1: Shift by one unit
 - Lag 2: Shift by two unit
 - By differencing, non-stationary time series become more stationary
 - Mean and variance become almost same in the different parts
 - Example: Daily temperature recorded in IIT Mandi for 3 years



Time Series Data and Dependence

- Time series data is given as:

$$\mathbf{X} = (x_1, x_2, \dots, x_t, \dots, x_T)$$

– x_t is the observation at time t

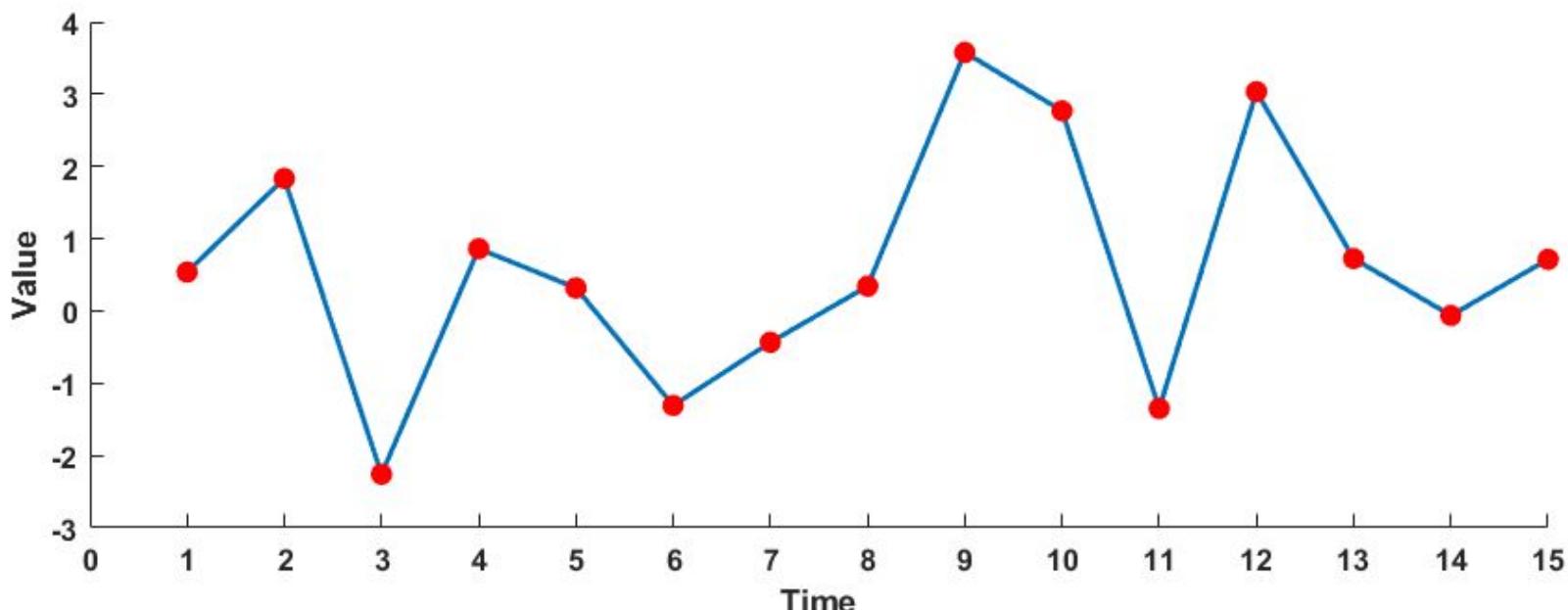
– T be the number of observations

- In time series data, value of each element at time t (x_t) is dependent on the values elements at previous p time steps ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) – p time lag
 - Lag is the shift in the time series by a given number of observations

Time Series Data and Dependence

- Example: Data series in i.i.d
 - x_t is a random number drawn from $N(0,1)$
- Each element at time t (x_t) is **not dependent** on the values elements at previous p time steps ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) – p time lag

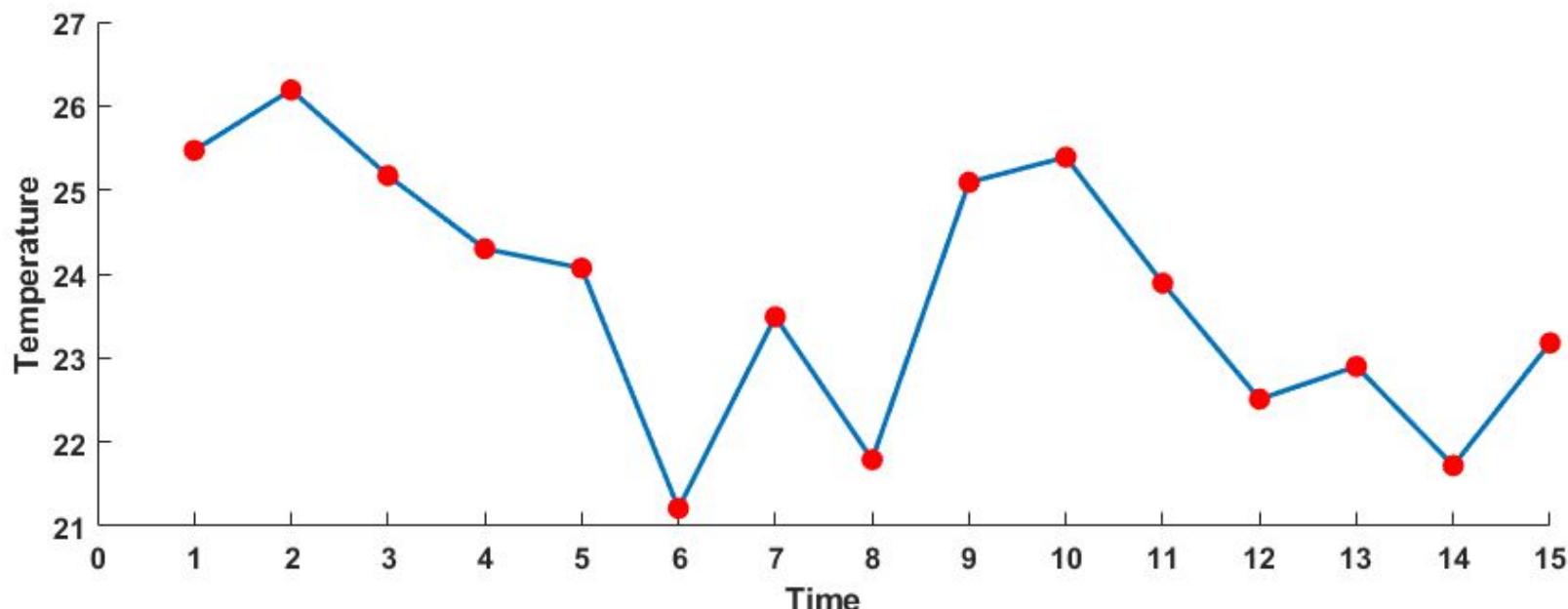
0.54	1.83	-2.26	0.86	0.32	-1.31	-0.43	0.34	3.58	2.77	-1.35	3.03	0.73	-0.06	0.71
------	------	-------	------	------	-------	-------	------	------	------	-------	------	------	-------	------



Time Series Data and Dependence

- Example: Daily temperature at IIT Mandi
- Each element at time t (x_t) is dependent on the values elements at previous p time steps ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) – p time lag
 - Temperature recorded for 15 days (1 Sept. 2019 – 15 Sept. 2019)

25.47	26.19	25.17	24.3	24.07	21.21	23.49	21.79	25.09	25.39	23.89	22.51	22.9	21.72	23.18
-------	-------	-------	------	-------	-------	-------	-------	-------	-------	-------	-------	------	-------	-------



Checking Dependency

- It's not always easy to just look at a time-series plot and say whether or not the series is independent
- x_t in a series is **independent** means that knowing previous values doesn't help you to predict the next value
 - Knowing x_{t-1} doesn't help to predict x_t
 - More generally, knowing $x_{t-1}, x_{t-2}, \dots, x_{t-p}$ doesn't help to predict x_t
 - p is the number of previous time step (time lag)
- Dependency of each element at time t (x_t) with the values of elements at previous p time steps ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) is observed using **autocorrelation**

Checking Dependency - Autocorrelation

- The relationship between variables is called **correlation**
- Autocorrelation:** The correlation calculated between the variable and itself at previous time steps
- Example:** Data series in i.i.d
 - Autocorrelation between x_t and x_{t-p} – Pearson correlation coefficient between original series and lag- p series

*Original
Series*

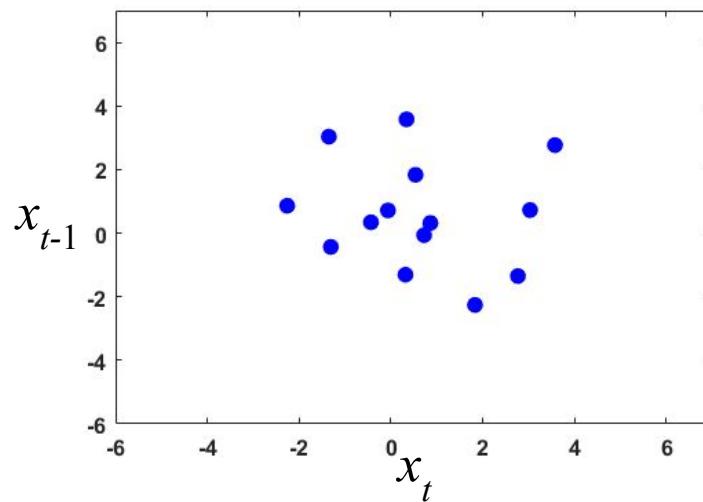
x_t	0.54	1.83	-2.26	0.86	0.32	-1.31	-0.43	0.34	3.58	2.77	-1.35	3.03	0.73	-0.06	0.71
-------	------	------	-------	------	------	-------	-------	------	------	------	-------	------	------	-------	------

Lag-1 Series

x_{t-1}	0.54	1.83	-2.26	0.86	0.32	-1.31	-0.43	0.34	3.58	2.77	-1.35	3.03	0.73	-0.06
-----------	------	------	-------	------	------	-------	-------	------	------	------	-------	------	------	-------

- Autocorrelation:

	x_t	x_{t-1}
x_t	1	-0.1242
x_{t-1}	-0.1242	1



Checking Dependency - Autocorrelation

- The relationship between variables is called correlation
- Autocorrelation:** The correlation calculated between the variable and itself at previous time steps
- Example:** Daily temperature at IIT Mandi
 - Autocorrelation between x_t (original series) and x_{t-1} (Lag-1 series)

Original Series

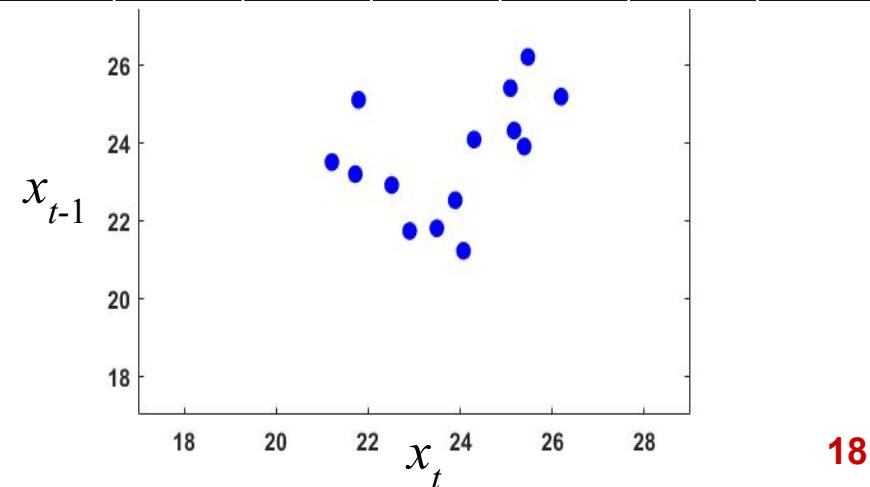
x_t	25.4 7	26.19 7	25.1 7	24.3 7	24.0 7	21.2 1	23.49 9	21.7 9	25.09 9	25.3 9	23.8 9	22.51 9	22.9 9	21.72 8	23.1 8
-------	-----------	------------	-----------	-----------	-----------	-----------	------------	-----------	------------	-----------	-----------	------------	-----------	------------	-----------

Lag-1 Series

x_{t-1}	25.47 26.19 25.17 24.3 24.07 21.21 23.49 21.79 25.09 25.39 23.89 22.51 22.9 21.72
-----------	--

- Autocorrelation:

	x_t	x_{t-1}
x_t	1	0.405 4
x_{t-1}	0.405 4	1



Autoregression (AR)

Autoregression (AR)

- Regression on the values of same attribute
- Autoregression is a time series model that
 - uses observations from previous time steps as input to a linear regression equation to predict the value at the next time step
 - Output variable: value at next time step
 - Input variable: observations from previous time step
 - Output variable is a linear function of input variables

Autoregression (AR)

- Autoregression (AR): Regression on the values of same attribute
 - It is a time series model
 - Linear regression model that uses observations from previous p time steps as input to predict the value at the next time step
 - It makes an assumption that the observations at previous time steps are useful to predict the value at the next time step
 - The autocorrelation statistics help to choose which lag variables (p) will be useful in a model
 - Dependency of each element at time t (x_t) with the values of elements at previous p time steps ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) is observed using autocorrelation
 - Autocorrelation: The correlation calculated between the variable and itself at previous time steps

Autoregression (AR) Model

- Autoregression (AR) is a linear regression model that uses observations from previous time steps as input to predict the value at the next time step
- An autoregression (AR) model makes an assumption that the observations at previous time steps are useful to predict the value at the next time step
- The autocorrelation statistics help to choose which lag variables (p) will be useful in a model
- Interestingly, if all lag variables ($x_{t-1}, x_{t-2}, \dots, x_{t-p}$) show low or no correlation with the output variable (x_t), then it suggests that the time series problem may not be predictable
- This can be very useful when getting started on a new dataset

Autoregression (AR) Model

- Building an AR model depends on how many time lag (p) is considered
- AR(1) model: AR model using one time lag ($p=1$)
 - uses x_{t-1} i.e. value of previous time step to predict x_t

Illustration AR(1) Model – Prediction of Temperature

Date	Temp (x_{t-1})	Temp (x_t)	Date
		25.47	Sept 1
Sept 1	25.47	26.19	Sept 2
Sept 2	26.19	25.17	Sept 3
Sept 3	25.17	24.30	Sept 4
Sept 4	24.30	24.07	Sept 5
Sept 5	24.07	21.21	Sept 6
Sept 6	21.21	23.49	Sept 7
Sept 7	23.49	21.79	Sept 8
Sept 8	21.79	25.09	Sept 9
Sept 9	25.09	25.39	Sept 10
---	---	---	---
Oct 28	22.76	23.06	Oct 29
Oct 29	23.06	23.72	Oct 30
Oct 30	23.72	23.02	Oct 31

- T , the number of observations = 61

- Independent variable:
 - Temperature at the time $t-1$
- Dependent variable:
 - Temperature at the time t

AR(1) Model

- AR(1) model: AR model using one time lag ($p=1$)
 - uses x_{t-1} i.e. value of previous time step to predict x_t
- Given: Time series data: $\mathbf{X} = (x_1, x_2, \dots, x_t, \dots, x_T)$
 - x_t is the observation at time t
 - T be the number of observations
- AR(1) model is given as: $x_t = f(x_{t-1}, w_0, w_1) = w_0 + w_1 x_{t-1}$
 - The coefficients w_0 and w_1 are parameters of straight-line (regression coefficients)
- **Unknown**
- The regression coefficients are obtained as seen in simple linear regression (straight-line regression) using least square method

AR(1) Model - Training

- The regression coefficients are obtained as seen in simple linear regression (straight-line regression) using least square method
- Minimize the squared error between the actual data (x_t) at time t and the estimate of linear function (predicted variable (\hat{x}_t)) i.e. the function $f(x_{t-1}, w_0, w_1)$

$$\hat{x}_t = f(x_{t-1}, w_0, w_1) = w_0 + w_1 x_{t-1}$$

$$\underset{w, w_0}{\text{minimize}} E(w_0, w_1) = \frac{1}{2} \sum_{t=2}^T (\hat{x}_t - x_t)^2$$

- The optimal \hat{w}_0 and \hat{w}_1 is given as

$$\hat{w}_1 = \frac{\sum_{t=1}^T (x_{t-1} - \mu_{t-1})(x_t - \mu_t)}{\sum_{t=1}^T (x_{t-1} - \mu_{t-1})^2}$$

$$\hat{w}_0 = \mu_t - w_1 \mu_{t-1}$$

- μ_{t-1} : sample mean of variables at time $t-1$, x_{t-1}
- μ_t : sample mean of variables at time t , x_t

AR(1) Model: Testing

- For any test example at time $t-1$, x_{t-1} , the predicted value at time t , \hat{x}_t is given by:

$$\hat{x}_t = f(x_{t-1}, w_0, w_1) = \hat{w}_0 + \hat{w}_1 x_{t-1}$$

Evaluation Metrics for Time Series Prediction: Squared Error and Root Mean Squared Error

- The prediction accuracy is measured in terms of squared error: $E = (\hat{x}_t - x_t)^2$
 - x_t : actual value
 - \hat{x}_t : predicted value
- Let T_{test} be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error (RMSE):

$$E_{\text{RMS}} = \sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (\hat{x}_t - x_t)^2}$$

- RMSE expressed in % as:

$$E_{\text{RMS}} = \sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (\hat{x}_t - x_t)^2} * 100$$

Evaluation Metrics for Time Series Prediction: Absolute Error and Mean Absolute Percentage Error (MAPE)

- **Absolute error:**
 - x_t : actual value
 - \hat{x}_t : predicted value
- Let T_{test} be the total number of test samples
- The prediction accuracy of regression model is measured in terms of **mean absolute percentage error**:

$$E_{MAP} = \left(\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} \frac{|x_t - \hat{x}_t|}{x_t} \right) * 100$$

Illustration AR(1) Model – Prediction of Temperature: Training

Temp (x_{t-1})	Temp (x_t)	Date
	25.47	Sept 1
25.47	26.19	Sept 2
26.19	25.17	Sept 3
25.17	24.30	Sept 4
24.30	24.07	Sept 5
24.07	21.21	Sept 6
21.21	23.49	Sept 7
23.49	21.79	Sept 8
21.79	25.09	Sept 9
25.09	25.39	Sept 10
---	---	---
22.76	23.06	Oct 29
23.06	23.72	Oct 30
23.72	23.02	Oct 31

- T , the number of observations = 61

$$\hat{w}_1 = \frac{\sum_{t=1}^{60} (x_{t-1} - \mu_{t-1})(x_t - \mu_t)}{\sum_{t=1}^{60} (x_{t-1} - \mu_{t-1})^2}$$

$$\hat{w}_0 = \mu_t - w_1 \mu_{t-1}$$

- μ_{t-1} : 22.81 • \hat{w}_1 : 0.523
- μ_t : 22.85 • \hat{w}_0 : 10.861

Illustration AR(1) Model – Prediction of Temperature: Test

- Predict Temperature for Nov 2

Nov 1

- \hat{w}_1 : 0.523
- \hat{w}_0 : 10.861

Temp (x_{t-1})	Temp (x_t)
22.30	-

- Predicted Temperature for Nov 2 : 22.52
- Actual Temperature on Nov 2 : 21.43
- Squared error : 1.19
- Absolute error : 0.0509

Autoregression Model

- AR(p) model: AR model using p time lags ($p < T$)
 - uses $x_{t-1}, x_{t-2}, \dots, x_{t-p}$ i.e. value of previous p time step to predict x_t

Illustration AR(p) Model – Prediction of Temperature

Temp (x_{t-3})	Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)	Date
			25.47	Sept 1
		25.47	26.19	Sept 2
	25.47	26.19	25.17	Sept 3
25.47	26.19	25.17	24.30	Sept 4
26.19	25.17	24.30	24.07	Sept 5
25.17	24.30	24.07	21.21	Sept 6
24.30	24.07	21.21	23.49	Sept 7
24.07	21.21	23.49	21.79	Sept 8
21.21	23.49	21.79	25.09	Sept 9
---	---	---	---	---
22.83	23.98	24.47	22.76	Oct 28
23.98	24.47	22.76	23.06	Oct 29
24.47	22.76	23.06	23.72	Oct 30
22.76	23.06	23.72	23.02	Oct 31

- T , the number of observations = 61
- $p = 3$
- Independent variable:
 - Temperature at the time $t-1$, $t-2$ and $t-3$
- Dependent variable:
 - Temperature at the time t

Autoregression Model

- AR(p) model: AR model using p time lags ($p < T$)
 - uses $x_{t-1}, x_{t-2}, \dots, x_{t-p}$ i.e. value of previous p time step to predict x_t
- Given: Time series data: $\mathbf{X} = (x_1, x_2, \dots, x_t, \dots, x_T)$
 - x_t is the observation at time t
 - T be the number of observations
- AR(p) model is given as:

$$x_t = f(x_{t-1}, x_{t-2}, \dots, x_{t-p}, w_0, w_1, \dots, w_p) = w_0 + w_1 x_{t-1} + \dots + w_p x_{t-p}$$

$$x_t = f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^p w_j x_{t-j} = \mathbf{w}^\top \mathbf{x}$$

where $\mathbf{w} = [w_0, w_1, \dots, w_p]^\top$ and $\mathbf{x} = [1, x_{t-1}, x_{t-2}, \dots, x_{t-p}]^\top$

- The coefficients w_0, w_1, \dots, w_p are parameters of hyperplane (regression coefficients) - **Unknown**

AR (p) Model - Training

- The regression coefficients are obtained as seen in **multiple linear regression** with p input variables using least square method
- Minimize the squared error between the actual data (x_t) at time t and the estimate of linear function (predicted variable (\hat{x}_t)) i.e. the function $f(\mathbf{x}, \mathbf{w})$

$$\hat{x}_t = f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^p w_j x_{t-j} = w_0 + \mathbf{w}^\top \mathbf{x}$$

$$\underset{\mathbf{w}}{\text{minimize}} E(\mathbf{w}) = \frac{1}{2} \sum_{t=p+1}^T (\hat{x}_t - x_t)^2$$

- The **autocorrelation statistics** help to choose which lag variables (p) will be useful in a model

AR (p) Model - Training

- The optimal $\hat{\mathbf{w}}$ is given as
$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{x}^{(t)}$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{t-p} & \dots & x_{t-3} & x_{t-2} & x_{t-1} \\ 1 & x_{(t+1)-p} & \dots & x_{t-2} & x_{t-1} & x_t \\ \hline & \cdots & & & & \\ 1 & x_{(t+n)-p} & \dots & x_{t+n-3} & x_{t+n-2} & x_{t+n-1} \\ \hline & \cdots & & & & \\ 1 & x_{T-p} & \dots & x_{T-3} & x_{T-2} & x_{T-1} \end{bmatrix} \quad \mathbf{x}^{(t)} = \begin{bmatrix} x_t \\ x_{t+1} \\ \vdots \\ x_{t+n} \\ \vdots \\ x_T \end{bmatrix}$$

\mathbf{X} is data matrix with time lag p

- The **autocorrelation statistics** help to choose which lag variables (p) will be useful in a model

AR (p) Model: Testing

- The value at time t , \hat{x}_t is predicted by taking values from past p time steps $(x_{t-1}, x_{t-2}, \dots, x_{t-p})$ as input:

$$\hat{x}_t = f(\mathbf{x}, \hat{\mathbf{w}}) = \hat{w}_0 + \sum_{j=1}^p \hat{w}_j x_{t-j} = \hat{\mathbf{w}}^\top \mathbf{x}$$

- The prediction accuracy is measured in terms of squared error:

$$E = (\hat{x}_t - x_t)^2$$

- Let T_{test} be the total number of test samples
- The prediction accuracy of regression model is measured in terms of root mean squared error:

$$E_{\text{RMS}} = \sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (\hat{x}_t - x_t)^2}$$

- Mean absolute percentage error (MAPE) is also used as a measure

Illustration AR(p) Model – Prediction of Temperature: Checking Dependency

Temp (x_{t-3})	Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)	Date
			25.47	Sept 1
		25.47	26.19	Sept 2
	25.47	26.19	25.17	Sept 3
25.47	26.19	25.17	24.30	Sept 4
26.19	25.17	24.30	24.07	Sept 5
25.17	24.30	24.07	21.21	Sept 6
24.30	24.07	21.21	23.49	Sept 7
24.07	21.21	23.49	21.79	Sept 8
21.21	23.49	21.79	25.09	Sept 9
---	---	---	---	---
22.83	23.98	24.47	22.76	Oct 28
23.98	24.47	22.76	23.06	Oct 29
24.47	22.76	23.06	23.72	Oct 30
22.76	23.06	23.72	23.02	Oct 31

- $p = 3$
- T , the number of observations = 61
- Autocorrelation between x_t and x_{t-1} : 0.54
- Autocorrelation between x_t and x_{t-2} : 0.25
- Autocorrelation between x_t and x_{t-3} : -0.08
- An autocorrelation is deemed significant if

$$|\text{autocorrelation}| > \frac{2}{\sqrt{T}} = 0.25$$

- Time lag $p=2$ is sufficient as x_t is significant with x_{t-1} and x_{t-2}

Illustration AR(p) Model – Prediction of Temperature: Training

Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)	Date
		25.47	Sept 1
	25.47	26.19	Sept 2
25.47	26.19	25.17	Sept 3
26.19	25.17	24.30	Sept 4
25.17	24.30	24.07	Sept 5
24.30	24.07	21.21	Sept 6
24.07	21.21	23.49	Sept 7
21.21	23.49	21.79	Sept 8
23.49	21.79	25.09	Sept 9
---	---	---	---
23.98	24.47	22.76	Oct 28
24.47	22.76	23.06	Oct 29
22.76	23.06	23.72	Oct 30
23.06	23.72	23.02	Oct 31

- $p = 2$
- T , the number of observations = 59
- Multiple linear regression with number of input variables = 2

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{x}^{(t)} ; \quad \hat{\mathbf{w}} \in \mathbf{R}^3$$

Illustration AR(p) Model – Prediction of Temperature: Test

- Predict Temperature for Nov 2

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{x}^{(t)} ; \quad \hat{\mathbf{w}} \in \mathbf{R}^3$$

Oct 31 Nov 1

Temp (x_{t-2})	Temp (x_{t-1})	Temp (x_t)
23.02	22.30	--

- AR(2) model:**

- Predicted Temperature for Nov 2 : 22.49
- Actual Temperature on Nov 2 : 21.43
- Squared error : 1.13
- Absolute error : 0.0495

- AR(1) model:**

- Predicted Temperature for Nov 2 : 22.52
- Actual Temperature on Nov 2 : 21.43
- Squared error : 1.19
- Absolute error : 0.0509

Summary: Autoregression

- Autoregression (AR): Regression on the values of same attribute
 - It is a time series model
 - Linear regression model that uses observations from previous p time steps as input to predict the value at the next time step
 - It makes an assumption that the observations at previous time steps are useful to predict the value at the next time step
 - The autocorrelation statistics help to choose which lag variables (p) will be useful in a model
- AR model can be performed on time series data with single variable or with multiple variables
- In this course we are limited only on the time series data with single variable

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

Clustering

Clustering

- Process of grouping a set of examples (samples)
- Clustering generates **a partition** consisting of **cohesive groups or clusters** from given collection of examples (samples)



- For example:
 - Grouping students in a class based on gender
 - Grouping students in a class based the month of birth
 - Grouping the students based on the place of sitting

Clustering

- Process of grouping a set of examples
- Clustering generates **a partition** consisting of **cohesive groups or clusters** from given collection of examples



- The examples to be clustered are either **labelled** or **unlabelled**
 - Algorithms which cluster labelled examples:
 - Supervised clustering
 - Classification: **Learning by examples**
 - Algorithms which cluster unlabelled examples:
 - Unsupervised clustering
 - Do not rely on predefined classes
 - **Learning by observation**, rather than learning by examples.

Clustering

- Clustering is a two step process
 - Step1: Partition the collection of examples (clustering)
 - Learning by observation (training phase)
 - Group the collection of examples into finite number of clusters such that the examples that are **similar** to one another within the same cluster and are **dissimilar** to examples in other clusters
 - Obtaining cluster labels
 - **Unsupervised learning:** Do not rely on predefined classes and class-labelled training examples
 - Step2: Assign cluster labels to examples
 - Testing phase

Categorization of Clustering Methods

- Partitioning methods
- Hierarchical methods
- Density-based methods

Categorization of Clustering Methods

- Partitioning methods:
 - These methods construct K partitions of the data, where each partition represents a cluster
 - Idea: Cluster the collection of examples based on the distance between examples
 - Results in spherical shaped cluster
- 1. K -means algorithm
- 2. K -medoids algorithm
- 3. Gaussian mixture model
- Hierarchical methods:
 - These methods create a hierarchical decomposition of the collection of examples
 - Results in spherical shaped cluster
- 1. Agglomerative approach (bottom-up approach)
- 2. Divisive approach (top-down approach)

Categorization of Clustering Methods

- Density-based methods:
 - These methods cluster collection of examples based on the notion of **density**
 - **General idea:** To continue growing the given cluster as long as density (number of examples) in the neighbourhood exceeds some threshold
 - Example:
 - DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Partitioning Method based Clustering

Classical Portioning Methods

- Centroid-based technique:
 - Partition the collection of examples into K clusters based on the distance between examples
 - Cluster similarity is measured in regard to the sample mean of the examples within a cluster
 - Cluster centroid or center of gravity: Sample mean value of the examples within a cluster
 - Cluster center is used to represent the cluster
 - Example: K -means algorithm
- Representative object-based technique:
 - Actual example is considered to represent the cluster
 - One representative example per cluster
 - Example: K -medoids algorithm

K-Means Clustering Algorithm

- Dividing the data into K groups or partitions
- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and K
- Target: Partition the set D into K clusters (disjoint subsets), $\{D_k\}_{k=1}^K$
 - Each of the clusters is associated with centers, μ_k , $k=1, 2, \dots, K$
 - Come up with the centers of clusters
 - Cluster center acts as a cluster representative
- Euclidean distance with center of a cluster can be used as a measure of dissimilarity

K-Means Clustering Algorithm: Training Phase

- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$ and K
1. Initialize the cluster center, $\mu_k, k=1, 2, \dots, K$ using randomly selected K data points in D
 2. Assign each data point \mathbf{x}_n to cluster center k^*
$$k^* = \arg \min_k \|\mathbf{x}_n - \mu_k\|^2 \quad \text{Squared Euclidean distance}$$
 3. Update $\mu_k, k=1, 2, \dots, K$: Re-compute μ_k after assigning all the data points.
$$\hat{\mu}_k = \frac{\sum_{D_k} \mathbf{x}_n}{N_k} \quad \begin{array}{l} D_k: \text{Data for cluster } k \\ N_k: \text{Number of examples in cluster } k \end{array}$$
 4. Repeat the steps 2 and 3 until the convergence

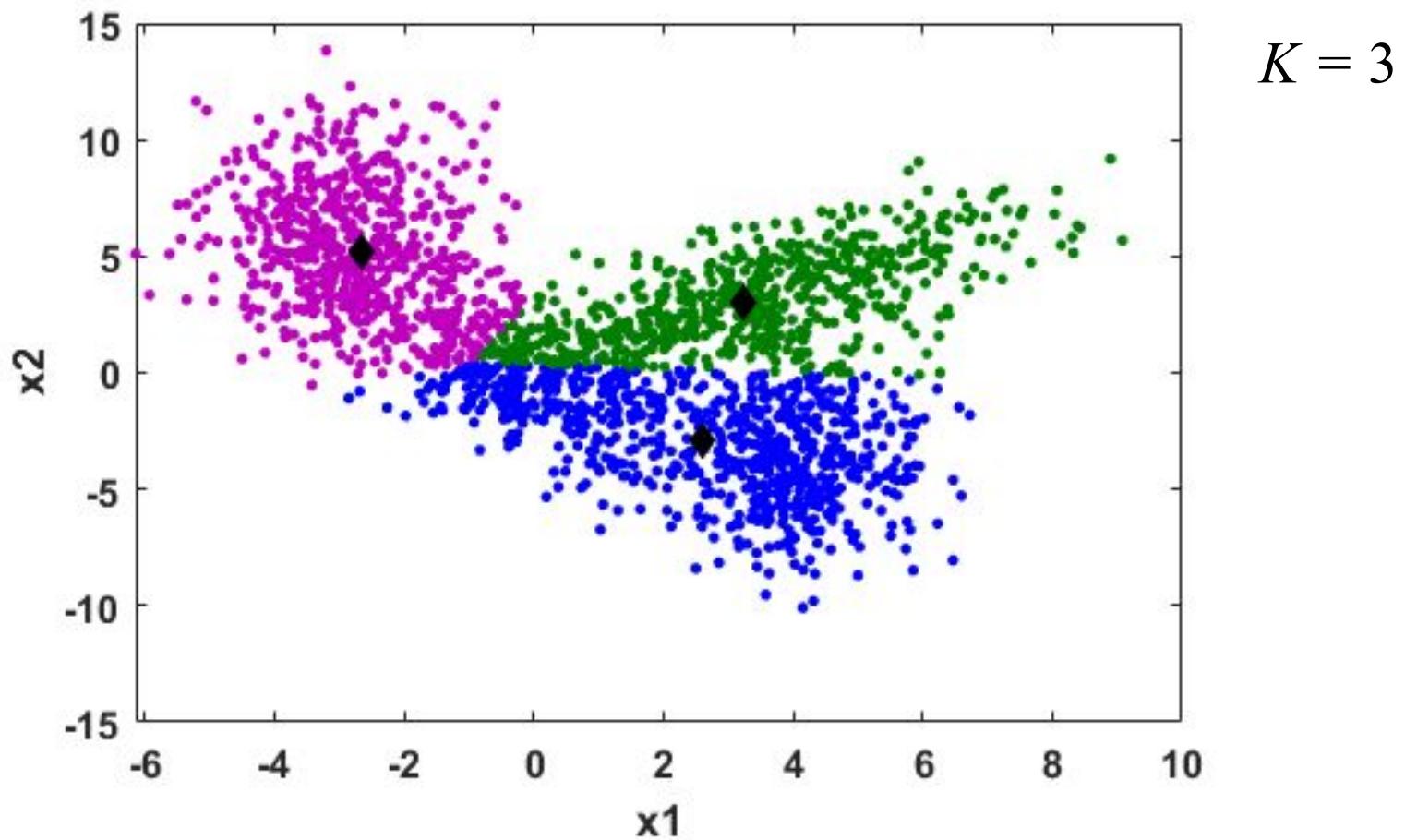
K-Means Clustering Algorithm: Training Phase

- Convergence criteria:
 - No change in the cluster assignment **OR**
 - The difference between the **distortion measure (J)** in the successive iteration falls below the threshold
 - **Distortion measure (J)** : Sum of the squares of the distance of each example to its assigned cluster center

$$J = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

z_{nk} is 1 if \mathbf{x}_n belongs to cluster k , otherwise 0

Illustration of K -Means Clustering



- Boundary between the cluster is linear
- Hard clustering: Each example must belong to exactly one group

Modified K-Means Clustering Algorithm

- Dividing the data into K groups or partitions
- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$ and K
- Target: Partition the set D into K clusters (disjoint subsets), $\{D_k\}_{k=1}^K$
 - Each of the clusters is associated with centers, $\mu_k, k=1, 2, \dots, K$
 - **Better representative for a cluster**
 - Come up with the centers of clusters and variance & covariance (covariance matrix)
 - Cluster center and covariance matrix act as cluster representatives
 - For $d=2$
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} E[x_1] \\ E[x_2] \end{bmatrix}$$
$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} E[(x_1 - \mu_1)^2] & E[(x_1 - \mu_1)(x_2 - \mu_2)] \\ E[(x_2 - \mu_2)(x_1 - \mu_1)] & E[(x_2 - \mu_2)^2] \end{bmatrix}$$
- Mahalanobis distance with cluster representatives can be used as a measure of dissimilarity

Modified K-Means Clustering Algorithm: Training Phase

- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^d$ and K
1. Initialize the cluster center, $\boldsymbol{\mu}_k$, $k=1, 2, \dots, K$ using randomly selected K data points in D
 2. Initialize the covariance matrix, $\boldsymbol{\Sigma}_k$, $k=1, 2, \dots, K$ using unit matrix
 3. Assign each data point \mathbf{x}_n to cluster center k^*

$$k^* = \arg \min_k (\mathbf{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) \quad \begin{matrix} \textit{Squared Mahalanobis} \\ \textit{distance} \end{matrix}$$

4. Update $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, $k=1, 2, \dots, K$: Re-compute $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ after assigning all the data points.

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{D_k} \mathbf{x}_n}{N_k} \quad \hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{D_k} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k)^\top}{N_k} \quad \begin{matrix} D_k: \text{Data for cluster } k \\ N_k: \text{Number of examples in cluster } k \end{matrix}$$

5. Repeat the steps 3 and 4 until the convergence

Modified K-Means Clustering Algorithm: Training Phase

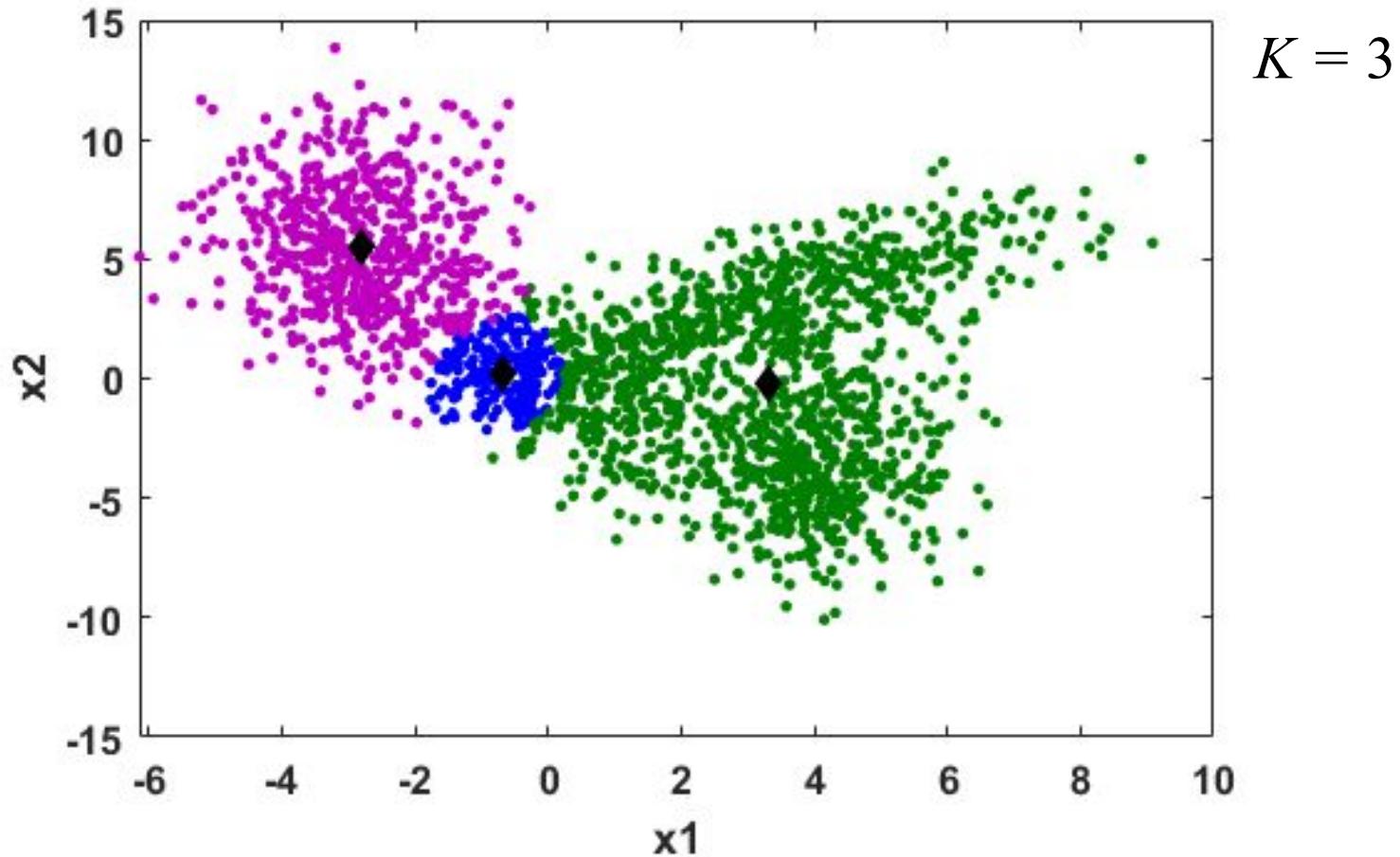
- Convergence criteria:
 - No change in the cluster assignment **OR**
 - The difference between the distortion measure (J) in the successive iteration falls below the threshold

$$J = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \left[(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) \right]$$

z_{nk} is 1 if \mathbf{x}_n belongs to cluster k , otherwise 0

- Hard clustering: Each example must belong to exactly one group

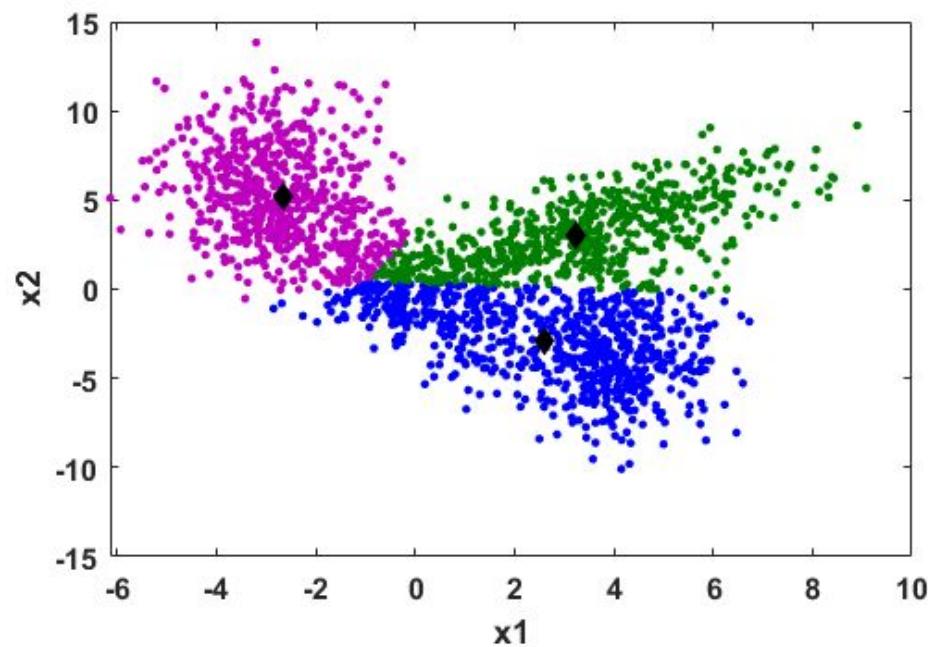
Illustration of K -Means Clustering



- Boundary between the cluster is quadratic
- Hard clustering: Each example must belong to exactly one group

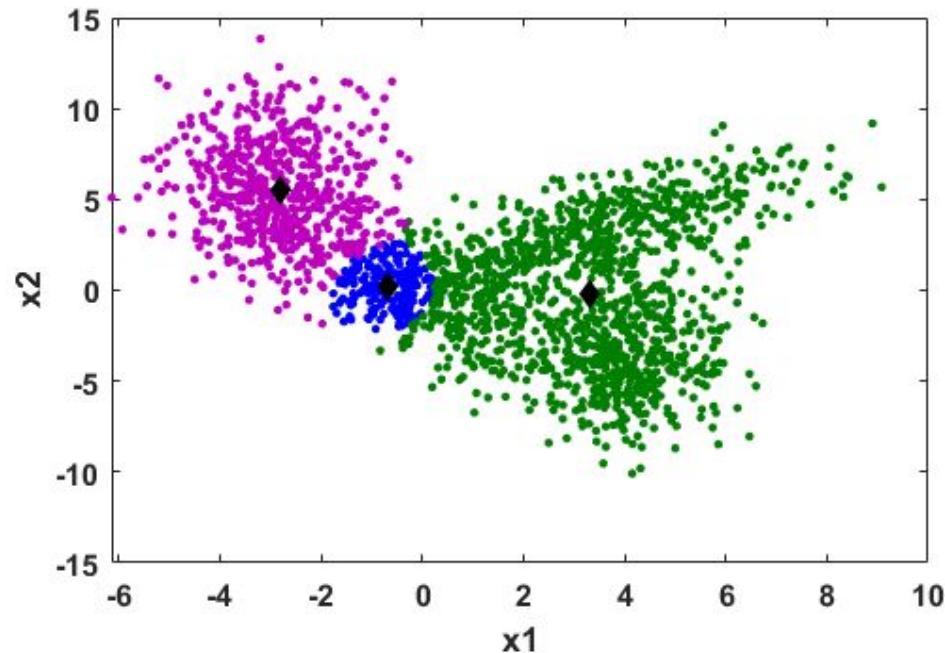
Illustration of K -Means Clustering

$$K = 3$$



*Measure of
Dissimilarity:*

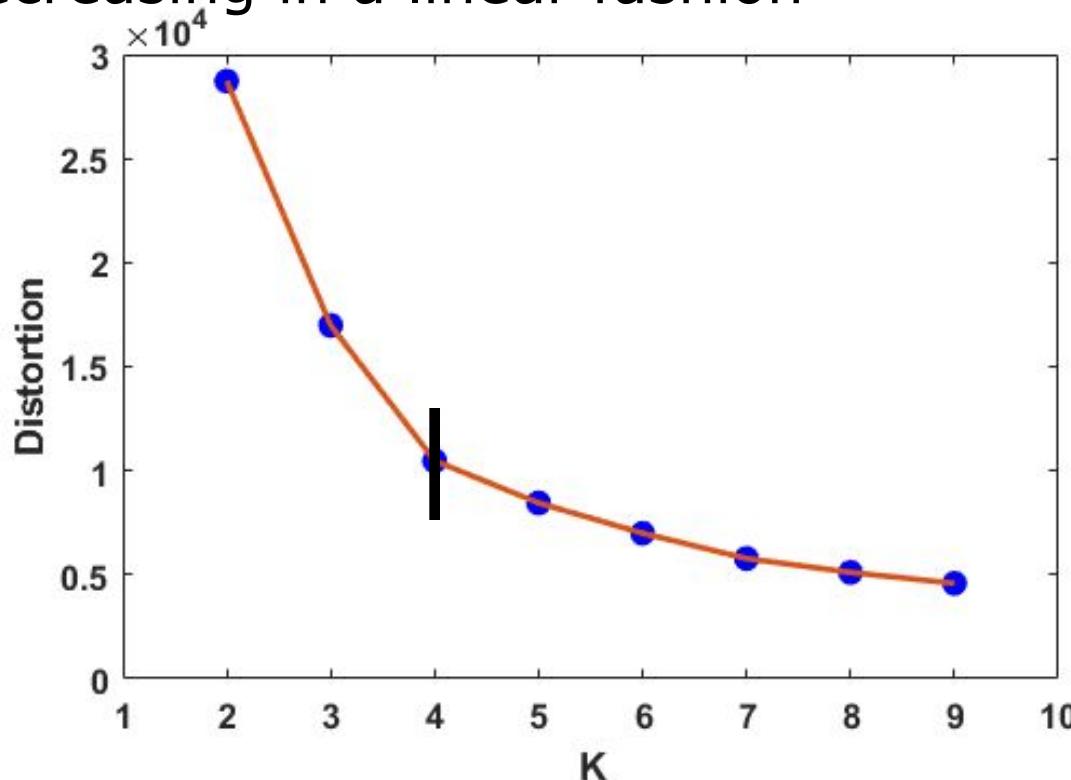
*Euclidian
Distance*



*Mahalanobis
Distance*

Elbow Method to Choose K

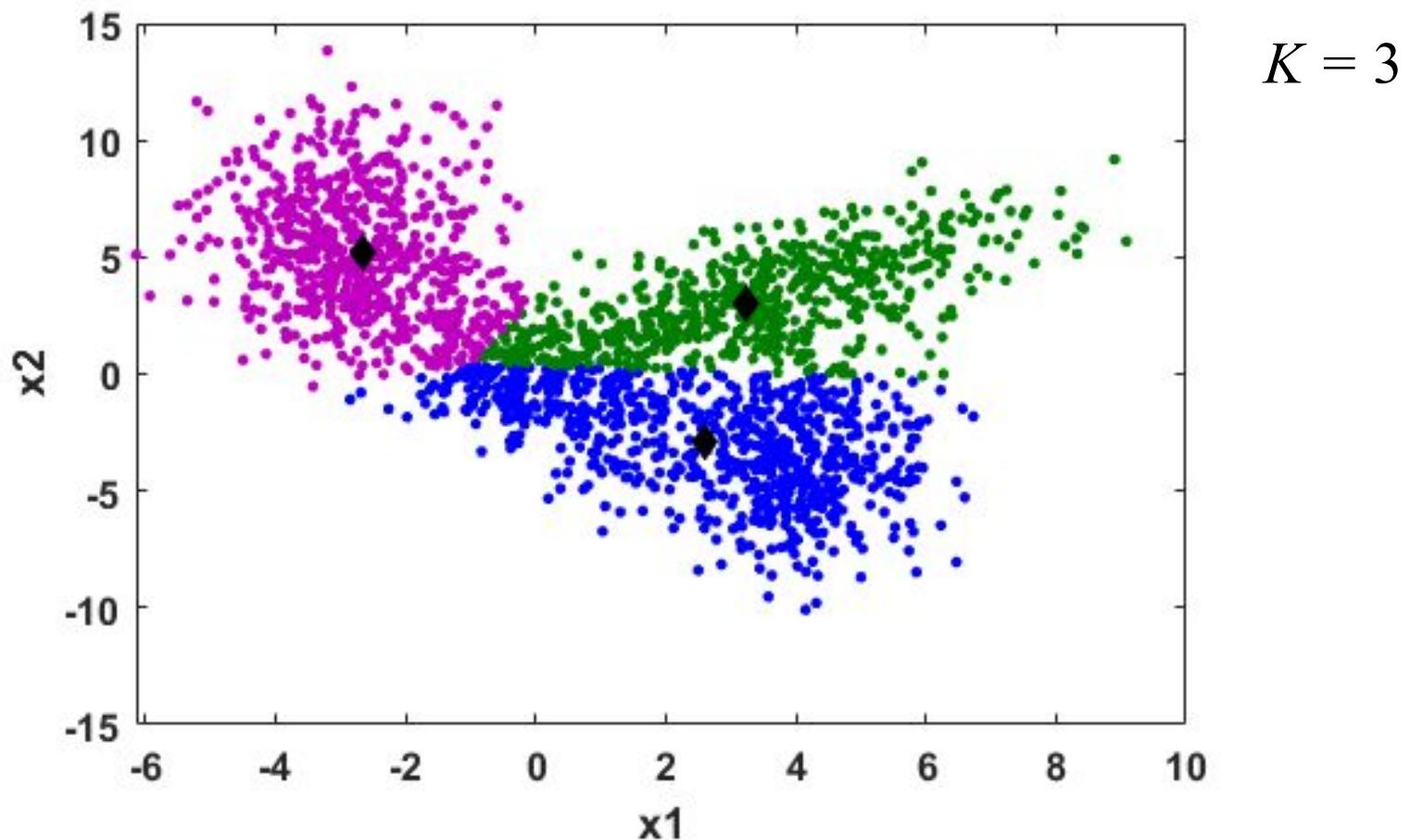
- Determine the distortion measure for different values of K
- Plot the K vs Distortion
- Optimal number of clusters: Select the value of K at the “elbow” i.e. the point after which the distortion start decreasing in a linear fashion



K-Means Clustering Algorithm: Training Phase

- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$ and K
1. Initialize the cluster center, $\boldsymbol{\mu}_k, k=1, 2, \dots, K$ using randomly selected K data points in D
 2. Assign each data point \mathbf{x}_n to cluster center k^*
$$k^* = \arg \min_k \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \quad \text{Squared Euclidean distance}$$
 3. Update $\boldsymbol{\mu}_k, k=1, 2, \dots, K$: Re-compute $\boldsymbol{\mu}_k$ after assigning all the data points.
$$\boldsymbol{\mu}_k = \frac{\sum_{D_k} \mathbf{x}_n}{N_k} \quad \begin{array}{l} D_k: \text{Data for cluster } k \\ N_k: \text{Number of examples in cluster } k \end{array}$$
 4. Repeat the steps 2 and 3 until the convergence

Illustration of K -Means Clustering



- Boundary between the cluster is linear
- Hard clustering: Each example must belong to exactly one group

***K*-Medoid Clustering Algorithms**

- Related to K -means clustering
- The K -means algorithm is sensitive to outliers because an example with extremely large value may substantially distort the distribution of data
- Solution: One of the data points is chosen as representative of cluster, instead of mean value of the cluster
- It replaces the means of cluster with modes
- Partitioning around medoids
- A medoid of a finite dataset: The data point from the set, whose average dissimilarity (distance) to all the points is minimal
 - The most centrally located point in the set

K-Medoid Clustering Algorithm

- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$ and K
1. Initialize the medoid, $\mathbf{x}_k^{\text{init}}, k=1, 2, \dots, K$ using randomly selected K data points in D
 2. Assign each data point \mathbf{x}_n to the closest medoid
$$k^* = \arg \min_k \|\mathbf{x}_n - \mathbf{x}_k\|^2 \quad \text{Squared Euclidian distance}$$
 3. Update medoids $\mathbf{x}_k, k=1, 2, \dots, K$
 - For each data point \mathbf{x}_n assigned to a cluster k compute the average dissimilarity (distance) of \mathbf{x}_n to all the data points assigned to cluster k
$$\text{Average dissimilarity for } \mathbf{x}_n = \frac{\sum_{m \in D_k} \|\mathbf{x}_n - \mathbf{x}_m\|^2}{N_k} \quad N_k: \text{Number of examples in cluster } k$$
 - Select the example with minimum average dissimilarity as medoid
 1. Repeat the steps 2 and 3 until the convergence

K-Medoid Clustering Algorithm

- Convergence criteria:
 - No change in the cluster assignment **OR**
 - The difference between the distortion measure (absolute-error) (J) in the successive iteration falls below the threshold
 - Distortion measure (J) : Sum of the squares of the distance of each example to its corresponding reference point (medoid)

$$J = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \hat{\mathbf{x}}_k\|^2$$

z_{nk} is 1 if \mathbf{x}_n belongs to cluster k , otherwise 0

- Optimal number of clusters (k) can obtained using elbow method

Evaluation of Clustering: Purity Score

- Lets us assume that class index for each example is given
- **Purity score:** Purity is a measure of the extent to which clusters contain a single class
 - For each cluster, count the number of data points from the most common class
 - Take the sum over all clusters and divide by the total number of data points
- Let M be the number of classes, $C_1, C_2, \dots, C_m, \dots, C_M$
- Let K be the number of clusters, $k = 1, 2, \dots, K$
- Let N be the number of data points

Evaluation of Clustering: Purity Score

- For each cluster k ,
 - Count the number of data points from each class
 - Consider the number of data points of most common class

$$\max_m |N_k \cap C_m|$$

$|N_k \cap C_m|$ is the number of data points in k^{th} cluster belonging to class m

- Take the sum over all clusters, k
- Divide by the total number of data points (N)

$$\text{Purity Score} = \frac{1}{N} \sum_{k=1}^K \max_m |N_k \cap C_m|$$

Illustration of Computing Purity Score

- Number of data points, $N = 25$
- number of classes, $M = 3$
- number of clusters, $K = 3$

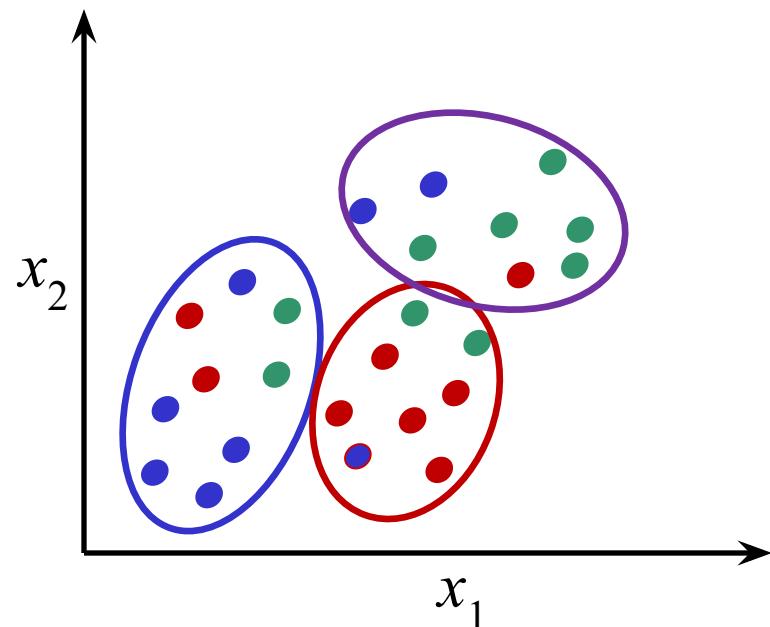


Illustration of Computing Purity Score

- Number of data points, $N = 25$
- number of classes, $M = 3$
- number of clusters, $K = 3$
- *Cluster 1:*
 - Number of examples of Blue Class are more, i.e. **5**

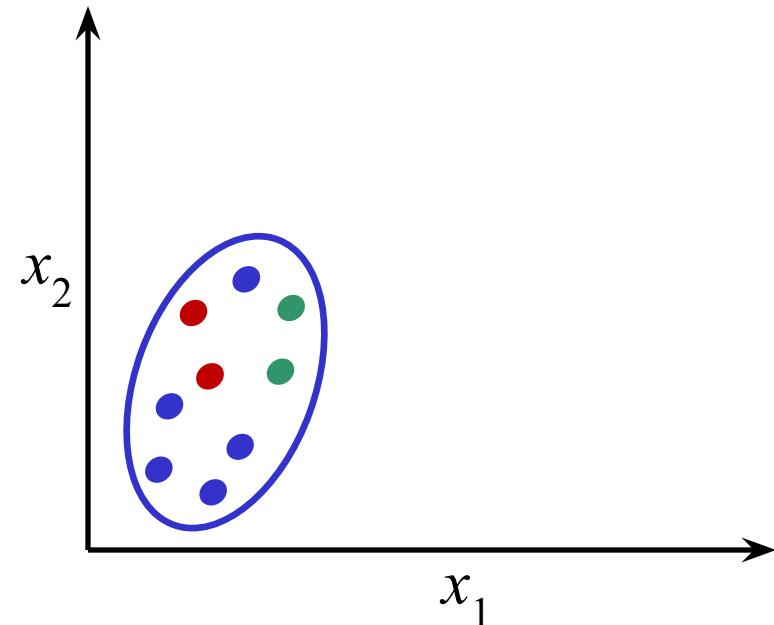


Illustration of Computing Purity Score

- Number of data points, $N = 25$
- number of classes, $M = 3$
- number of clusters, $K = 3$
- *Cluster 1*:
 - Number of examples of **Blue Class** are more, i.e. **5**
- *Cluster 2*:
 - Number of examples of **Red Class** are more, i.e. **5**

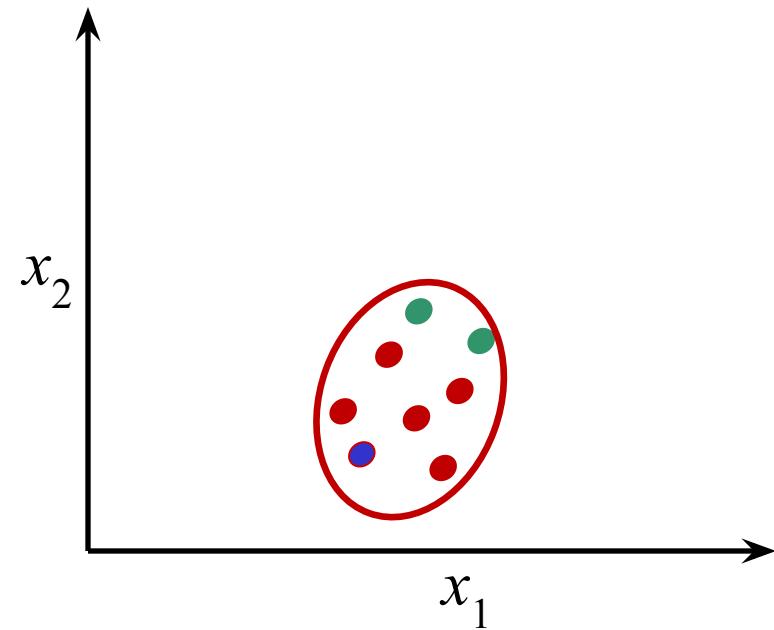


Illustration of Computing Purity Score

- Number of data points, $N = 25$
- number of classes, $M = 3$
- number of clusters, $K = 3$
- *Cluster 1*:
 - Number of examples of **Blue Class** are more, i.e. **5**
- *Cluster 2*:
 - Number of examples of **Red Class** are more, i.e. **5**
- *Cluster 3*:
 - Number of examples of **Green Class** are more, i.e. **5**

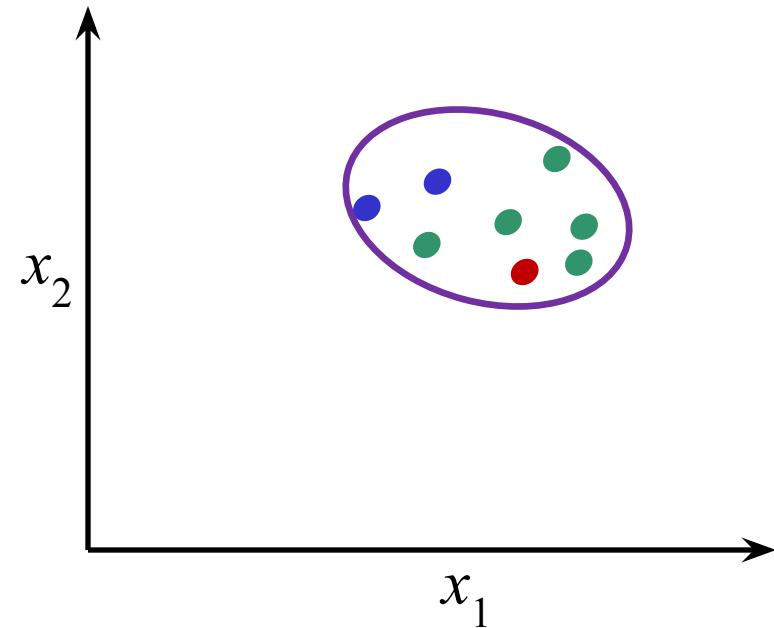
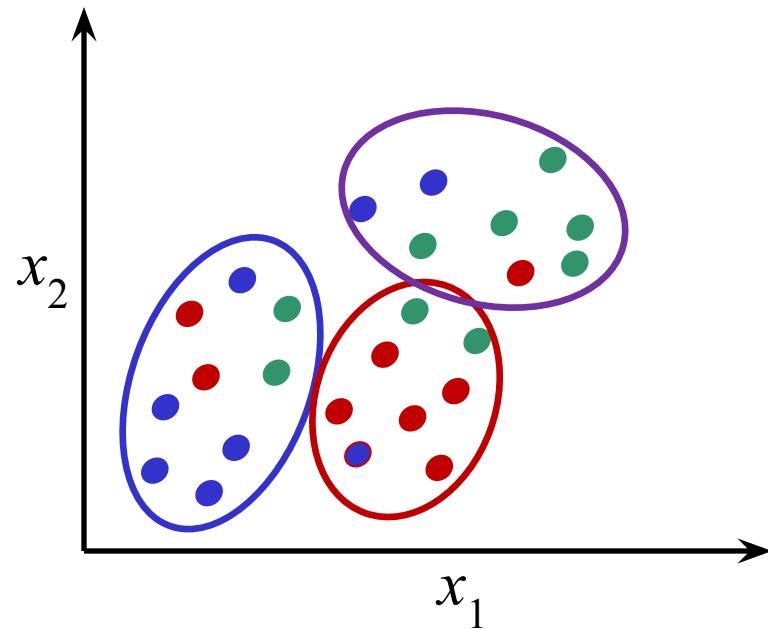


Illustration of Computing Purity Score

- Number of data points, $N = 25$
- number of classes, $M = 3$
- number of clusters, $K = 3$
- *Cluster 1*:
 - Number of examples of **Blue Class** are more, i.e. **5**
- *Cluster 2*:
 - Number of examples of **Red Class** are more, i.e. **5**
- *Cluster 3*:
 - Number of examples of **Green Class** are more, i.e. **5**
- **Purity score: $(5+5+5)/25 = 0.60$**



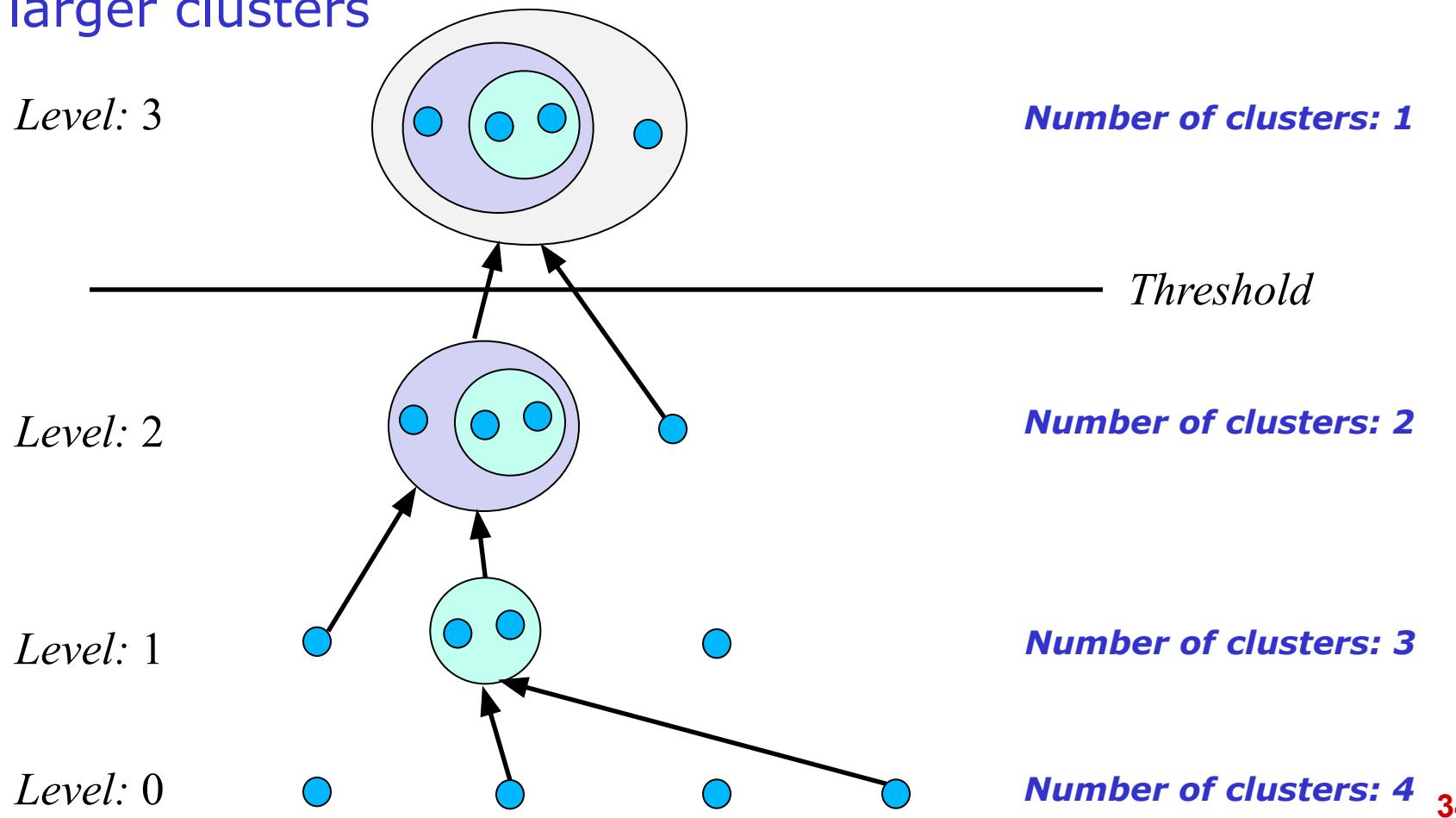
Hierarchical Clustering

Hierarchical Clustering Algorithms

- These methods create a **hierarchical decomposition** of the collection of examples
- Produce nested sequence of data partitions
- These sequence can be depicted using a tree structure
- Hierarchical clustering method works by grouping data points into a **tree of clusters**
- Hierarchical algorithms are either **agglomerative** or **divisive**
- This classification of hierarchical clustering is depending on whether the hierarchical decomposition is formed in a
 - Bottom-up (merging) OR
 - Top-down (splitting) fashion
- Need not have to specify the number of clusters

Agglomerative Hierarchical Clustering

- Bottom-up approach
- This strategy starts by placing each example in its own cluster (atomic clusters or singleton clusters) and then merges these atomic clusters into larger and larger clusters



Agglomerative Hierarchical Clustering

- Bottom-up approach
- This strategy starts by placing each example in its own cluster (atomic clusters) and then merges these atomic clusters into larger and larger clusters
- Starts with N clusters where each example is a cluster
- At each successive step (level), the most similar pair of clusters are merged
 - The measure of closeness (intercluster similarity) is considered to decide which two clusters are merged
 - At each level, number of clusters is reduced by one
- The process continues till all the examples are in a single cluster or until certain termination conditions are satisfied
 - Termination condition could be
 - Number of clusters
 - Intercluster similarity between each pair of cluster is within a certain threshold

Agglomerative Hierarchical Clustering

- **Once two examples are placed in the same cluster at a level, they remain in same cluster at all subsequent levels**
- Example: AGglomerative NESting (AGNES)
- Most hierarchical clustering methods belong to this category
- They differ only in their definition of **inter cluster similarity**
- Intercluster similarity is to identifying two closest cluster for merging
- When there is **one example** in a cluster, two closest clusters are found by computing **minimum Euclidean distance** between two clusters
- However, there is **no unique way** to find the two closest clusters when there are **more than one data points** in each clusters

Agglomerative Hierarchical Clustering

- Different **intercluster similarities** to find similarity between the clusters having **more than one examples**:

1. **Minimum distance** between any two examples from two clusters C_i and C_j

$$d_{\min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

- Select a pair of clusters for merging whose **minimum distance between any two examples** is **minimum of all the pair of clusters**

2. **Distance between the centers** of two clusters C_i and C_j

$$d_{mean}(C_i, C_j) = \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$$

- Where $\boldsymbol{\mu}_i$ is the center of C_i and $\boldsymbol{\mu}_j$ is the center of C_j
- Select a pair of clusters for merging whose **distance between the centers** is **minimum of all the pair of clusters**

Agglomerative Hierarchical Clustering

- Different **intercluster similarities** to find similarity between the clusters having **more than one examples**:
 3. Average distance of all the points in one cluster (C_i) to all the points in another cluster (C_j)

$$d_{avg}(C_i, C_j) = \frac{1}{N_i N_j} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

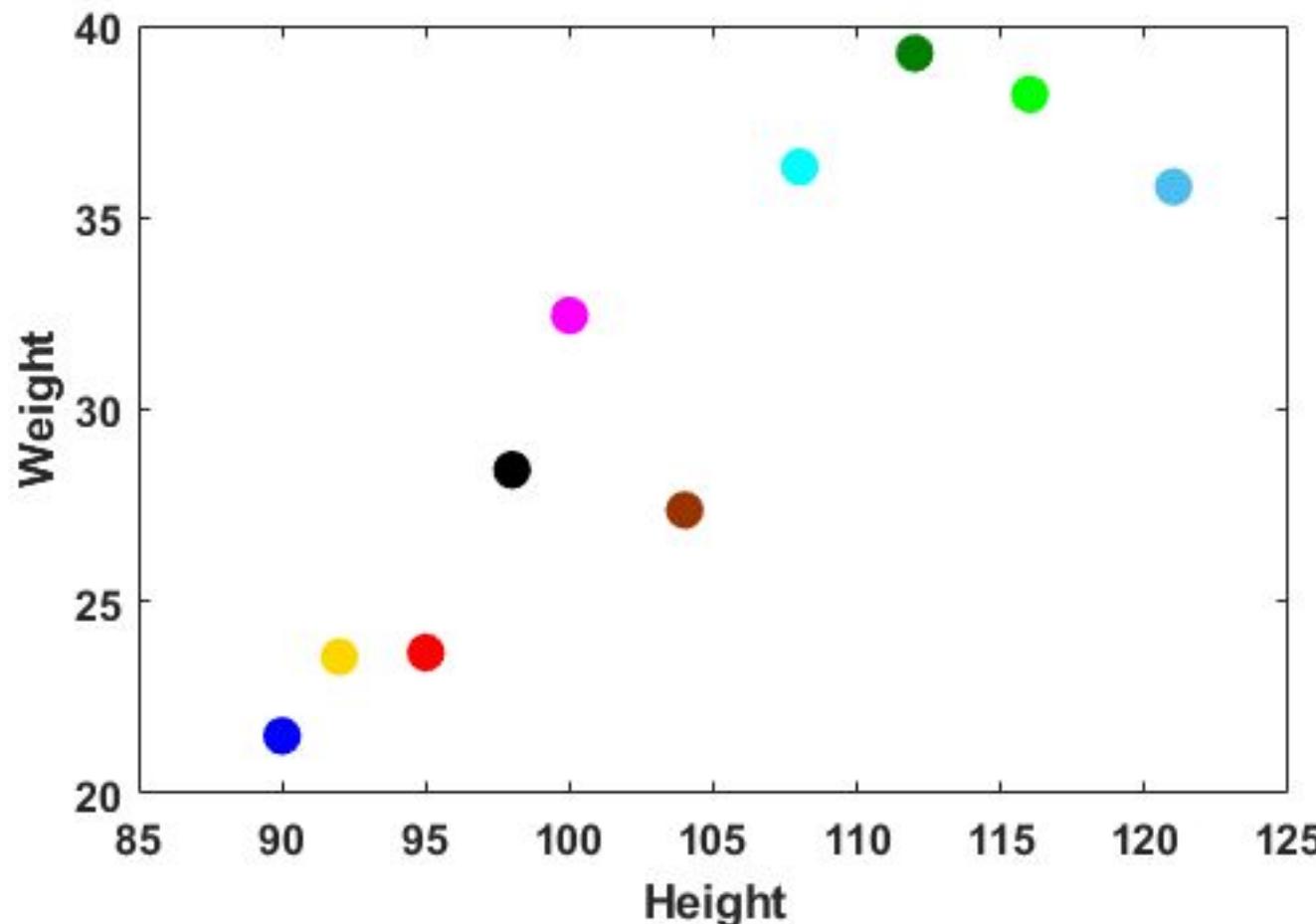
- Where N_i and N_j are the number of examples in clusters C_i and C_j respectively
- Select a pair of clusters for merging whose average distance is **minimum than that of all the pair of clusters**

Agglomerative Hierarchical Clustering

- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$
- Target: Partition the data
- Step1: N clusters where each example is a cluster
- Step2: Compute intercluster similarity between each pair of clusters
- Step3: Choose a pair of clusters that are most similar (minimum intercluster distance) and merge them
- Step4: Repeat Step2 and Step3 until all the examples are in a single cluster or until certain termination conditions are satisfied

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

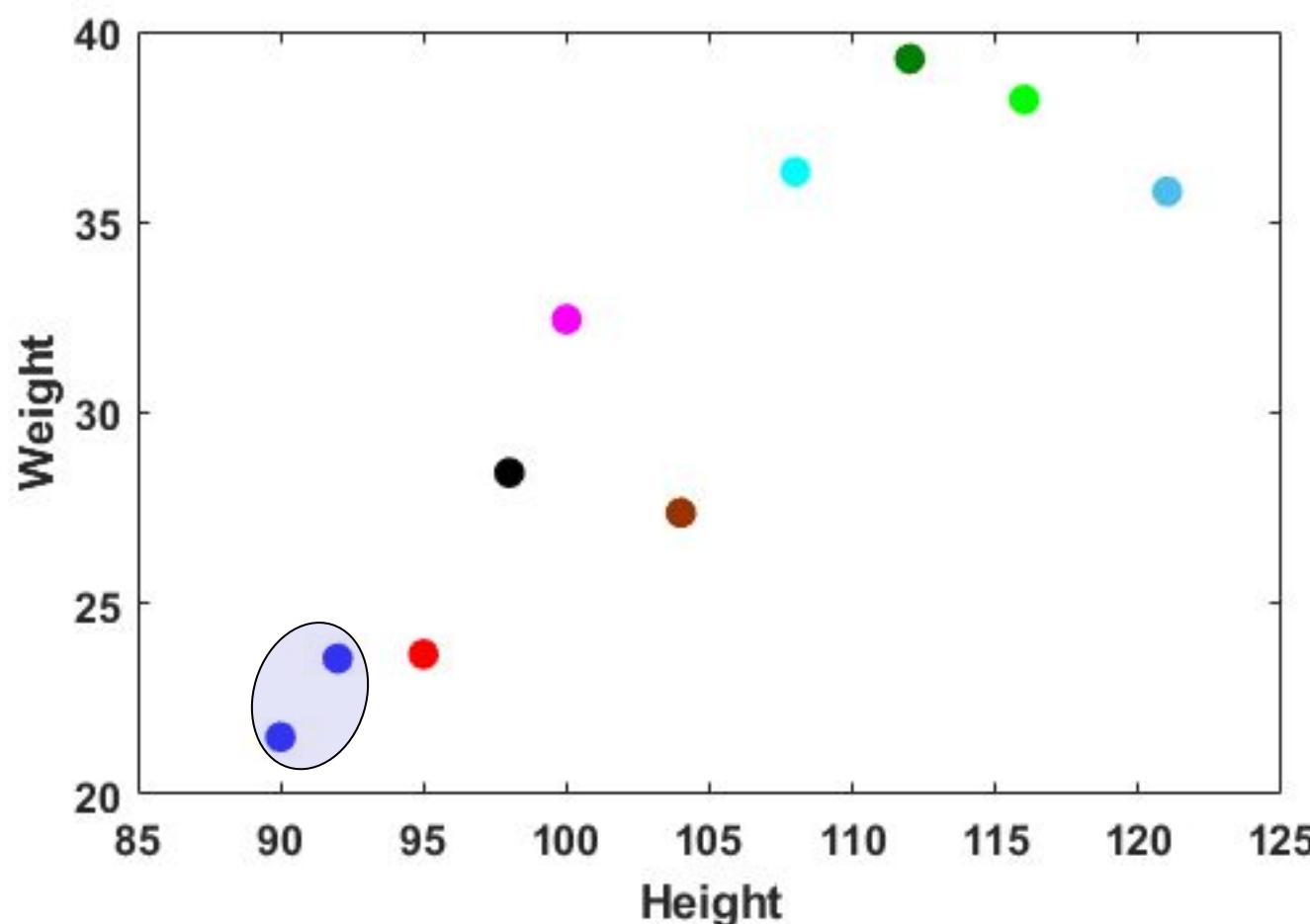


Level: 0

Number of clusters: 10

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

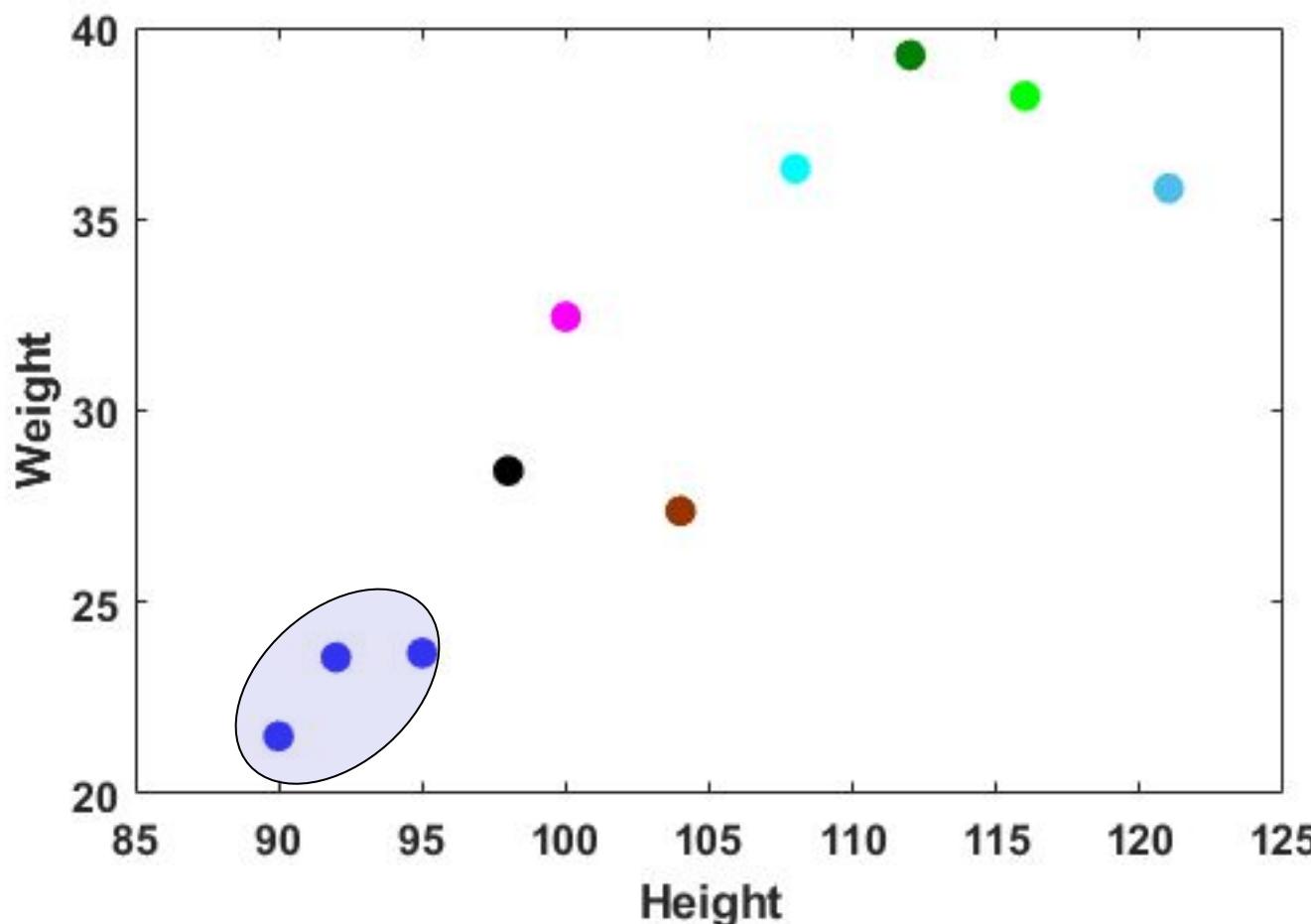


Level: 1

Number of clusters: 9

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

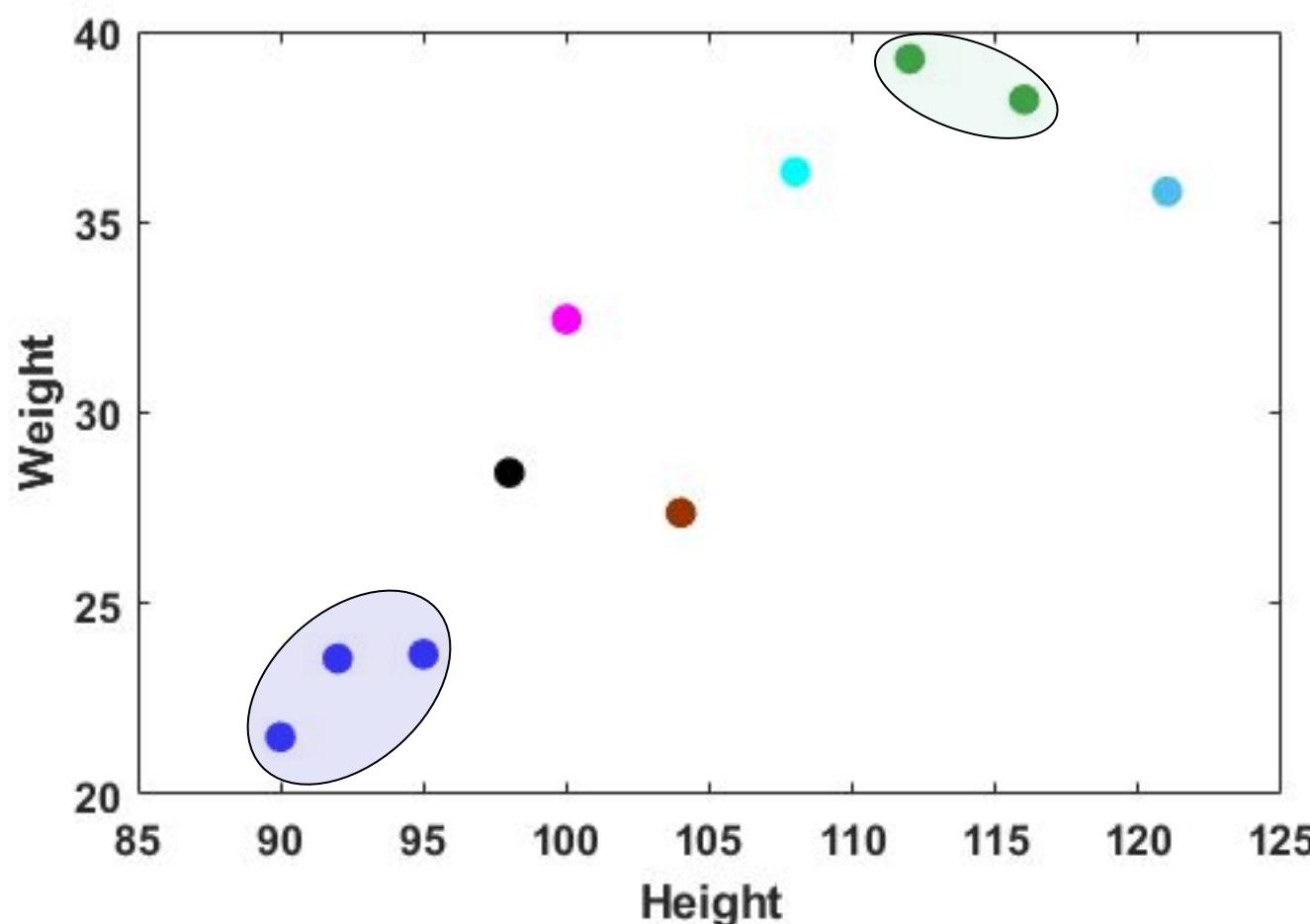


Level: 2

Number of clusters: 8

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

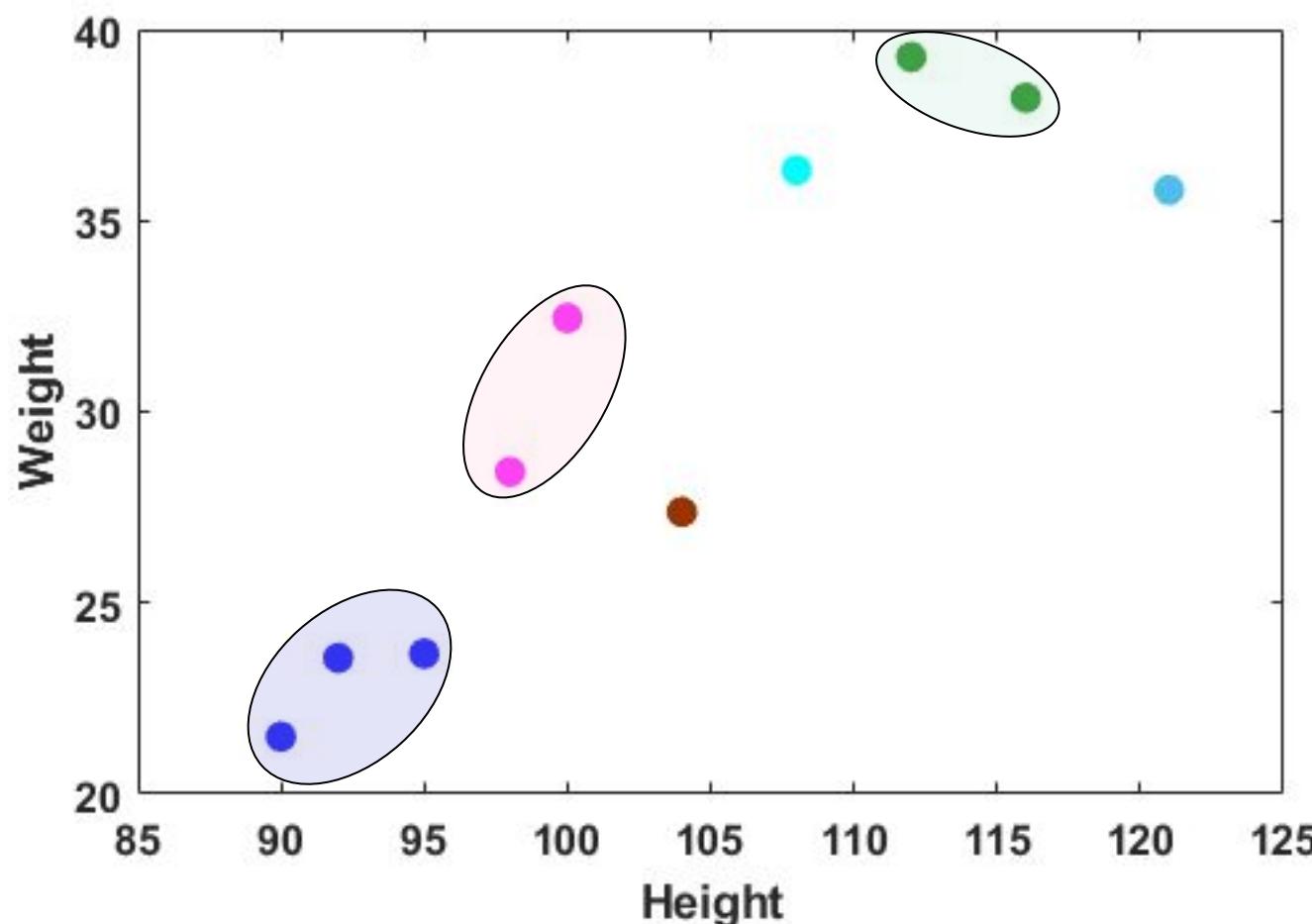


Level: 3

Number of clusters: 7

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

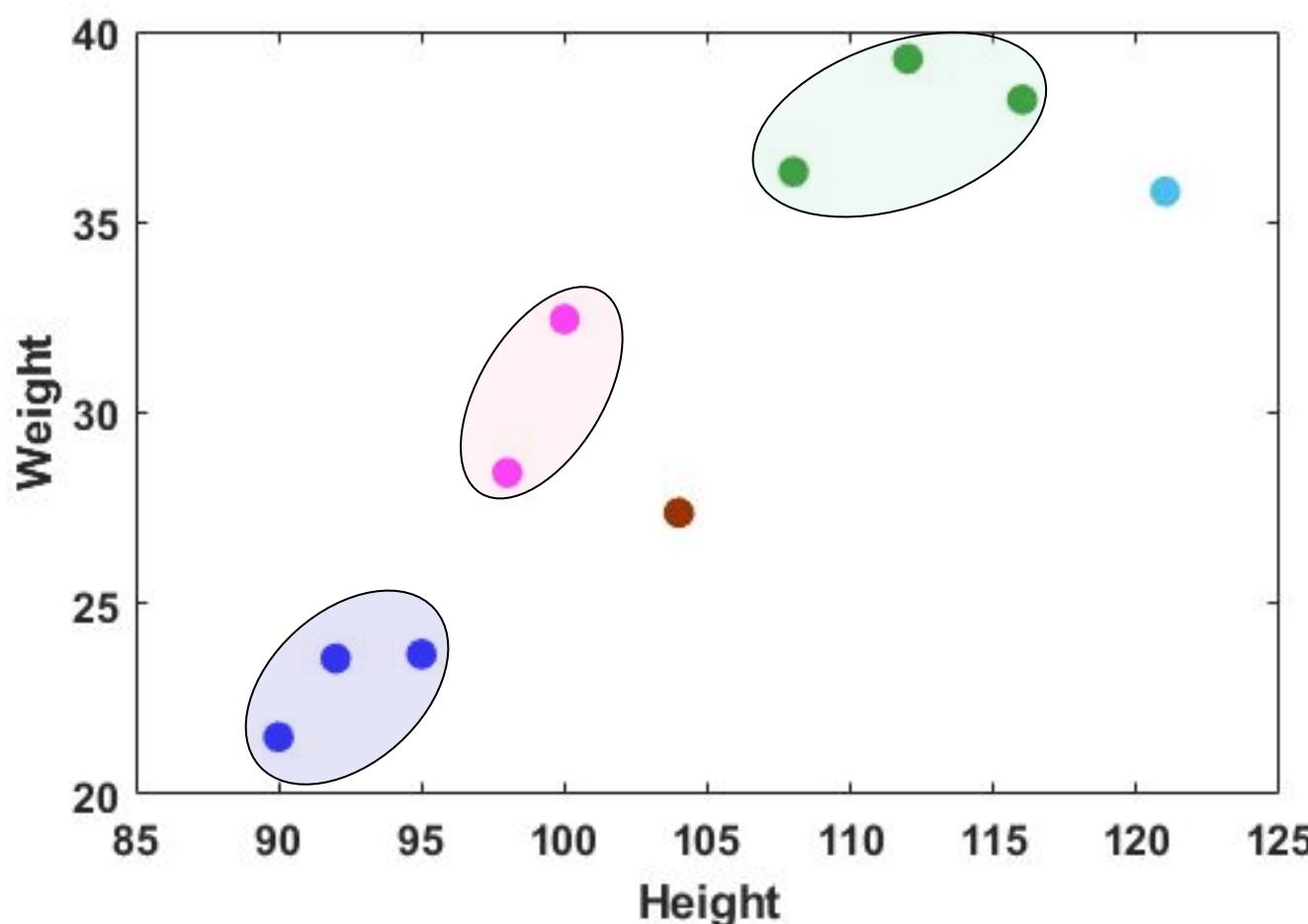


Level: 4

Number of clusters: 6

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

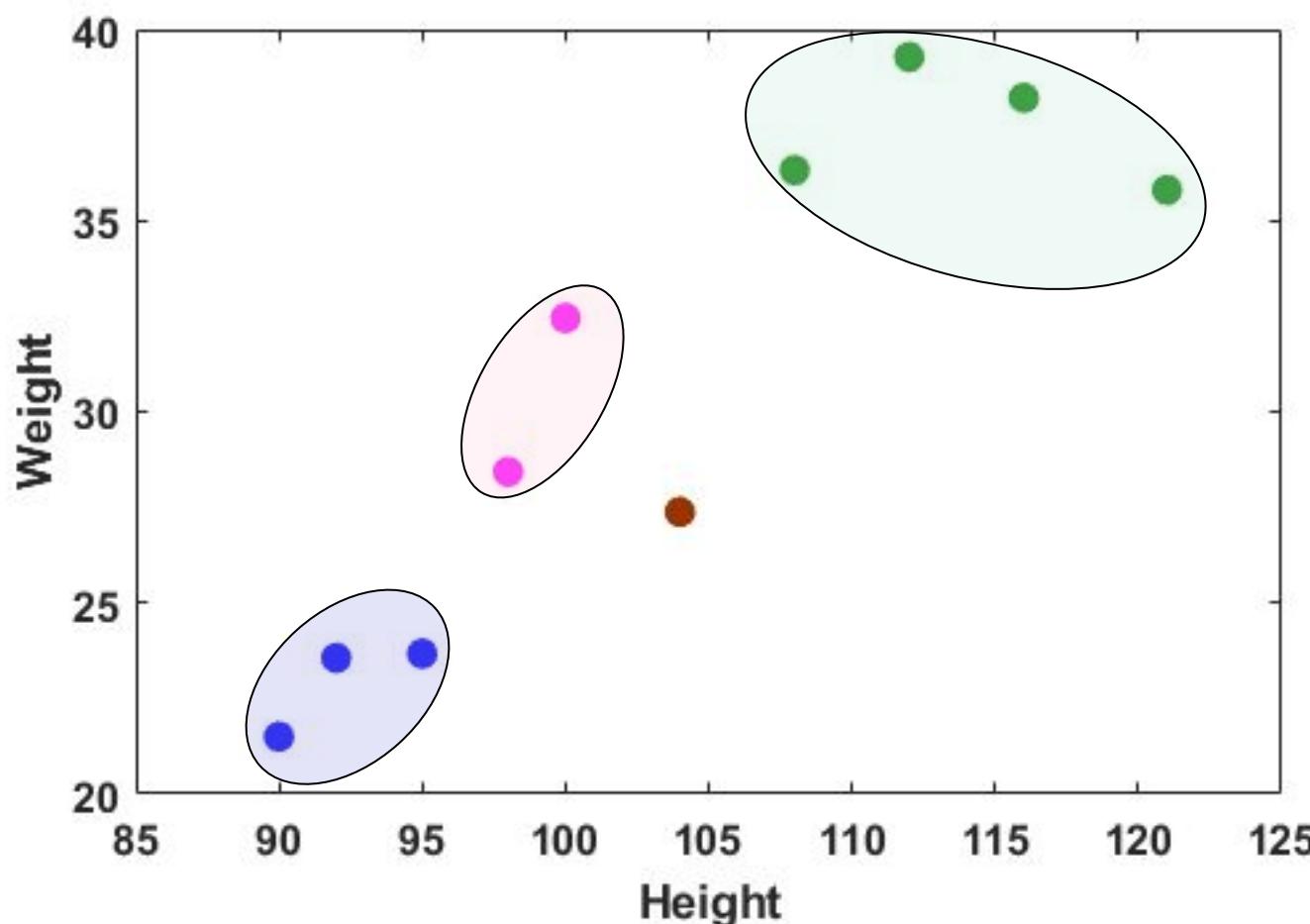


Level: 5

Number of clusters: 5

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

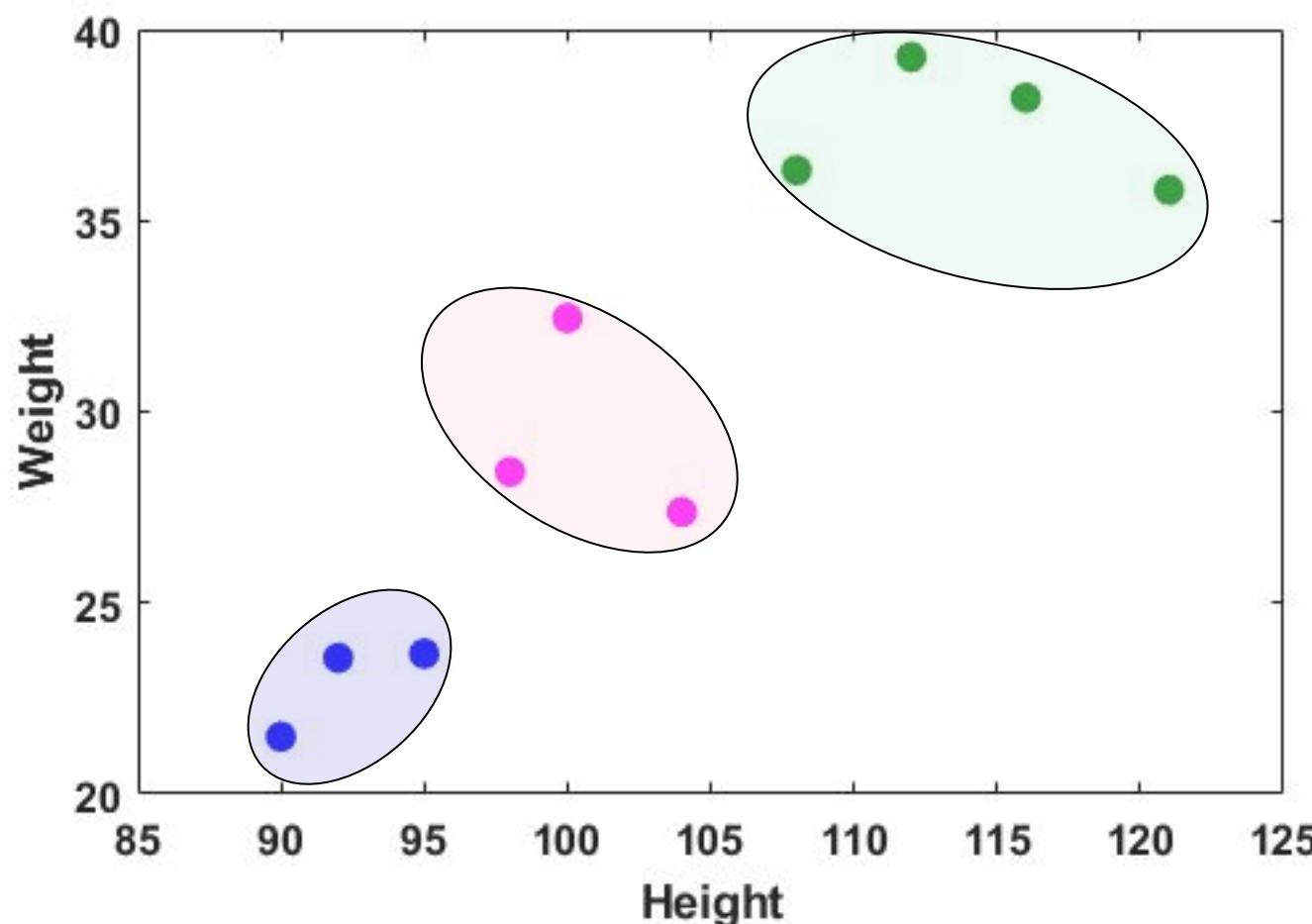


Level: 6

Number of clusters: 4

Illustration: Agglomerative Hierarchical Clustering

- Intercluster similarity:
 - Single example in clusters: Euclidian distance
 - More than one examples in cluster: Distance between the centres of two clusters

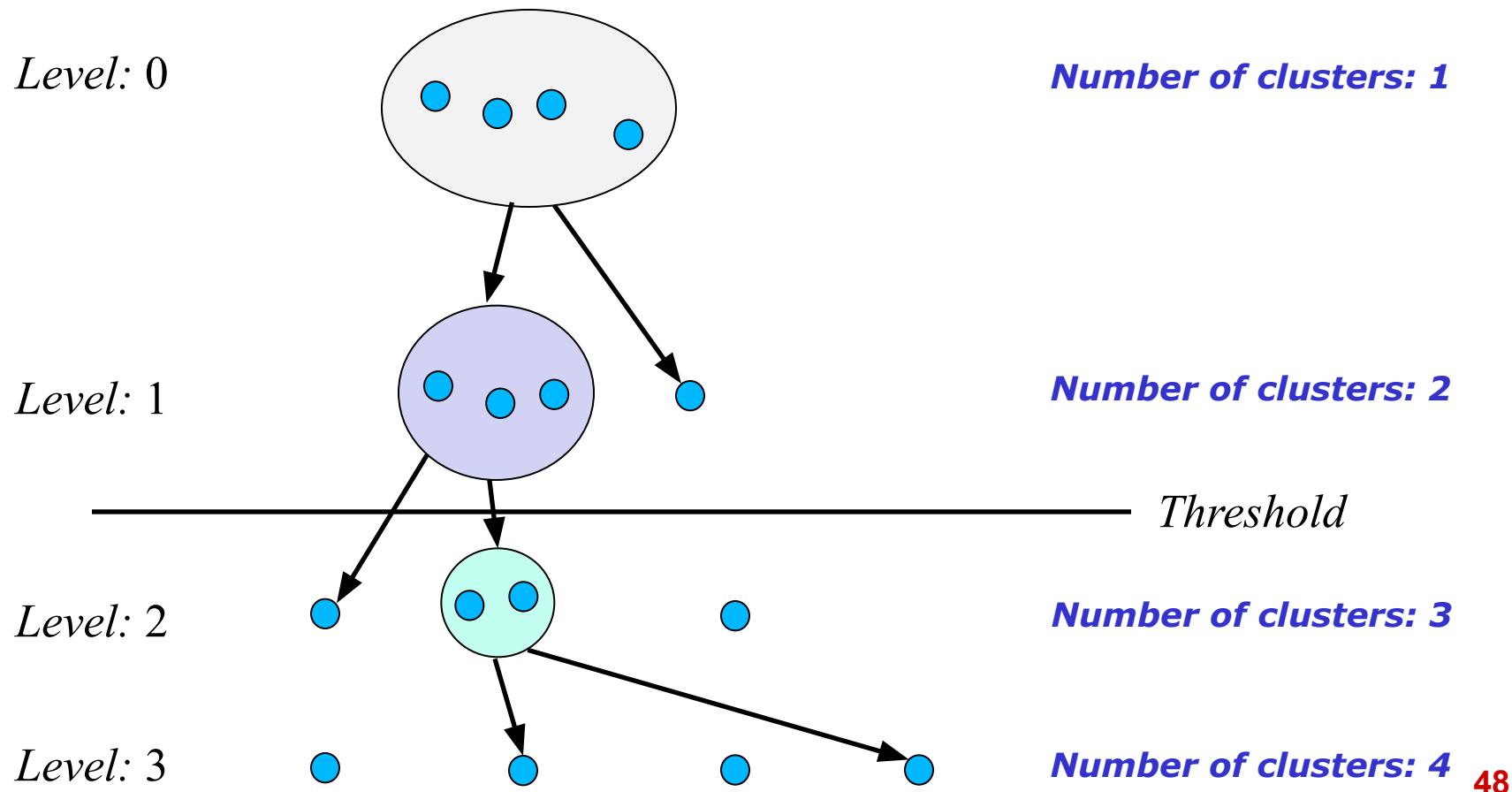


Level: 7

Number of clusters: 3

Divisive Hierarchical Clustering

- Top-down approach
- Starts with single cluster having all the examples
- It subdivides the cluster into smaller and smaller clusters in the successive step



Divisive Hierarchical Clustering

- Top-down approach
- Starts with single cluster having all the examples
- It subdivides the cluster into smaller and smaller clusters in the successive step
- At each successive step, a compactness measure is used to choose which cluster to split
 - Compactness measure: Average value of distance between the data points of a cluster
 - Compactness measure (CM_i) of a cluster C_i :

$$CM_i = \frac{1}{N_i^2} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_i} \|\mathbf{x} - \mathbf{x}'\|$$

- Where N_i is the number of examples in cluster C_i
- Choose the cluster with larger value of compactness measure to split

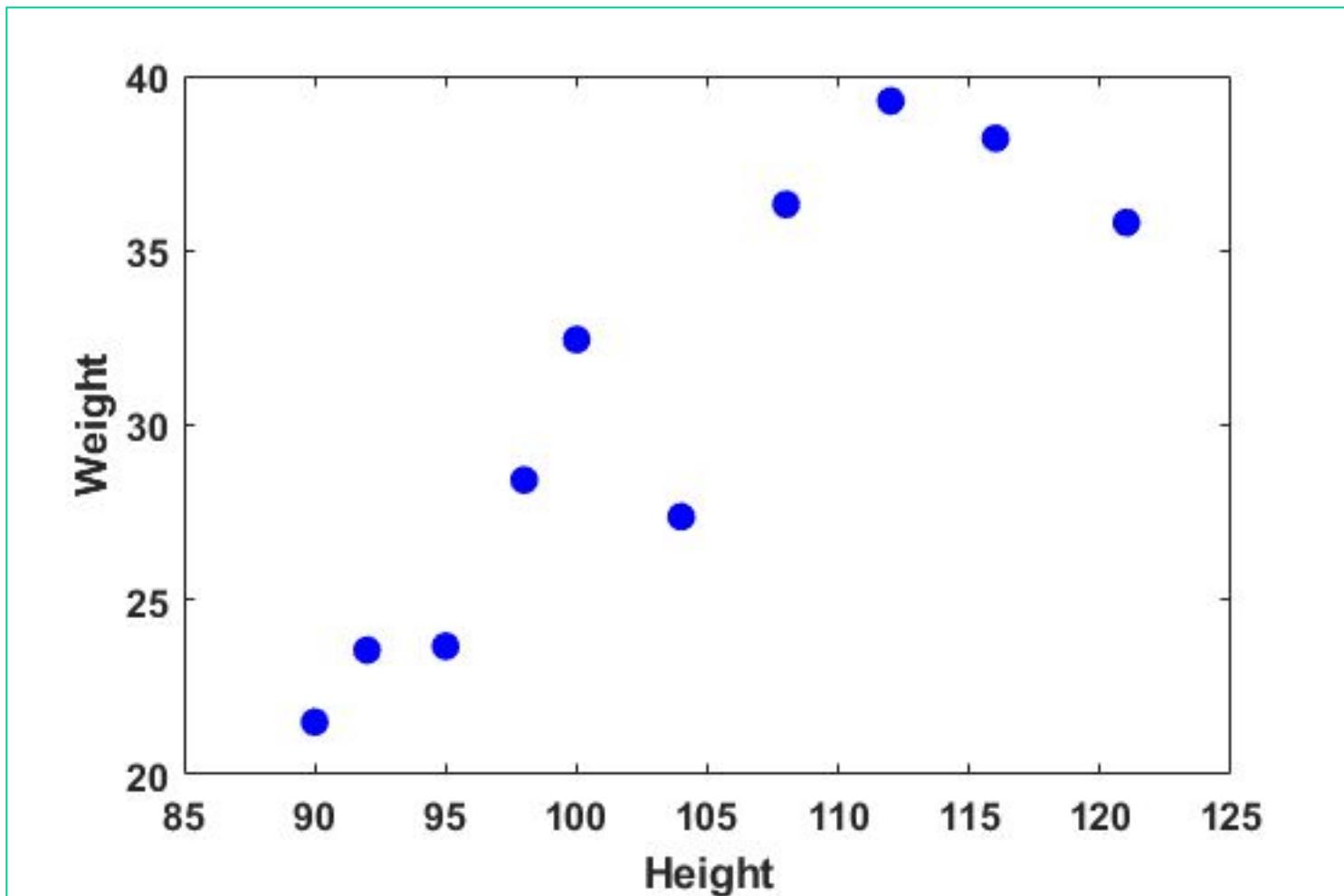
Divisive Hierarchical Clustering

- To split a cluster, find a pair of examples having maximum Euclidian distance and split around these two examples (keeping them as centroids)
- At each level, number of clusters is increases by one
- The process continues until each example forms a cluster (atomic or singleton cluster) or until it satisfies certain termination condition
- Termination condition could be
 - Number of clusters
 - Compactness measure of each cluster is within a certain threshold
- ***Once two examples are placed in two different clusters at a level, they remain in different clusters at all subsequent levels***
- Example: DIvisive ANAlysis (DIANA)

Divisive Hierarchical Clustering

- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$
- Target: Partition the data
- Step1: Single cluster having all the examples
- Step2: Find a pair of examples with in a cluster having maximum Euclidian distance
 - These examples act as centroid
- Step3: Split into two clusters by assigning each data point to one of these two examples using Euclidian distance
- Step4: Compute compactness measure for each cluster
 - Step5: Choose the cluster with larger value of compactness measure to split
- Step6: Repeat Step2 to Step5 until each example forms a cluster (atomic or singleton cluster) or until it satisfies certain termination condition

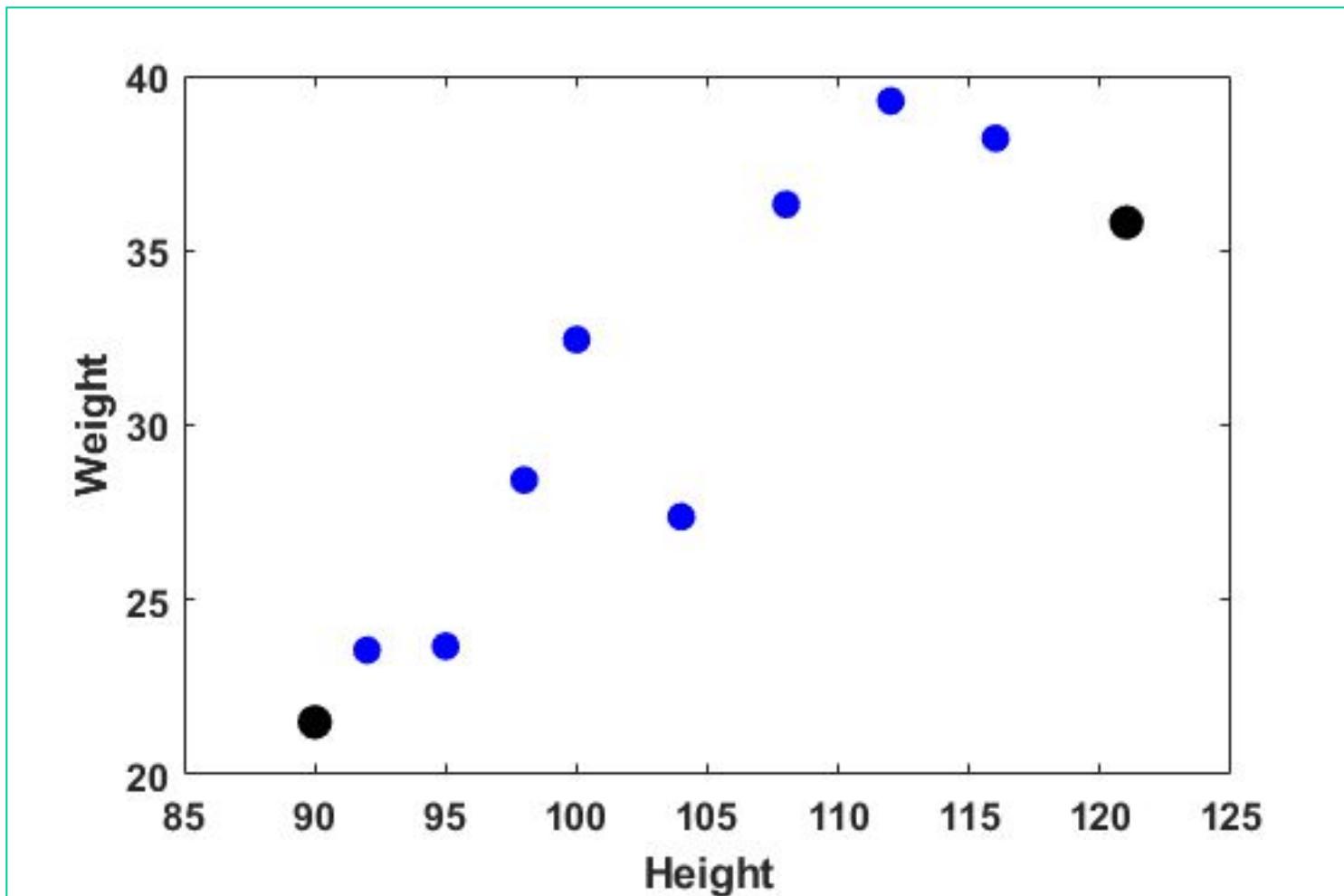
Divisive Hierarchical Clustering



Level: 0

Number of clusters: 1

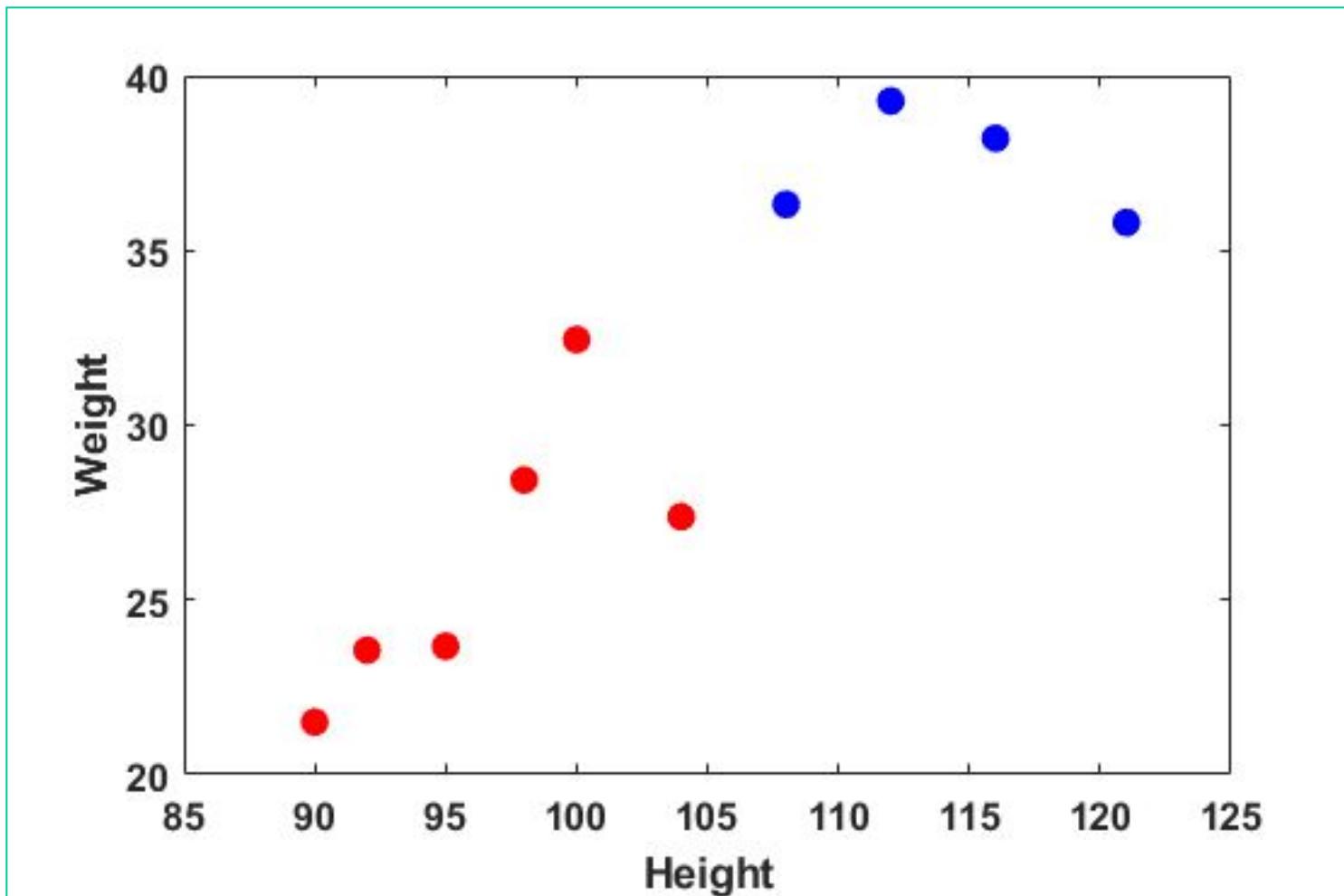
Divisive Hierarchical Clustering



Level: 0

Number of clusters: 1

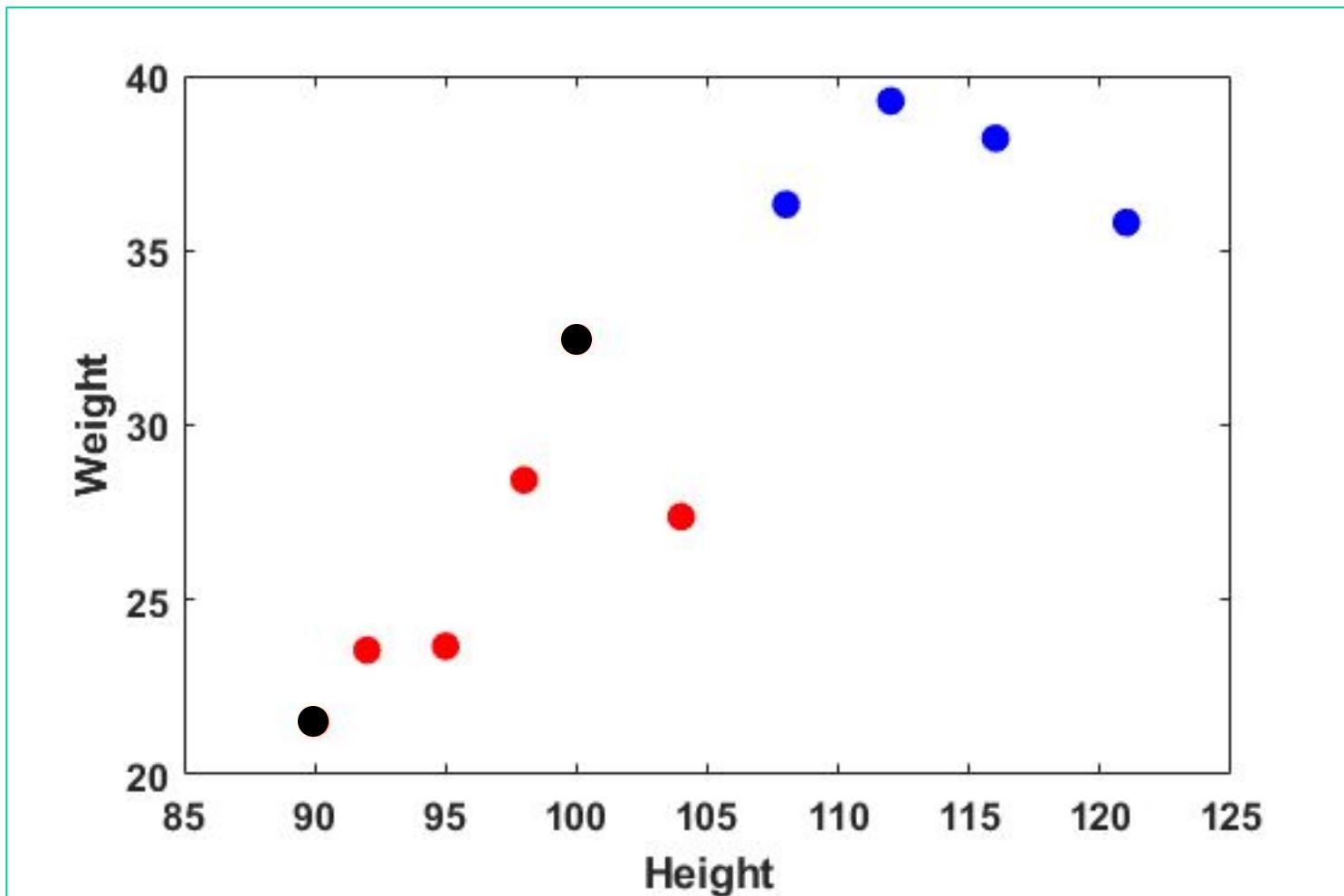
Divisive Hierarchical Clustering



Level: 1

Number of clusters: 2

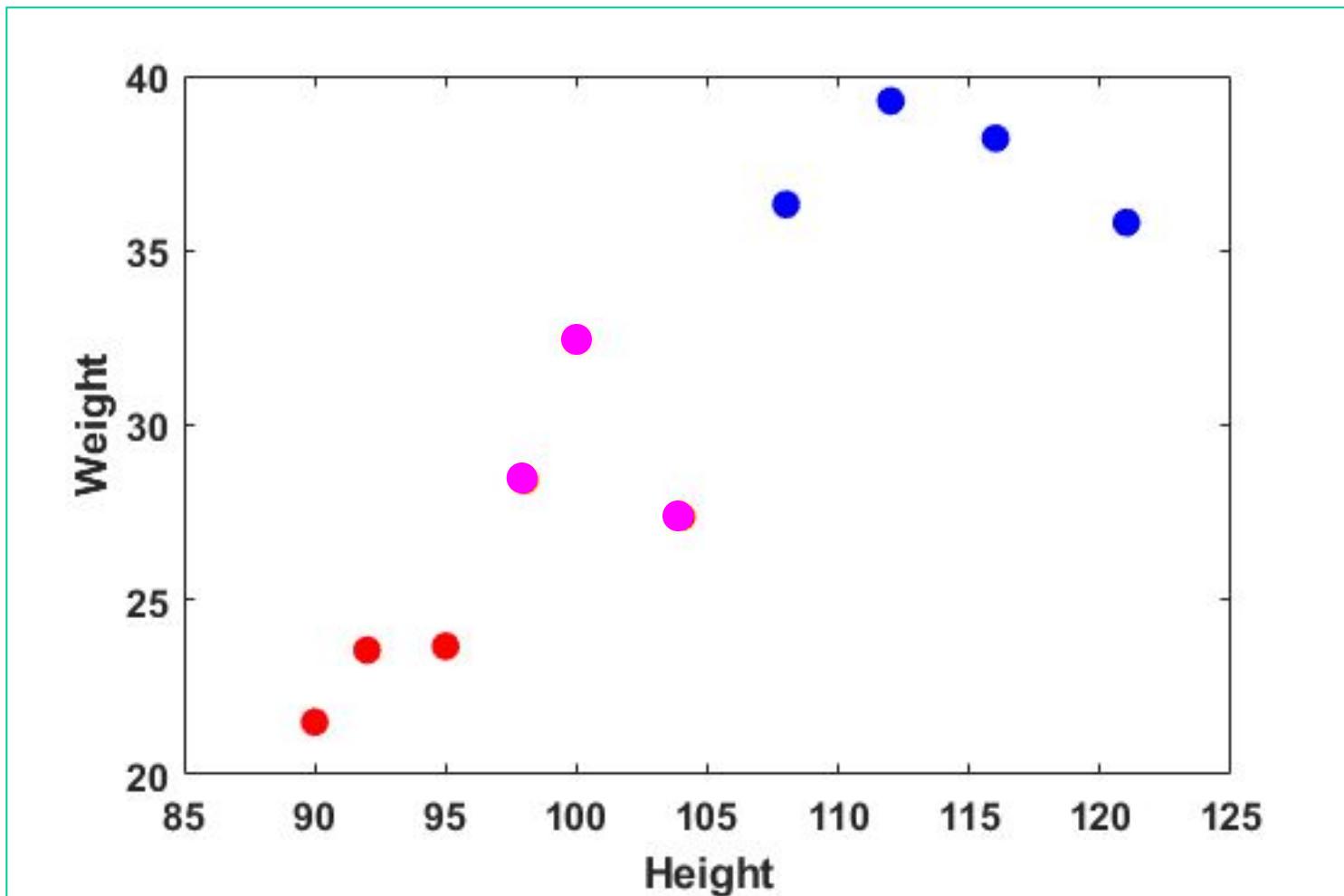
Divisive Hierarchical Clustering



Level: 1

Number of clusters: 2

Divisive Hierarchical Clustering



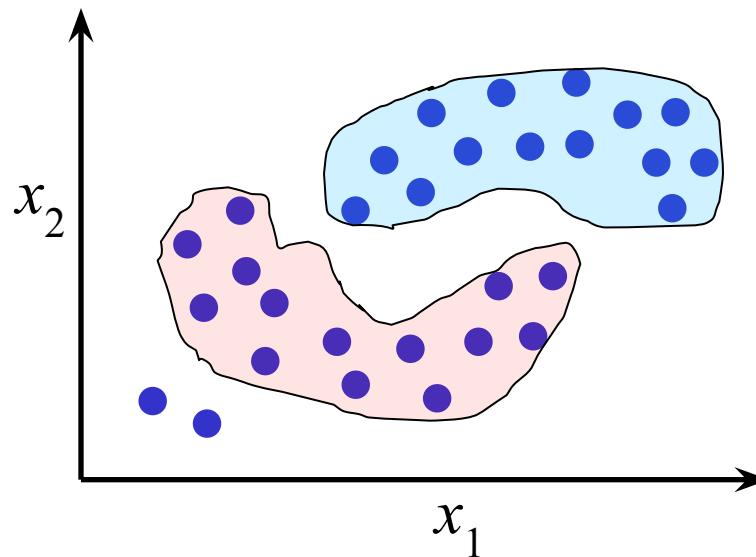
Level: 2

Number of clusters: 3

Density-Based Clustering

Density-Based Clustering

- These methods cluster collection of examples based on the notion of **density**
- These methods regard **clusters** as **dense regions** of **examples** in the data space that are separated by regions of low density (i.e. noise)
- They discover clusters with **arbitrary shape**

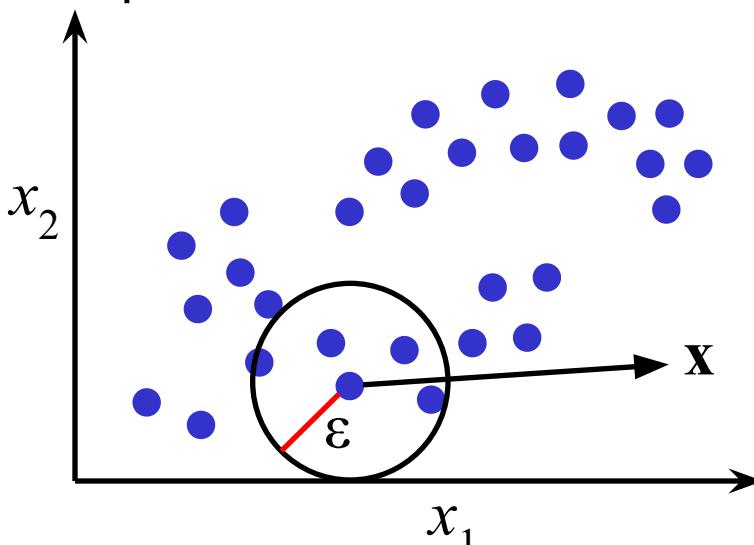


Density-Based Clustering

- These methods cluster collection of examples based on the notion of **density**
- These methods regard **clusters** as **dense regions of examples** in the data space that are separated by regions of low density (i.e. noise)
- They discover clusters with **arbitrary shape**
- They automatically identifies the number of clusters
- **General idea:** To continue growing the given cluster as long as density (number of examples) in the neighbourhood exceeds some threshold
- Example: **Density-based Spatial Clustering of Applications with Noise (DBSCAN)**
 - It grows the clusters according to a density-based connectivity analysis

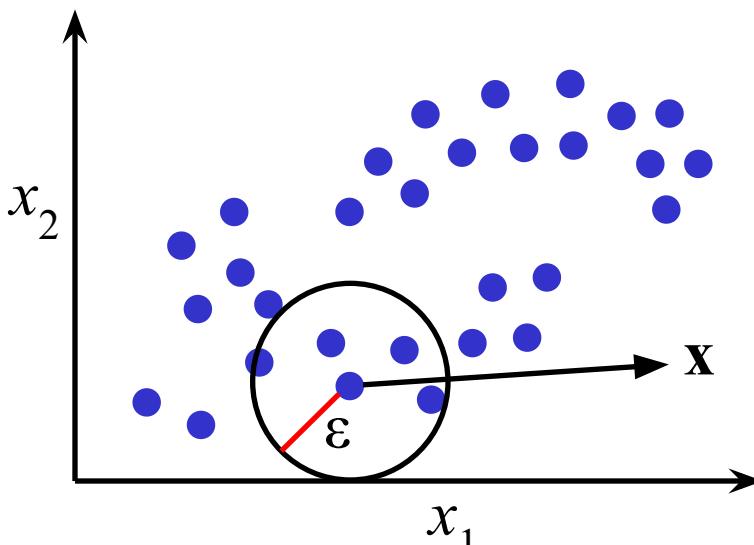
Density-based Spatial Clustering of Applications with Noise (DBSCAN)

- DBSCAN is a density-based clustering included with noise
- It grows the regions with sufficiently high density (neighbors) into clusters with arbitrary shape
- It defines a cluster as a maximal set of density-connected points
- DBSCAN has 5 important components:
 1. **Epsilon (ε)**: It is a value of radius of boundary from every example



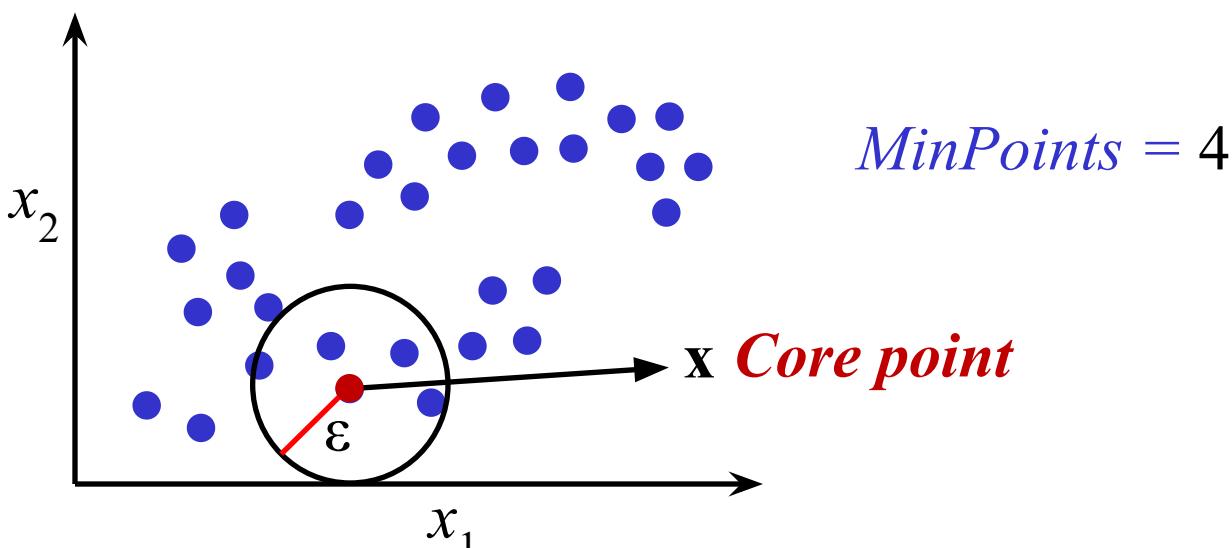
Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:
 1. **Epsilon (ε)**: It is a value of radius of boundary from every example
 2. ***MinPoints***: Minimum number of examples present inside the boundary with radius of ε from an example x
 - These examples with in a boundary are neighbors to x and called as ε -neighborhood of an example, x



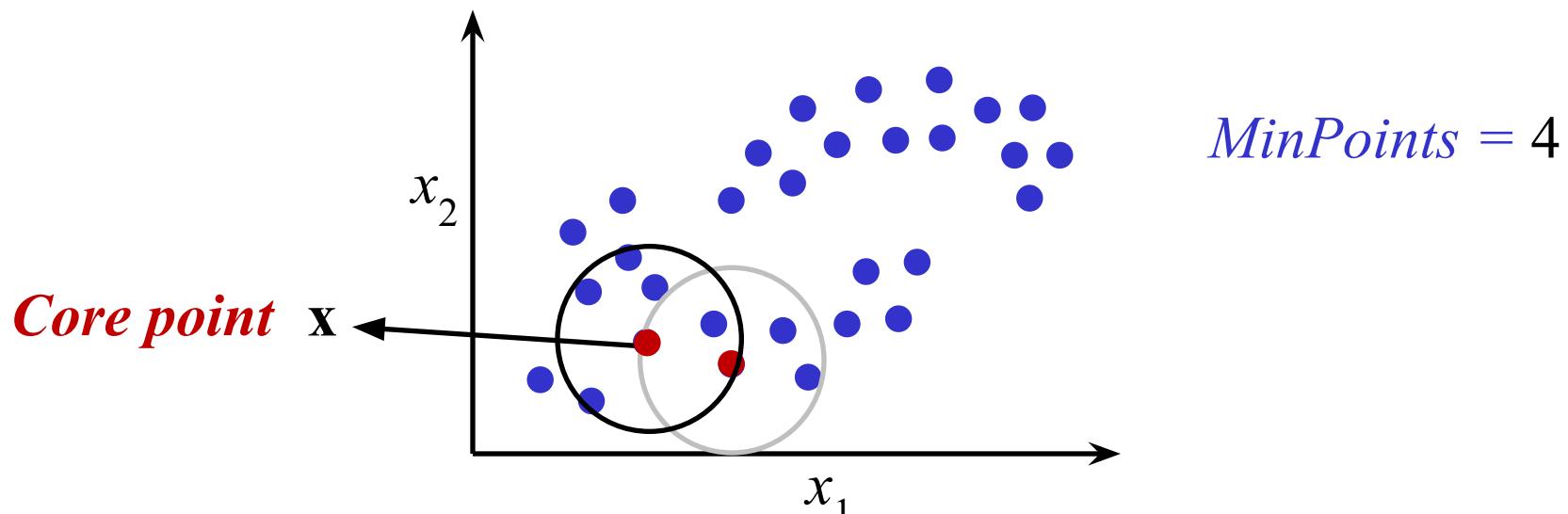
Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:
 1. **Epsilon (ε)**: It is a value of radius of boundary from every example
 2. **$MinPoints$** : Minimum number of examples present inside the boundary with radius of ε from an example x
 3. **Core point**: If there are atleast $MinPoints$ number of examples are with in ε -radius from x , then x is called as **core point**



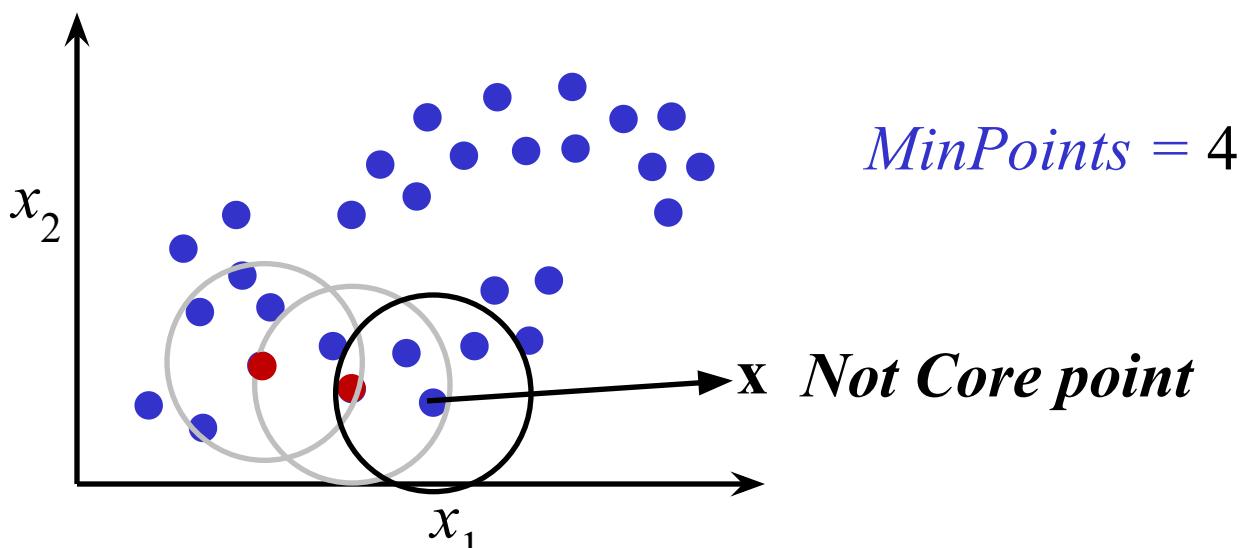
Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:
 1. **Epsilon (ε)**: It is a value of radius of boundary from every example
 2. **$MinPoints$** : Minimum number of examples present inside the boundary with radius of ε from an example x
 3. **Core point**: If there are atleast $MinPoints$ number of examples are with in ε -radius from x , then x is called as **core point**



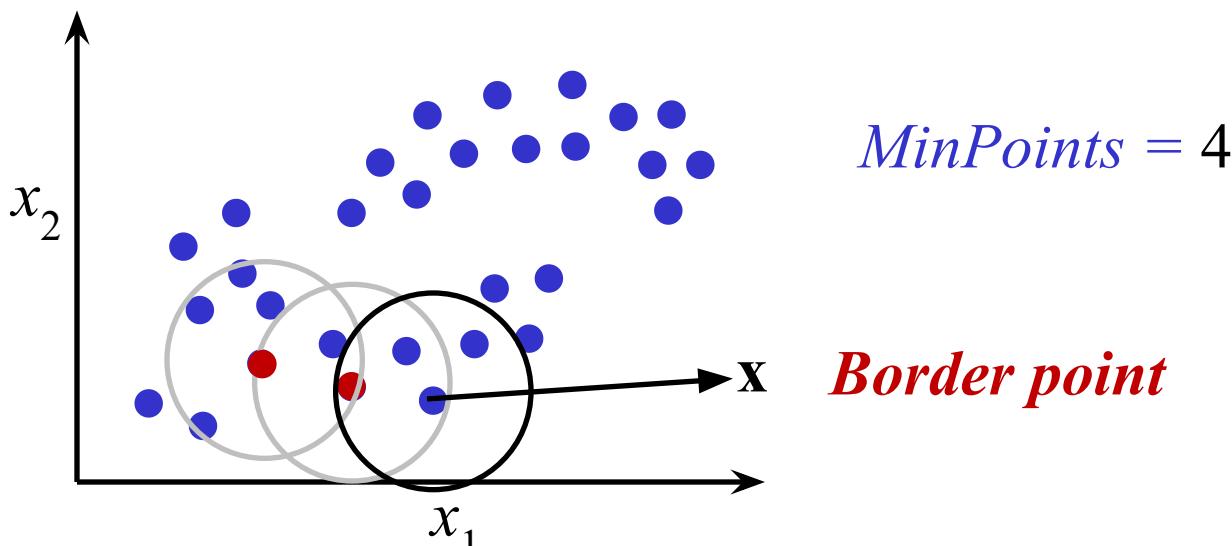
Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:
 1. **Epsilon (ε)**: It is a value of radius of boundary from every example
 2. **$MinPts$** : **Minimum number of examples** present inside the boundary with radius of ε from an example x
 3. **Core point**: If there are atleast **$MinPoints$** number of examples are with in ε -radius from x , then x is called as **core point**



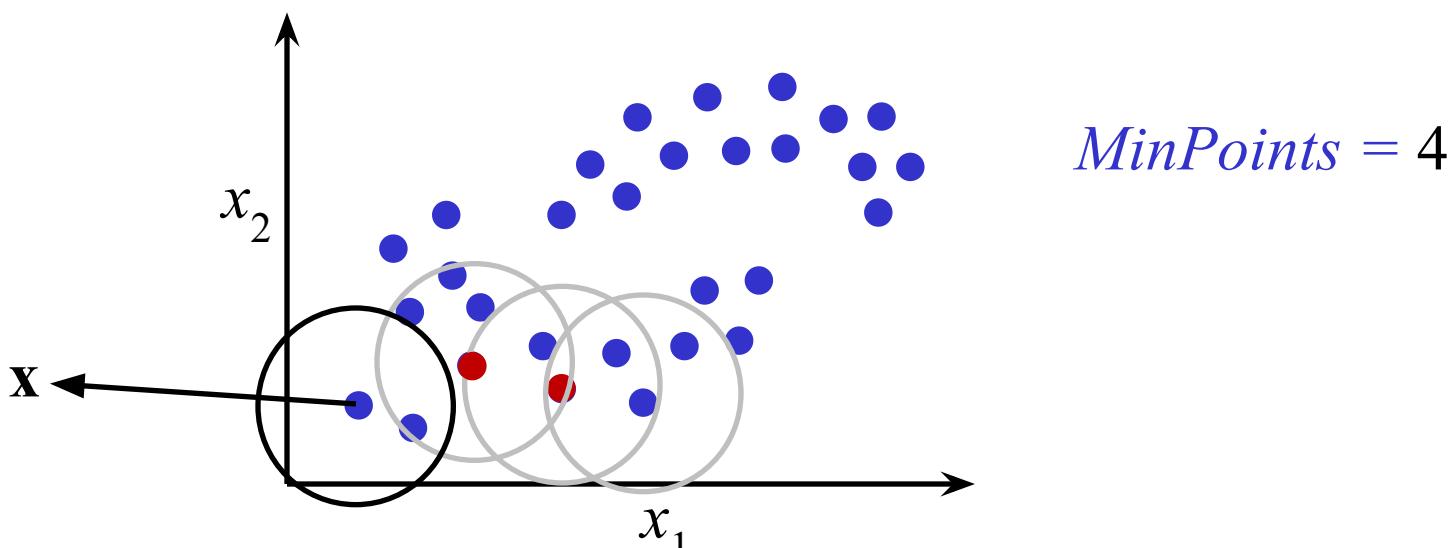
Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:
 3. Core point: If there are atleast *MinPoints* number of examples are within ε -radius from x , then x is called as core point
 4. Border point:
 - The number of examples within ε -radius from x is less than *MinPoints* AND atleast one of the example in neighborhood is core point, then x is called as border point



Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:
 3. Core point: If there are atleast *MinPoints* number of examples are within ϵ -radius from x , then x is called as core point
 4. Border point:
 - The number of examples within ϵ -radius from x is less than *MinPoints* AND atleast one of the example in neighborhood is core point, then x is called as border point

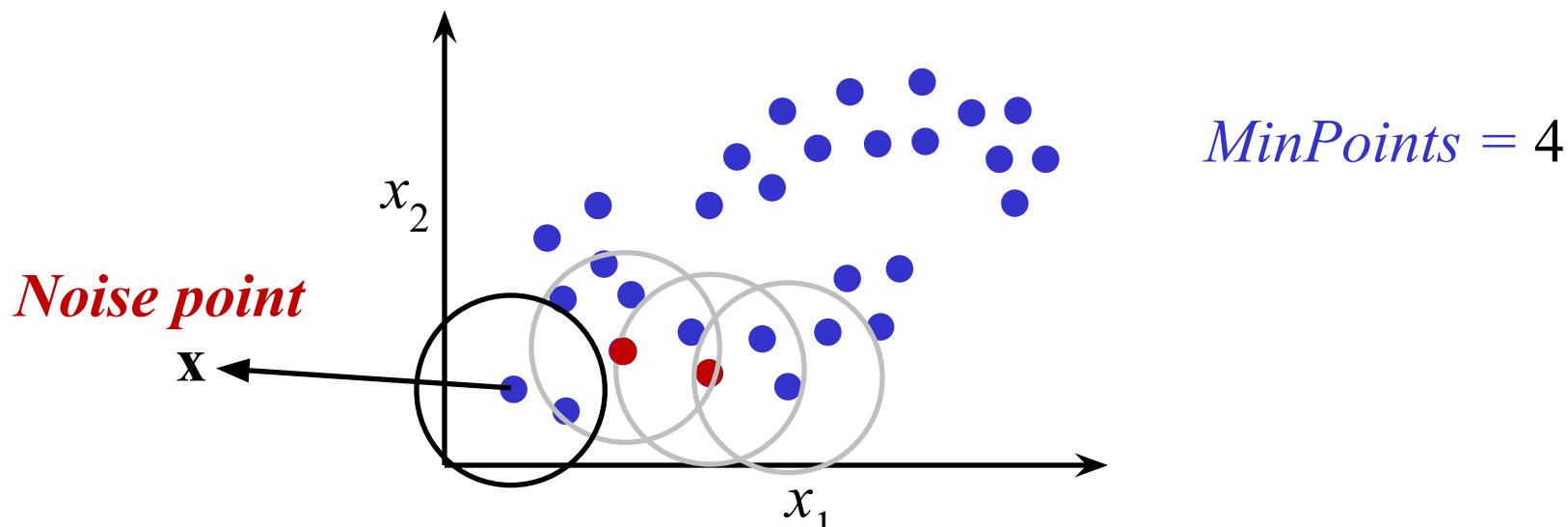


Density-based Special Clustering of Applications with Noise (DBSCAN)

- DBSCAN has 5 important components:

5. Noise point:

- The number of examples within ε -radius from x is **less than *MinPoints*** **AND** no example in neighborhood is **core point**
- The noise point is similar to outlier



Clustering using DBSCAN

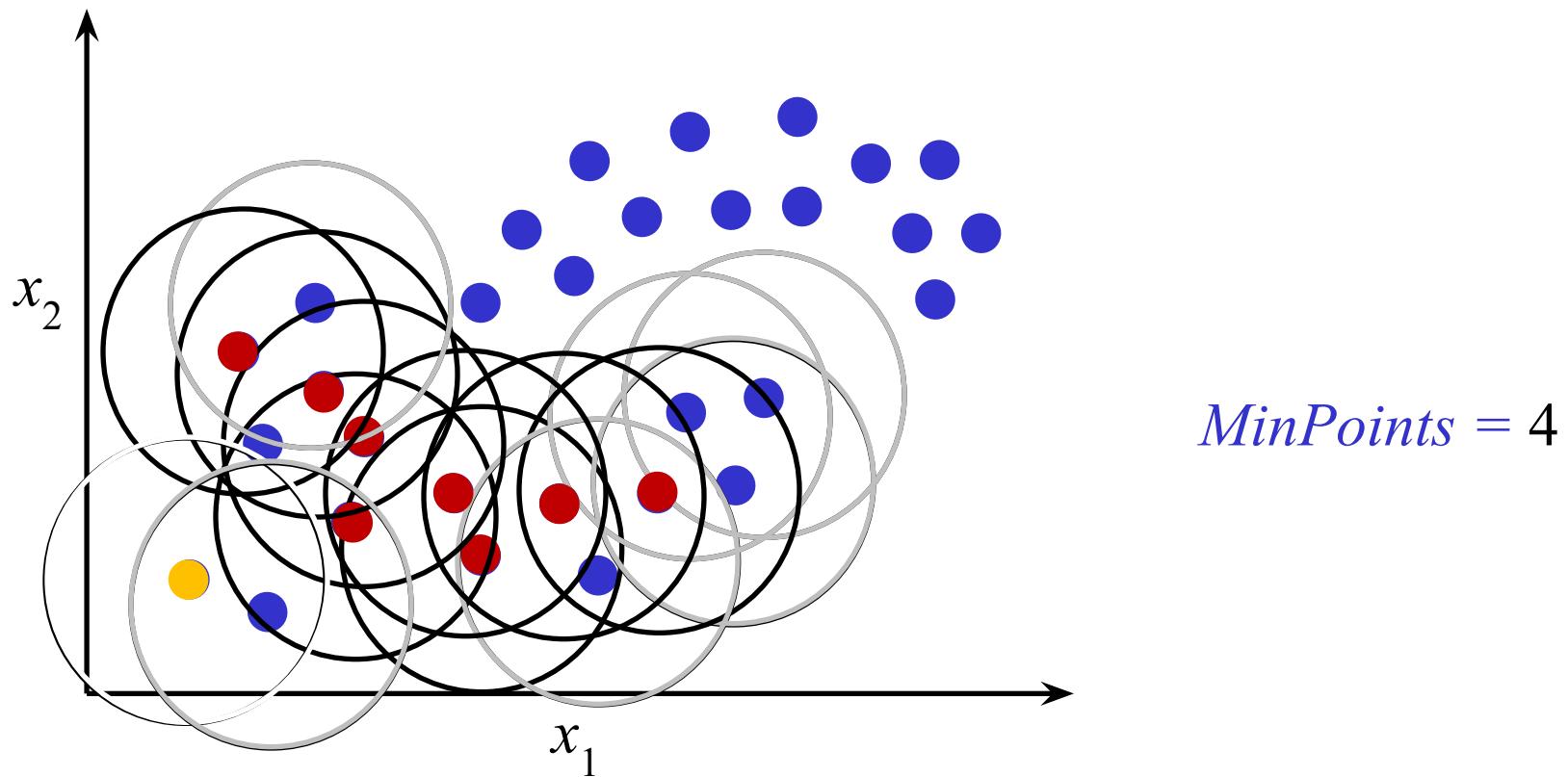
- Given: Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$
- First step is to identify core points, border points and noise points
 - Only core points and border points are considered inside the cluster
 - Noise points are not taken into the cluster
 - Thus DBSCAN is robust to outliers
- Next step is to find the connected components of core points
- Connected component of core points: Connecting the core points that are reachable from any point
 - All the connected (reachable) core points form a cluster

Clustering using DBSCAN

- The connected component of core points is obtained by understanding following *two* definitions.
- **Directly density-reachable:** A core point \mathbf{x}_i is directly density-reachable to a core point \mathbf{x}_j , if the core point \mathbf{x}_j is within ε -distance from core point \mathbf{x}_i
- **Density-reachable:** A core point \mathbf{x}_i is indirectly reachable to another core point \mathbf{x}_j through other core points, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k, \mathbf{x}_{k+1}, \dots, \mathbf{x}_K$ such that
 - \mathbf{x}_i is directly density-reachable to \mathbf{x}_1
 - \mathbf{x}_1 is directly density-reachable to \mathbf{x}_2
 -
 - \mathbf{x}_k is directly density-reachable to \mathbf{x}_{k+1}
 -
 - \mathbf{x}_K is directly density-reachable to \mathbf{x}_j

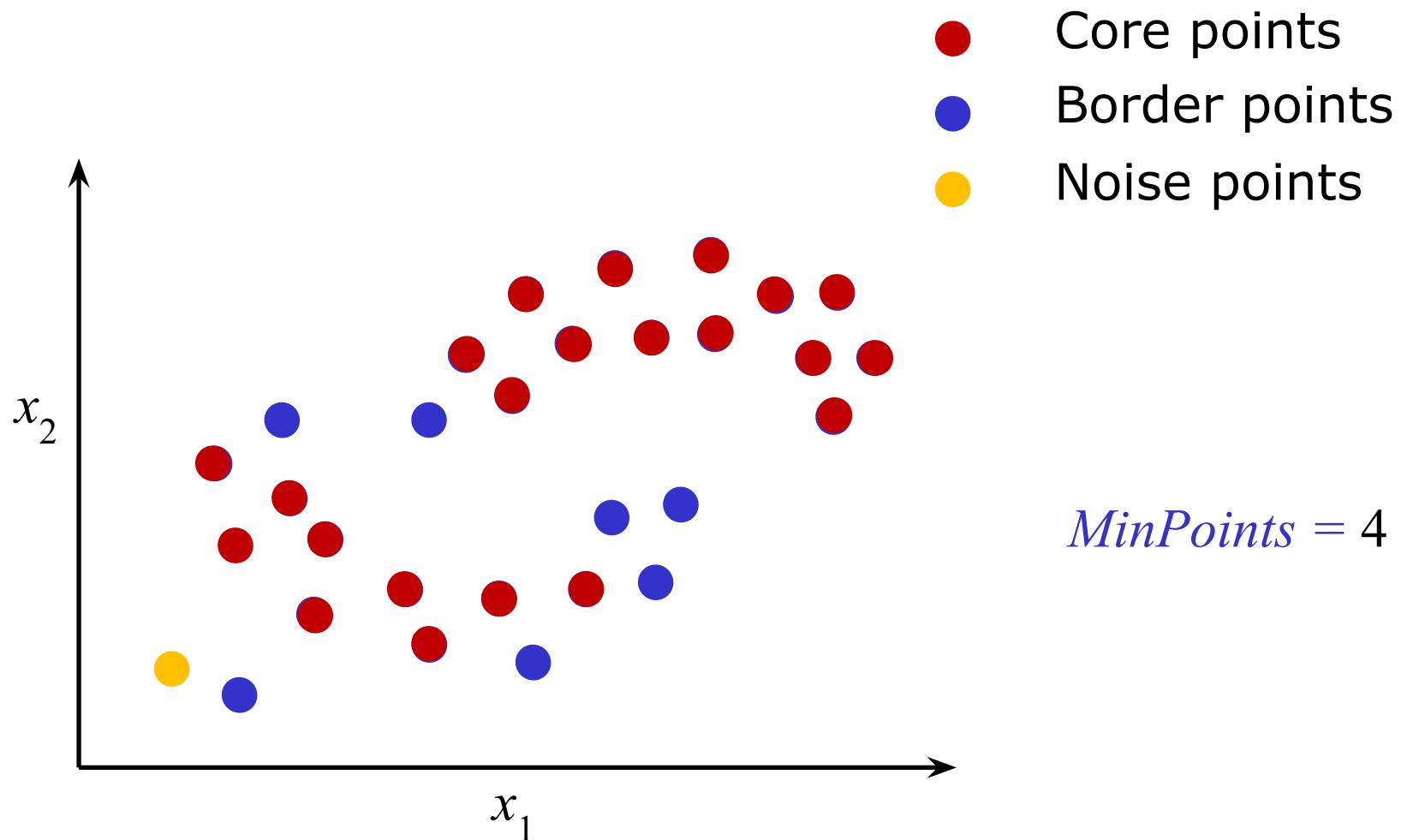
Clustering using DBSCAN

- Identify core points, border points and noise points



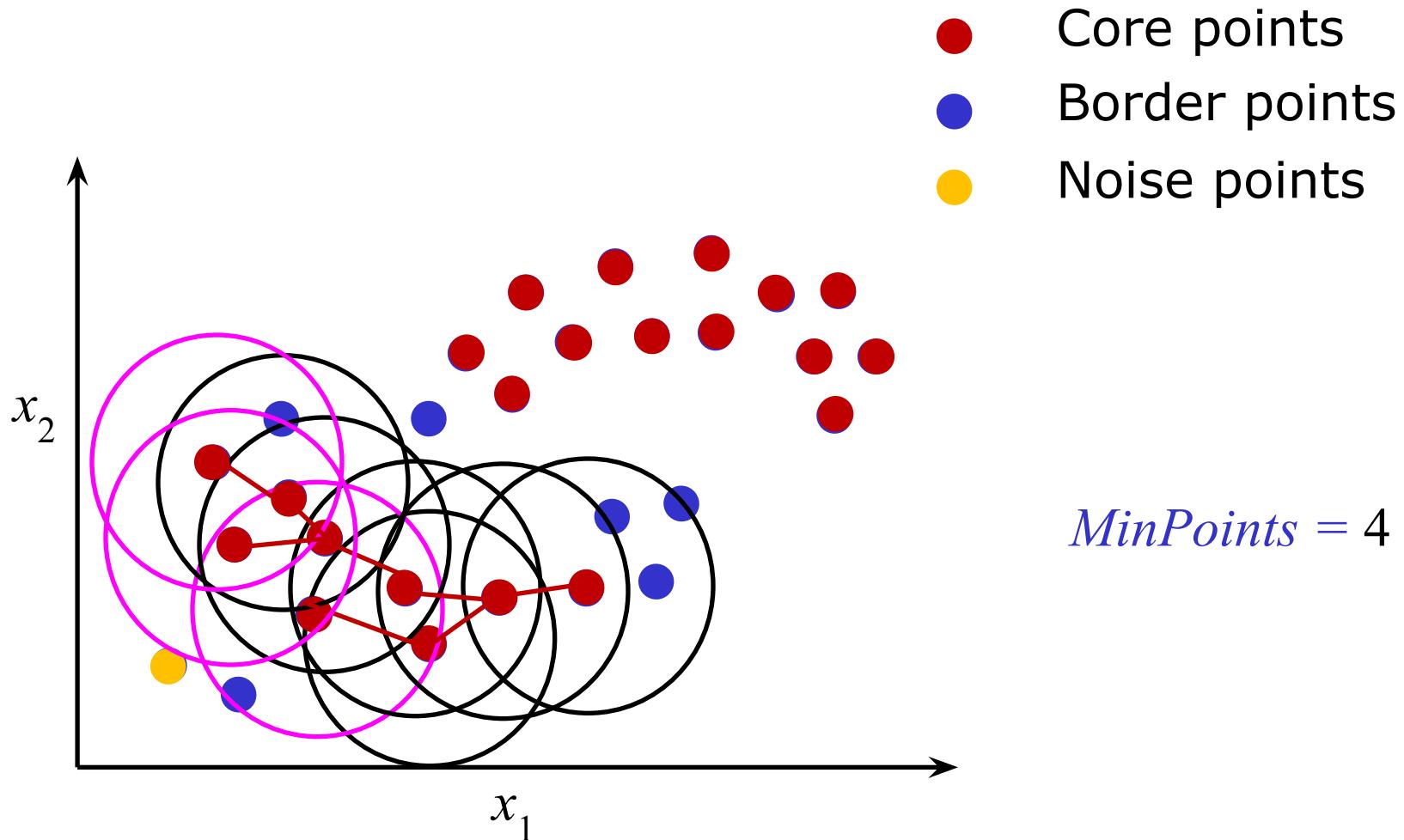
Clustering using DBSCAN

- Identify core points, border points and noise points



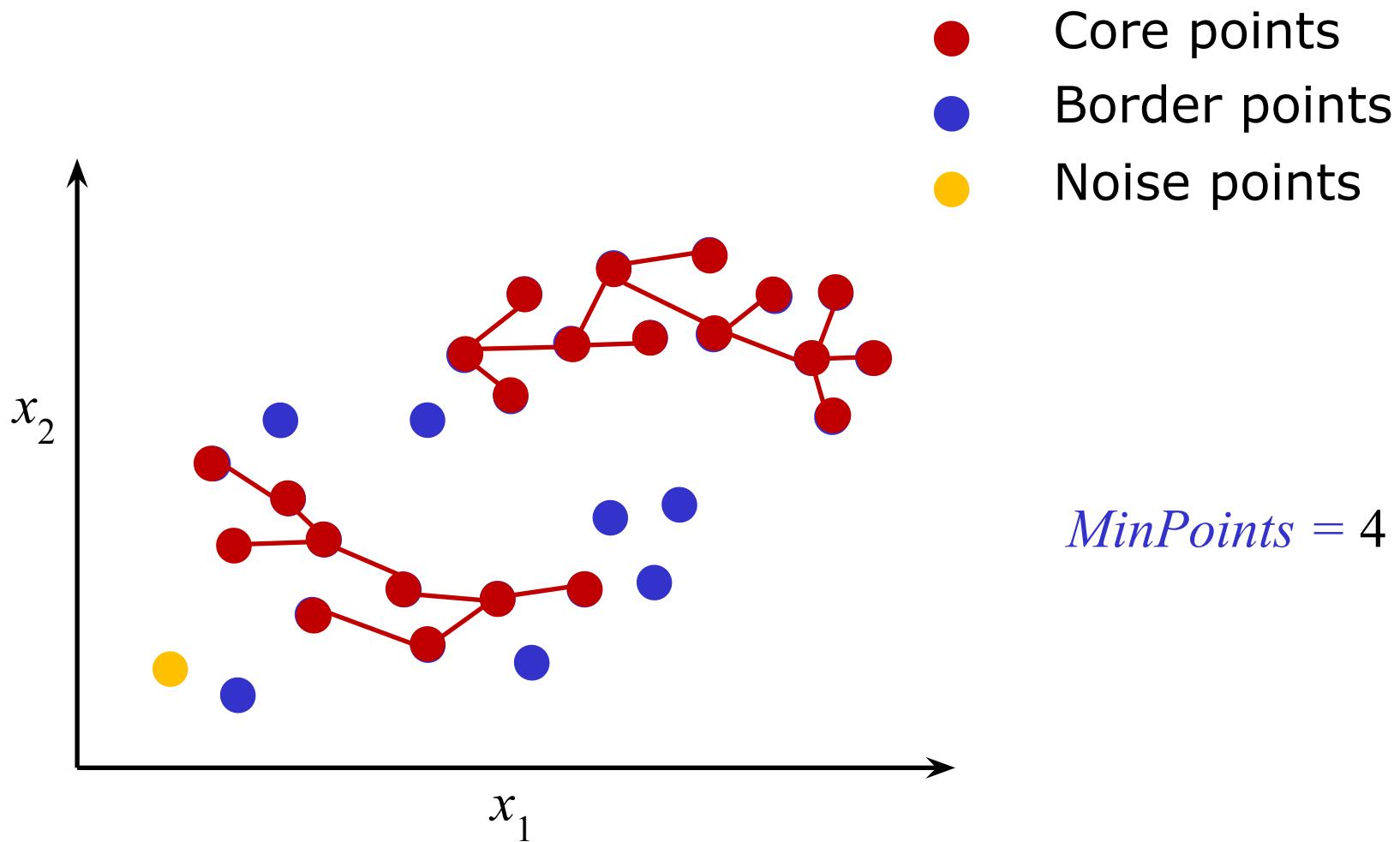
Clustering using DBSCAN

- Obtain the connected component of core points
 - Directly density-reachable:
 - Density-reachable:



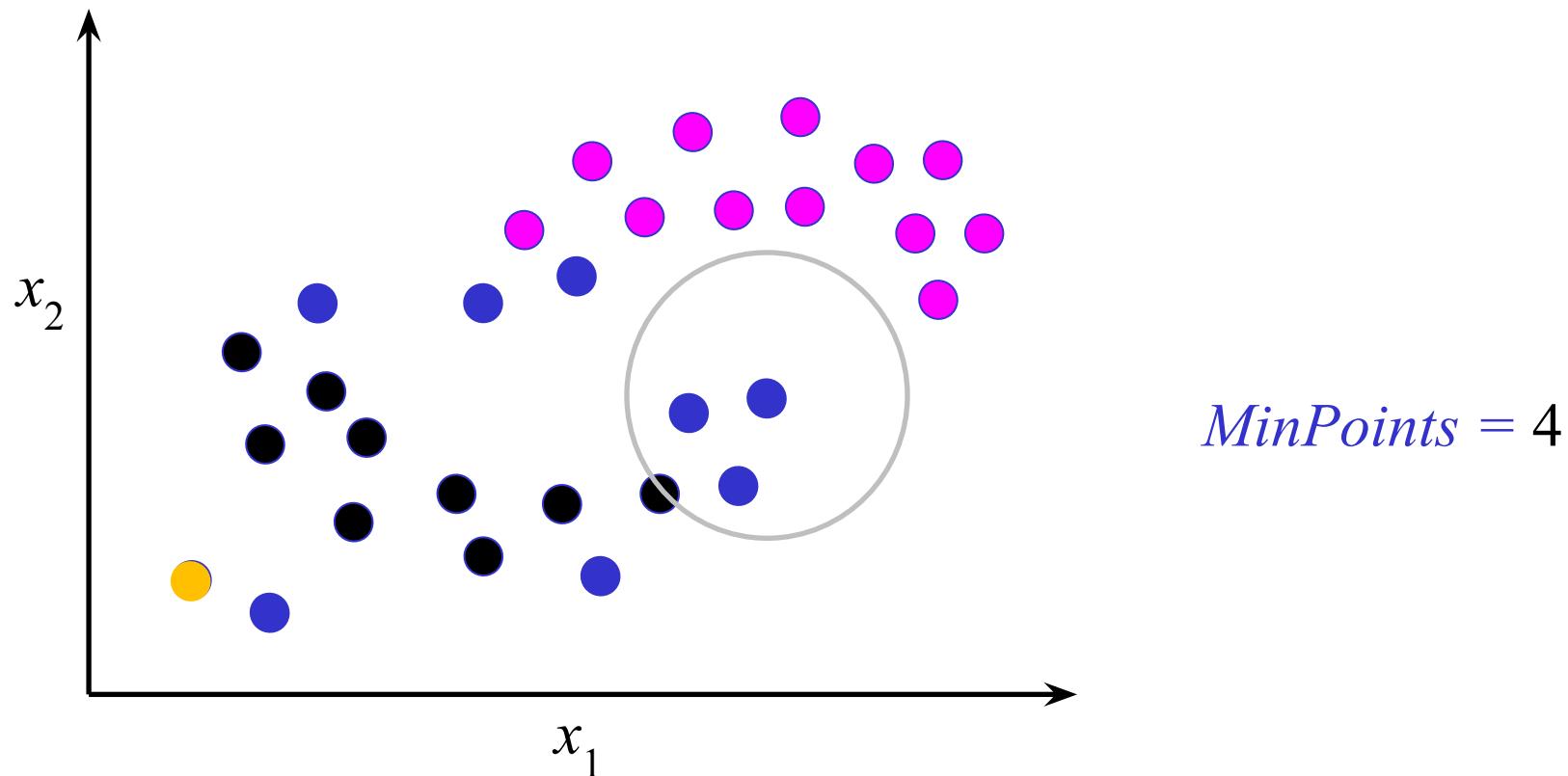
Clustering using DBSCAN

- All the core points with connected component forms a cluster



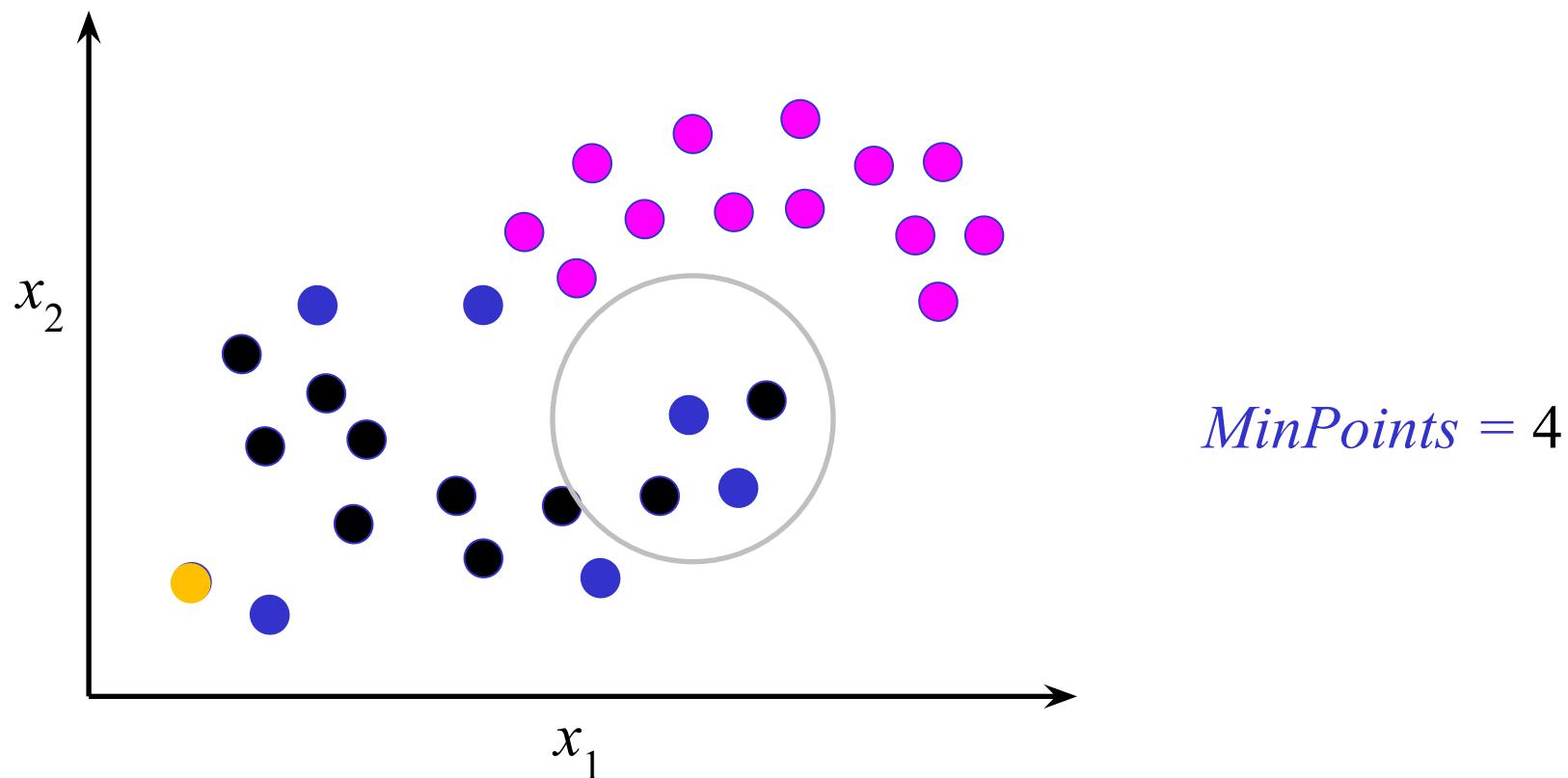
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



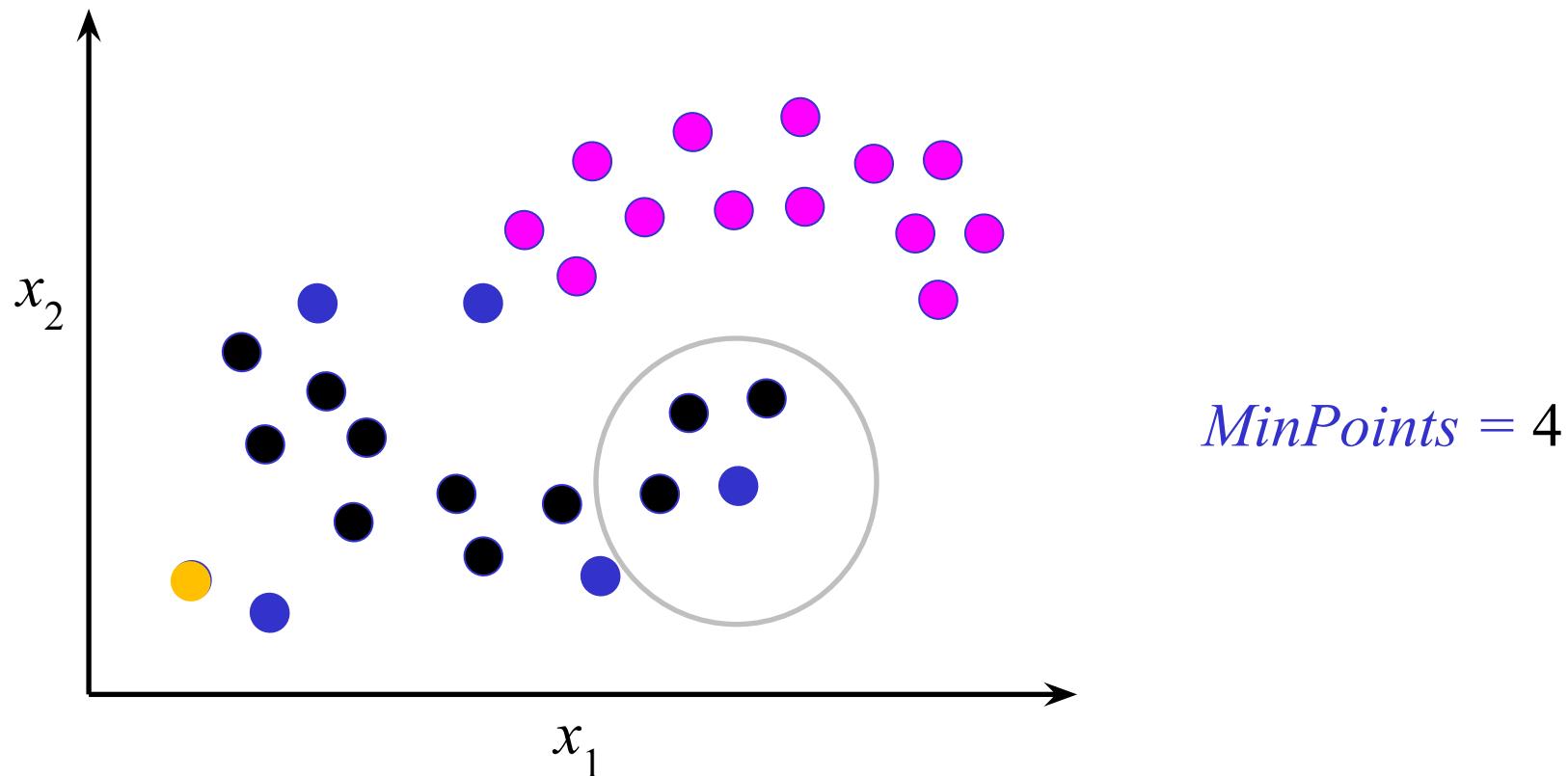
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



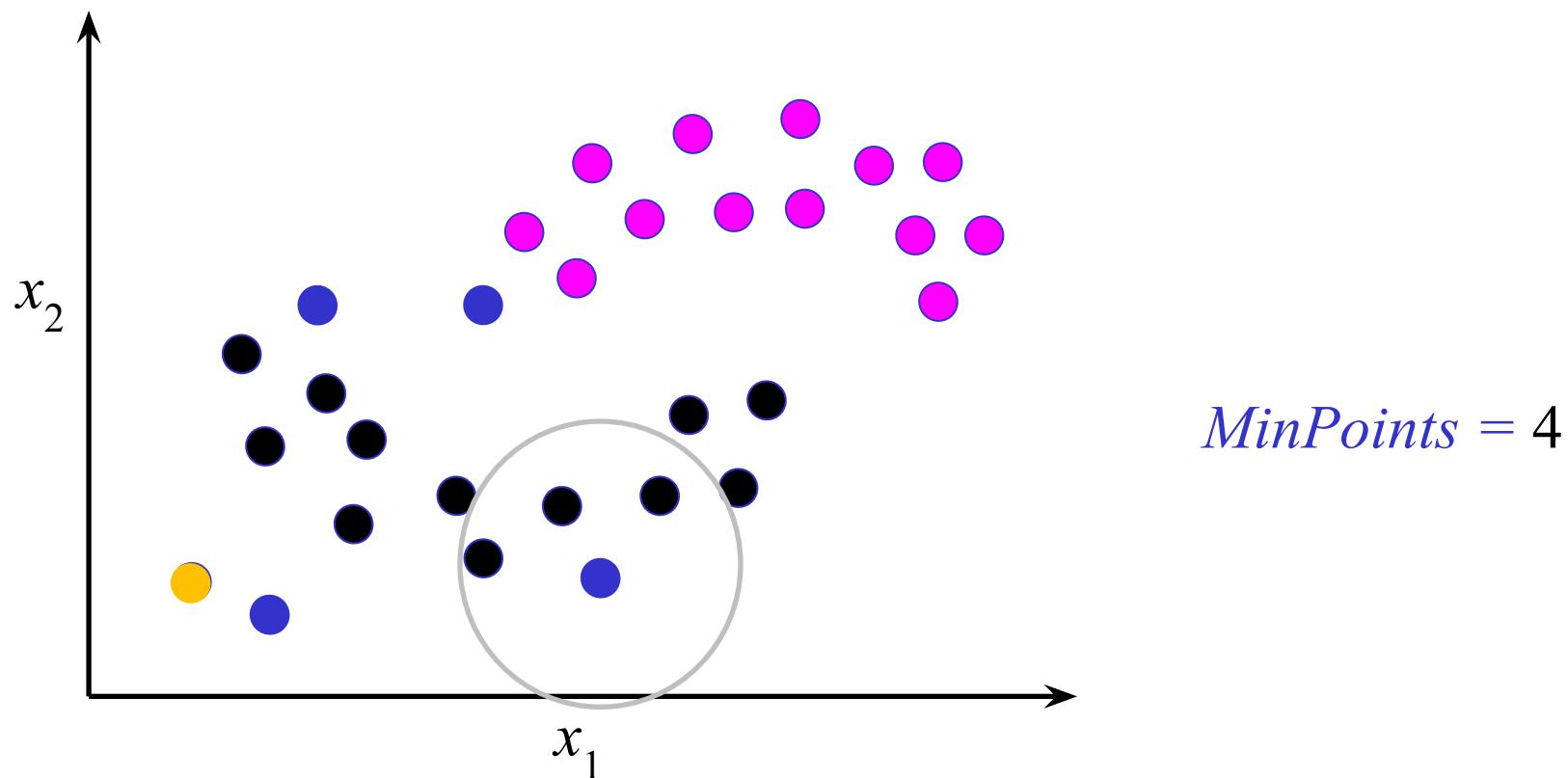
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



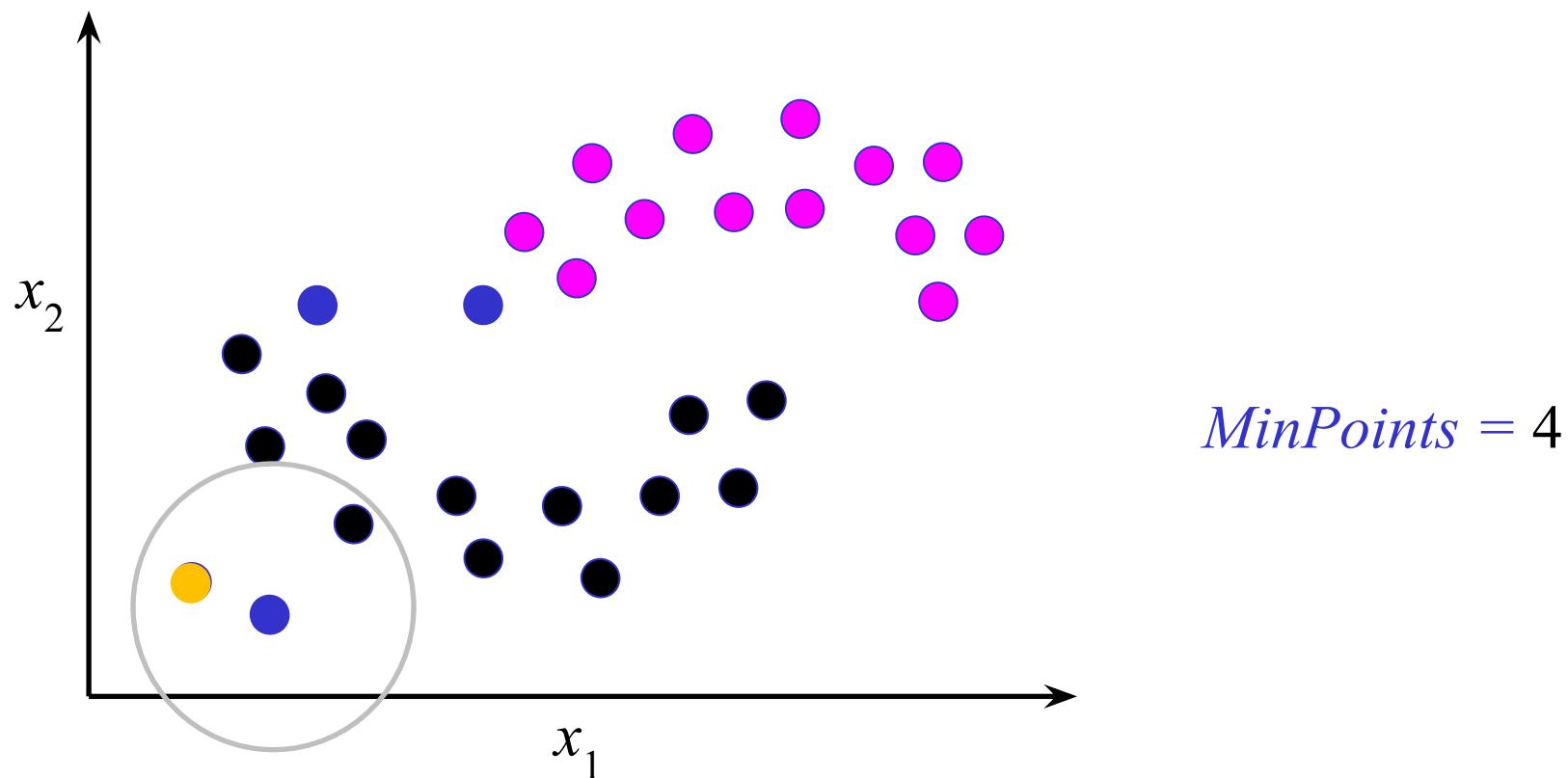
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



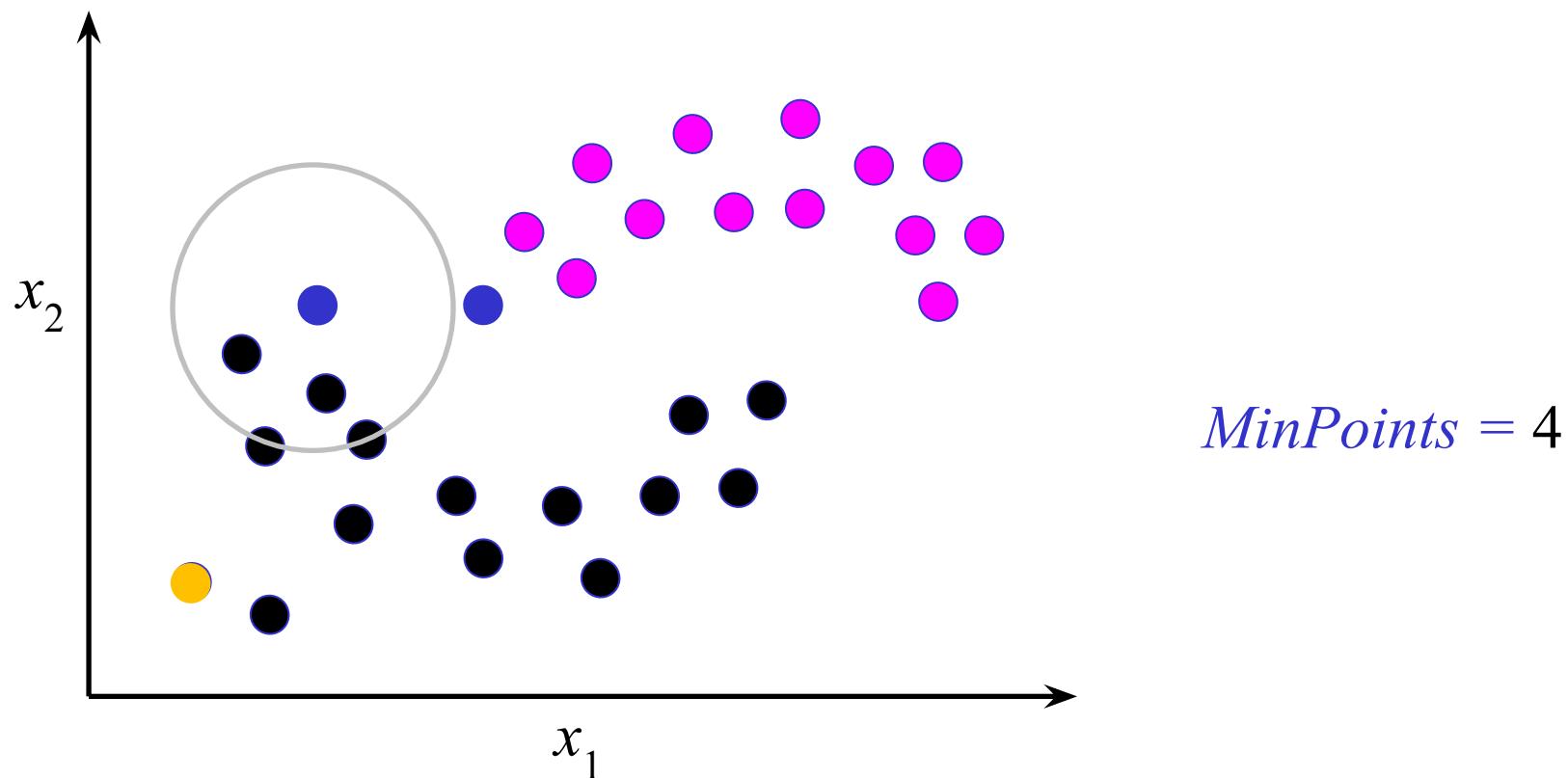
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



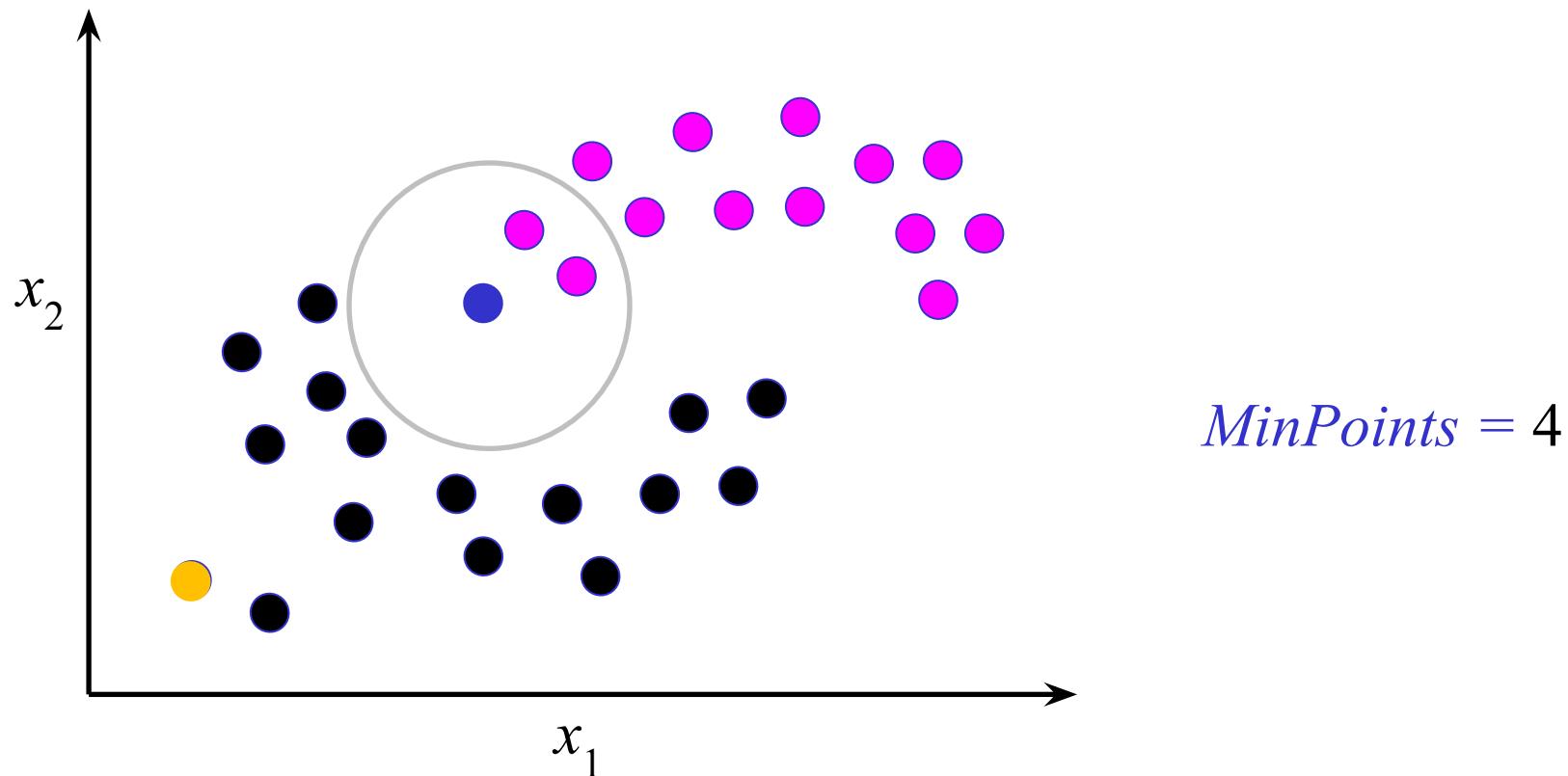
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



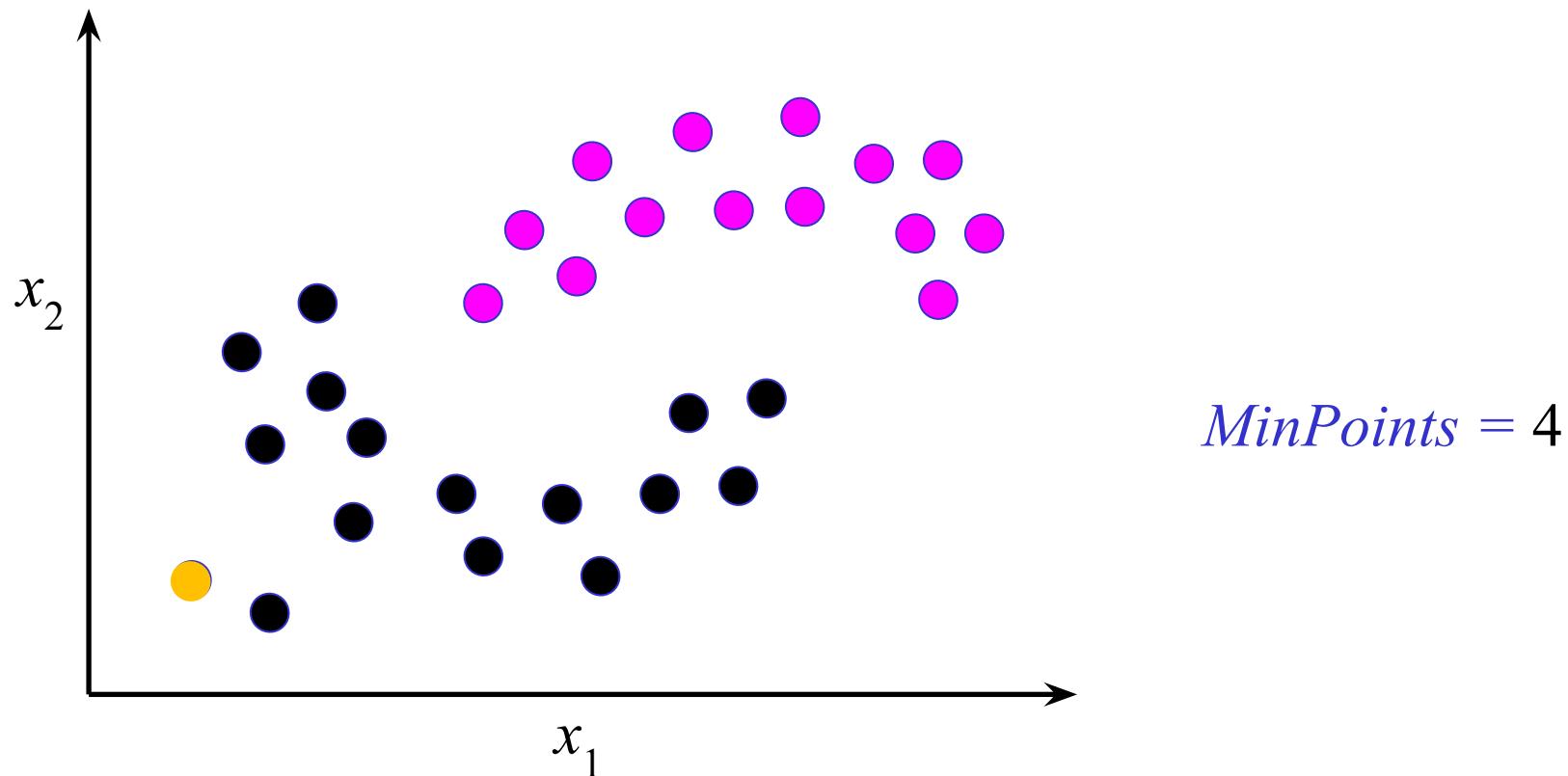
Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



Clustering using DBSCAN

- All the core points with connected component forms a cluster
- Assign the **border points** to nearby cluster which is at ε -radius from that border point



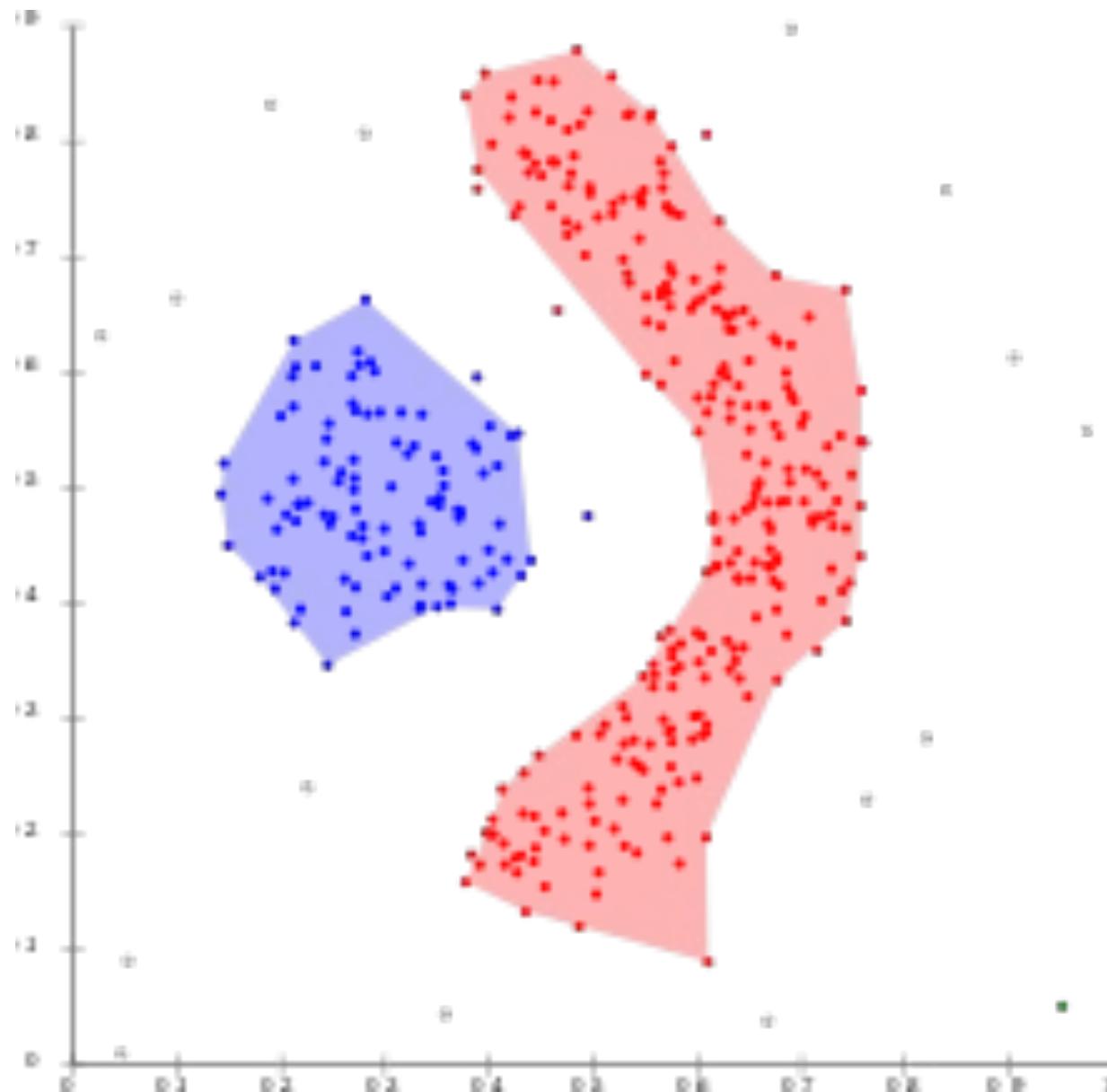
Clustering using DBSCAN

- **Training process:**
- **Given:** Training data, $D = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathbb{R}^d$
- Identify the core points, border points and noise points
- Find the connected components of core points
- Each connected component forms a cluster
- Assign each of the border points to a nearby cluster which is at ε -radius from that border point
- Noise points are not assigned to any clusters
- Training process, stores the core points as model

Clustering using DBSCAN

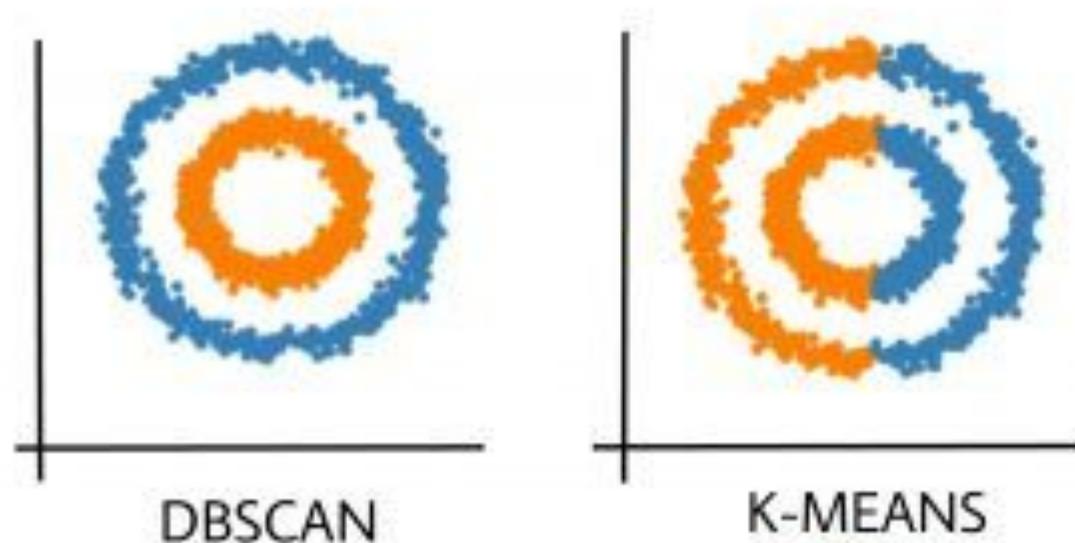
- **Test process:**
- For a test example, identify it as **core point** or **border point** or **noise point**
- If it is a **core point**, assign it to a cluster to which it is **directly density-reachable** or **density-reachable**
- If it is a **border point**, assign it to a **nearby cluster** which is at ε -radius from that border point
- If it is a **noise point**, do not assign to any cluster

Clustering using DBSCAN



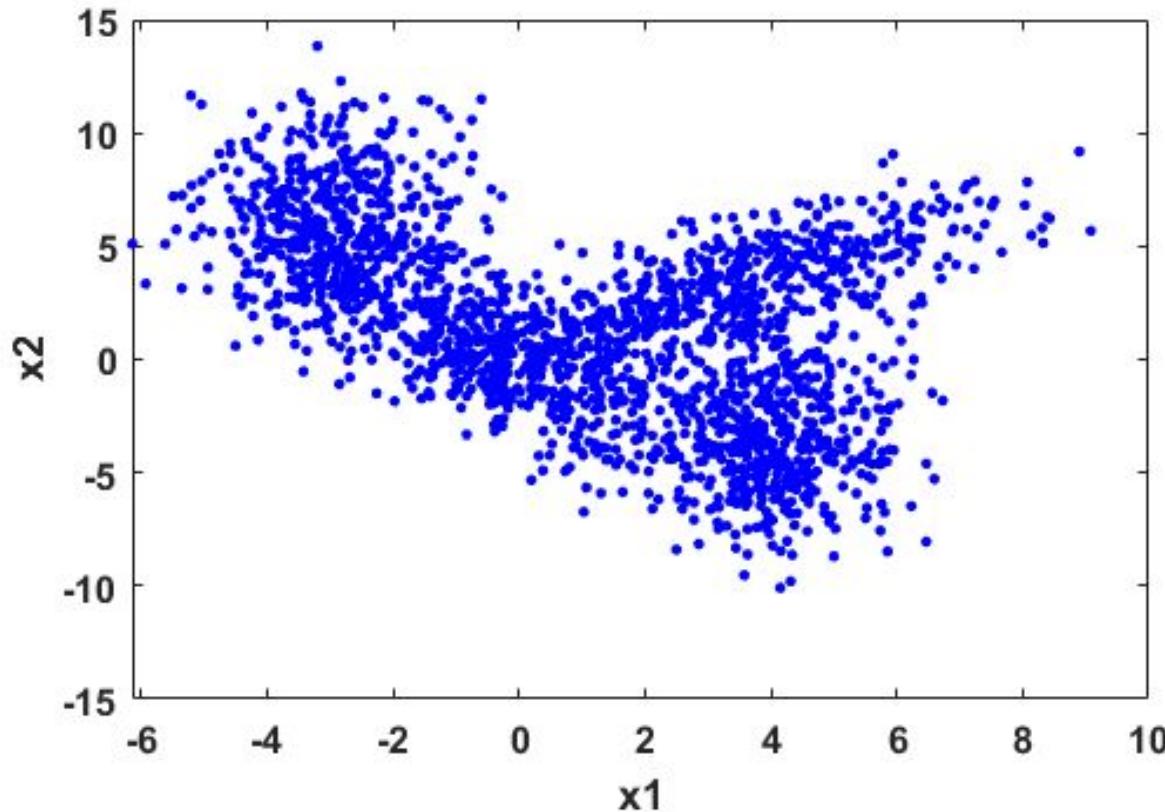
Advantages of DBSCAN

- DBSCAN does not require to specify the number of clusters in the data a priori
- DBSCAN can find arbitrarily shaped clusters
- DBSCAN has a notion of noise, and is robust to outliers
- The parameters ϵ and *MinPoints* are experimentally set by the users



Limitation of DBSCAN

- DBSCAN is not suitable when the data is completely dense and there is no low dense area to separate



- The parameters ε and $MinPoints$ should be chosen carefully

Text Books

1. J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Third Edition, Morgan Kaufmann Publishers, 2011.
2. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2009.