CIS 678 Machine Learning

Special Topics

Course Review Week(s)

Convolution

What will be the convolution output

No padding

$g(t-\tau)\,d\tau.$	0	1	0		
$f(\tau).$	2	5	0	1	3

Filter/Kernel

$(f * g)(t)$

? ? ?

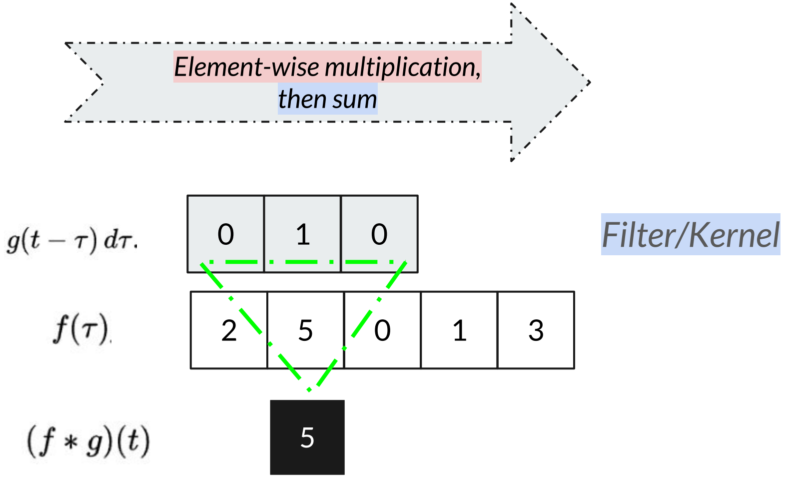
ref

Answer?

Convolution

What will be the convolution output

No padding

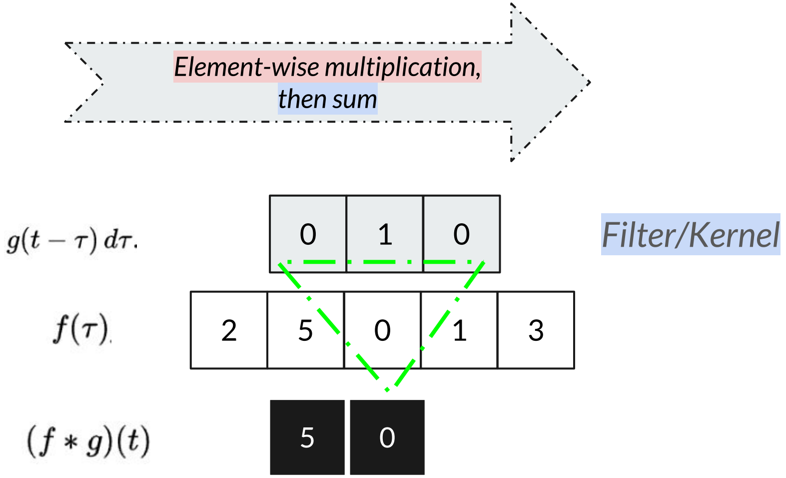


[ref]

Convolution

What will be the convolution output

No padding

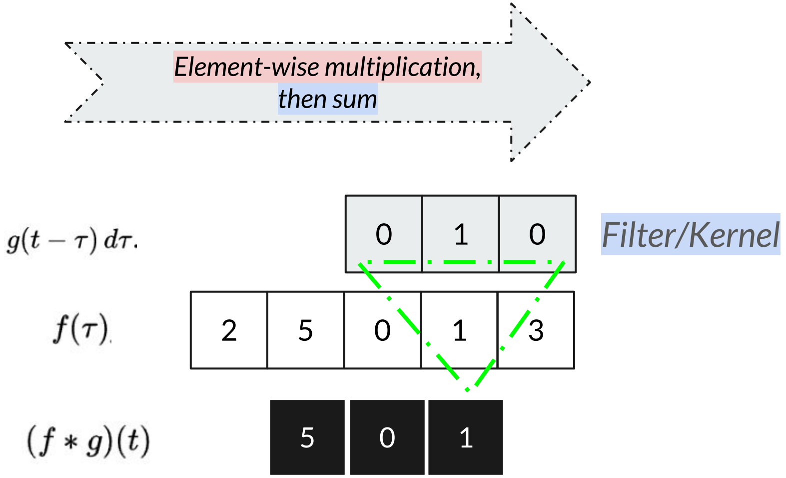


[ref]

Convolution

What will be the convolution output

No padding



[ref]()

2D Convolution

Data matrix (D):

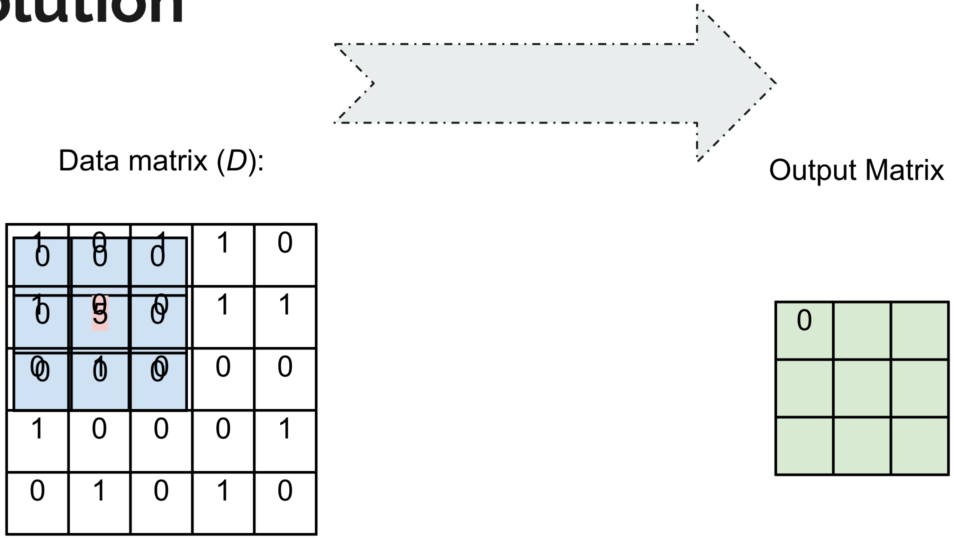
1	0	1	1	0
1	0	0	1	1
0	1	0	0	0
1	0	0	0	1
0	1	0	1	0

Kernel matrix (D):

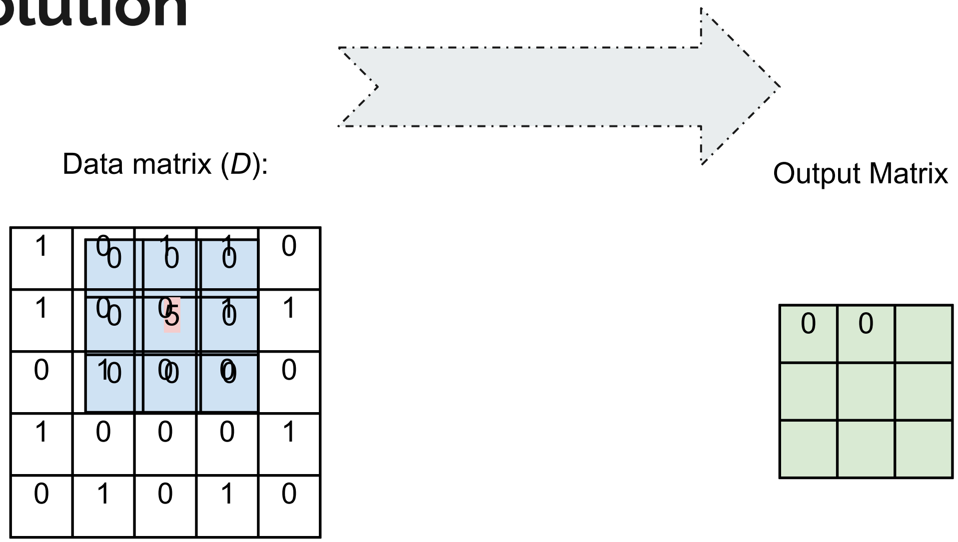
0	0	0
0	5	0
0	0	0

Answer?

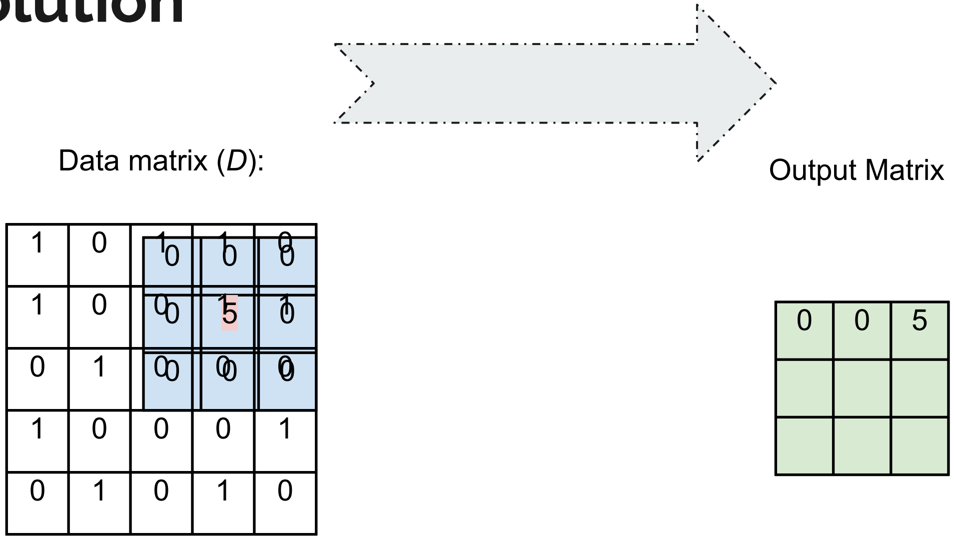
2D Convolution



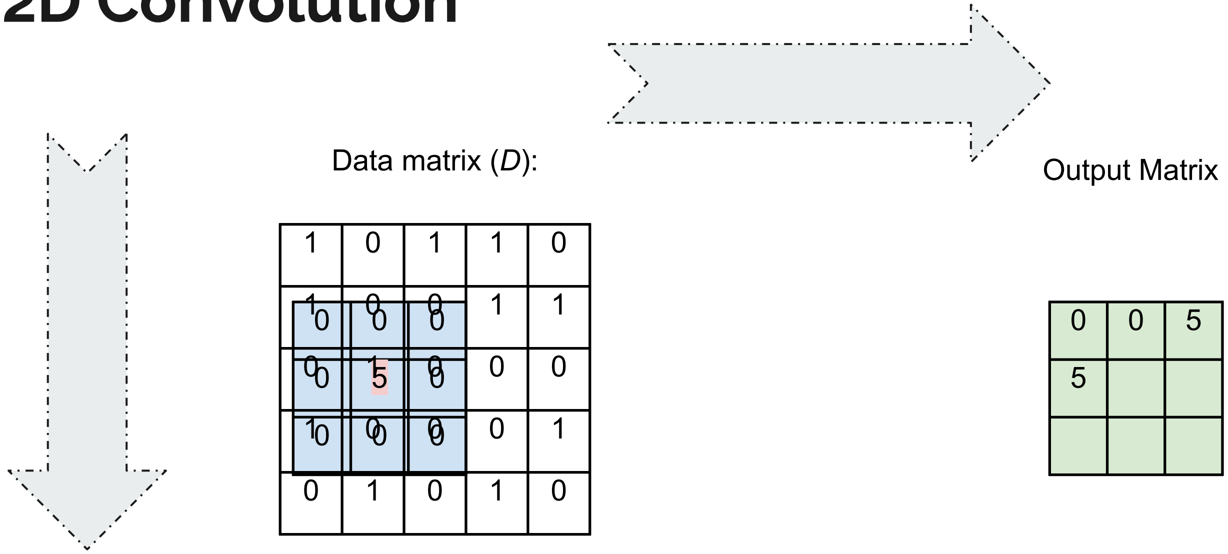
2D Convolution



2D Convolution



2D Convolution



Answer?

2D Convolution

Data matrix (D):

1	0	1	1	0
1	0	0	1	1
0	1	0	0	0
1	0	0	0	1
0	1	0	1	0

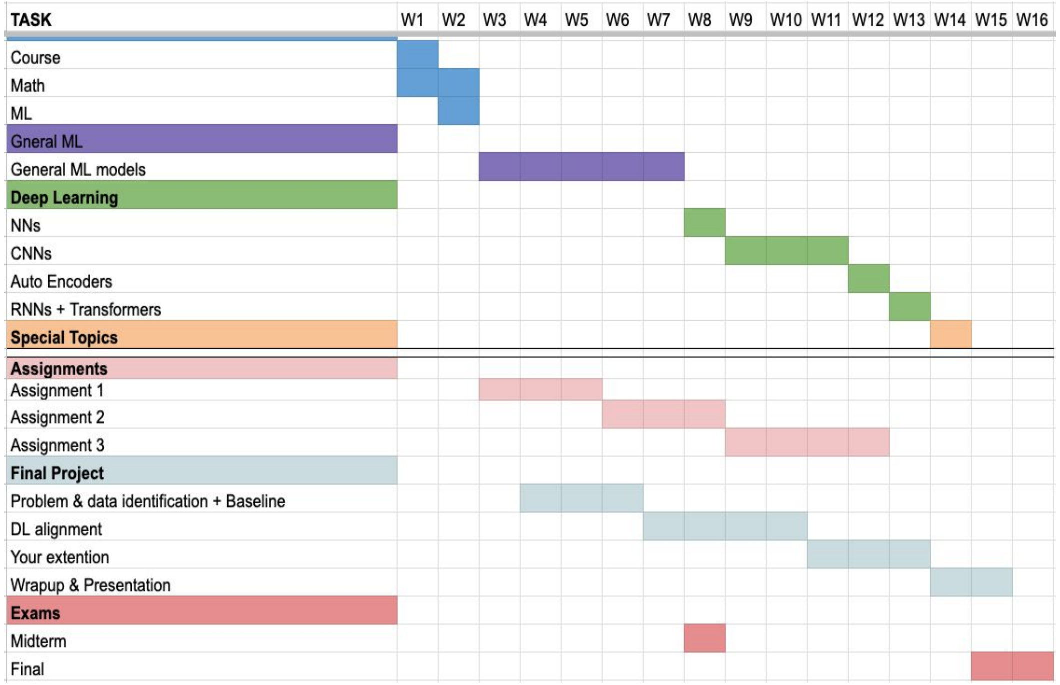
Kernel matrix (D):

0	0	0
0	5	0
0	0	0

Output Matrix

0	0	5
5	0	0
0	0	0

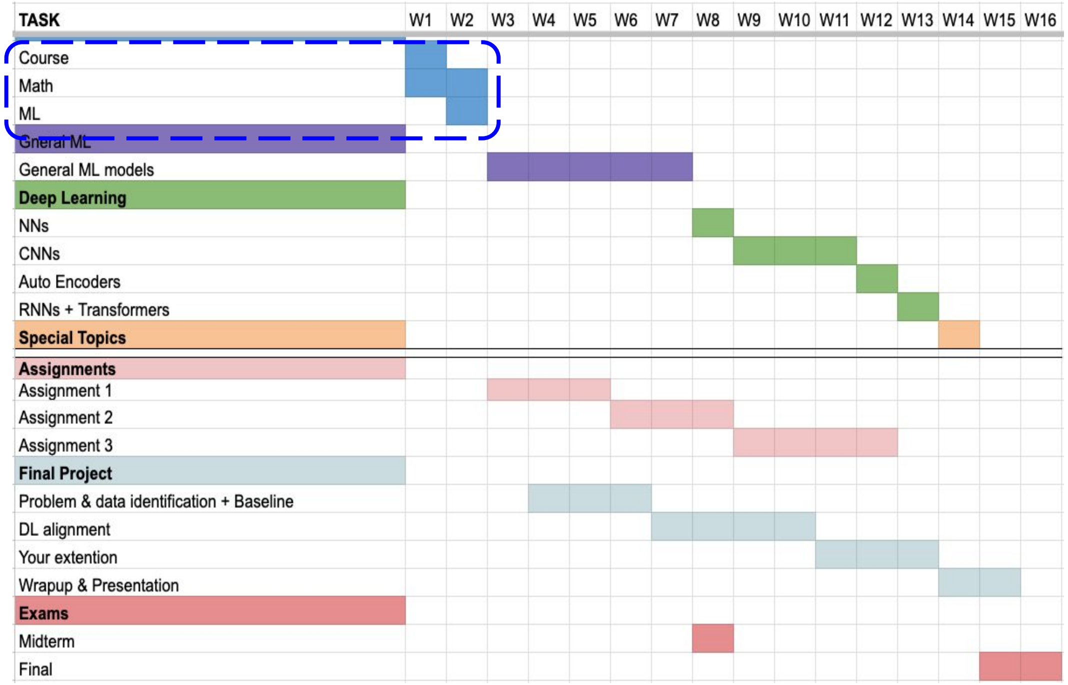
Our Plan


TASK
W1
W2
W3
W4
W5
W6
W7
W8
W9
W10
W11
W12
W13
W14
W15
W16
Course
Math
ML
Gneral ML
General ML models
Deep Learning
NNs
CNNs
Auto Encoders
RNNs + Transformers
Special Topics
Assignments
Assignment 1
Assignment 2
Assignment 3
Final Project
Problem & data identification + Baseline
DL alignment
Your extention
Wrapup & Presentation
Exams
Midterm
Final

Our Plan

Vector space
Proximity or distance metric
- L1/Manhattan distance
- L2/Euclidean distance,
- Cosine distances

kNN model:
- Distance based
- Can be applied to both Regression and Classification tasks

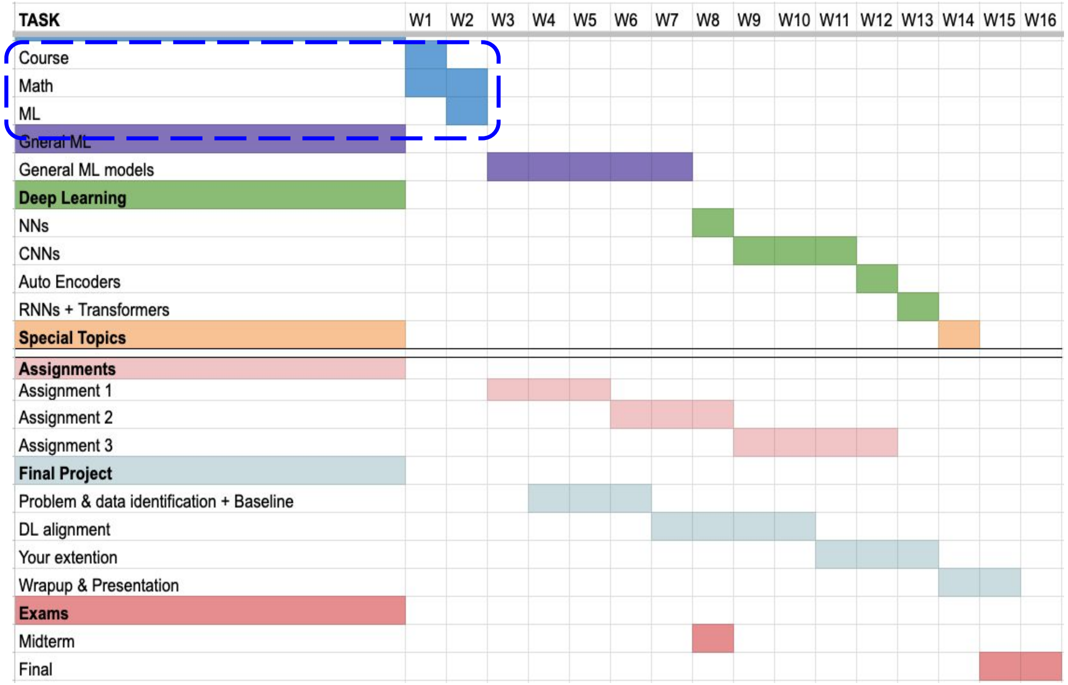


Our Plan

Vector space
Proximity or distance metric
- L1/Manhattan distance
- L2/Euclidean distance,
- Cosine distances

kNN model:
- Distance based
- Can be applied to both Regression and Classification tasks

Probability (measuring uncertainty)
Probability distributions



Our Plan

Vector space
Proximity or distance metric

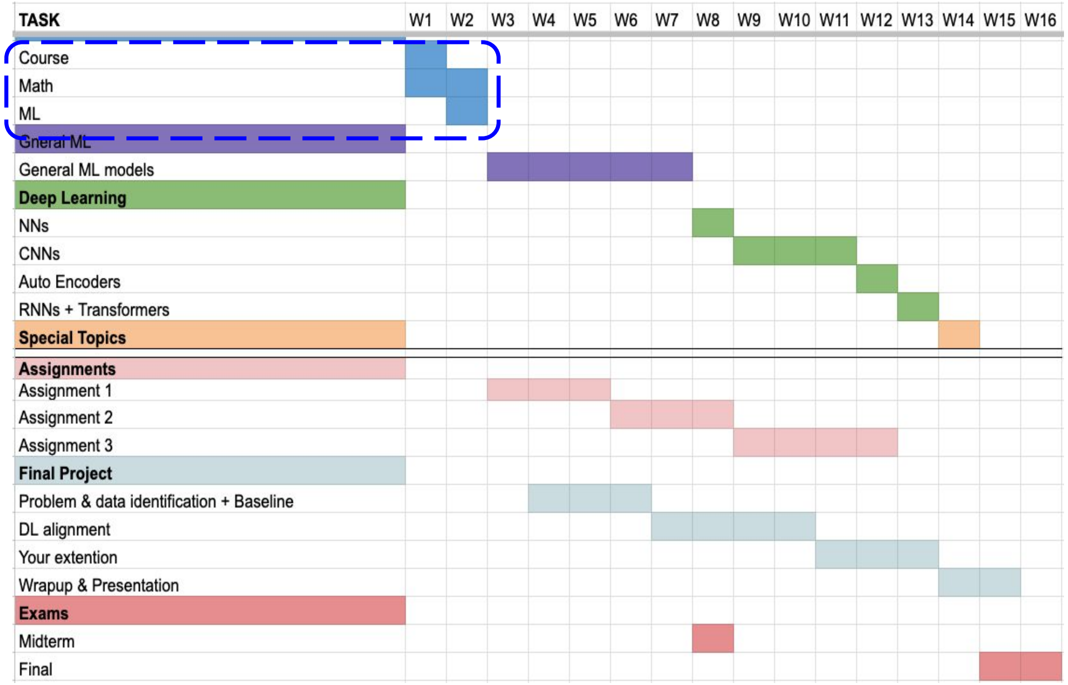
- L1/Manhattan distance
- L2/Euclidean distance,
- Cosine distances

kNN model:

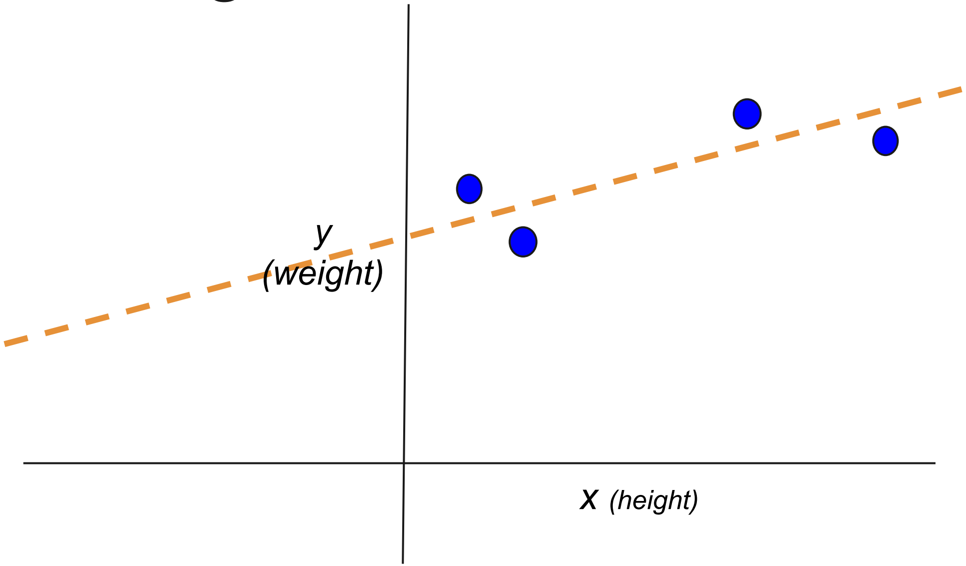
- Distance based
- Can be applied to both Regression and Classification tasks

Probability (measuring uncertainty)
Probability distributions

ML introduction:
Linear to Polynomial Regression



Linear Regression



Model

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\Theta = \{\beta_0, \beta_1\}$$

Fitting Error

$$\epsilon = |\hat{y} - y|$$

Optimization function

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\boxed{\Theta^* = \mathrm{argmin}_\Theta\, E\{(x_i, y_i)\}_{i=1,\cdots,N}}$$

Linear to Polynomial Regression

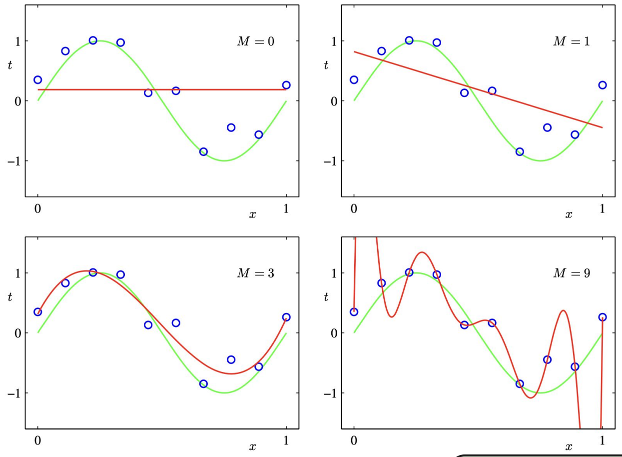


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Model generalization

Linear to Polynomial Regression

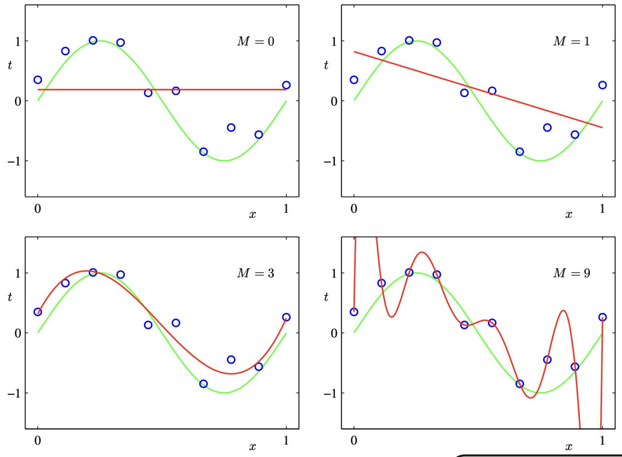


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Absolute values are increasing

Model generalization

Linear to Polynomial Regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Essentially, the same formulation

Generally ML vs Math conventions

$$\boxed{W^* = \text{argmin}_W\, E\{(x_i, t_i)\}_{i=1,\cdots,N}}$$

Model

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\Theta = \{\beta_0, \beta_1\}$$

$$\epsilon = |\hat{y} - y|$$

Optimization function

$$E_\Theta = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$\boxed{\Theta^* = \text{argmin}_\Theta\, E\{(x_i, y_i)\}_{i=1,\cdots,N}}$$

Linear to Polynomial Regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Regularizer

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{1}{2}\|\mathbf{w}\|^2$$

Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Abs values
Are increasing

Model generalization: Regularization

Linear to Polynomial Regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

||W||^2

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{1}{2}\|\mathbf{w}\|^2$$

decreasing

Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

How to control this?

Model generalization: Regularization

Linear to Polynomial Regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Hyperparameter

$||W||^2$

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Model generalization: Regularization

Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Who to control this?

Probabilistic Twin

Of

The Least Squares Solution:

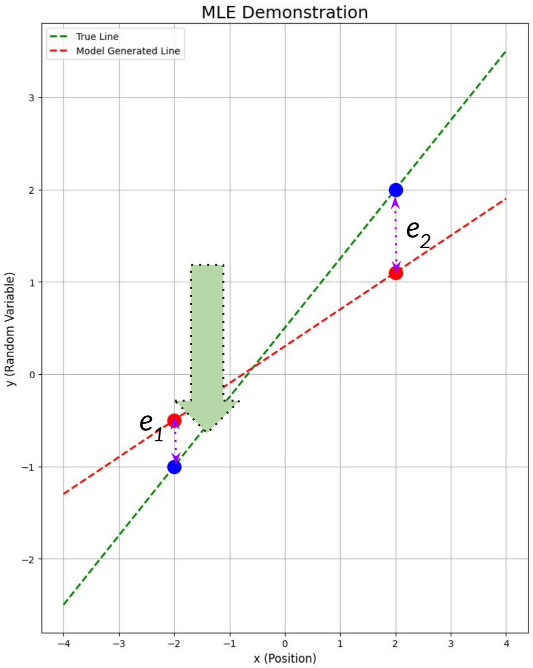
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

Probabilistic Twin

Of

The Least Squares Solution:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$



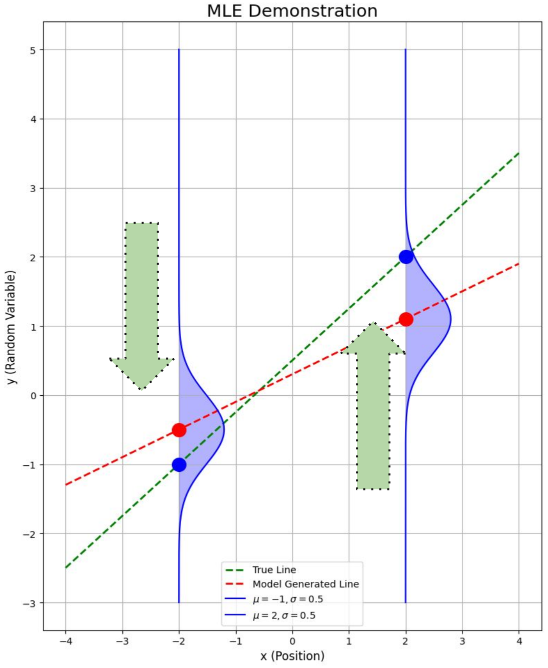
Probabilistic Twin

Probabilistic Formulation: Modeling Error Distribution

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n | y(x_n, \mathbf{w}), \beta^{-1}\right).$$

Taking the log

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi).$$



Probabilistic Twin

Probabilistic Formulation: Modeling Error Distribution

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n | y(x_n, \mathbf{w}), \beta^{-1}\right).$$

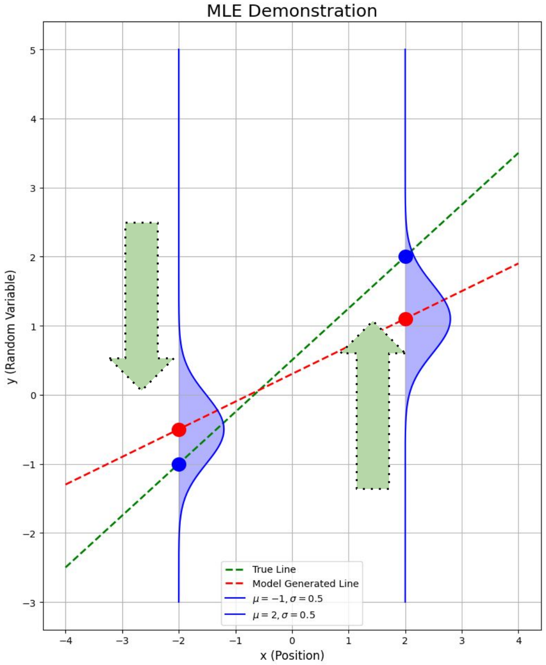
Taking the log

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi).$$

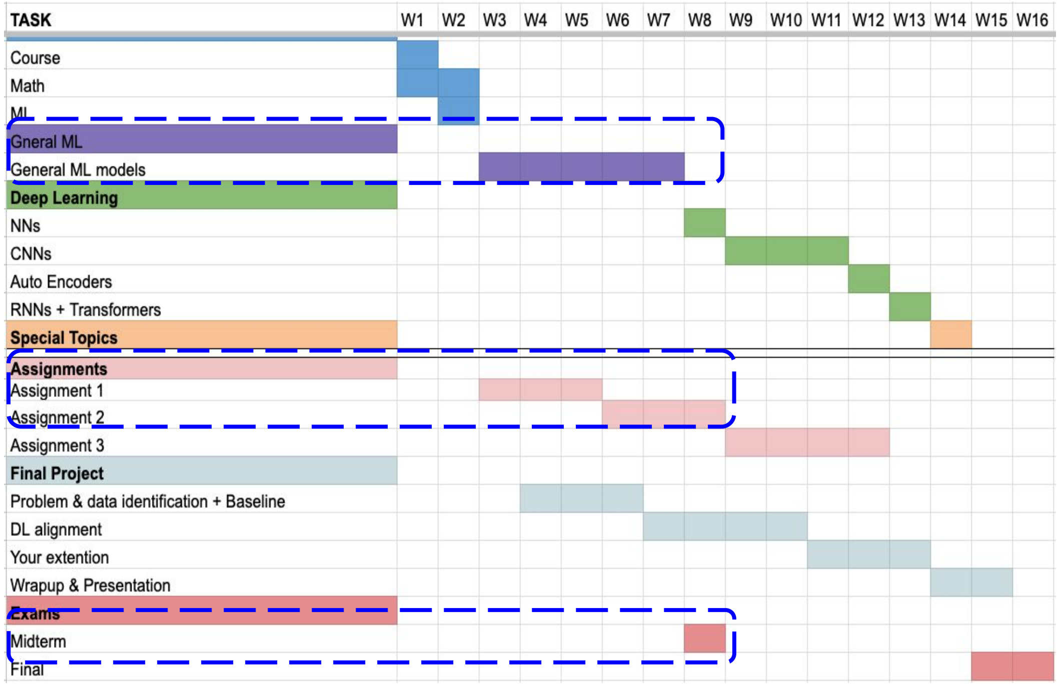
Does it look familiar???

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Maximizing Log likelihood is equivalent to minimizing the quadratic loss/error in the context of LR!



General ML models, Assignment 1,2, & Midterm



General ML models

Supervised

- kNN
- Linear Regression
- Decision Tree
- Meta learners
  - Random Forest Regressor
  - Boosting Regressor
- Support Vector Regressor (SVRs)

Regression

- kNN
- Logistic Regression
- Decision Tree
- Meta Learners
  - Random Forest Classifier
  - Boosting Classifiers
- Support Vector Classifiers (SVCs)
- Naive Bayes

Classification

General ML models

Supervised

Regression

- kNN
- Linear Regression
- Decision Tree
- Meta learners
  - Random Forest Regressor
  - Boosting Regressor
- Support Vector Regressor (SVRs)

Classification

- kNN
- Logistic Regression
- Decision Tree
- Meta Learners
  - Random Forest Classifier
  - Boosting Classifiers
- Support Vector Classifiers (SVCs)
- Naive Bayes

Parametric vs nonparametric

General ML models

Model generalization

- Universal concepts (applies to all models)
  - Cross validation
  - HP optimization

Universal concepts

- Overfitting
- Underfitting

Overfitting vs Under fitting

General ML models

Model generalization

- Training set, Validation set, Test set
- iid data

Training	Test / Hold out set

Data splits

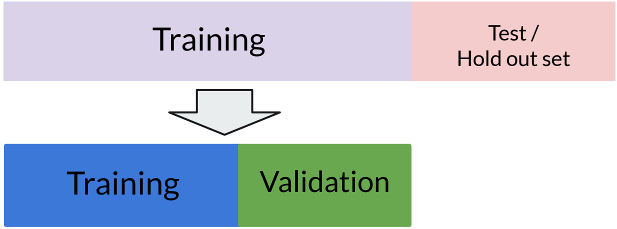
- Overfitting
- Underfitting

Overfitting vs Under fitting

General ML models

Model generalization

- Training set, Validation set, Test set
- iid data

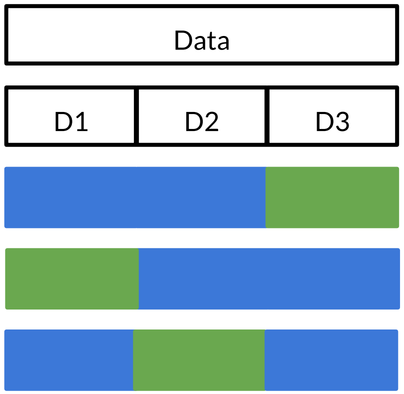


Data splits

- Overfitting
- Underfitting

Overfitting vs Under fitting

K-fold-cross validation



3-fold-cv

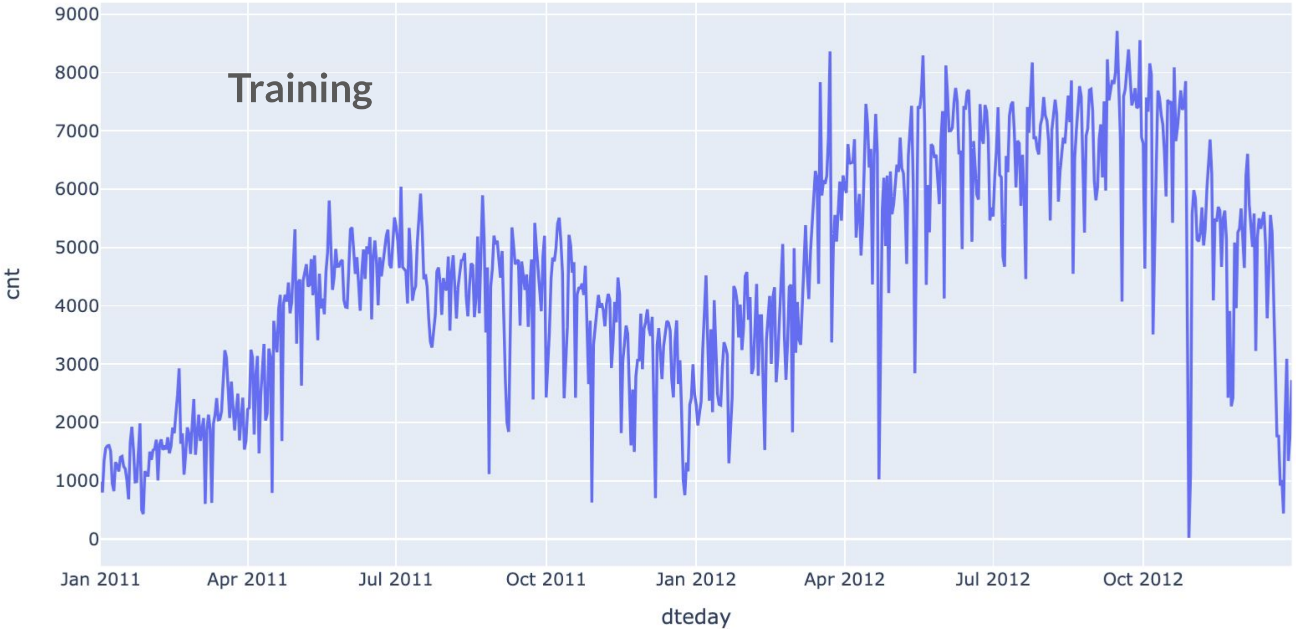
Train

validate

What HP gives the best validation score?

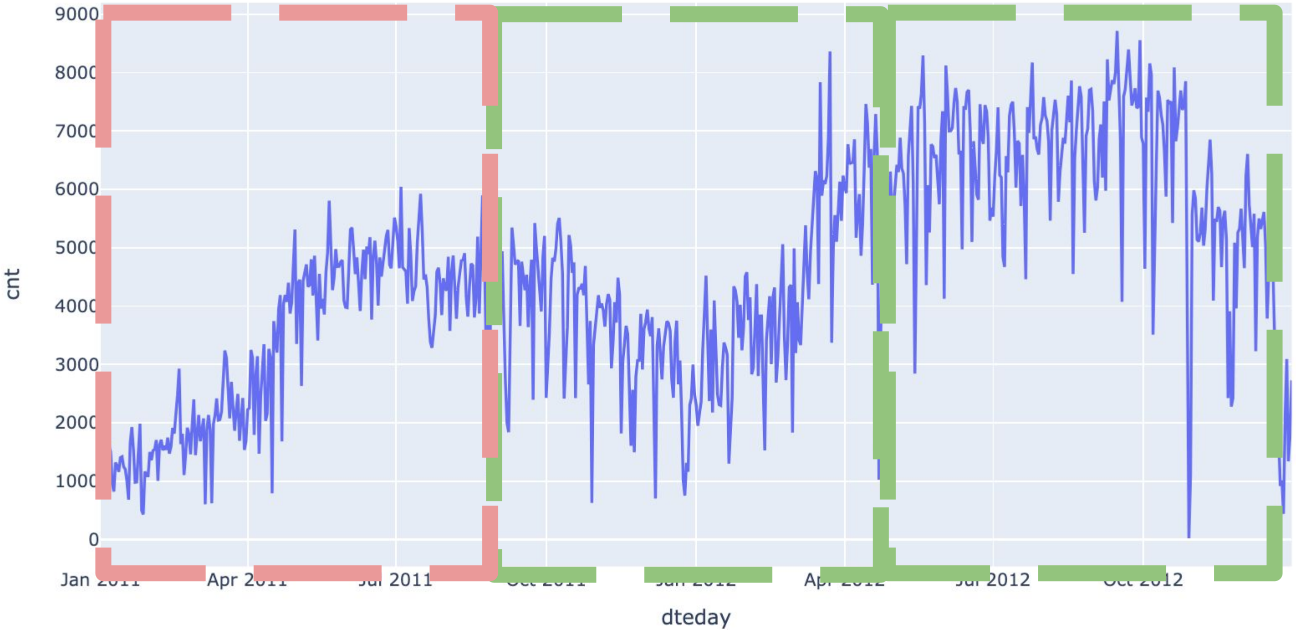
How to CV Sequential Data/Models

How to CV Sequential Data/Models



Always follow!

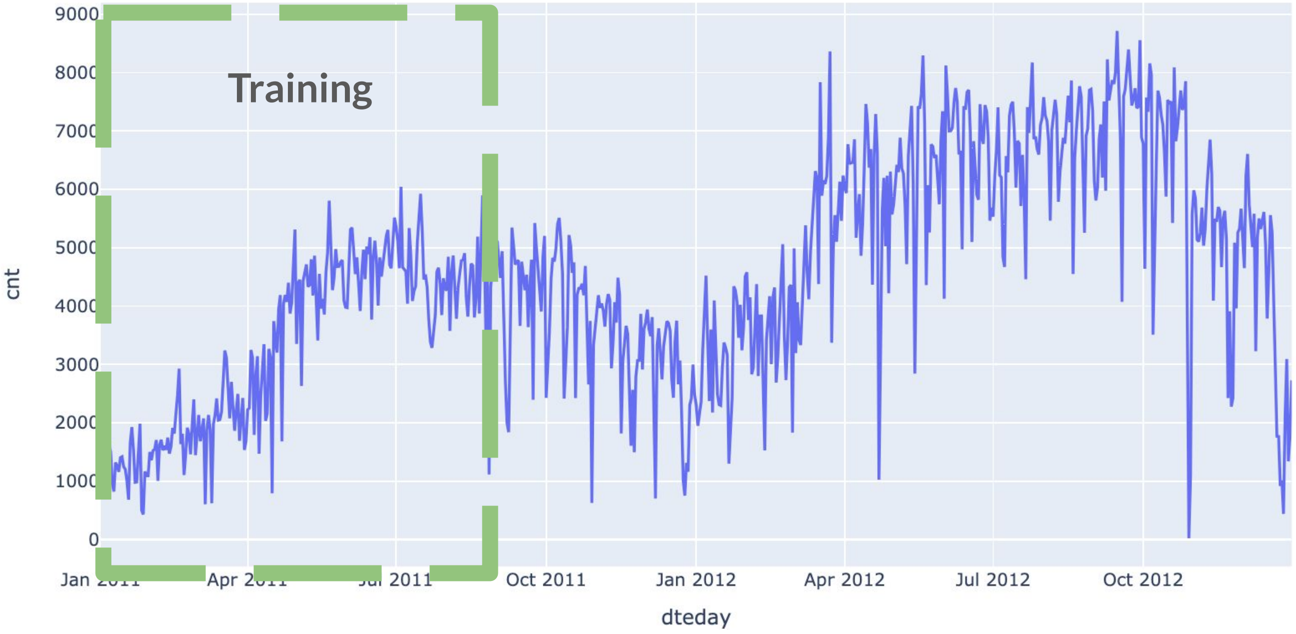
Sequential cross validation



Invalid configuration:

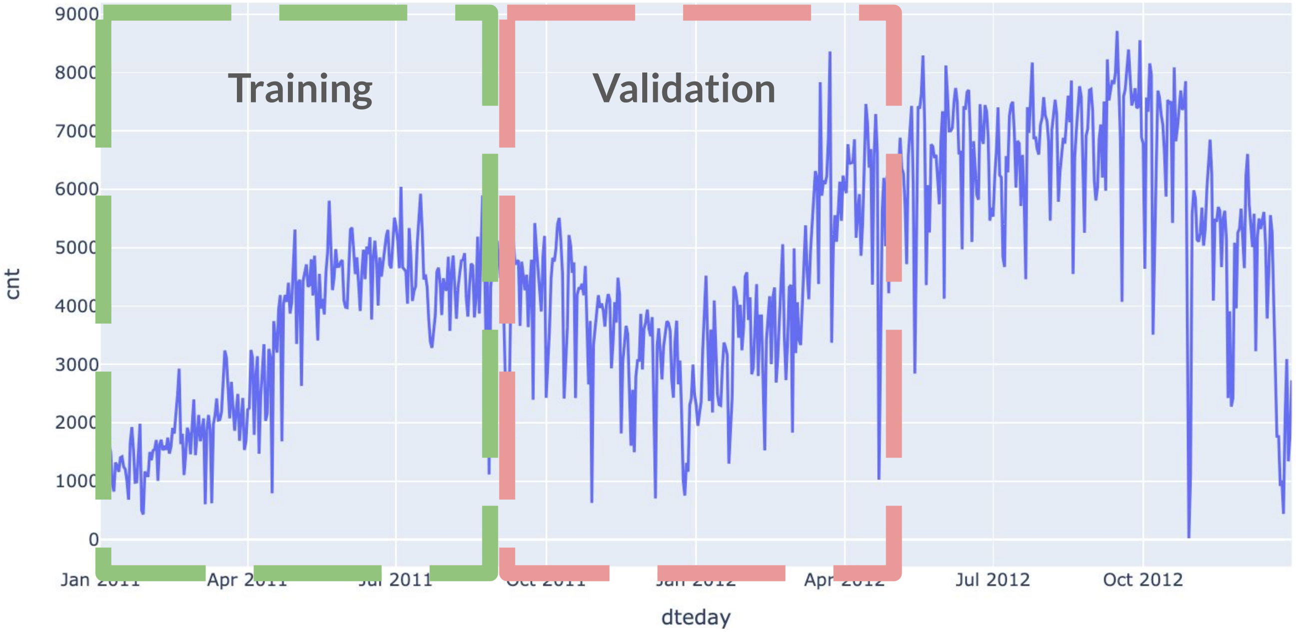
Training on the last two and validation on the ***first fold****.*

Sequential cross validation



Always follow!

Sequential cross validation



Always follow!

Sequential cross validation

Always follow!

Sequential cross validation

Always follow!

Unsupervised Learning

Supervised Learning

- We learned about **Classification** and **Regression**
- These are examples of **supervised** learning
- In your data you have both **X(features)** and **y(Labels)**

f(y|X)

Label (y) is predefined

Classification

X	$y \in \{cat,\ dog,\ rabbit\}$
$x_1,\ x_2,\ \cdots\ x_m$	cat
$x_1,\ x_2,\ \cdots\ x_m$	rabbit
$\cdots$	$\cdots$
$x_1,\ x_2,\ \cdots\ x_m$	dog

X	$y \in R$
$x_1,\ x_2,\ \cdots\ x_n$	1.9
$\cdots$	$\cdots$
$x_1,\ x_2,\ \cdots\ x_n$	2.5

Regression

Unsupervised Learning

- In contrast, in **Unsupervised learning**, we have to learn meaningful **representations/models** from **X(features)** only.
- **Clustering** is an example of unsupervised learning

X
$x_1, x_2, \cdots x_o$
$\cdots$
$x_1, x_2, \cdots x_o$

No concept of data label (y)

Unsupervised Learning

- In contrast, in **Unsupervised learning**, we have to learn meaningful **representations/models** from **X(features)** only.
- **Clustering** is an example of unsupervised learning

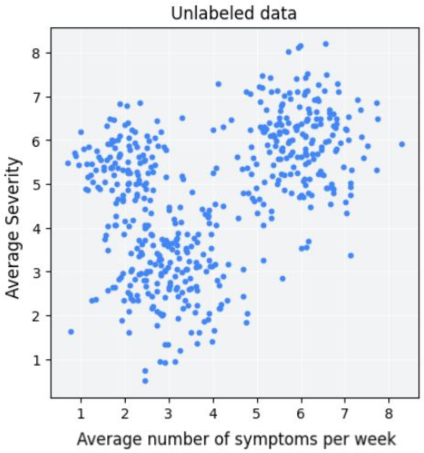
X
$x_1, x_2, \cdots x_o$
$\cdots$
$x_1, x_2, \cdots x_o$

No concept of data label (y)

$f(y|X)$

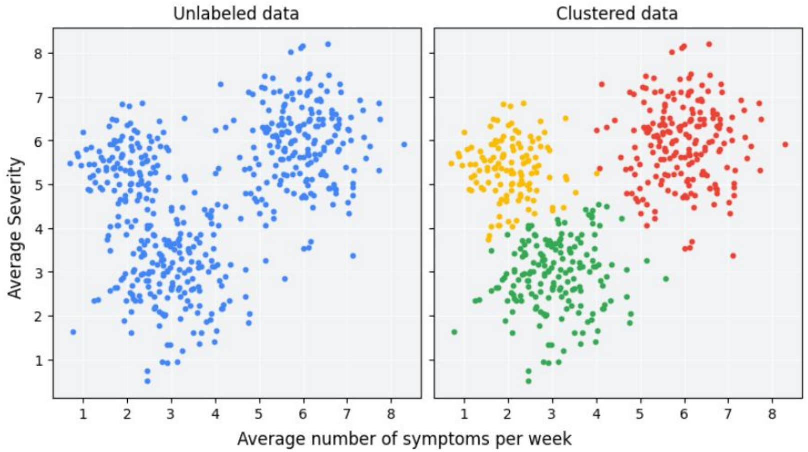
Clustering

- A medical study involving
  - Average number of symptoms per week
  - Average severity



Clustering

- A medical study involving
  - Average number of symptoms per week
  - Average severity



Usages of Clustering

Clustering has a myriad of uses in a variety of industries. Some common applications for clustering include the following:

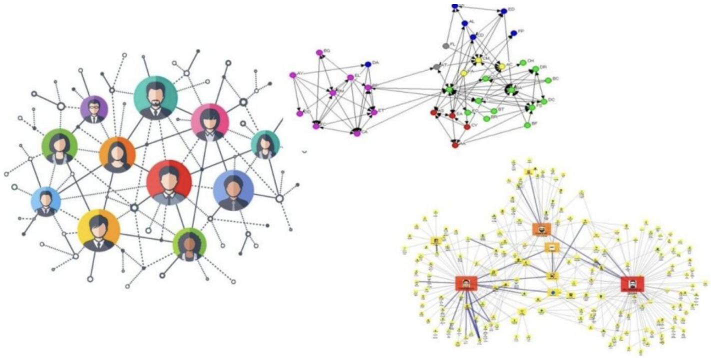
- **Market segmentation**
- Social network analysis
- Search result grouping
- Medical imaging
- Image segmentation
- Anomaly detection

ref

Usages of Clustering

Clustering has a myriad of uses in a variety of industries. Some common applications for clustering include the following:

- Market segmentation
- **Social network analysis**
- Search result grouping
- Medical imaging
- Image segmentation
- Anomaly detection



ref

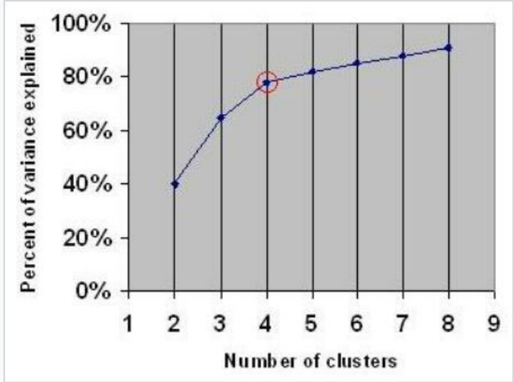
Clustering

- What is clustering?
- **Clustering algorithms:**
  - **K-Means**: Centroid Based
  - **Hierarchical clustering**: Distance connectivity based
  - **GMM**: Distribution based
  - **DBSCAN**: Density Based

- Identifying the number of clusters ?

Selecting the Number of clusters

- Elbow method
- Silhouette score based



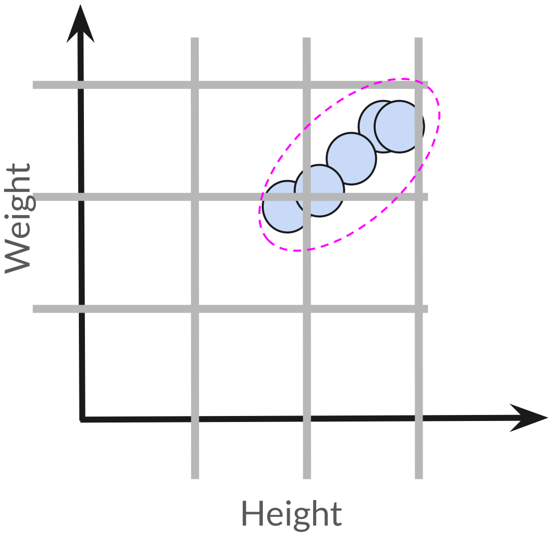
Explained Variance. The "elbow" is indicated by the red circle. The number of clusters chosen should therefore be 4.

Curse of Dimensionality

Curse of Dimensionality

- We have added Weight as our second feature dimension
- Do you find the samples represent our class?
- What's the domain coverage now?
- Is ~10% a reasonable assumption?
- See how numbers dropped from ~40% to ~10%
- The other way, the empty space grew up from ~60% to ~90%
- **What's the number would look like if we add 2 more dimensions?**

- Empty space grows exponentially with the increase in adding new features.
- **Data distribution becomes sparse, and difficult to learn a good model.**



Dimensionality Reduction

<u>General idea</u>

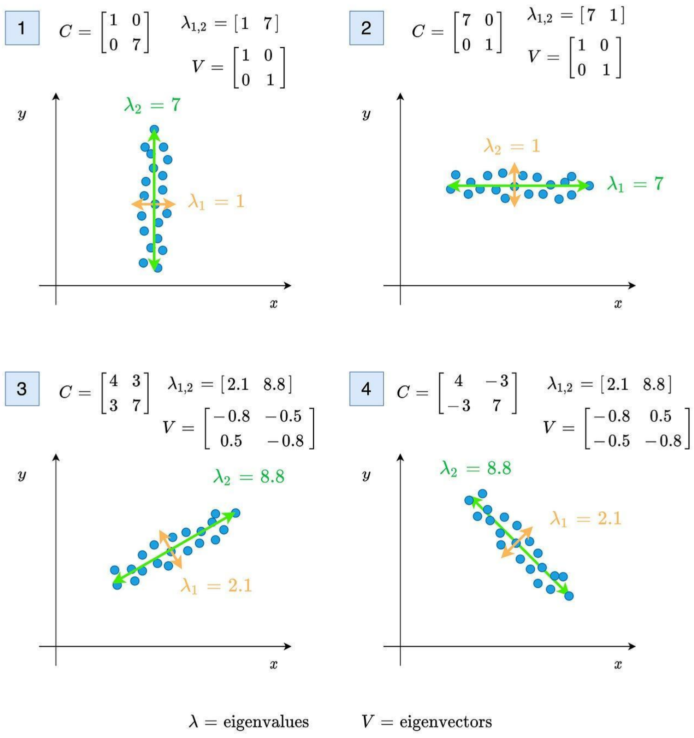
- You have some data of feature dimension size, $|D|$
- You want to compress them to of size, $|d|$
- $|d| < |D|$

Dimensionality Reduction

- Linear
    - Principal Component Analysis (PCA)
    - SVD
- Neural Networks
    - Auto Encoders
    - RBMs

PCA

[Notebook presentation]

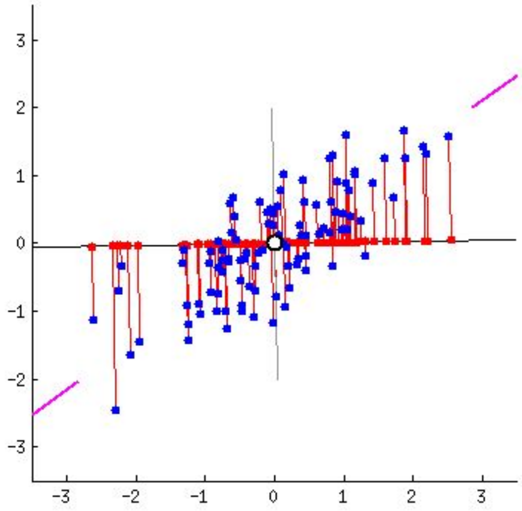


[ref]

What are we optimizing?

Two techniques (in general):

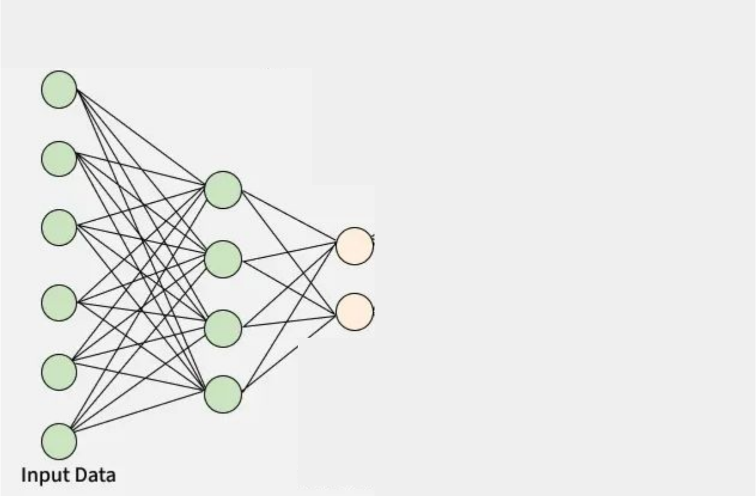
- Maximize variance
- Minimize errors (distance between each green-red paris)



ref

Unsupervised learning (nonlinear)

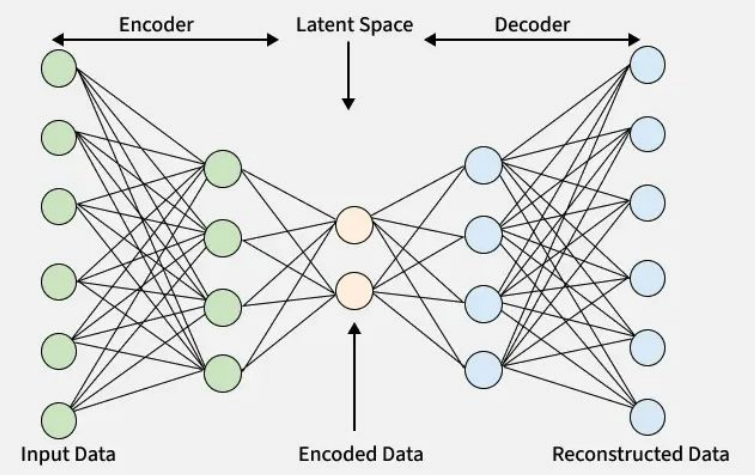
- **Auto Encoders**
- Restricted Boltzmann Machines (RBMs)



X Z

Unsupervised learning (nonlinear)

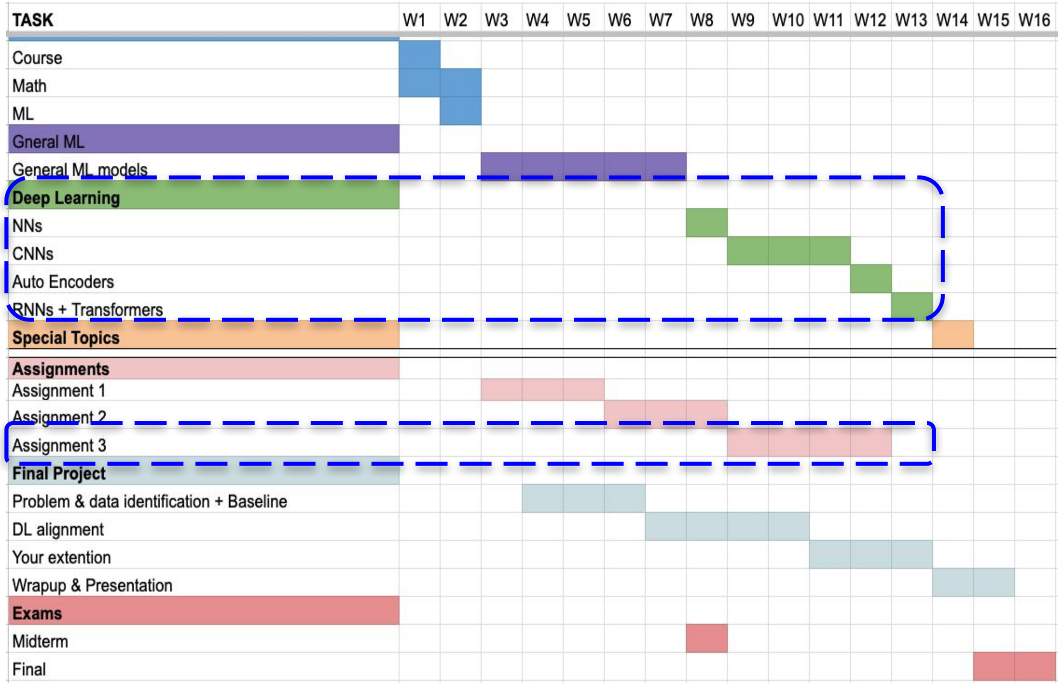
- **Auto Encoders**
- Restricted Boltzmann Machines (RBMs)



X Z X'

Deep Learning

Assignment 3

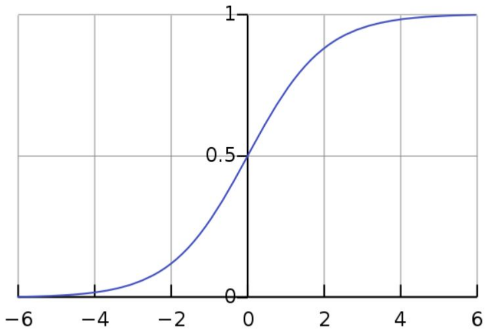


Logistic Regression

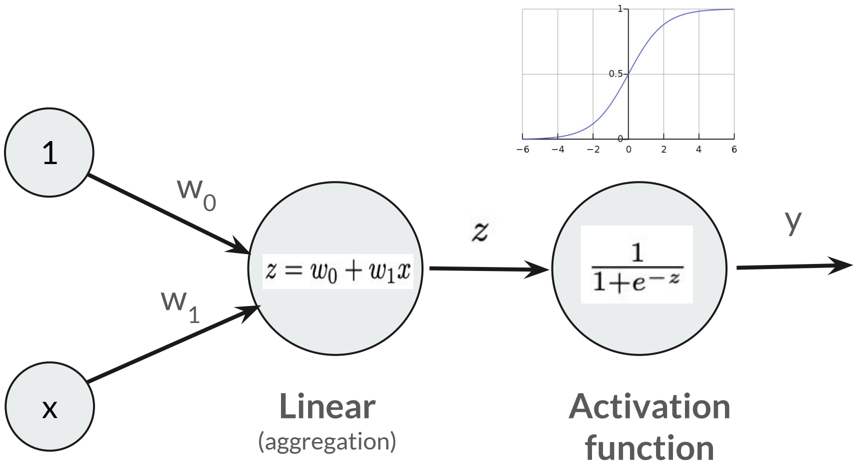
- Probabilistic classifier

$$p(x) = \frac{1}{1+e^{-(w_0+w_1x)}}$$

- Sigmoid function



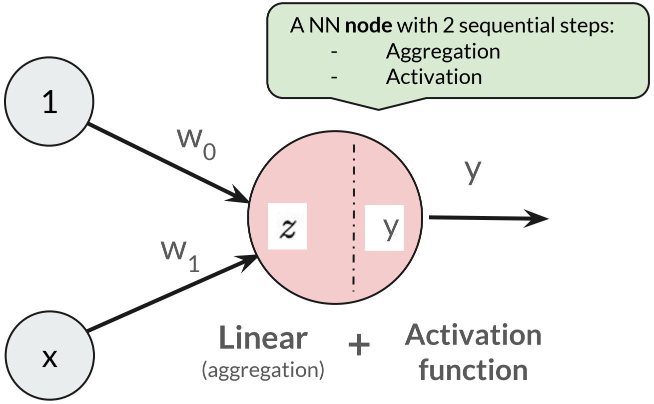
Neural Networks (Node)



$y \in \{0, 1\}$

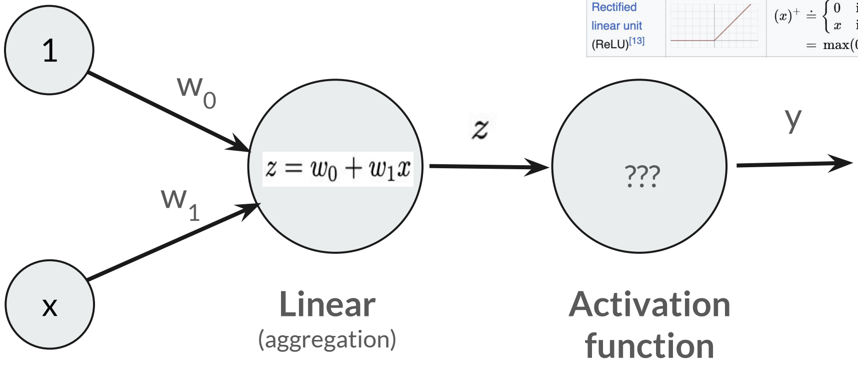
Logistic Regression

Neural Networks (Node)



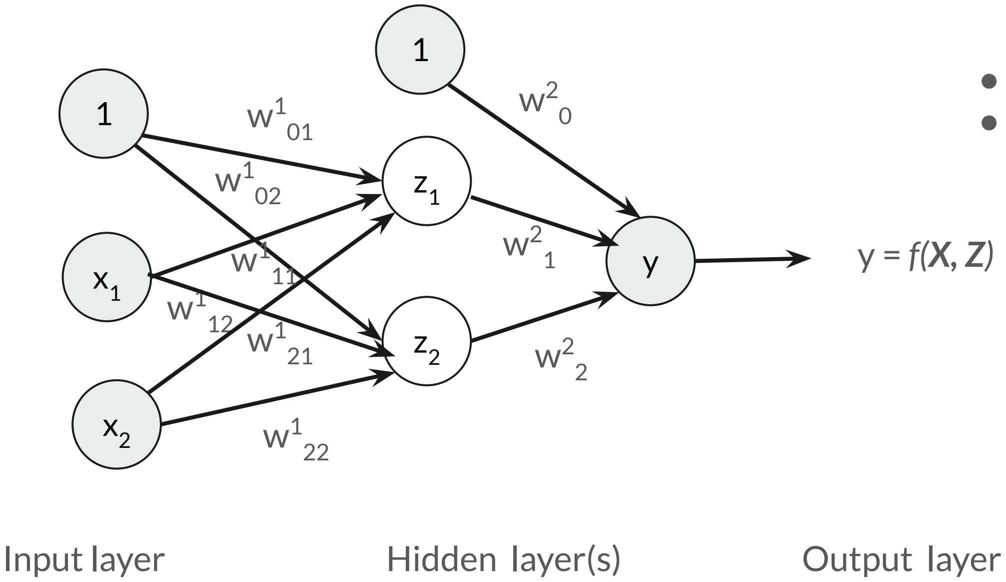
Neural Networks (Node)

Name	Plot	Function, $g(x)$
Identity		x
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Logistic, sigmoid, or soft step		$\sigma(x) \doteq \frac{1}{1+e^{-x}}$
Hyperbolic tangent (tanh)		$\tanh(x) \doteq \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Soboleva modified hyperbolic tangent (smht)		$\operatorname{smht}(x) \doteq \frac{e^{ax}-e^{-bx}}{e^{cx}+e^{-dx}}$
Rectified linear unit (ReLU)[13]		$(x)^{+} \doteq \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max(0, x) = x\mathbf{1}_{x>0}$



A NN with ???
Activation function

Feed-forward (FF) neural networks



$y = f(\boldsymbol{X}, \boldsymbol{Z})$

- Feedforward NN
- Perceptron

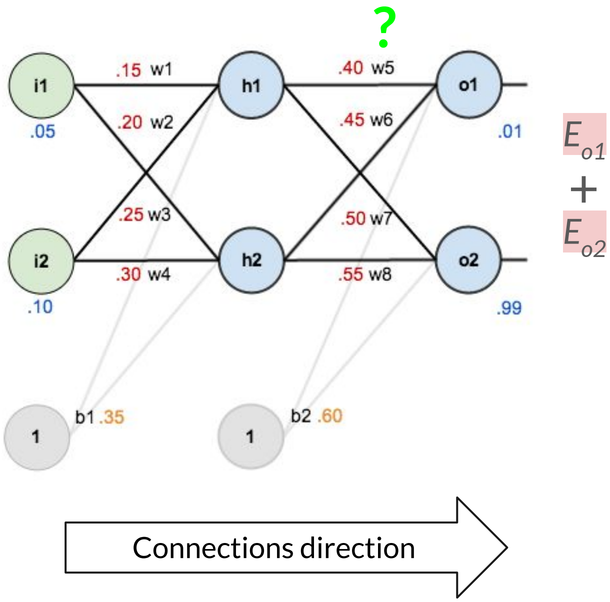
Error Back propagation

Gradient Descent (Error Back Propagation)

The Backwards Pass

Let’s focus on $\frac{\partial E_{total}}{\partial w_5}$

What would be the gradient update for w5?

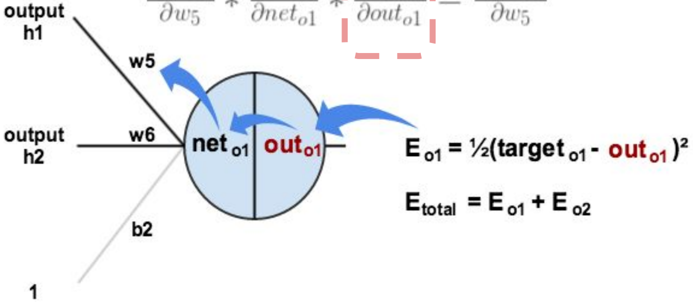


Adapted from

Gradient Descent (Error Back Propagation)

The Backwards Pass

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \boxed{\frac{\partial E_{total}}{\partial out_{o1}}} = \frac{\partial E_{total}}{\partial w_5}$$



$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

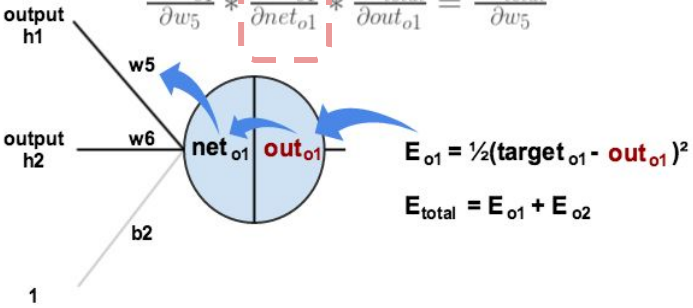
$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

[Adapted from](#)

Gradient Descent (Error Back Propagation)

The Backwards Pass

$$\frac{\partial net_{o1}}{\partial w_5} * \boxed{\frac{\partial out_{o1}}{\partial net_{o1}}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



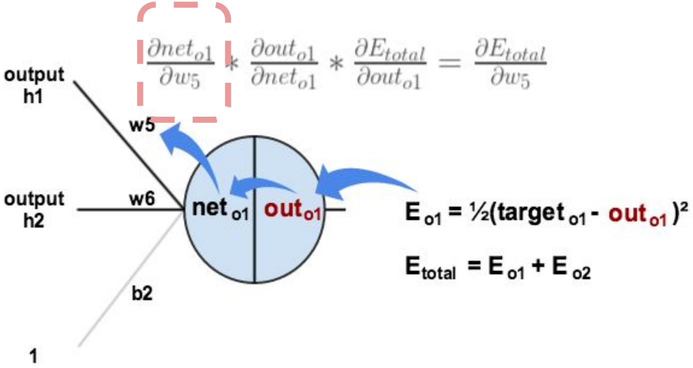
$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$
$$= 0.75136507(1 - 0.75136507)$$
$$= 0.186815602$$

[Adapted from]

Gradient Descent (Error Back Propagation)

The Backwards Pass



$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

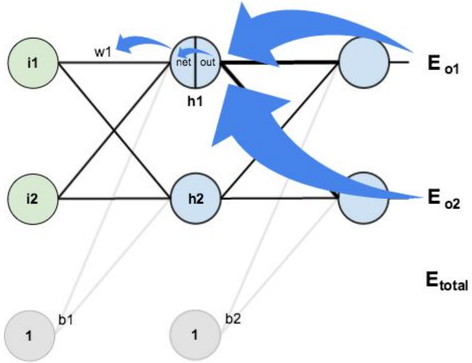
[Adapted from]

Gradient Descent (Error Back Propagation)

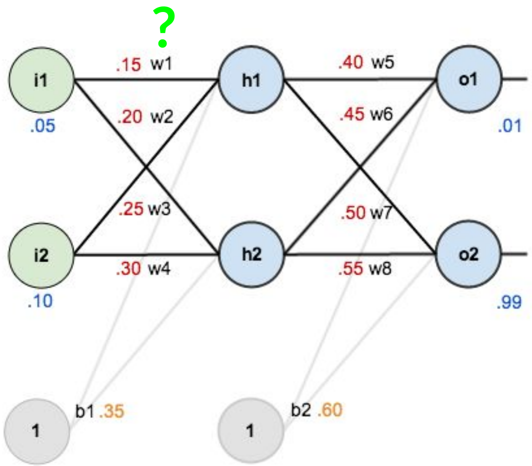
Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$E_{total} = E_{o1} + E_{o2}$



E_{o1}
$+$
E_{o2}

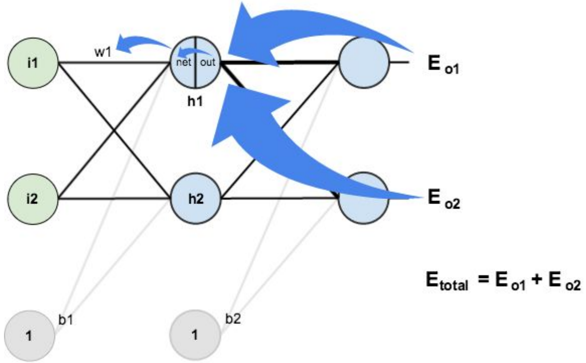
- While changing w5 affects only O1, a change in w1 will change both O1 and O2

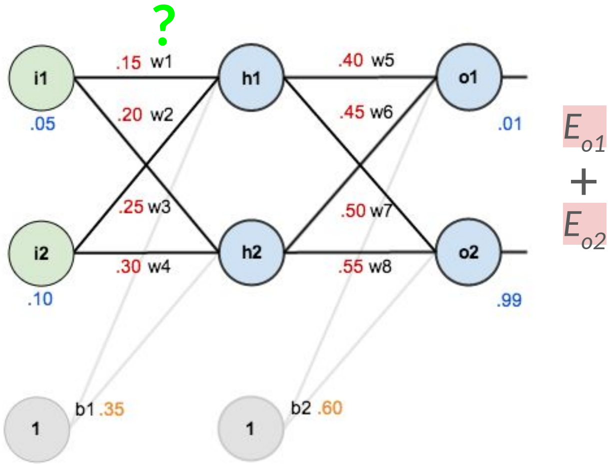
Gradient Descent (Error Back Propagation)

Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

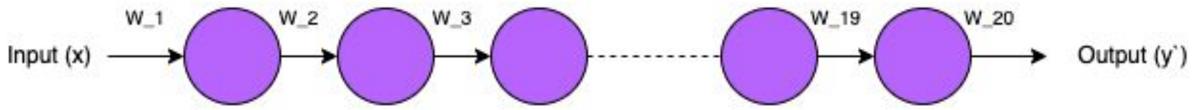
$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$





- Can you think of an arbitrary node in a giant and complex NN? What challenges we may encounter?

Vanishing and Exploding Gradients



20-Layer Neural Network

$$\boxed{o_1 = a_1(w_1.x)}$$

$$\boxed{o_2 = a_2(w_2.a_1(w_1.x))}$$

.... (l=20)

ref

Convolutional NNs

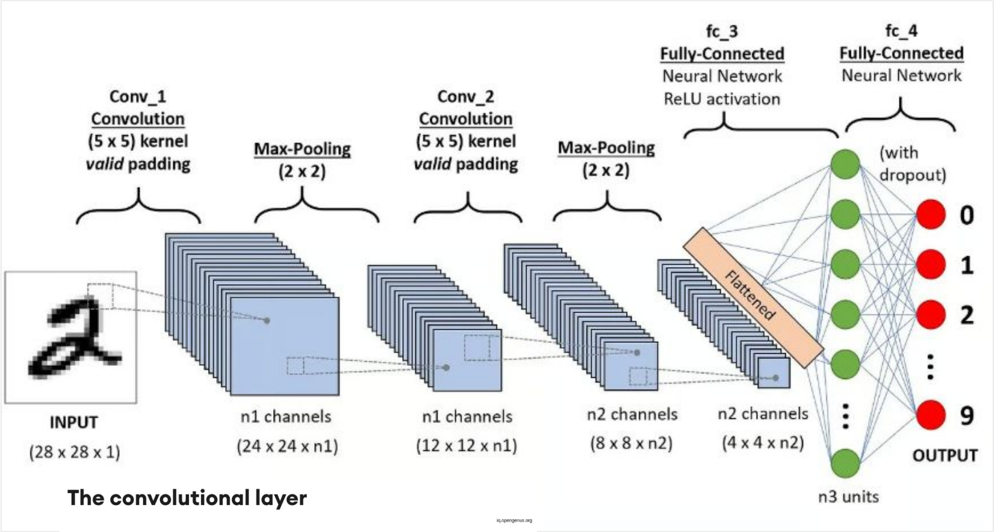
- State of the Art for CV and some other problems
- Filters/Convolutional Kernels

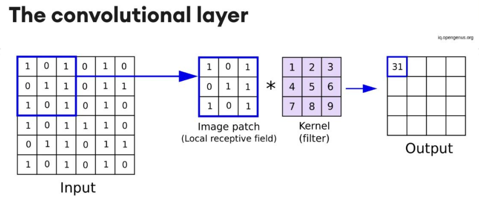
Examples:

- *Alexnet*
- *VGG*
- *ResNet*
- *GoogLeNet*
- ..

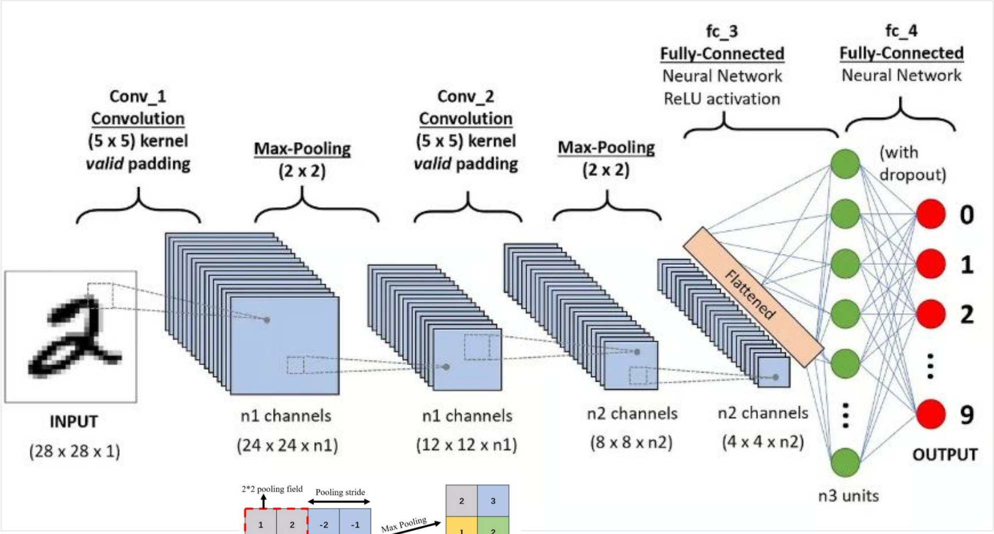
ref

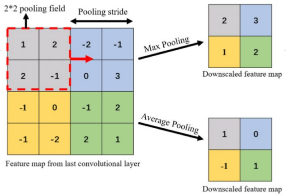
Convolutional NNs





Convolutional NNs





How are the time-series problems different?

- The models (Regression and Classification), we have learned so far are of the form:

  $f(y|X)$

- Purely time series models are of form :

  $f(y_t|y_{t-1}, y_{t-2, \ldots} y_0)$, where there is no explicit, X.

- For certain data, especially the time series, we can take advantage of the form:

  $f(y_t|X, y_{t-1}, y_{t-2, \ldots} y_0)$; essentially, here the input features are X plus the lagged instances of the target y.

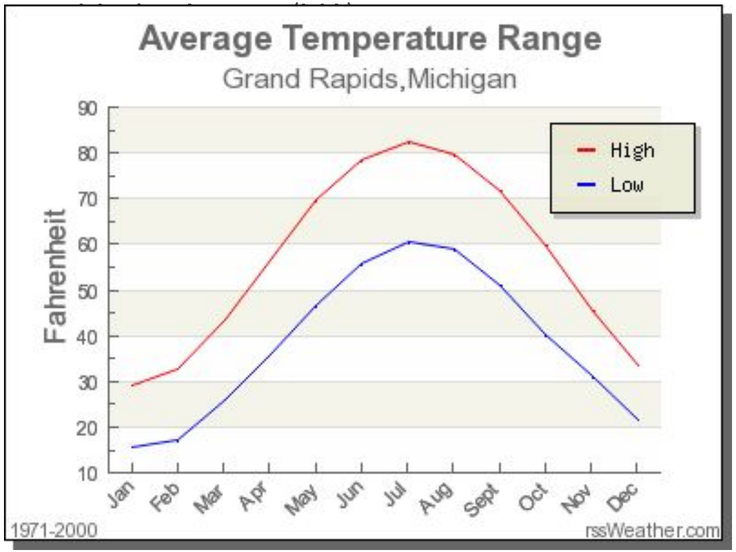
Statistical time series models

- Autoregressive Models (AR)
- Moving Average (MA)
- Autoregressive Moving Average (ARMA)
- Autoregressive Moving Integrated Average (ARIMA)

$$f(y_t | y_{t-1,}\ y_{t-2,} \ldots y_0)$$

Statistical time series models

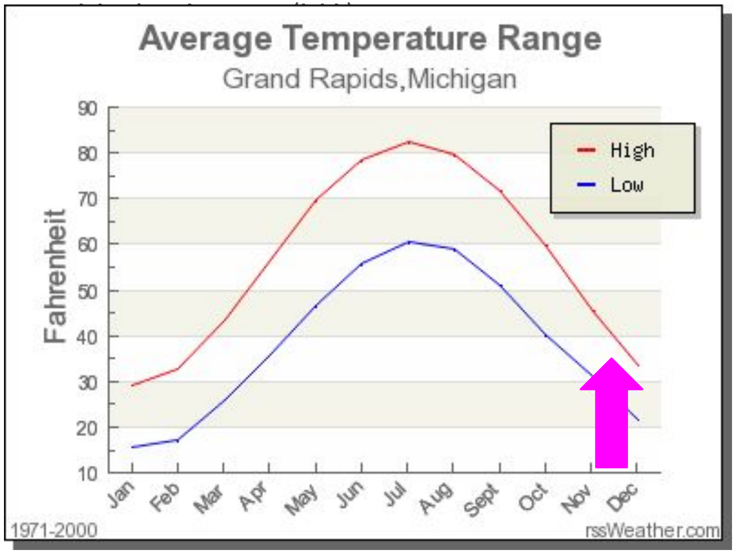
- Autoregressive Models (AR)



$$f(y_t|y_{t-1}, y_{t-2}, ...y_0)$$

Statistical time series models

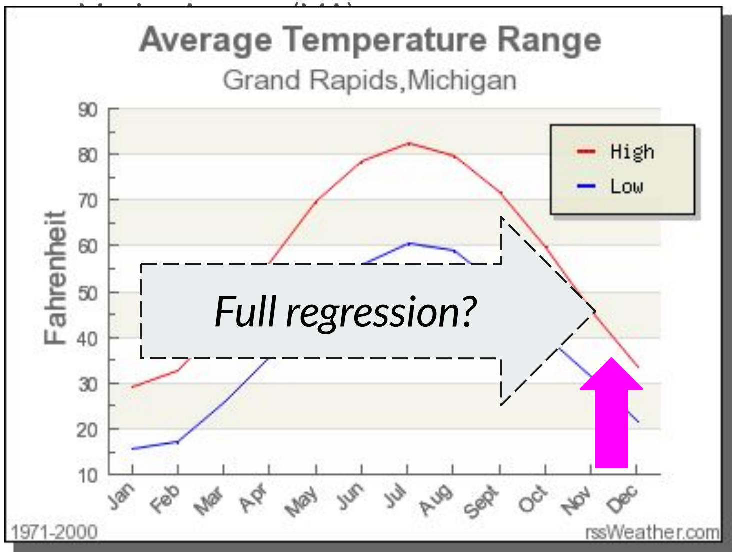
- Autoregressive Models (AR)



$$f(y_t|y_{t-1,}\, y_{t-2, \ldots\ldots}y_0)$$

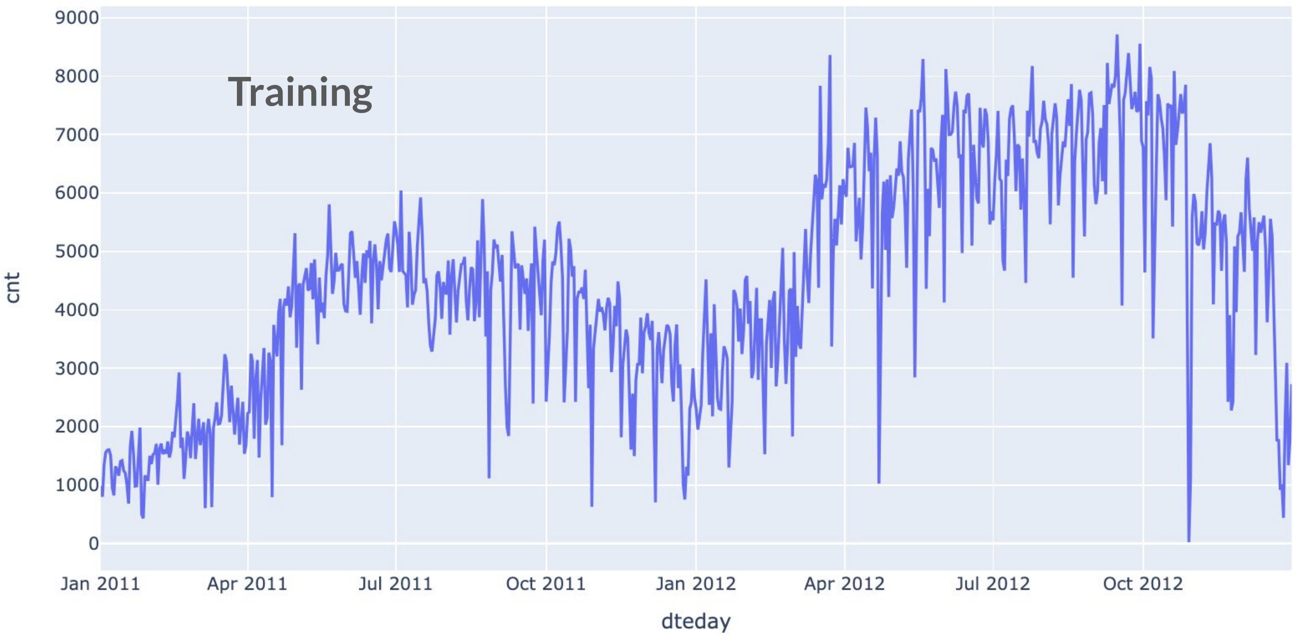
Statistical time series models

- Autoregressive Models (AR)



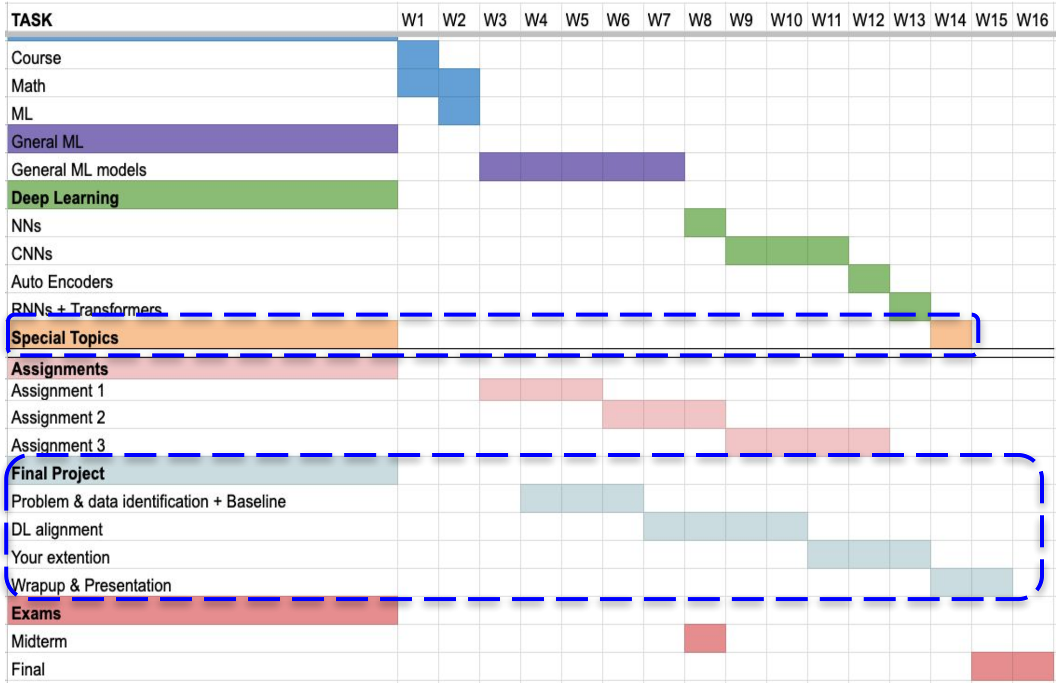
$$f(y_t | y_{t-1}, y_{t-2}, \ldots y_0)$$

How to CV Sequential Data/Models



Always follow!

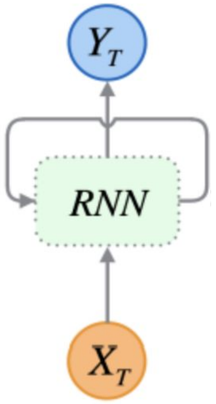
Final Project + Special Topics



Recurrent Neural Networks

Originally designed to Solve:

- Sequential Problems
    - *NLP (MT, ...), Speech Recognition,...*
    - *DNA Sequencing*
- Time-series Problems

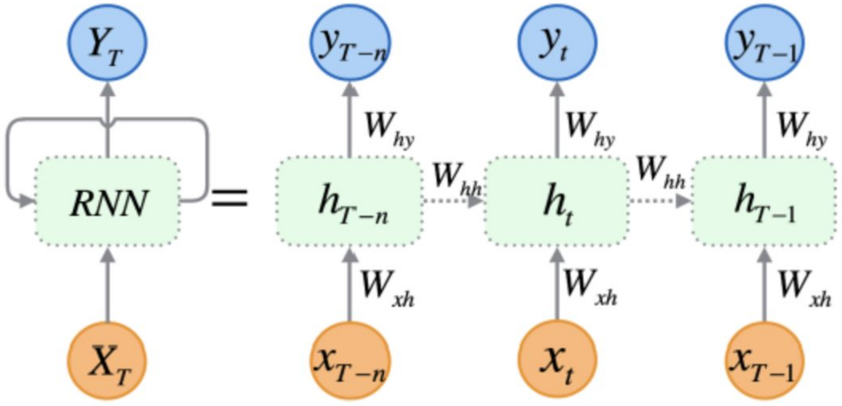


ref

Recurrent Neural Networks

Originally designed to Solve:

- Sequential Problems
    - *NLP (MT, ...), Speech Recognition,...*
    - *DNA Sequencing*
- Time-series Problems



ref

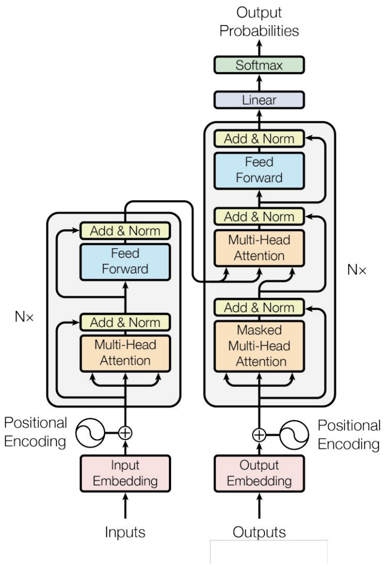
Transformers

Examples:

- *Encoder decoder pair*
- *GPT*
- *BERT*

BERT

Encoder



GPT

Decoder

[ref]

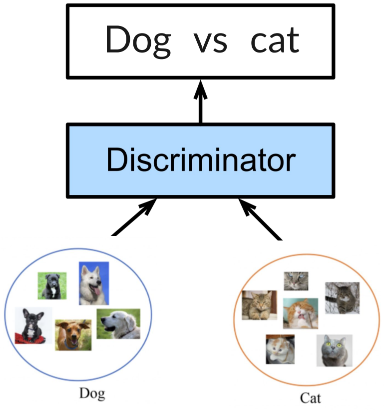
Generative AI

Examples:

- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)

ref

Generative AI

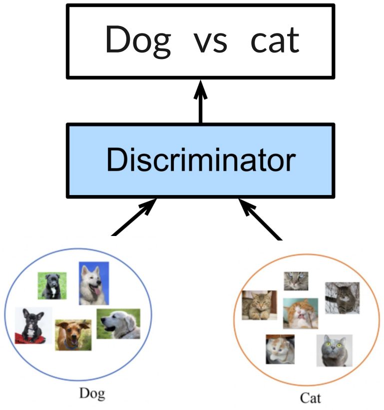


ref

Generative AI

Examples:

- Generative Adversarial Networks (GANs)

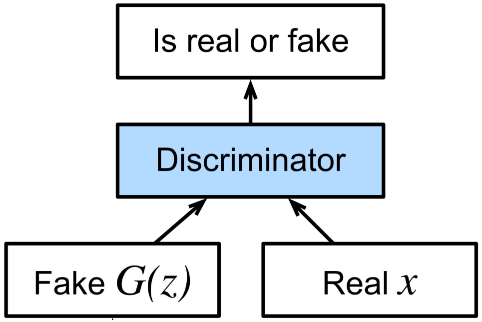


ref

Generative AI

Examples:

- Generative Adversarial Networks (GANs)

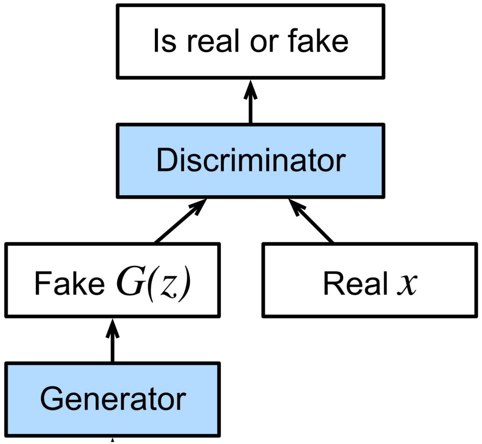


ref

Generative AI

Examples:

- Generative Adversarial Networks (GANs)

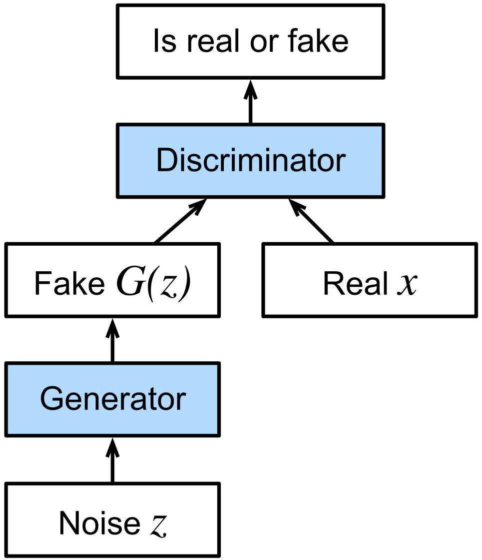


ref

Generative AI

Examples:

- Generative Adversarial Networks (GANs)

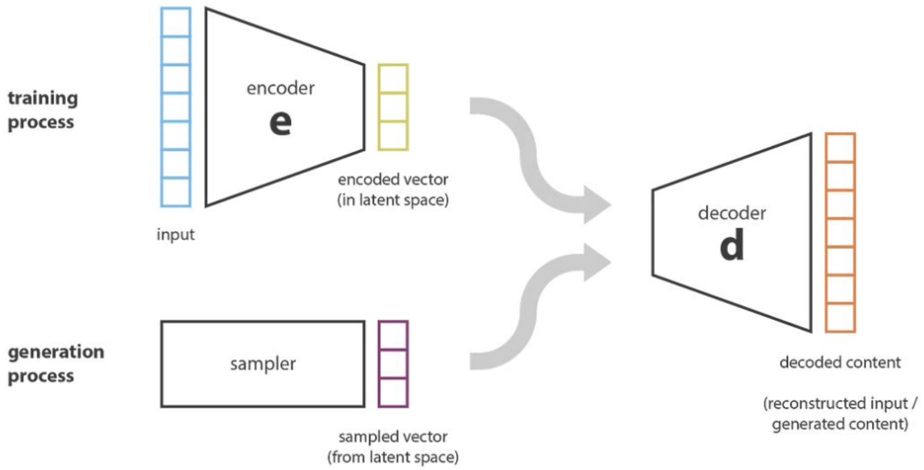


ref

Generative AI

Examples:

- Generative Adversarial Networks (G
- **Variational Autoencoders (VAEs)**



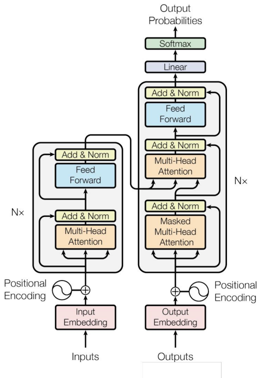
ref

Foundational models

- We don’t train independent models explicitly (say machine translation)

BERT

Encoder



Machine translation

GPT

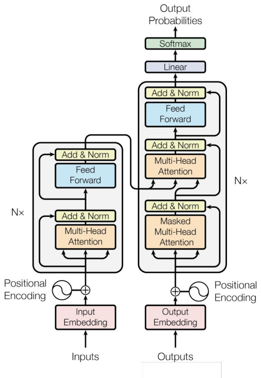
Decoder

Foundational models

- We don't train independent models explicitly (say machine translation)
- Or document summarization

BERT

Encoder



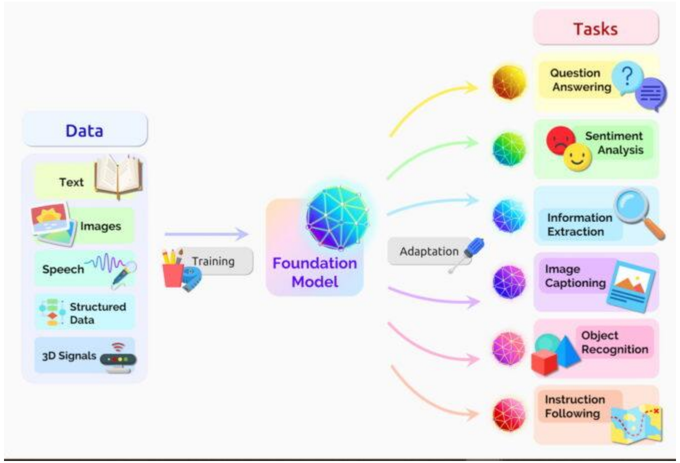
GPT

Decoder

Document summarization

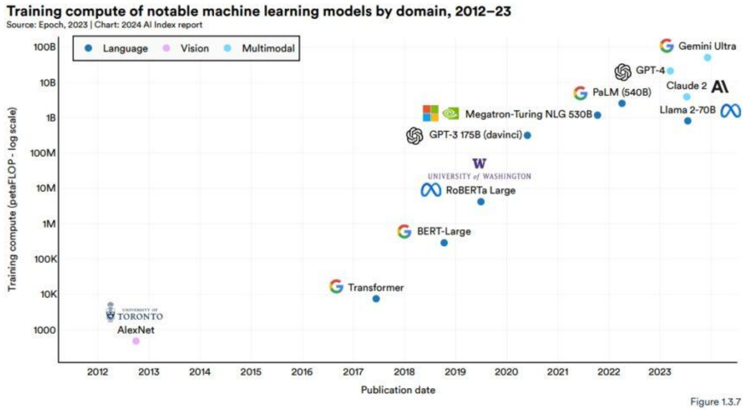
Foundational models

- We don't train independent models explicitly (say machine translation)
- Or document summarization
- **We train a Base model for multiple tasks jointly, and then fine tune for specific tasks**



Foundational models

- We don't train independent models explicitly (say machine translation)
- Or document summarization
- **We train a Base model for multiple tasks jointly, and then fine tune for specific tasks**



Training Neural Network Challenges

- Intractable gradients
  - Vanishing, and
  - Exploding gradients

Training Neural Network Challenges

- Intractable gradients
  - Vanishing, and
  - Exploding gradients
- Various normalizations
  - Input normalization (standard scalar)
  - Batch normalization
  - Layer normalization

Training Neural Network Challenges

- Intractable gradients
  - Vanishing, and
  - Exploding gradients
- Various normalizations
  - Input normalization (standard scalar)
  - Batch normalization
  - Layer normalization
- Controlling overfitting
  - Regularization
  - Early stopping
  - Drop out

QA