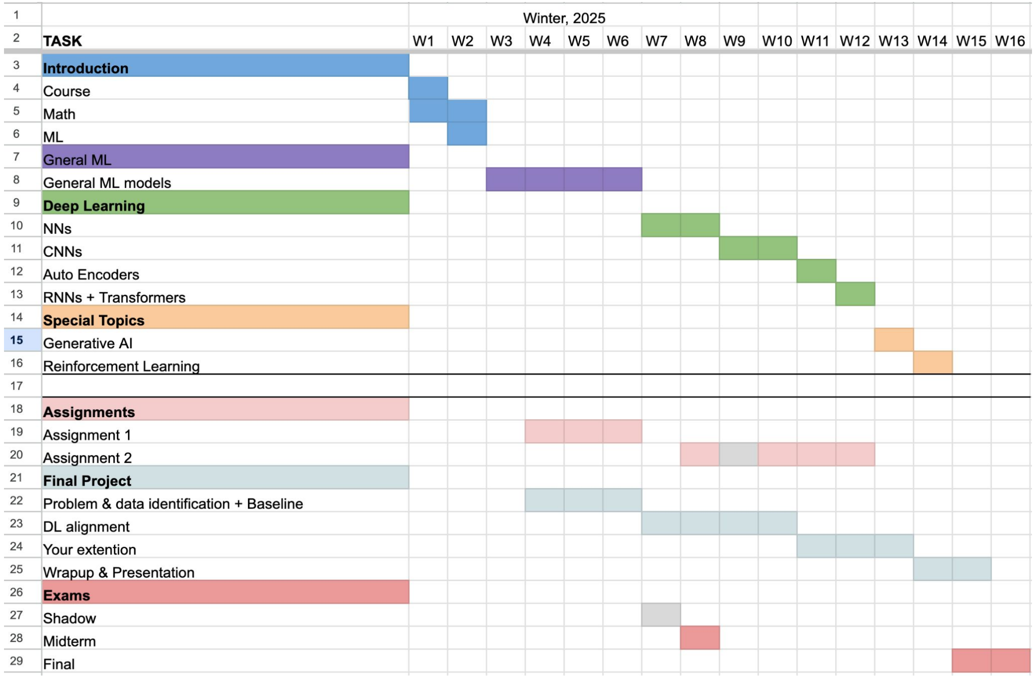
CIS 678 Machine Learning

Course Review Week

Our Plan


Winter, 2025
TASK
W1 W2 W3 W4 W5 W6 W7 W8 W9 W10 W11 W12 W13 W14 W15 W16
Introduction
Course
Math
ML
Gneral ML
General ML models
Deep Learning
NNs
CNNs
Auto Encoders
RNNs + Transformers
Special Topics
Generative AI
Reinforcement Learning
Assignments
Assignment 1
Assignment 2
Final Project
Problem & data identification + Baseline
DL alignment
Your extention
Wrapup & Presentation
Exams
Shadow
Midterm
Final

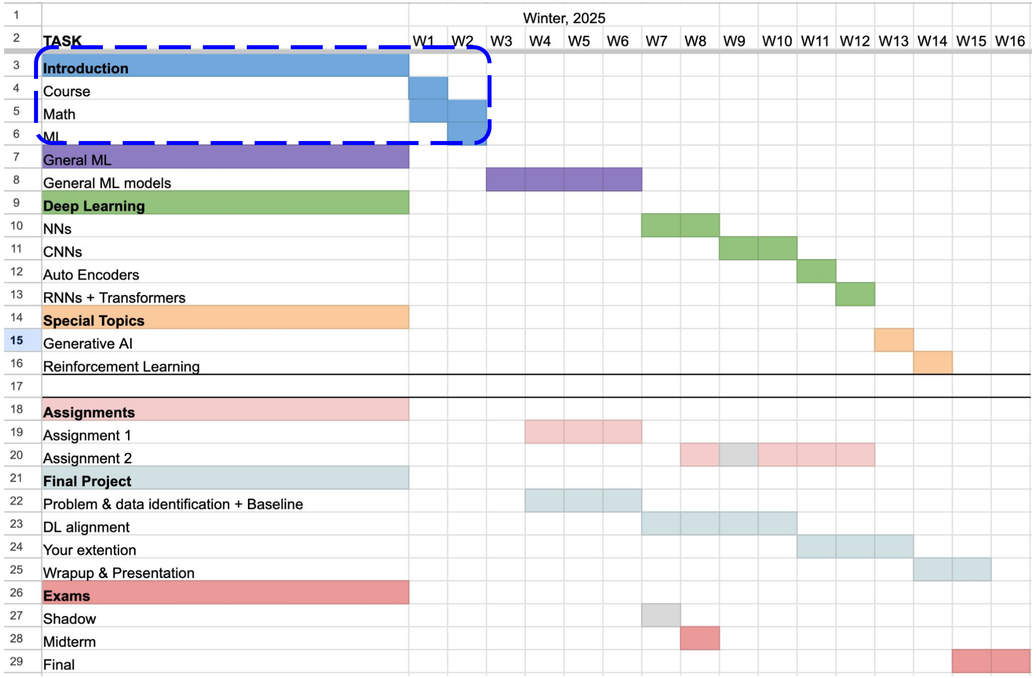
Background diagnostics + ML Introduction

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

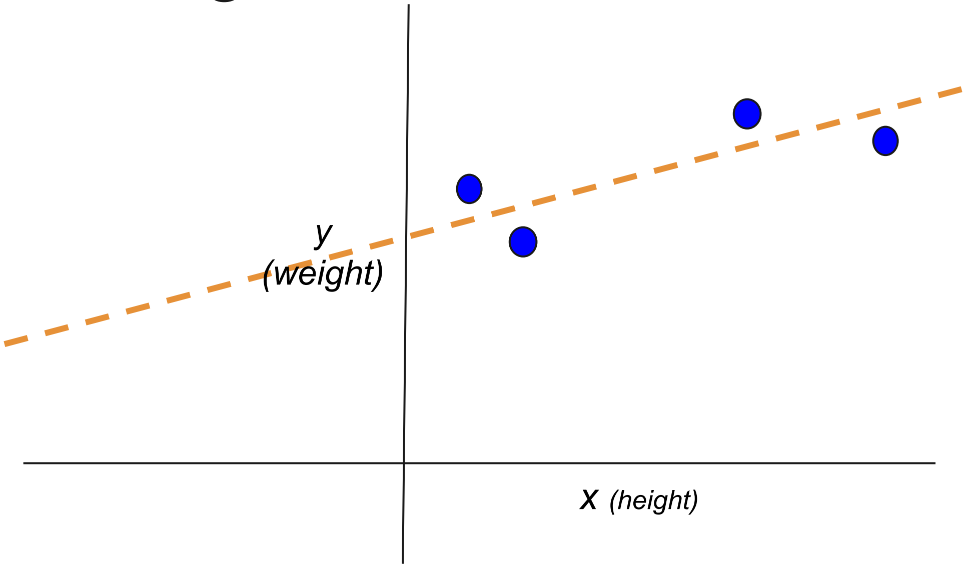
kNN model:
- Distance based
- Can be applied to both Regression and Classification tasks

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

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

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

Linear to Polynomial Regression

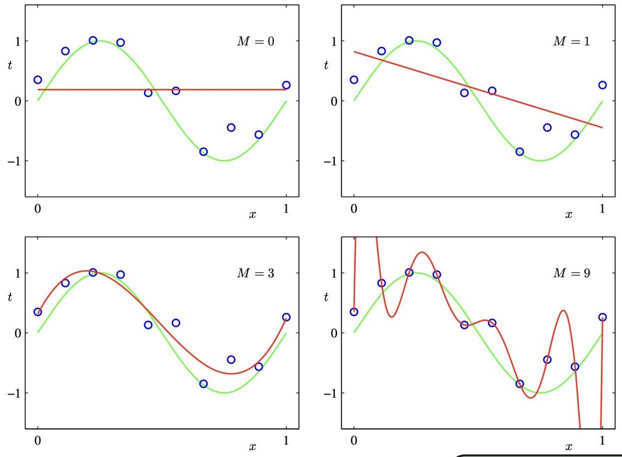


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	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

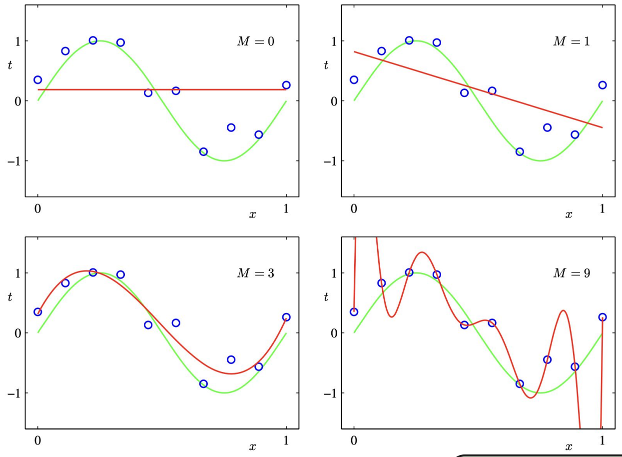


Table 1.1 Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	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

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

$||\mathbf{W}||^2$

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

decreasing

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

Abs values
Are increasing

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

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

How to control this?

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

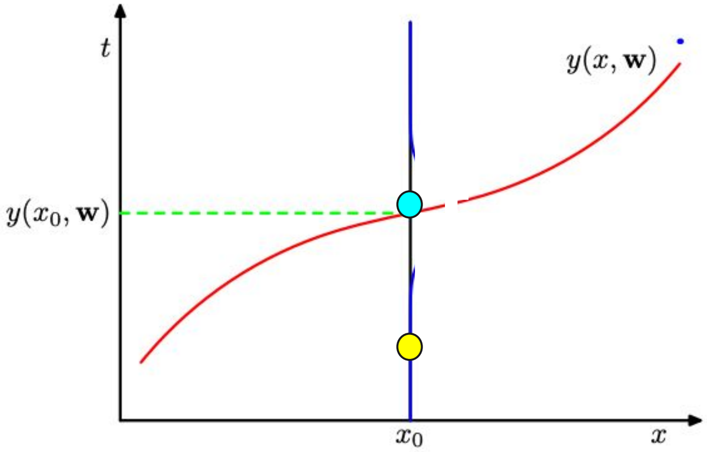
Who to control this?

Probabilistic equivalent

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

Probabilistic equivalent

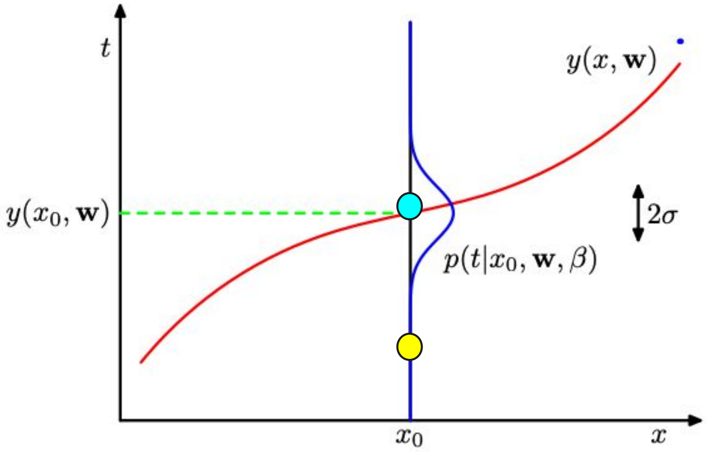
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$



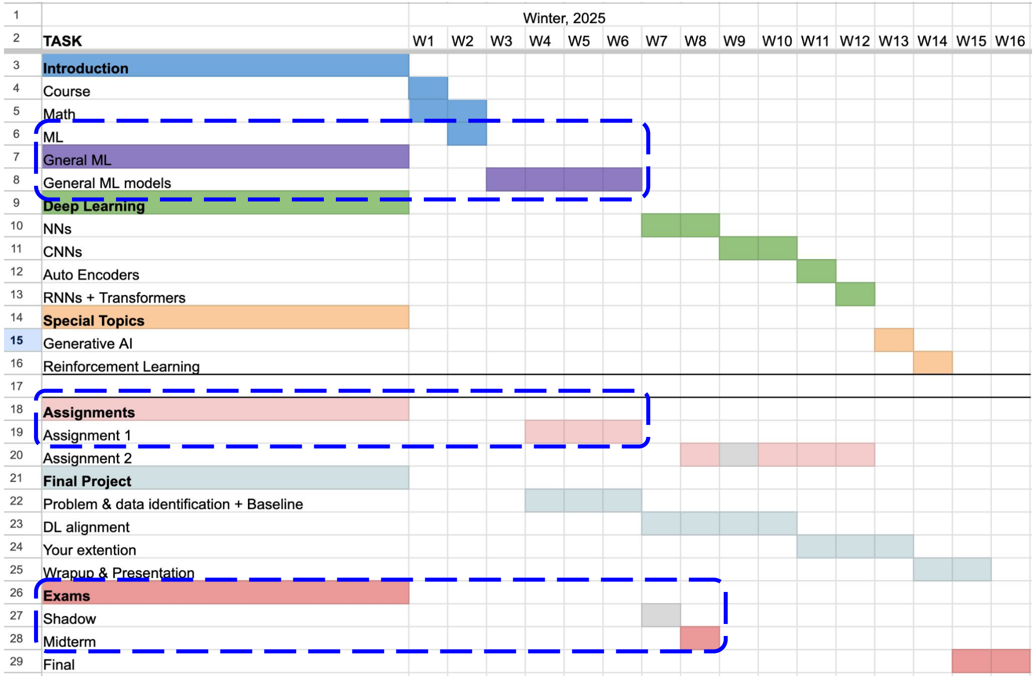
Probabilistic equivalent

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

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



General ML models, Assignment 1 & 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

Unsupervised

- Clustering algorithms
  - **k-means**: Centroid Based
  - k-modes: Mode Based (categorical)
  - **Hierarchical clustering**: Distance connectivity based
  - **GMM**: Distribution based
  - **DBSCAN**: Density Based
- How to choose the optimal number of clusters.

Clustering

- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)

Linear Dimensionality Reduction

General ML models

Model generalization

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

Universal concepts

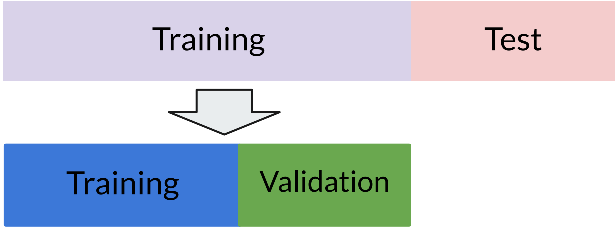
- Overfitting
- Underfitting

Overfitting vs Under fitting

General ML models

Model generalization

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



Data splits

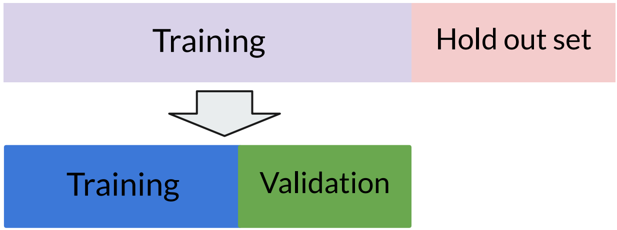
- Overfitting
- Underfitting

Overfitting vs Under fitting

General ML models

Model generalization

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

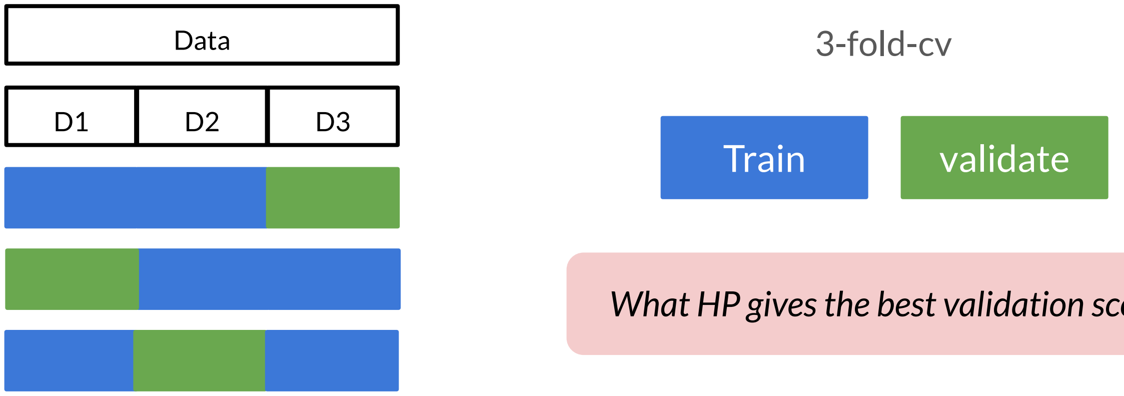


Data splits

- Overfitting
- Underfitting

Overfitting vs Under fitting

K-fold-cross validation



What HP gives the best validation score?

Assignment 1

Vector space

1. [**2x2 points**] For each dimension $d \in [2^1, 2^3, 2^5, ..., 2^{11}]$, sample 100 random points from corresponding vector spaces *(sample code to generate random samples is provided below)*, and

   - Record the $l2$ and the *cosine* distances between all pairs (of points); then
   - Fit two normal/Gaussian distributions, one for each distance metric. Share the mean (μ) and the standard distribution (σ) parameters of each distribution that you have learned.
   - Plot these normal/Gaussian distributions using your preferred visualization package(s).

   normal/Gaussian distribution: $p(x) \sim \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

   Sample code (to generate $n = 100$ random samples from a $d = 4$ dimensional vector space):

```
import numpy as np
d, n = 4, 100
sample_data = np.random.randn(n, d)
```

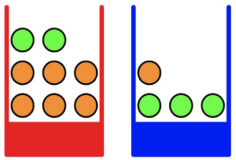


Figure 1: A read and a blue box containing a set of orange and green balls.

Probability & distribution

2. [**2x2 points**]

   There are some orange and green balls in a red and a blue box as illustrated in Figure 1. Someone (blinded by tying a piece of cloth around his/her eyes) is asked to pick up a ball twice (with replacement, i.e. the ball is placed back in the same box from where it was picked up).

   (a) What is the **log-probability** of the first ball to be **orange** and second ball to be **green**?

   (b) What is is **probability** that both the time the ball came from the **red** box?

   Note: We assume, the selection of the red and the blue boxes follow an **uniform** distribution (given that the person is not able to see the color of those boxes).

Assignment 1

Loss function

3. [**2 points**] Given a Linear Regression model, $\Theta = \{\beta_0 = 0.1, \beta_1 = 0.9, \beta_2 = -3.5\}$ (where β_0 is the bias and β_1 and β_2 are the parameters associated with two input features/variables), the regularizer parameter $\lambda = 1.5$ and the following data set (with y being the target variable), estimate the

   - Quadratic error or loss with l2 regularizer as defined below

$$E_\Theta = \frac{1}{2}\sum_{i=0}^{N-1}(y_i - \hat{y_i})^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2$$

Building your own model from scratch

4. [**4 points**] You are asked to fit a **second order/degree** polynomial regression model, $y = \beta_0 + \beta_1 x + \beta_2 x^2$ with parameter $\Theta = \{\beta_0, \beta_1, \beta_2\}$ (where β_0 is the bias of your model) on the following dataset.

   data file:
   https://raw.githubusercontent.com/mdkamrulhasan/data-public/refs/heads/main/miscellaneous/second_degree_polynomial_regression_data.csv

   For the setup below, we ask you to find out the updated version of Θ after two(2) iterations of any gradient descent algorithm (you can use the algorithm that we shared as a part of our linear regression model illustration in the class):

   Setup details:

   - Use the quadratic error/loss function
   - Initialize, $\Theta_0 = \{\beta_0 = 0.0, \beta_1 = 0.0, \beta_2 = 1.0\}$
   - Use learning rate (parameter), $L = 0.001$

   Note: You have to define your error/loss function and also will need to estimate partial derivatives of your loss function.

Assignment 1

Regularization:
model generalization

5. [**3x2 points**] For the given dataset below, we ask you develop, test and compare following models (follow the instructions under the **Detailed specification** section below).

   (a) Linear Regression

   (b) Linear Regression with $l1$ regularizer

   (c) Linear Regession with $l2$ regularizer

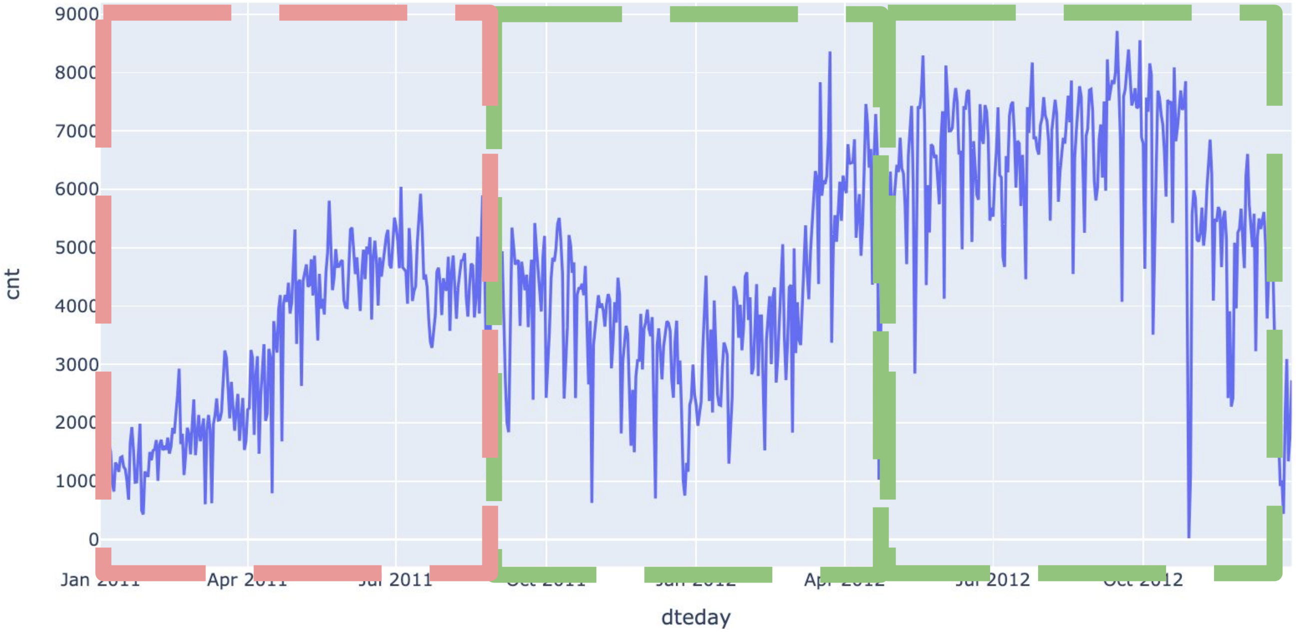
   data file:
   https://raw.githubusercontent.com/mdkamrulhasan/data_mining_kdd/main/data/medical-cost/insurance.csv

   Detailed specification:

   - Use a 50%-50% test/train setup.
   - Use Mean Squared Error (MSE) as your evaluation metric.
   - Visualize (using Bar charts) the model parameter **absolute** values (covert any negative values to positives before plotting).
   - You can use python packages such as **sklearn** for your solution.

Shadow test followed by Mid-term!

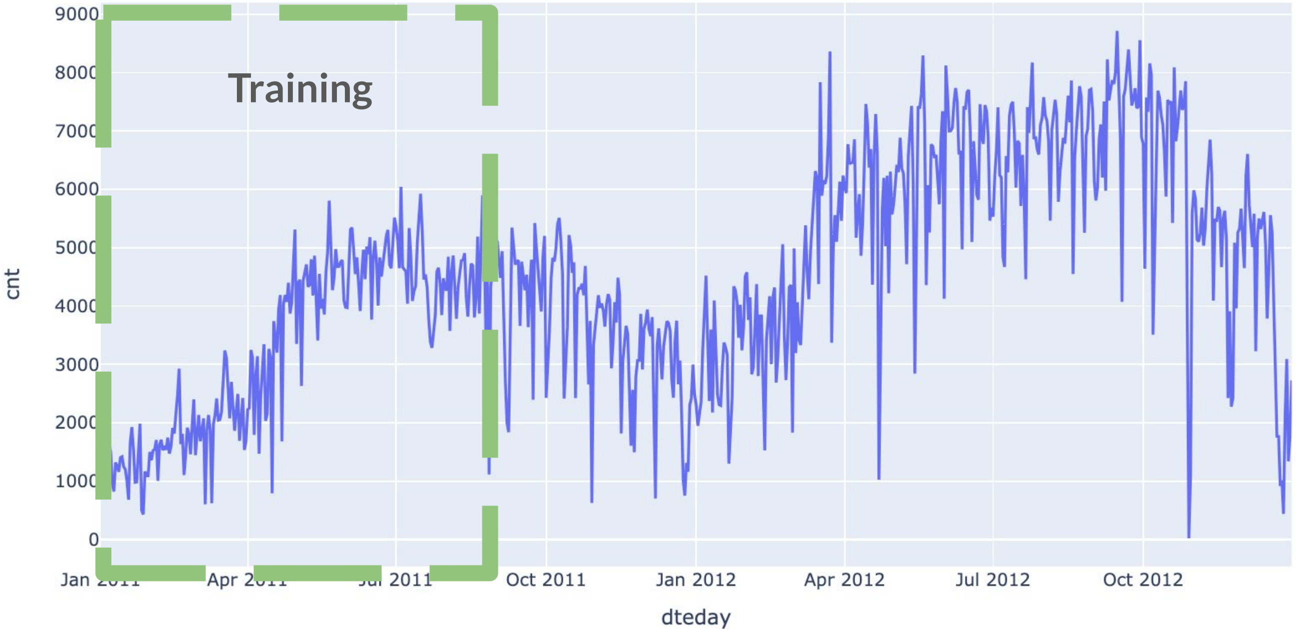
Sequential cross validation



Invalid configuration:

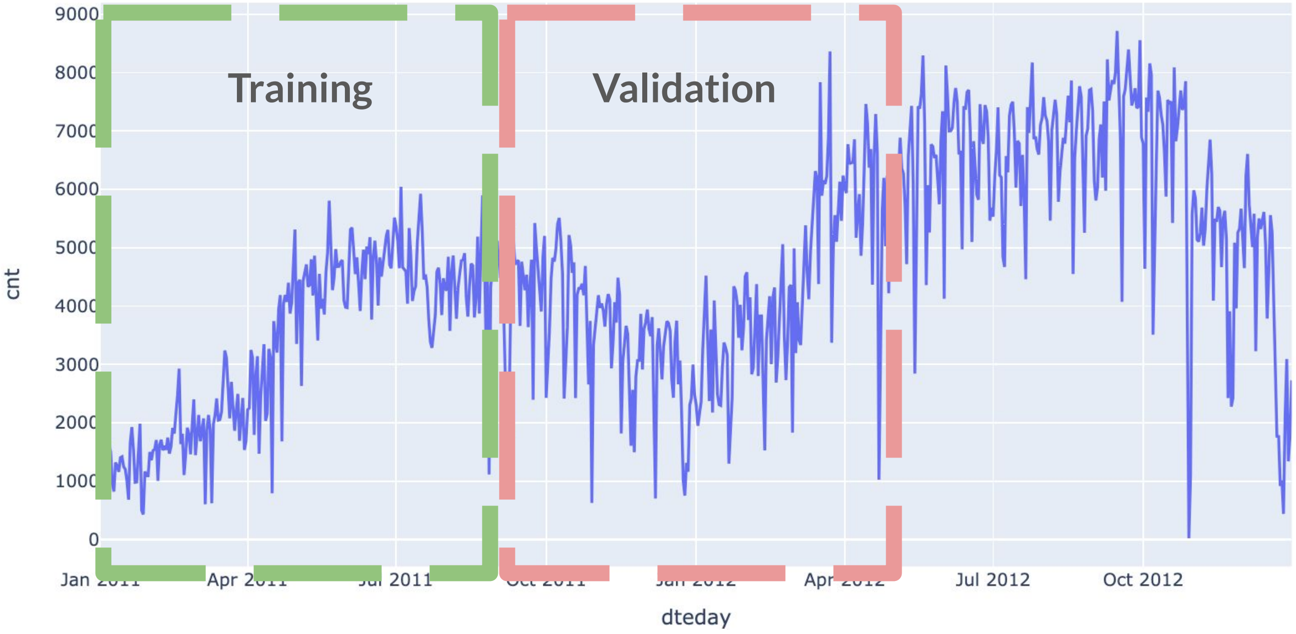
Training on the last two and validation on the ***first fold****.*

Sequential cross validation



Always follow!

Sequential cross validation



Always follow!

Sequential cross validation

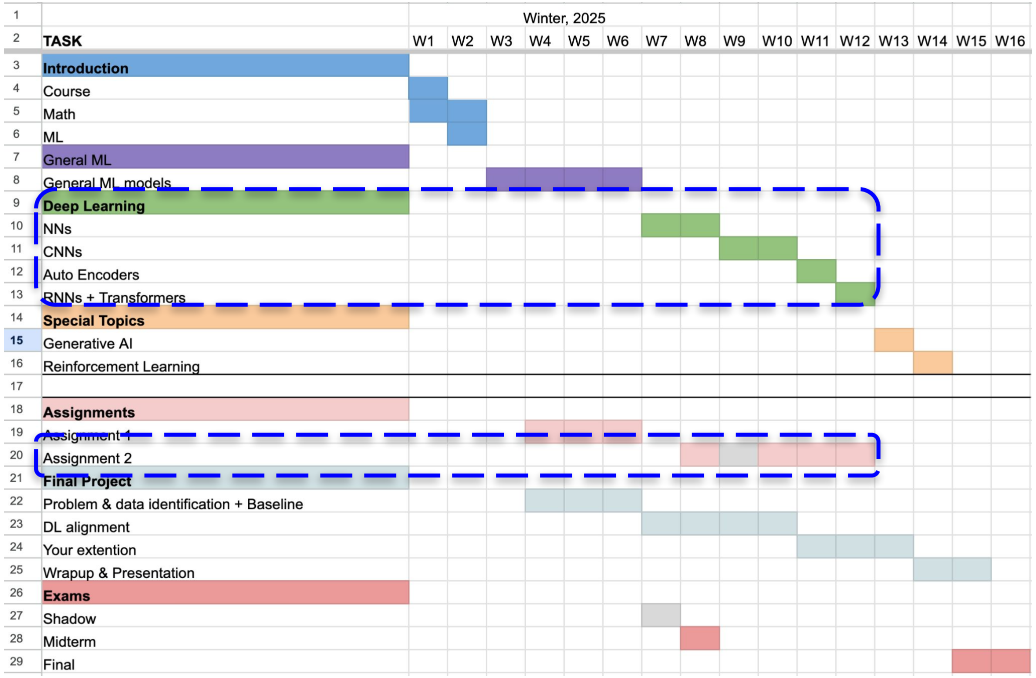
Always follow!

Sequential cross validation

Always follow!

Deep Learning

Assignment 2

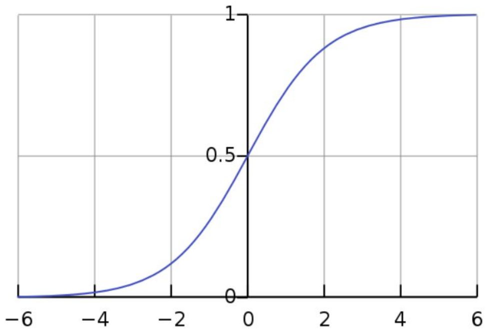


Logistic Regression

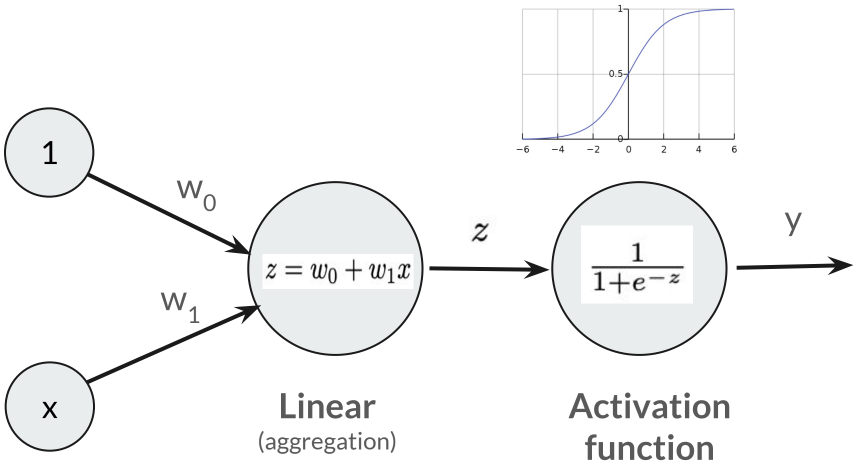
- Probabilistic classifier

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

- Sigmoid function



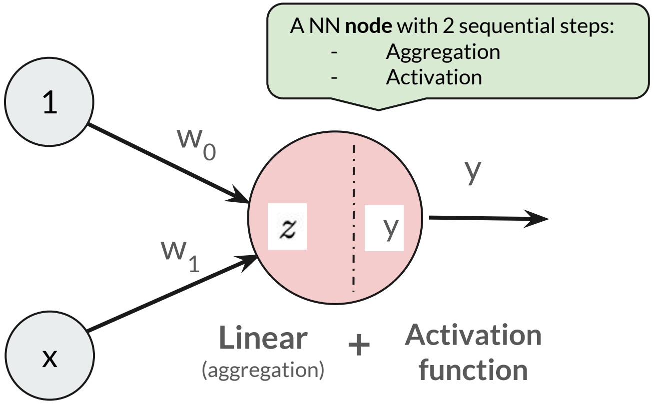
Neural Networks (Node)



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

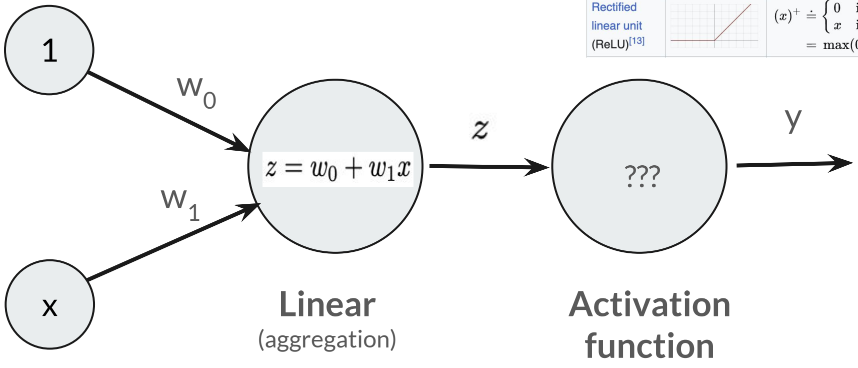
Logistic Regression

Neural Networks (Node)



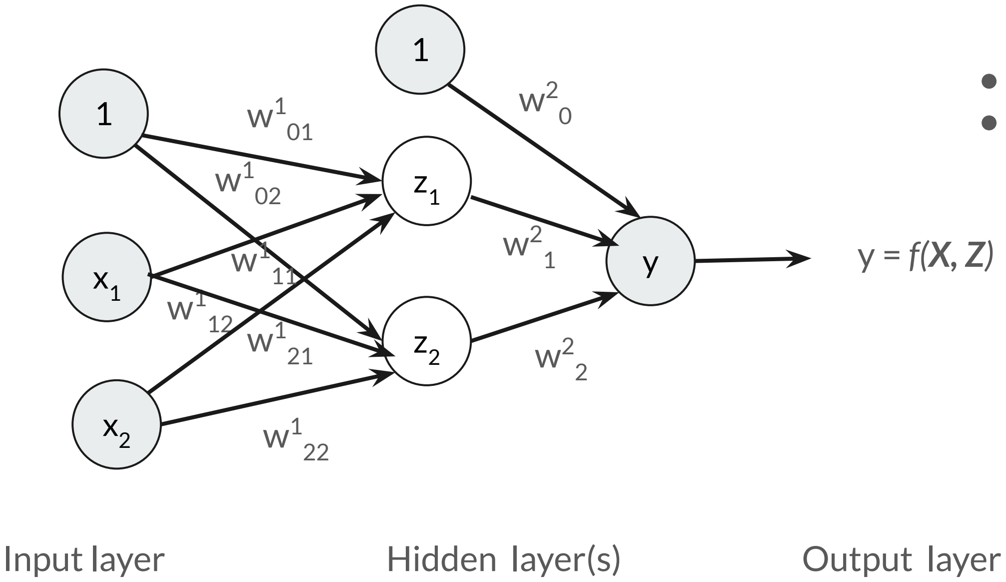
Neural Networks (Node)

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



- Feedforward NN
- Perceptron

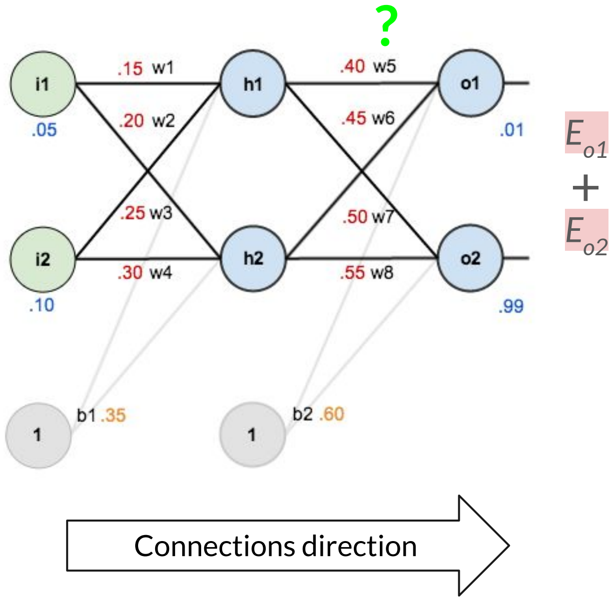
Error Back propagation

Gradient Descent (Error Back Propagation)

The Backwards Pass

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

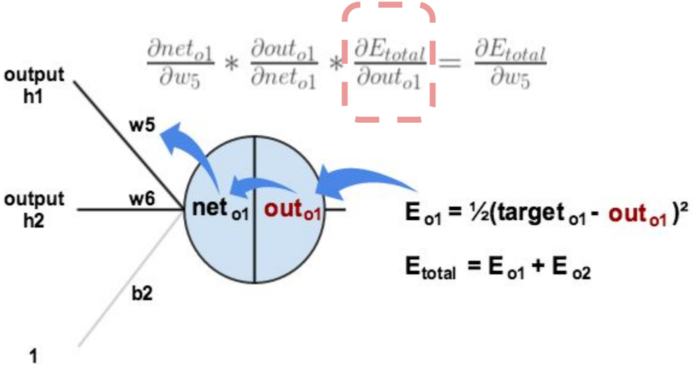
What would be the gradient update for w5?



Adapted from

Gradient Descent (Error Back Propagation)

The Backwards Pass



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

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

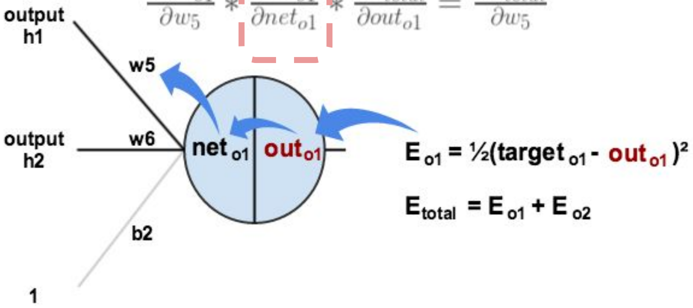
$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

[Adapted from]()

Gradient Descent (Error Back Propagation)

The Backwards Pass

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



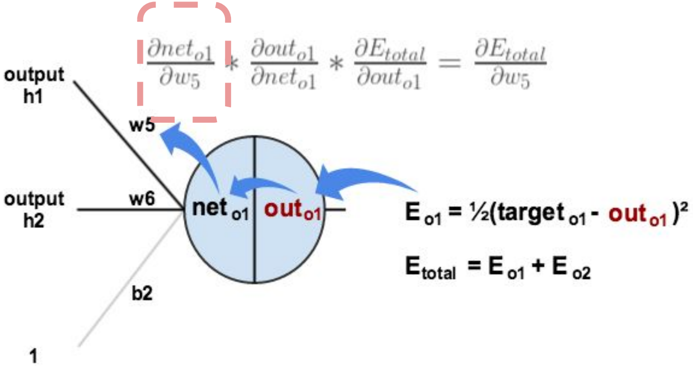
$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

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

[Adapted from]

Gradient Descent (Error Back Propagation)

The Backwards Pass



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

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

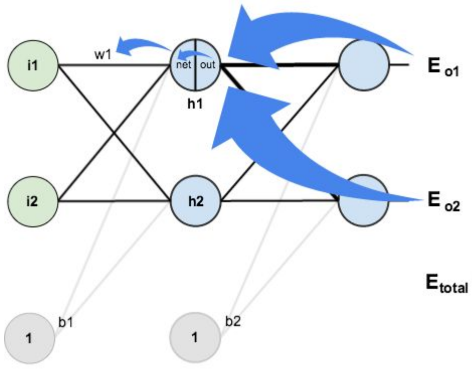
[Adapted from]()

Gradient Descent (Error Back Propagation)

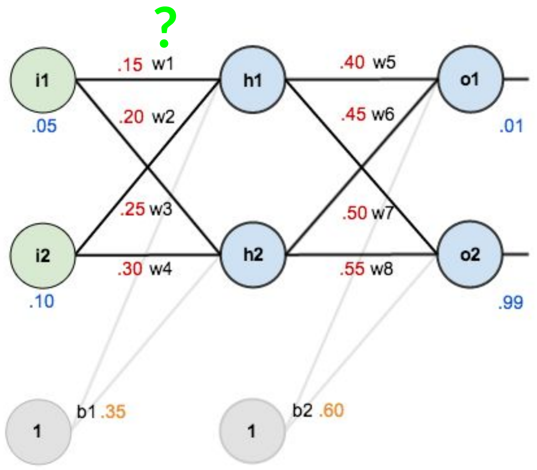
Hidden Layer

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

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



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



E_{o1} + E_{o2}

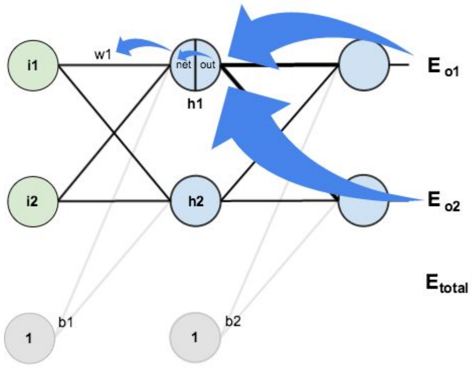
- While changing w5 affects only O1, a change in w1 will change both O1 and O2

Gradient Descent (Error Back Propagation)

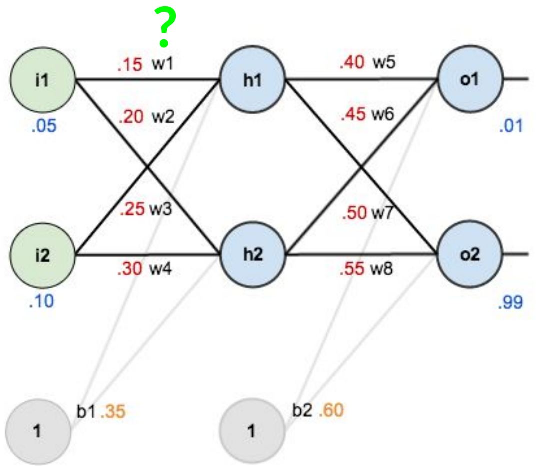
Hidden Layer

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

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



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



E_{o1} + E_{o2}

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

Convolutional NNs

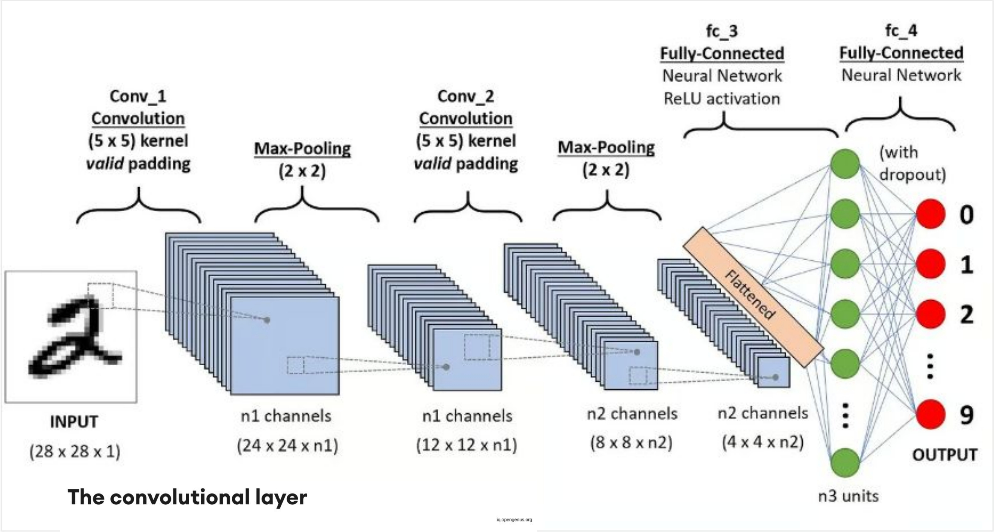
- State of the Art for CV and some other problems
- Filters/Convolutional Kernels

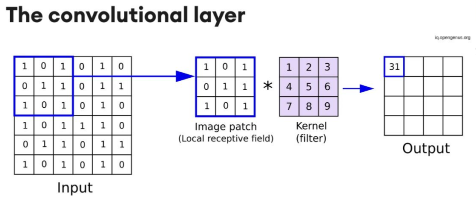
Examples:

- Alexnet
- VGG
- ResNet
- GoogLeNet
- ..

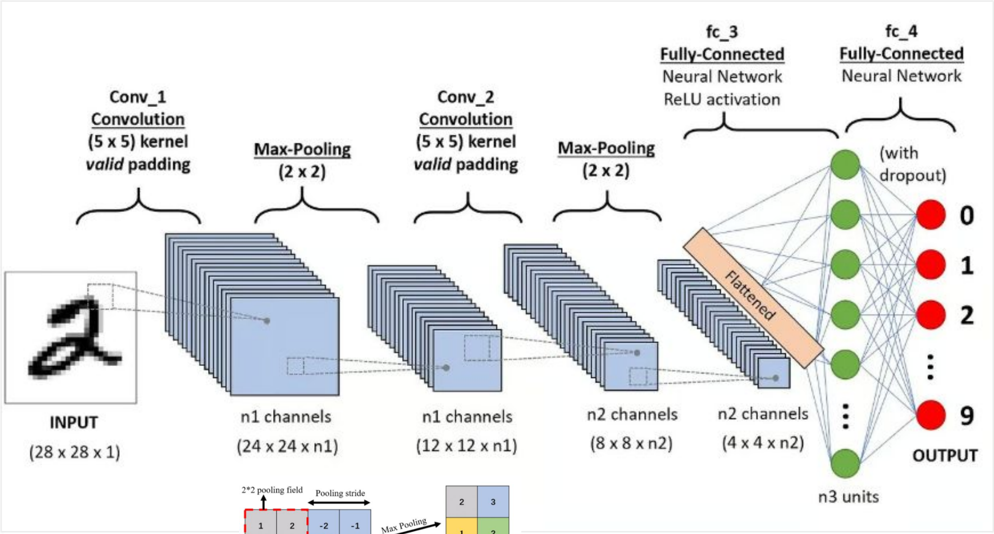
ref

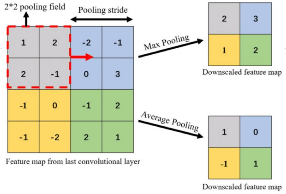
Convolutional NNs





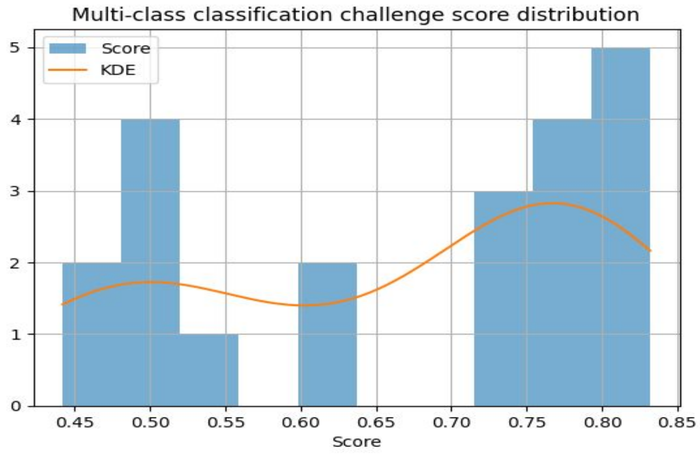
Convolutional NNs



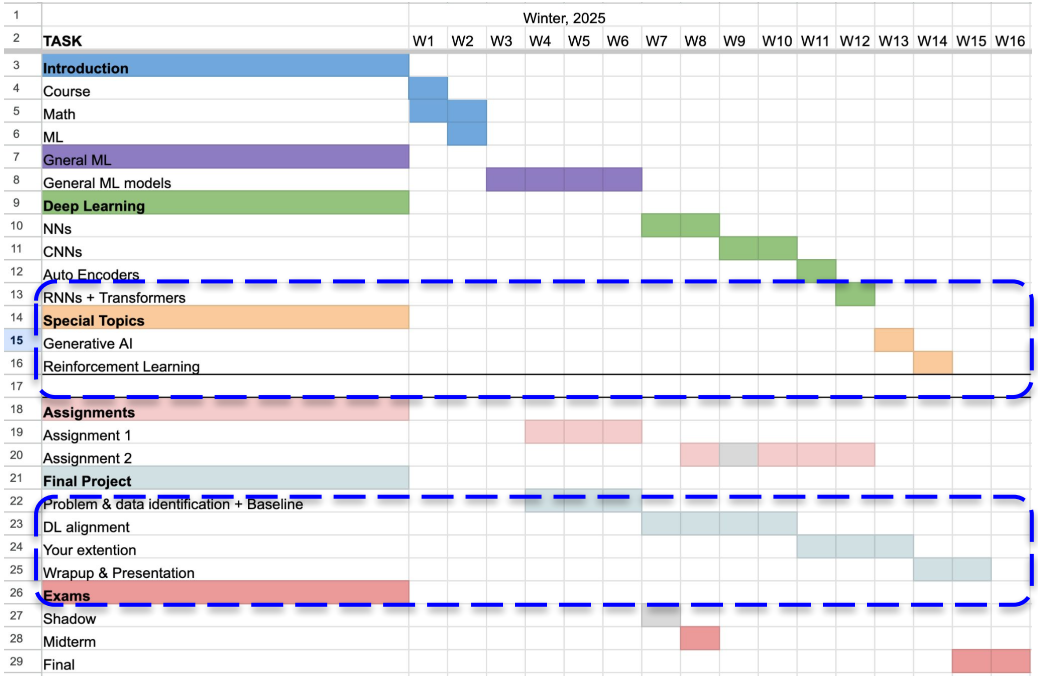


Assignment 1 (classification challenge)

- Deep learning models doubled the performance to general ML models



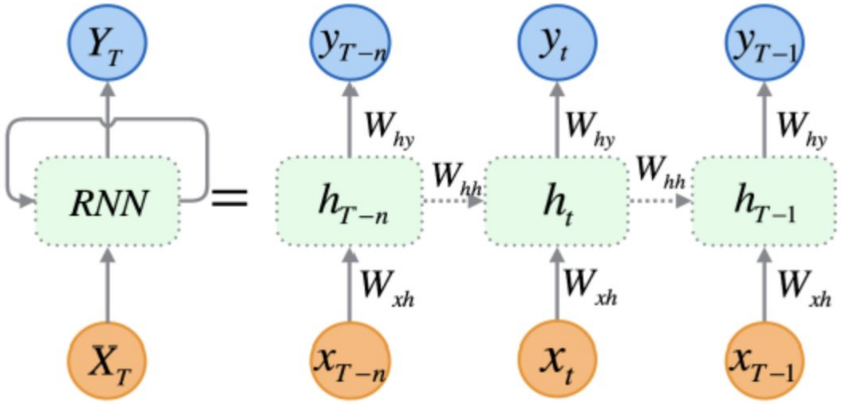
DL continuation + Project



Recurrent Neural Networks

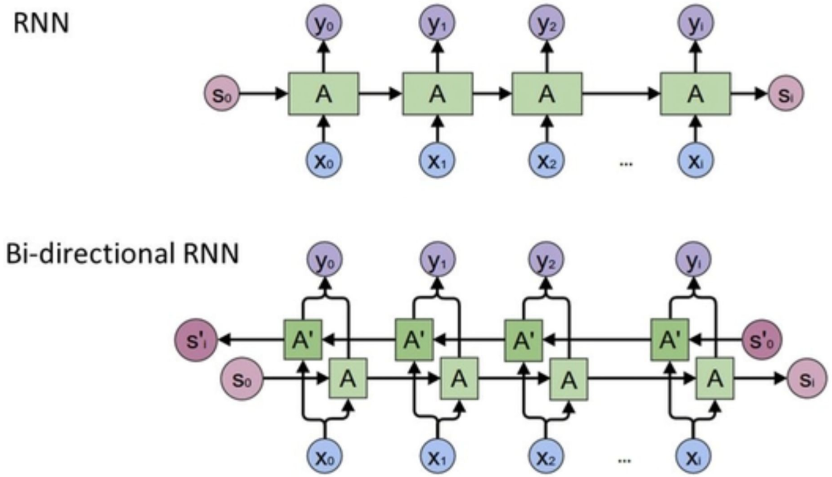
Examples:

- LSTMs
- GRU



ref

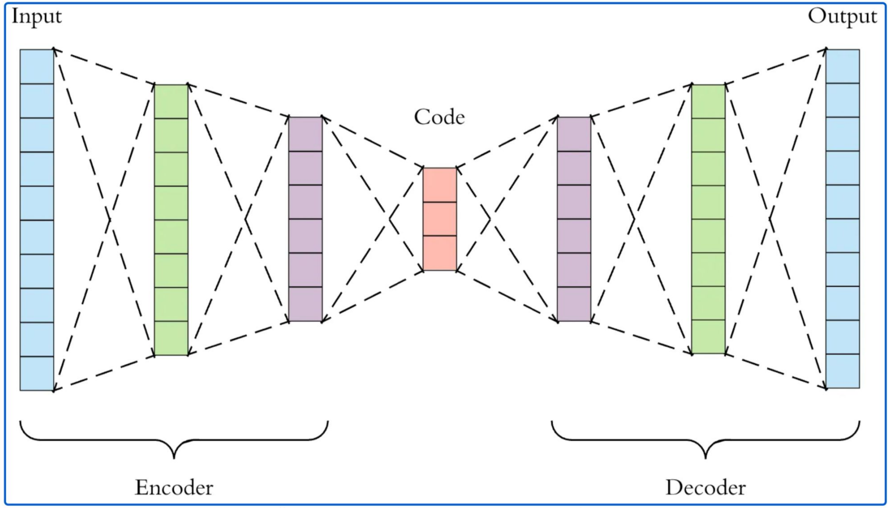
RNNs


RNN
y0
y1
y2
yi
s0
A
A
A
A
si
x0
x1
x2
...
xi
Bi-directional RNN
y0
y1
y2
yi
s'i
A'
A'
A'
A'
s'0
s0
A
A
A
A
si
x0
x1
x2
...
xi

Src

Unsupervised learning (nonlinear)

- Auto Encoders
- Restricted Boltzmann Machines (RBMs)



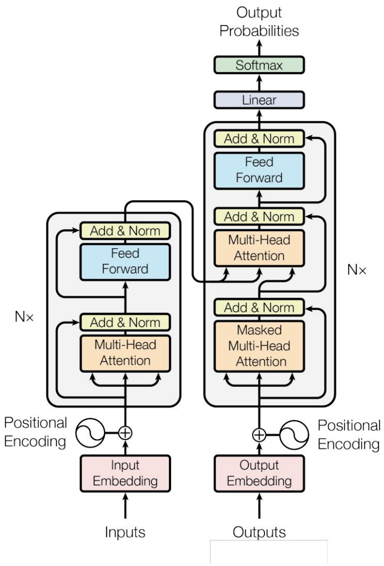
Transformers

Examples:

- Encoder decoder pair
- GPT
- BERT

BERT

Encoder



GPT

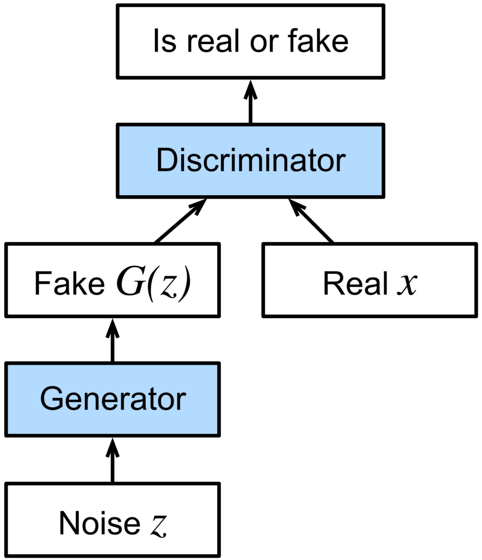
Decoder

ref

Generative AI

Examples:

- Generative Adversarial Networks (GANs)

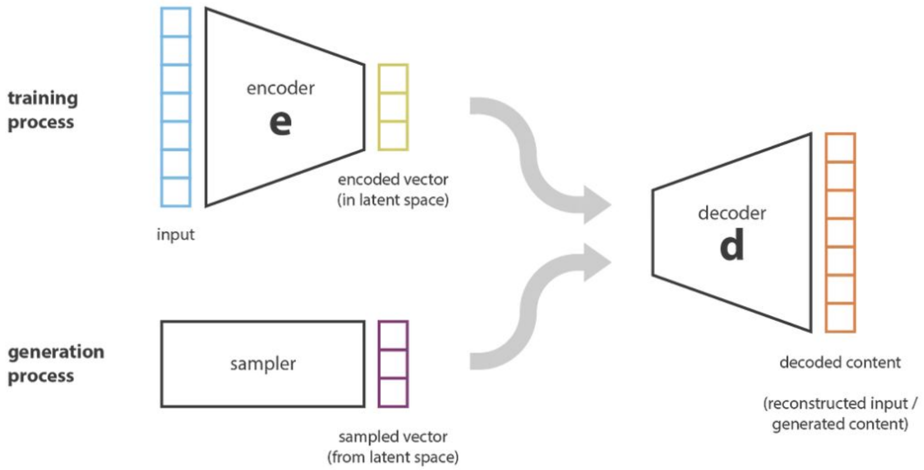


ref

Generative AI

Examples:

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



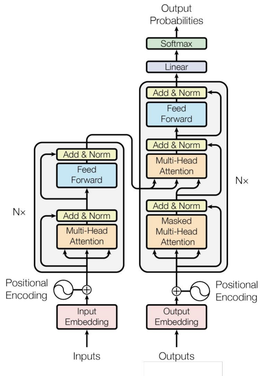
ref

Foundational models

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

BERT

Encoder



Machine translation

GPT

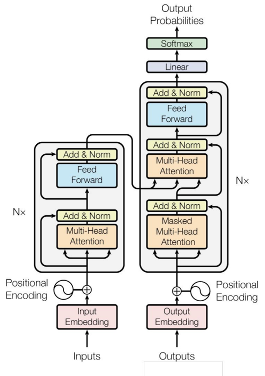
Decoder

Foundational models

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

BERT

Encoder



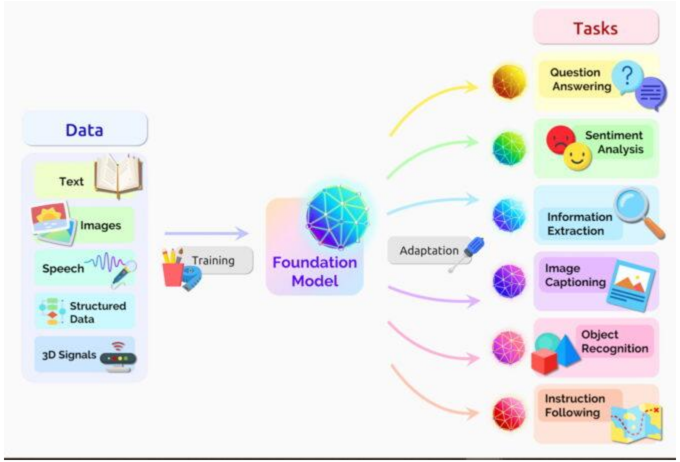
GPT

Decoder

Document summarization

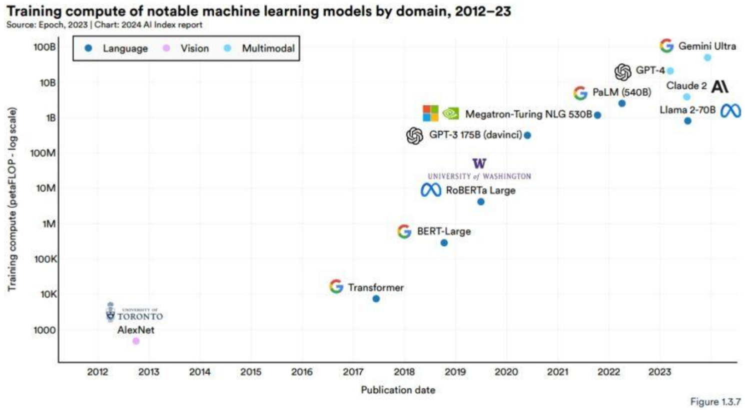
Foundational models

- 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