Departamento de Eletrónica, Telecomunicações e Informática

Complements of Machine Learning

Lecture 1 : Overview of ML course

Petia Georgieva
(petia@ua.pt)

universidade de aveiro

PROGRAM

Deep Learning Architectures

(continuation of Machine Learning course !!!)

- DenseNet - Medical Imaging & Explainable AI (XAI)
- Residual Networks (ResNet) & Inception Network
- MobileNet & EfficientNet
- YOLO (You Only Look Once) & Region proposal R-CNN
- U-net - Image semantic segmentation
- Autoencoders
- Language Models & Natural Language Processing (NLP)
- Transformers

Evaluation

Lectures & labs: 3 hours per week.

Practical component - 50% of the final grade

Practical component consists of 2 projects, developed in a group of two students.

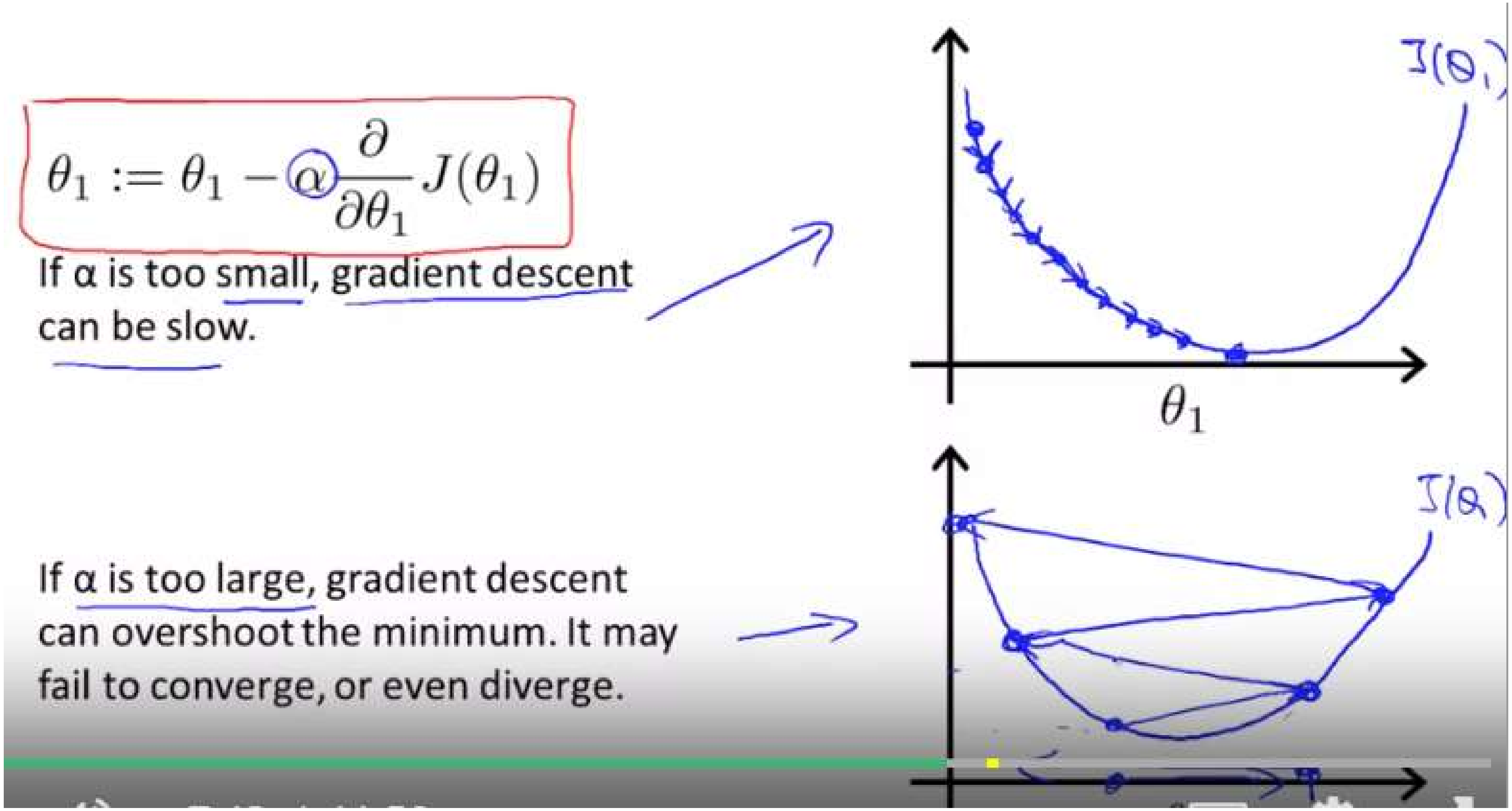
The first project is evaluated based on a submitted report (IEEE format) and a short (10-15 min.) oral presentation.

The second project is evaluated based on a submitted report (IEEE format).

The students are encouraged to use Latex text editor.
Overleaf is a convenient platform for collaborative writing and publishing using Latex (https://www.overleaf.com/) .

Theoretical Component – 50% of the final grade (Final exam).

Q2: Cost function convergence

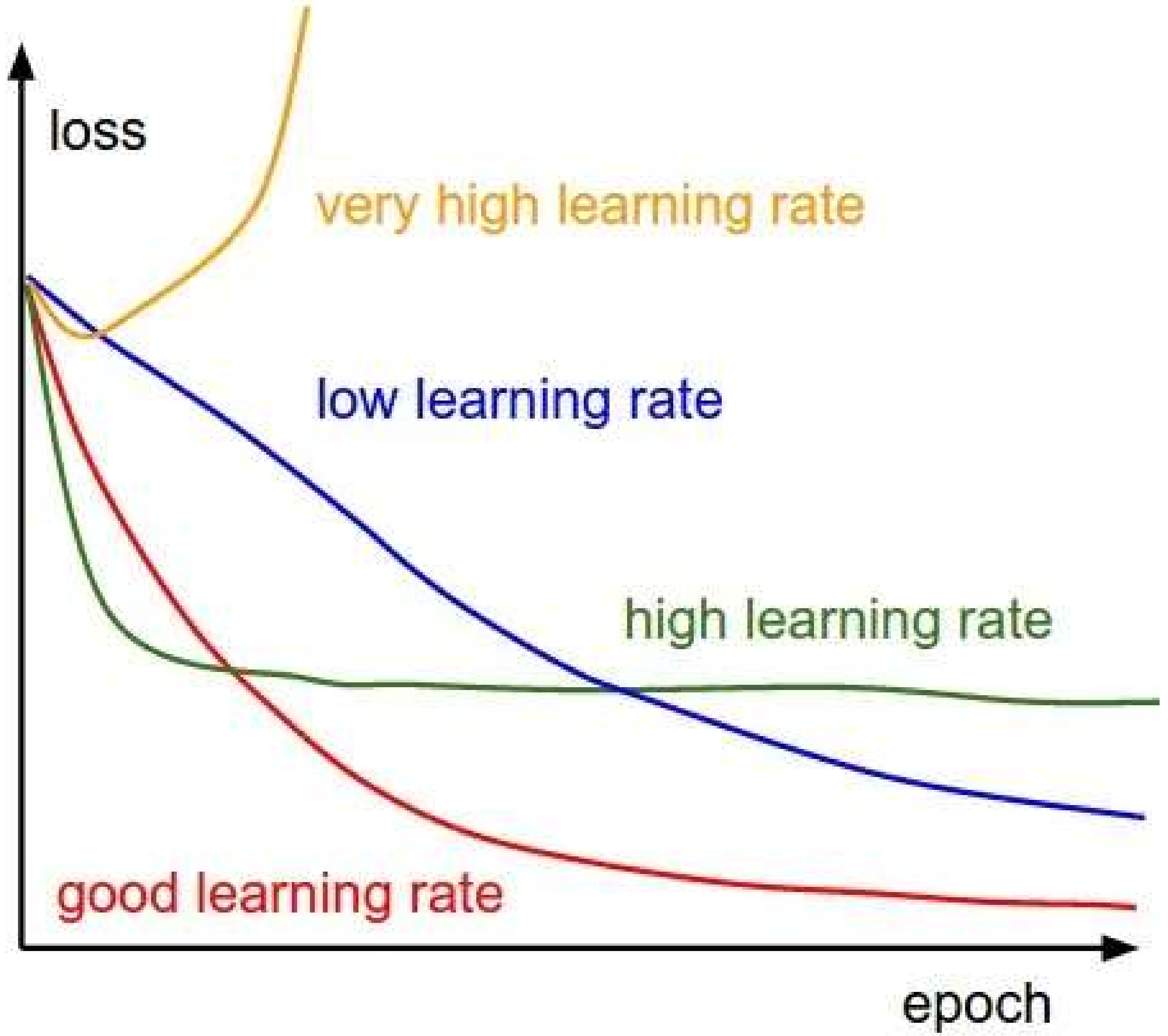


$$\boxed{\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)}$$

If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

Q2: Cost function convergence



Q3A: Standard Notations

x – input vector of features, attributes

y – output vector of labels, ground truth, target

m - number of training examples

n – number of features

$h_\theta(x)$ - model (hypothesis)

θ - vector of model parameters

Training set: data matrix X (m rows, n columns)

	feature x_1	feature x_2	.....	feature x_n	output(label) y
Example 1	$x_1^{(1)}$			$x_n^{(1)}$	$y^{(1)}$
Example 2	$x_1^{(2)}$			$x_n^{(2)}$	$y^{(2)}$
...					
Example i	$x_1^{(i)}$			$x_n^{(i)}$	$y^{(i)}$
...					
...					
Example m	$x_1^{(m)}$			$x_n^{(m)}$	$y^{(m)}$

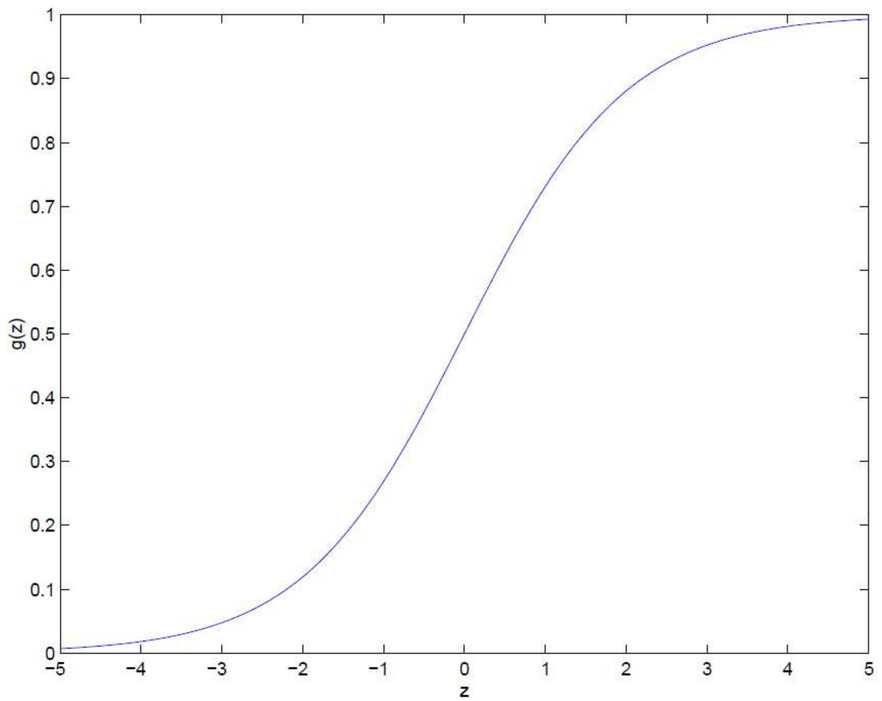
Q3B: Logistic Regression

Given labelled data of *m* examples and *n* features
Labels {0,1} => binary classification
x –vector of features; θ – vector of model parameters;
h(x) – logistic (sigmoid) function model (hypothesis)

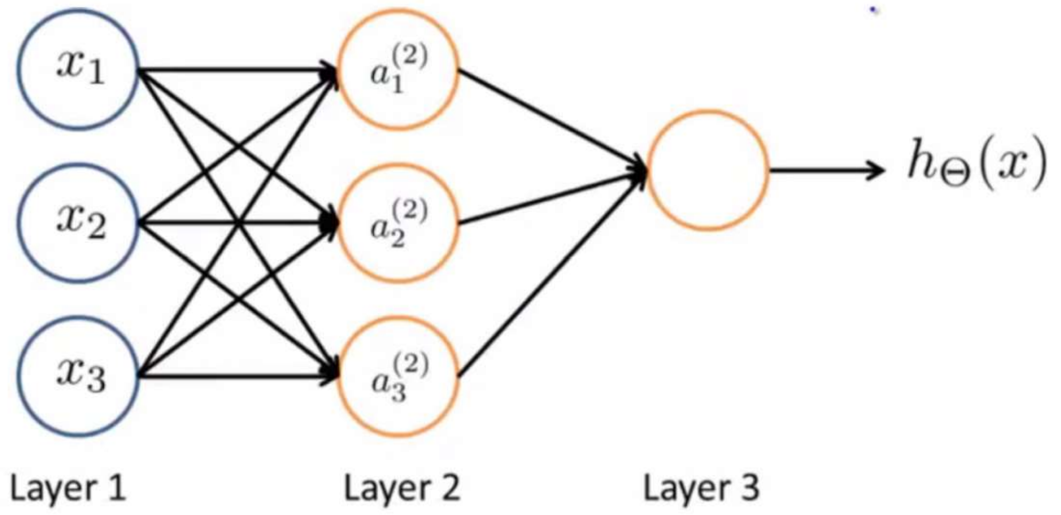
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-z}} = g(\theta^T x) = g(z)$$

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \theta_n x_n$$

Logistic (sigmoid) function



Q4: Neural Network



Input layer hidden layer output layer

$a_i^{(j)}$ = "activation" of unit i in layer j

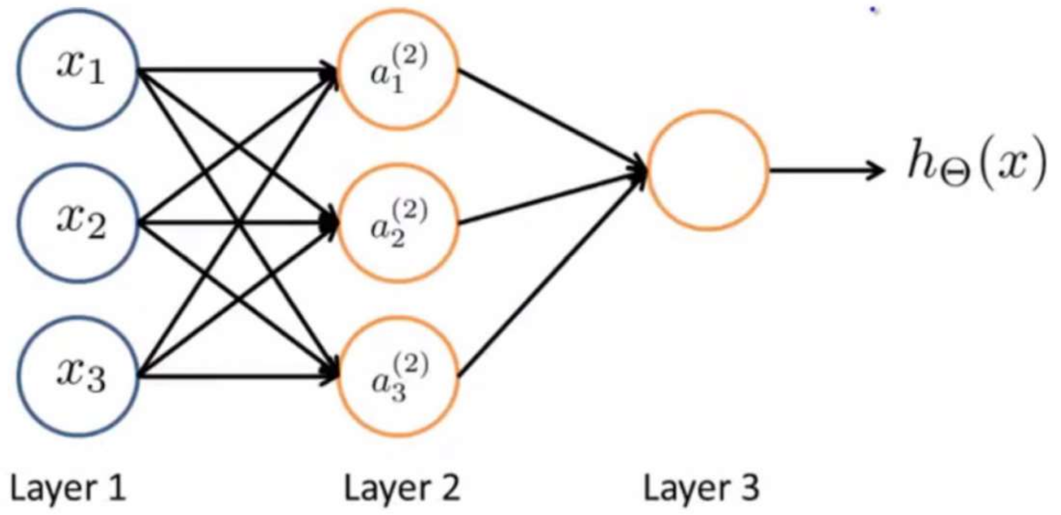
$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

universidade de aveiro

Q4: Neural Network –vectorized implementation

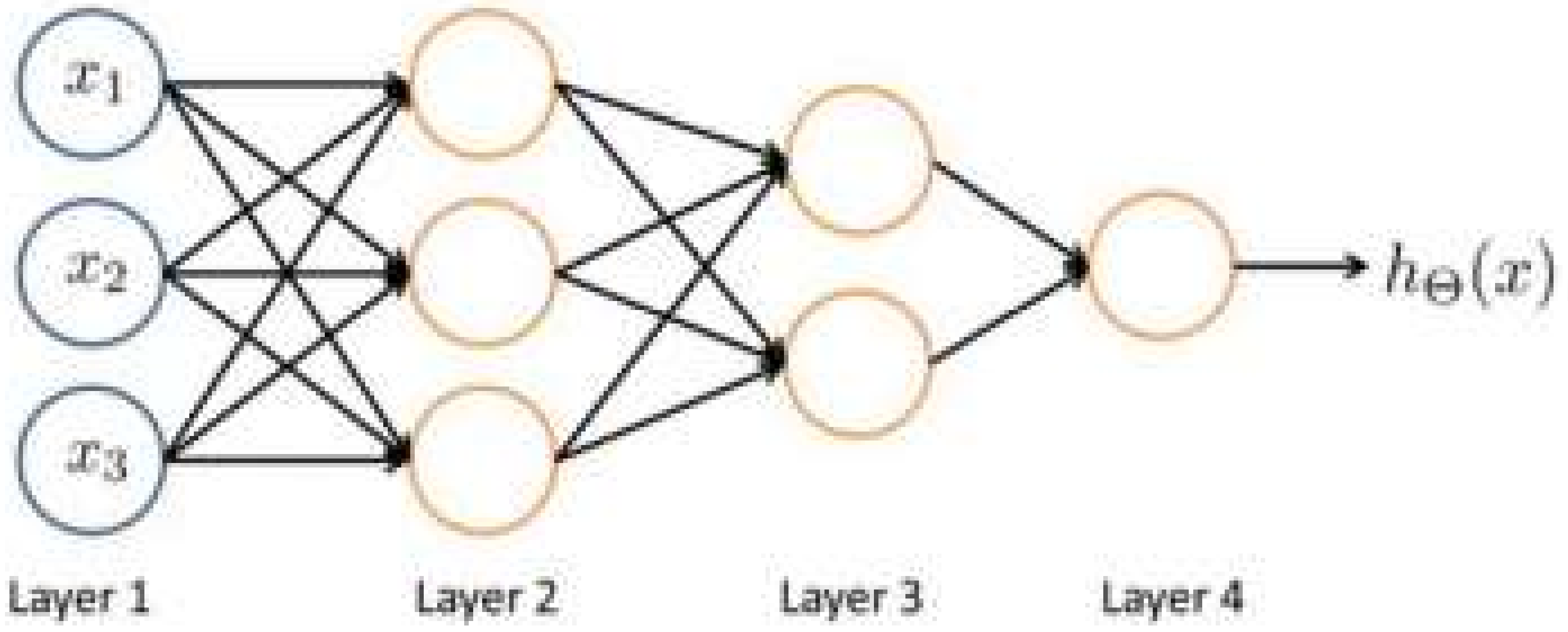


$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \underline{z^{(2)}} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)} x$$
$$a^{(2)} = g(z^{(2)})$$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

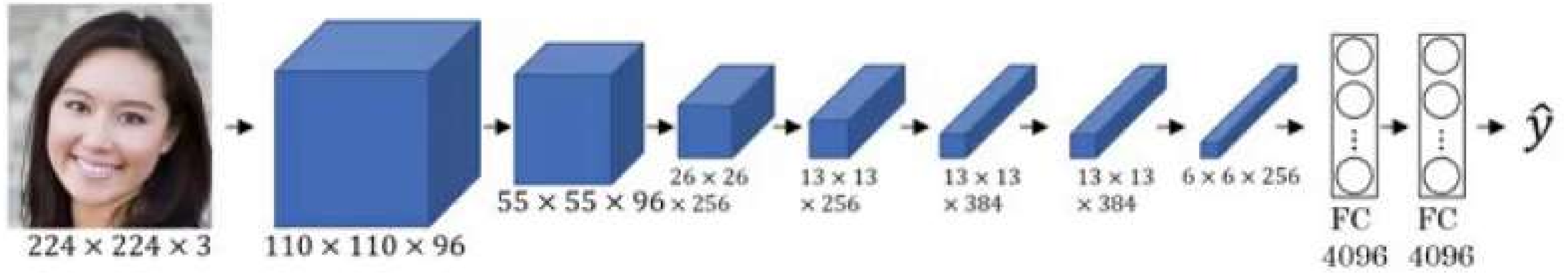
Q4: Neural Network dimension



Q5: Transfer Learning

If you have small training set: freeze earlier layers, train latter layers. The more data you have, the more layers you may train.

Intuition: Hidden layers earlier in the network extract much more general features not specific to the particular task.



Q6: Mean Normalization

Feature Normalization

Note that house sizes are much larger values (about 1000 times) than the number of bedrooms. When features differ by orders of magnitude, first performing feature scaling can make gradient descent converge much more quickly. To make sure features are on a similar scale apply Mean normalization.

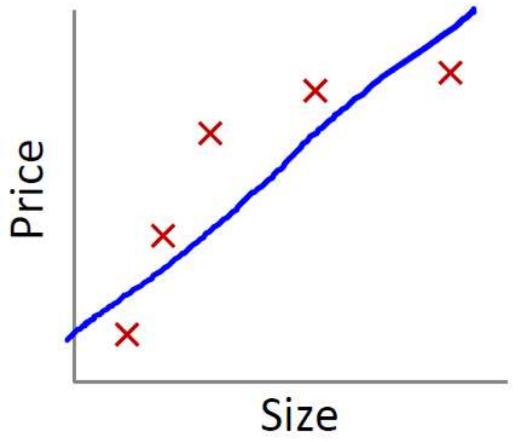
$$x_i = \frac{x_i - \mu_i}{\sigma_i}$$

```python
def featureNormalization(X):
    """
    Take in numpy array of X values and return normalize X values,
    the mean and standard deviation of each feature
    """
    mean=np.mean(X,axis=0)  #axis=0 are the rows, axis=1 are the columns
    std=np.std(X,axis=0)

    X_norm = (X - mean)/std

    return X_norm , mean , std
```

x1	x2	x3
69	4761	78
72	5184	74
94	8836	87
89	7921	96

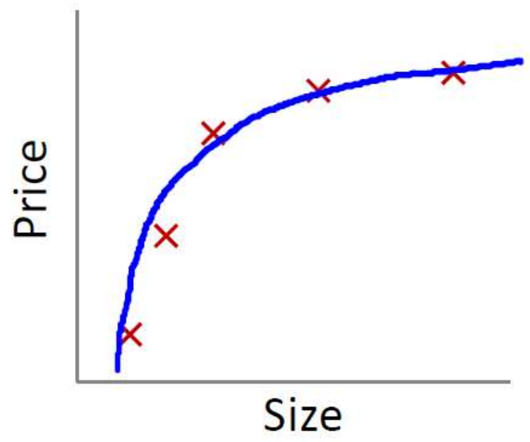
Q10: Overfitting problem

Overfitting: If we have too many features (high order polynomial model), the learned hypothesis may fit the training set very well but fail to generalize to new examples (predict prices on new examples).



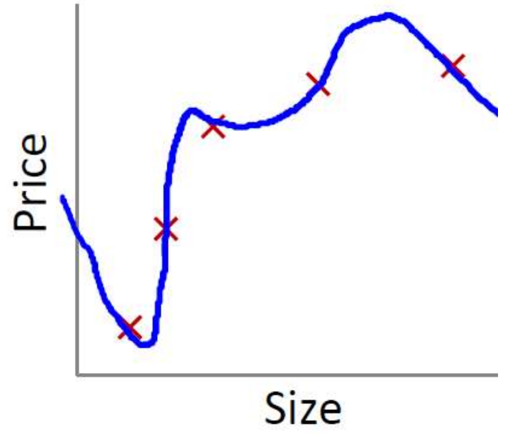
underfit- high bias
(1st order polin. model)

$$h_\theta(x) = \theta_0 + \theta_1 x$$



just right
(3rd order polinom. model)

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$



overfit- high variance
(higher ord. polinom. Model)

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta_{16} x^n$$

Q10: Overfitting problem

Overfitting: If we have too many different features (x1,…x100) the learned model may fit the training data very well but fail to generalize to new examples.

$x_1 =$ size of house

$x_2 =$ no. of bedrooms

$x_3 =$ no. of floors

$x_4 =$ age of house

$x_5 =$ average income in neighborhood

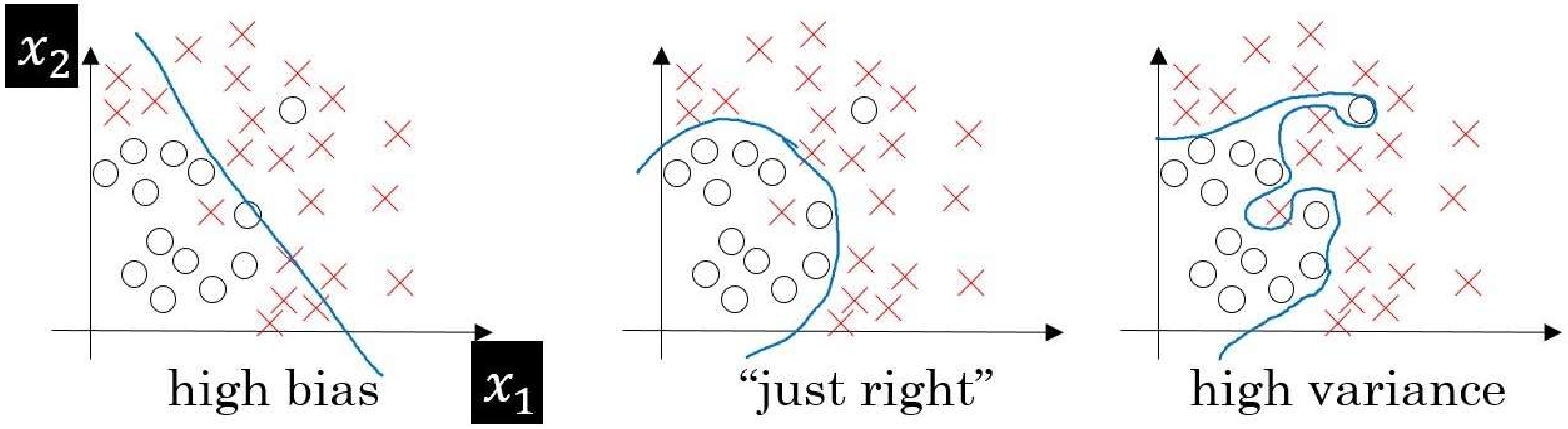
$x_6 =$ kitchen size

$\vdots$

x_{100}

Q10: Bias vs. Variance

An important concept in ML is the bias-variance tradeoff.
Models with **high bias** are not complex enough and **underfit** the training data.
Models with **high variance** are too complex and **overfit** the training data.



underfiting data
(very simple model)

(good model)

overfiting data
(very complex model)

Q10&28: Diagnosing Bias vs. Variance

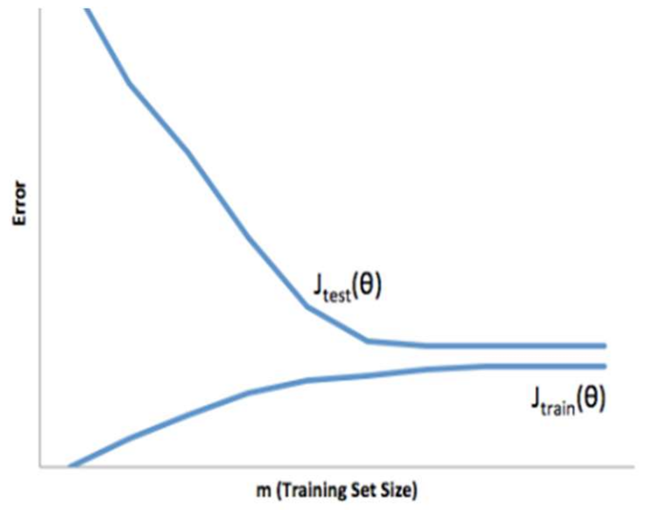
How to diagnose if we have a high bias problem or high variance problem ?

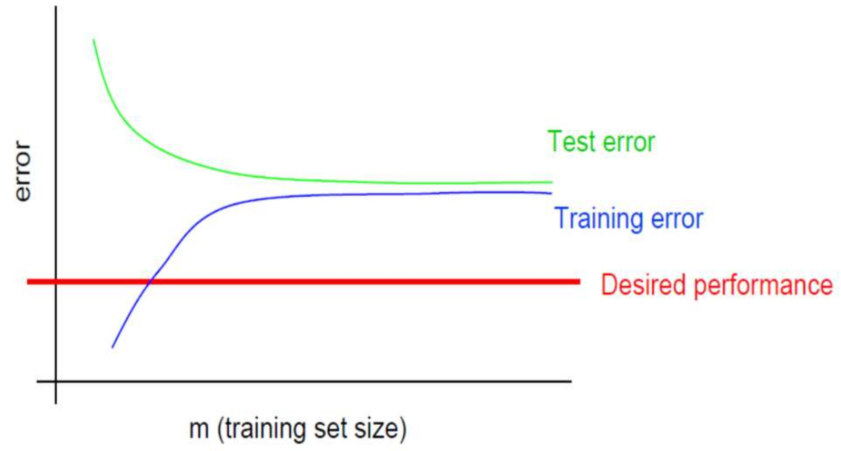
High Bias (underfiting) problem:

Training and Test errors are both high.

High Variance (overfiting) problem:

Training error is low and Test error is much higher.

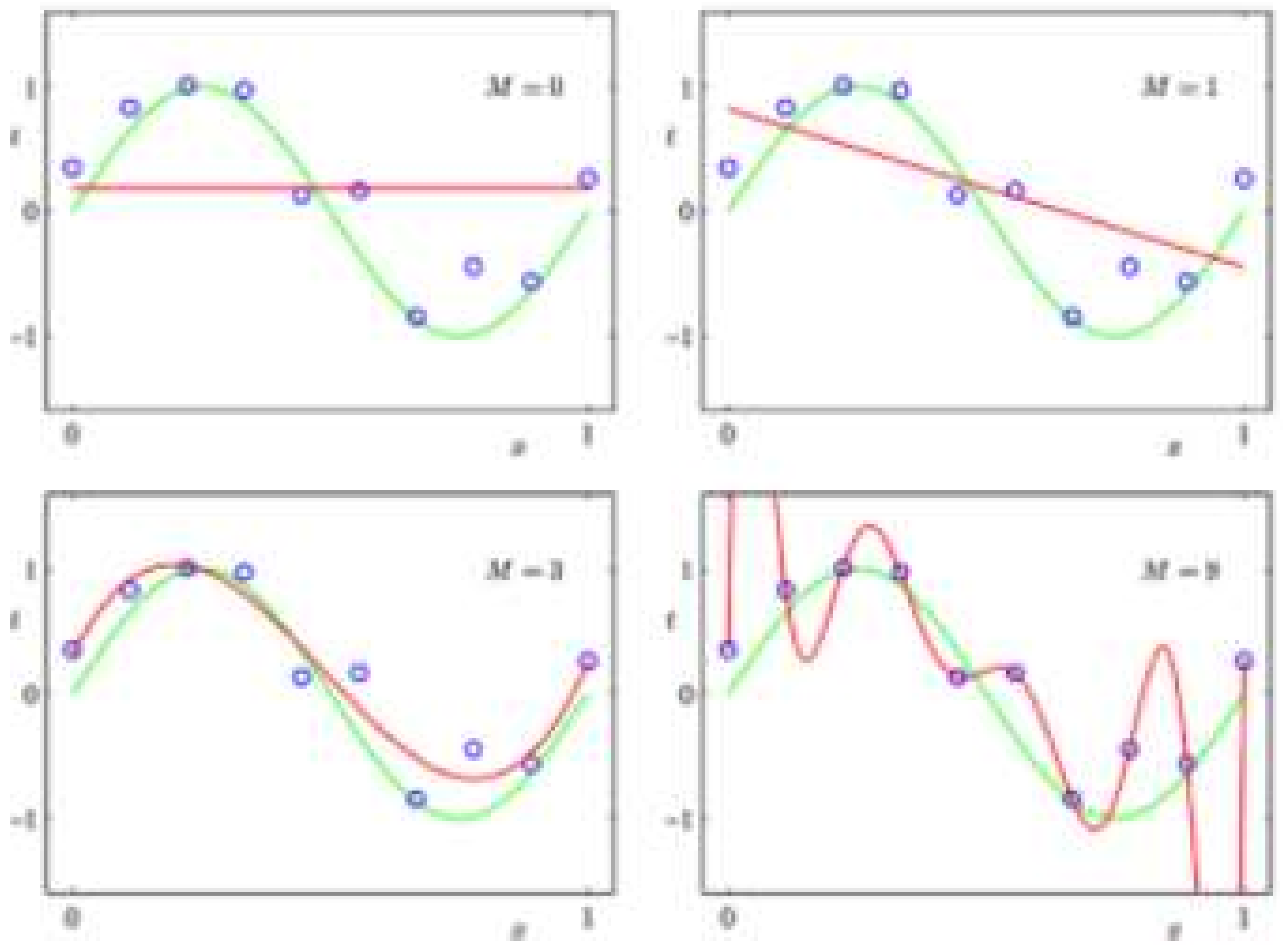




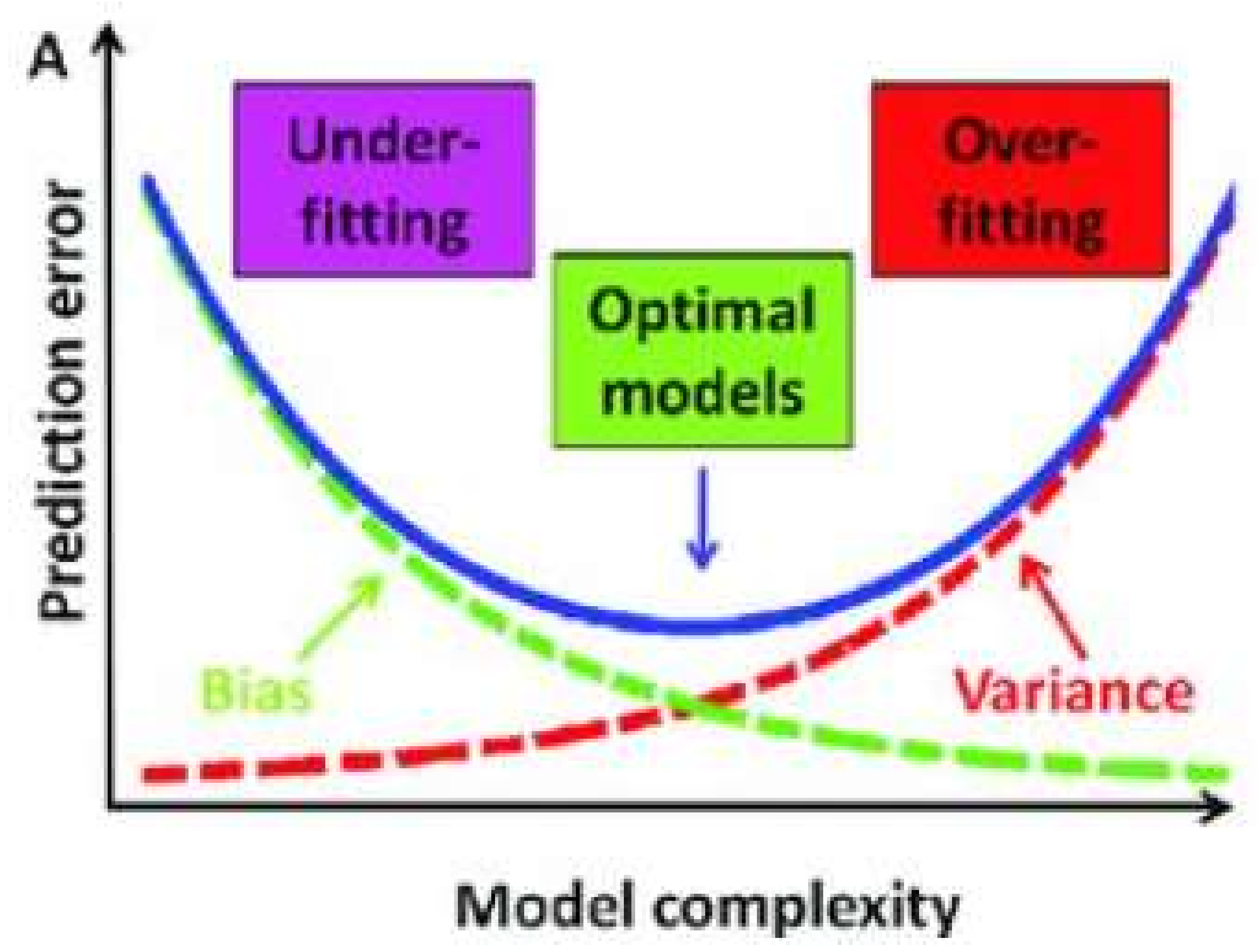
Bias-variance

Polynomials of different orders M to fit the data.

Which one overfitts the data?

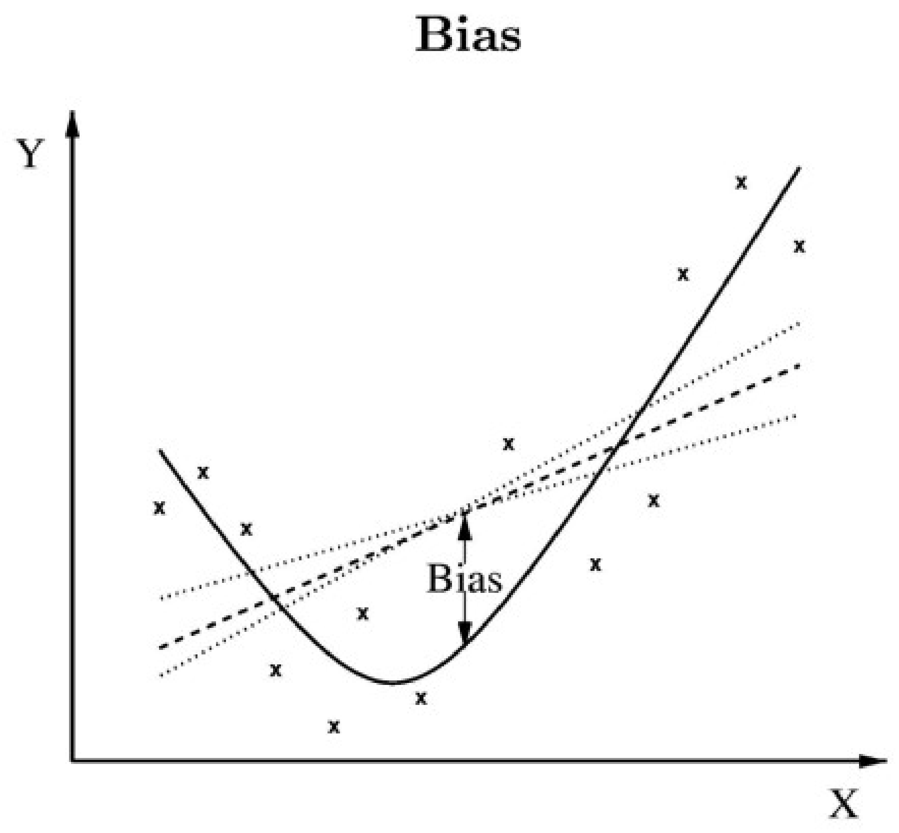


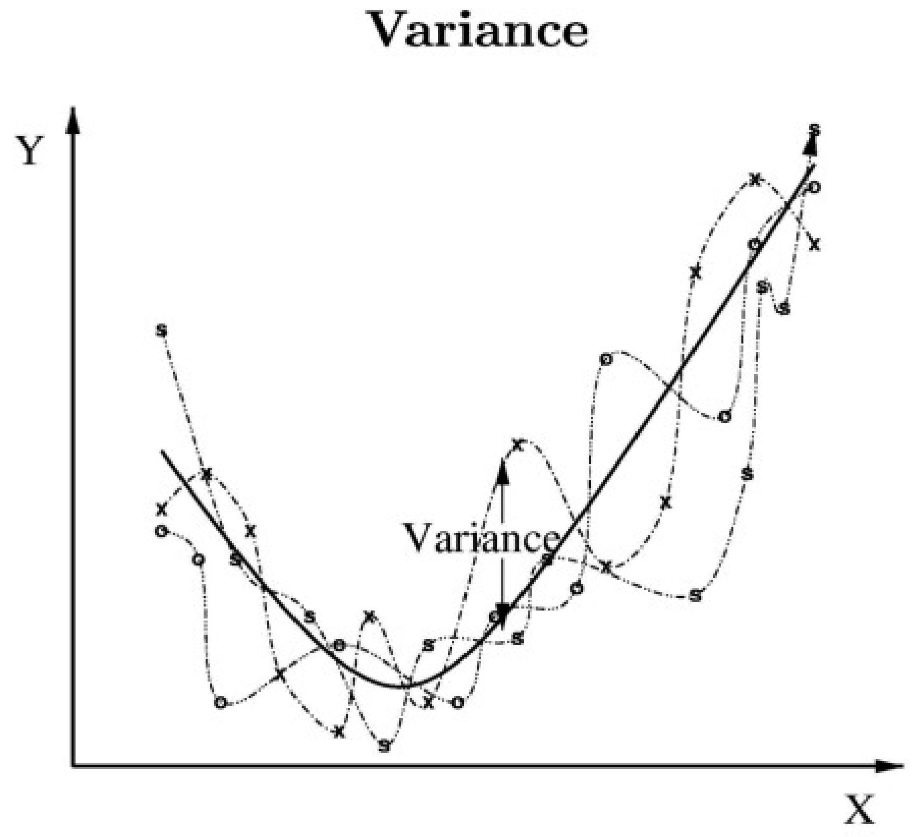
Bias-variance Trade-off



Regularization : tune the model to achieve a good bias-variance trade-off.

Bias-variance Trade-off





Regularization : tune the model to achieve a good bias-variance trade-off.

How to deal with overfitting problem ?

1. Reduce number of features.
— Manually select which features to keep.
— Algorithm to select the best model complexity.

2. Regularization (add extra term in cost function)
Regularization methods shrink model parameters θ towards zero to prevent overfitting by reducing the variance of the model.

2.1 Ridge Regression (L2 norm)
— Reduce magnitude of θ (but never make them =0) => keep all features
— Works well when all features contributes a bit to the output y.

2.2 Lasso Regression (L1 norm)
- May shrink some of the elements of vector θ to become 0.
- Eliminate some of the features => Serve as feature selection

Regularization -Ridge Regression (L2)

Unregularized cost function =>

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Regularized cost function
(add extra regularization term
don’t regularize θ_0

Ridge Regression

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

$$\min_\theta J(\theta)$$

from sklearn.linear_model import Ridge

Regularization: Lasso Regression (L1)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)-\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]+\boxed{\frac{\lambda}{2m}\sum_{j=1}^{n}\left|\theta_j\right|}$$

Ridge Regression shrinks θ towards zero, but never equal to zero => all features are included in the model no matter how small are the coefficients.

Lasso Regression is able to shrink coefficients to exactly zero => reduces the number of features. This makes Lasso Regression useful in cases with high dimension.

Lasso Regression involves absolute values (not differentiable)

from sklearn.linear_model import Lasso

Different Cost/Loss Functions

Training data MSE

- **Linear Regression Cost Function** with L2 Regularization (Ridge Regression)

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Ridge Regression

- **Logistic Regression Cost Function** with L2 Regularization

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- **SVM Cost Function** with L2 Regularization

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

Mean Squared Error (MSE) is NOT the cost function!!!

Q16&18: Hints to improve ML model

Suppose you have learned a data model (hypothesis). However, when you test your hypothesis on a new set of data, you find that it makes unacceptably large errors in its prediction (regression or classification). What should you try next?

-- Get more training examples – fixes high variance

-- Try smaller sets of features – fixes high variance

-- Try getting additional features – fixes high bias

-- Try adding polynomial features - fixes high bias

-- Try decreasing λ – fixes high bias

- Try increasing λ – fixes high variance

Evaluation Metrics for Regression

Goal: obtain reliable estimates of the error (in the validation set) and compare the performance of different regression models.

Mean Squared Error (MSE):

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

where

- $\hat{y}_i$ is the prediction of the model under evaluation for the case i
- y_i the respective true target variable value.

MSE is measured in a unit that is squared of the original variable scale.

Because of the this, Root Mean Squared Error is used as alternative :
RMSE =sqrt (MSE)

Evaluation Metrics for regression

Mean Absolute Error (MAE)

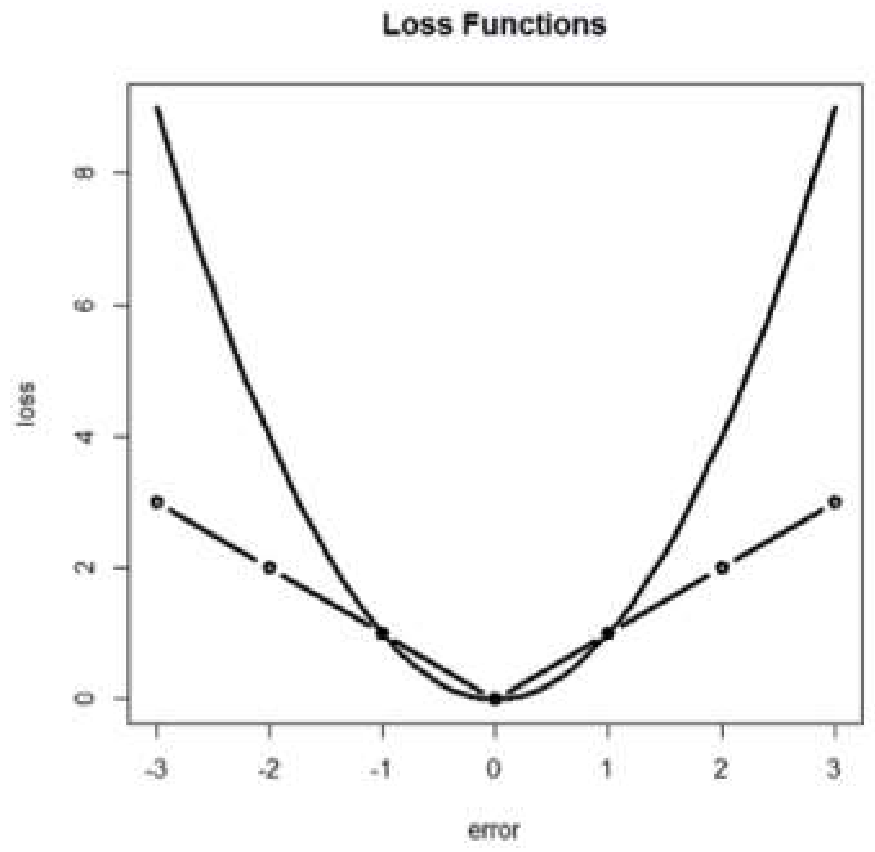
$$MAE = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|$$

where

- $\hat{y}_i$ is the prediction of the model under evaluation for the case i
- y_i the respective true target variable value.

MAE is measured in the same unit as the original variable scale.

MSE & MAE



Measures involving squared errors amplify the large errors.
Therefore MSE is good in areas where large errors are intolerable.

MAE is not as sensitive to large errors.
Gives a better indication of the “typical” error of the model because it treats all errors the same way and is expressed in the same units of the predicted output Y.

Relative Error Metrics

• **Relative Errors** are metrics without units which means that their scores can be compared across different domains.

• They are calculated by comparing the scores of the model under evaluation against the scores of some baseline model.

• The relative score is expected to be a value between 0 and 1.
If the values are near (or even above) 1 it means the model is as bad as the baseline model.
The baseline model is usually chosen as something too naive.

Relative Error Metrics

- The most common baseline model is the constant model consisting of predicting for all test cases the average target (output) value (¯y) calculated with the training data.

Normalized Mean Squared Error (NMSE)

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

Normalized Mean Absolute Error (NMAE)

$$NMAE = \frac{\sum_{i=1}^{N}|\hat{y}_i - y_i|}{\sum_{i=1}^{N}|\bar{y} - y_i|}$$

NMSE and NMAE usually vary between 0 an 1.
The relative measures (e.g. NMSE,NMAE) have the advantage of independence of the application domain.

Correlation Coefficient

Correlation is a commonly used method to examine the relationship between quantitative variables. The most commonly used statistic is the **linear correlation coefficient** (r) which is also known as the **Pearson product moment correlation coefficient** in honor of its developer, Karl Pearson.

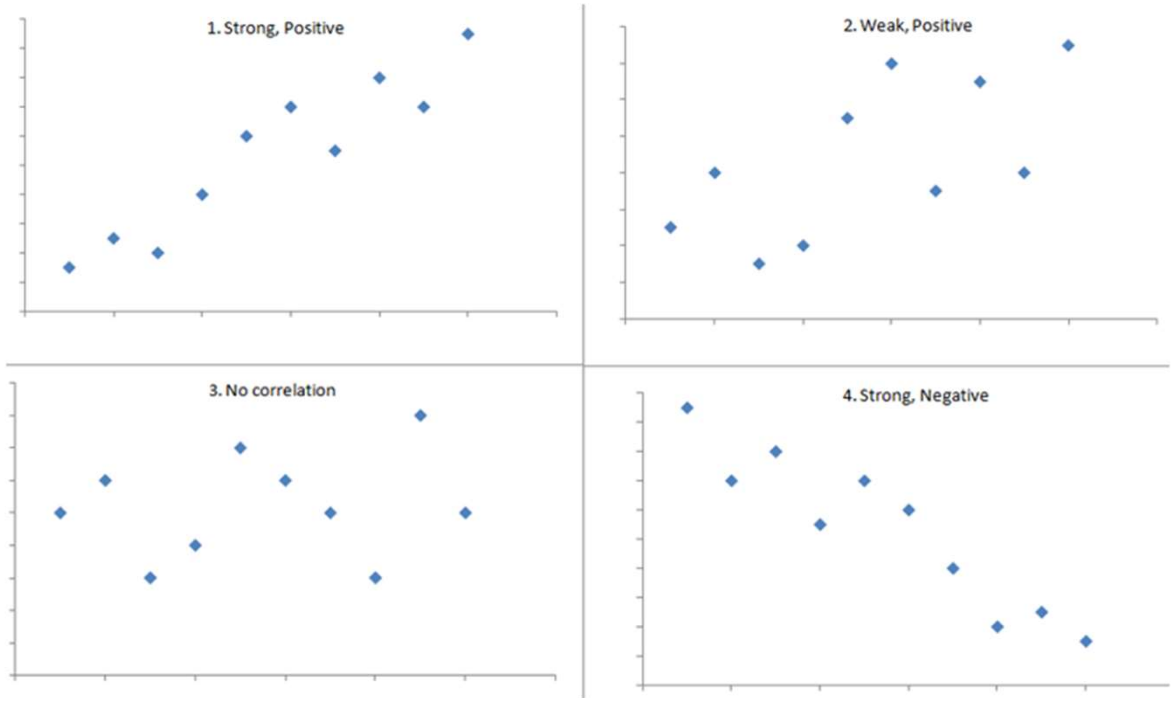
Correlation coefficient measures the strength of the relationship between two variables x and y.

It varies between -1 and 1.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Correlation Coefficient

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

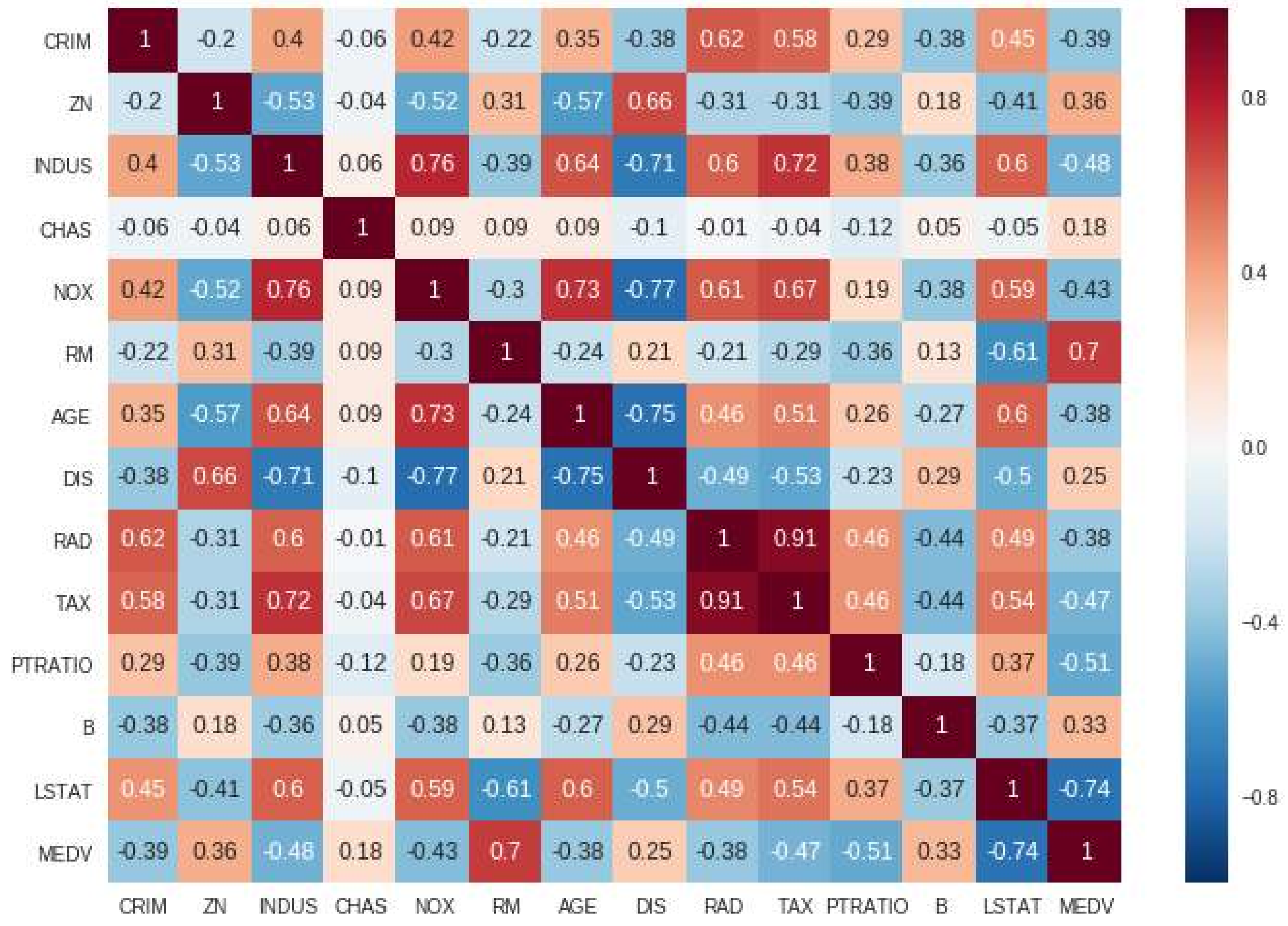


.

Strong linear relationship	$r > 0.9$
Medium linear relationship	$0.7 < r \leq 0.9$
Weak linear relationship	$0.5 < r \leq 0.7$
No or doubtful linear relationship	$0 < r \leq 0.5$

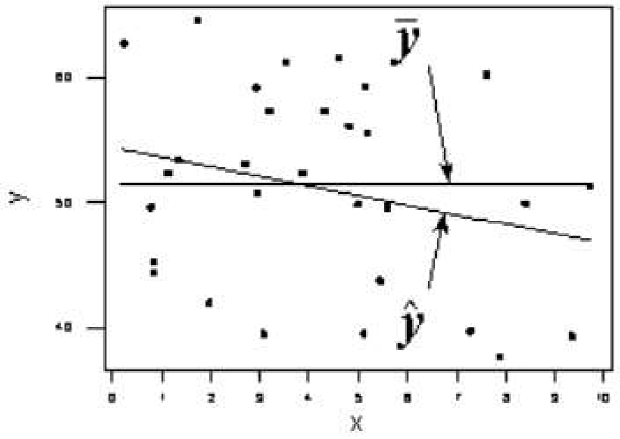
Correlation Matrix

Correlation Matrix meassures the correlation between the features and the predicted variable and also the correlation between the features
The goal is to detect the multicollinearity problem, that is if the features are highly correlated, this causes variances to be large, highly dependent on the training data.
Lab: Boston Housing dataset: 13 features, predict MEDV (median value)



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	-0.2	0.4	-0.06	0.42	-0.22	0.35	-0.38	0.62	0.58	0.29	-0.38	0.45	-0.39
ZN	-0.2	1	-0.53	-0.04	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
INDUS	0.4	-0.53	1	0.06	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
CHAS	-0.06	-0.04	0.06	1	0.09	0.09	0.09	-0.1	-0.01	-0.04	-0.12	0.05	-0.05	0.18
NOX	0.42	-0.52	0.76	0.09	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
RM	-0.22	0.31	-0.39	0.09	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
AGE	0.35	-0.57	0.64	0.09	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
DIS	-0.38	0.66	-0.71	-0.1	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
RAD	0.62	-0.31	0.6	-0.01	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
TAX	0.58	-0.31	0.72	-0.04	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.38	0.18	-0.36	0.05	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
LSTAT	0.45	-0.41	0.6	-0.05	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
MEDV	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

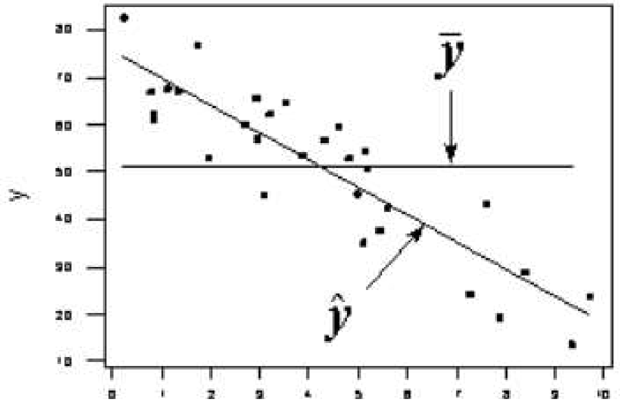
Coefficient of determination R^2



$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 = 119.1$

$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = 1708.5$

$SSTO = \sum_{i=1}^{n} (y_i - \bar{y})^2 = 1827.6$



$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 = 6679.3$

$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = 1708.5$

$SSTO = \sum_{i=1}^{n} (y_i - \bar{y})^2 = 8487.8$

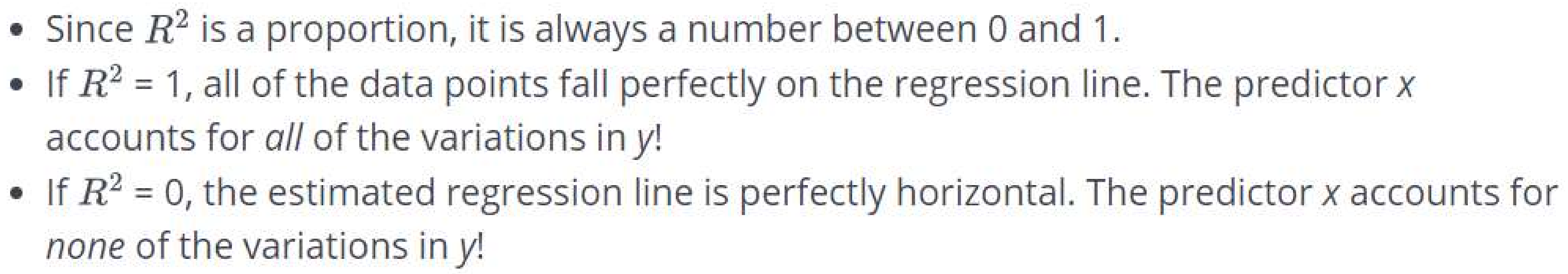
- SSR is the "regression sum of squares" and quantifies how far the estimated sloped regression line, $\hat{y}_i$, is from the horizontal "no relationship line," the sample mean or $\bar{y}$.
- SSE is the "error sum of squares" and quantifies how much the data points, y_i, vary around the estimated regression line, $\hat{y}_i$.
- SSTO is the "total sum of squares" and quantifies how much the data points, y_i, vary around their mean, $\bar{y}$.

SSTO=SSR+SSE

R^2 = SSR/SSTO=1-SSE/SSTO

Coefficient of determination R^2

Coefficient of determination (R squared value; R^2 score, R-Sq)

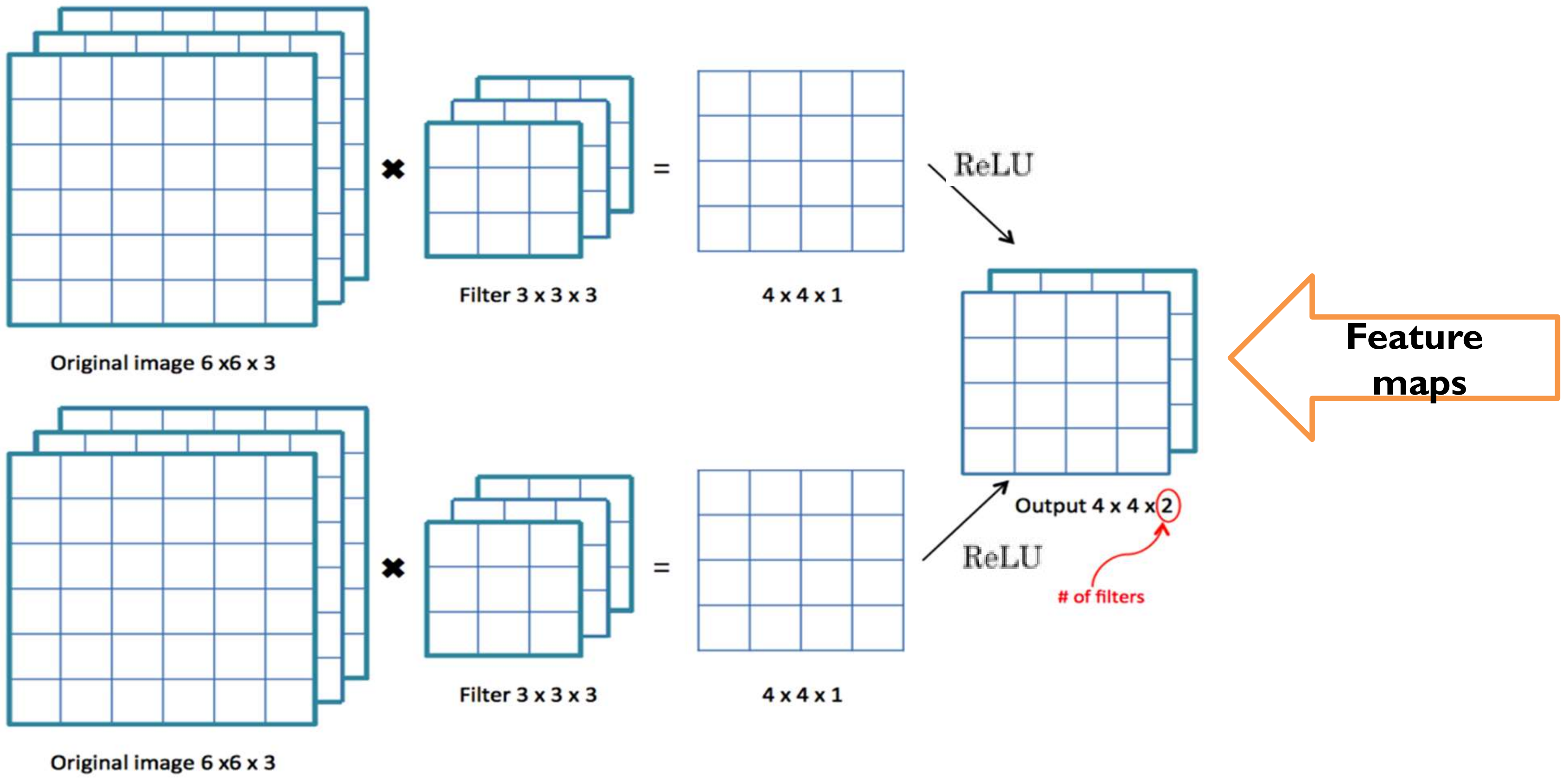


- Since R^2 is a proportion, it is always a number between 0 and 1.
- If R^2 = 1, all of the data points fall perfectly on the regression line. The predictor *x* accounts for *all* of the variations in *y*!
- If R^2 = 0, the estimated regression line is perfectly horizontal. The predictor *x* accounts for *none* of the variations in *y*!

The closer to 1, the better.
R^2 is indicative of the level of explained variability in the data set.
If R^2 = 0.5 the predictor x can explain half of the observed variation in y.

R^2 is the performance metrics (the score) by default in LinearRegression, Ridge, Lasso

Q7&17&29: ONE CONV LAYER OF CNN

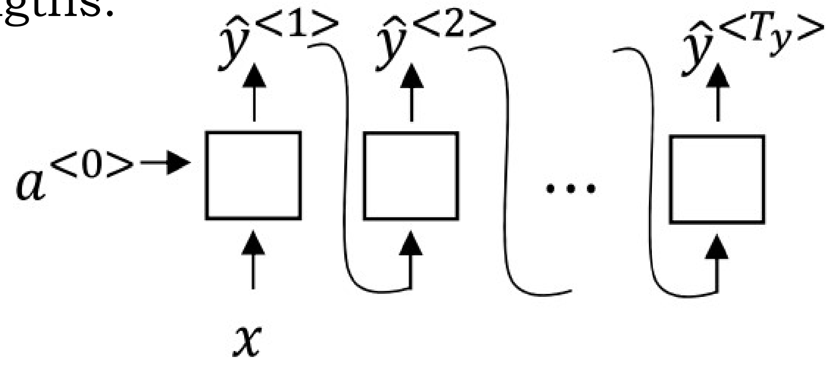


RGB images have 3 dimensions: height, width, and number of channels (3D volume). The conv filter will be also a 3D volume : $f \times f \times 3$ (number of channels has to be equal in the image and the filter)

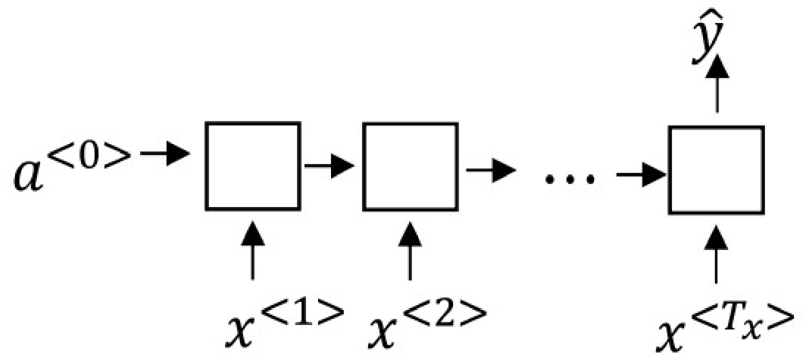
Feature map = $\lfloor \frac{n+2p-f}{s} + 1 \rfloor$ by $\lfloor \frac{n+2p-f}{s} + 1 \rfloor$ * number of filters

Q13: Different Types of RNN

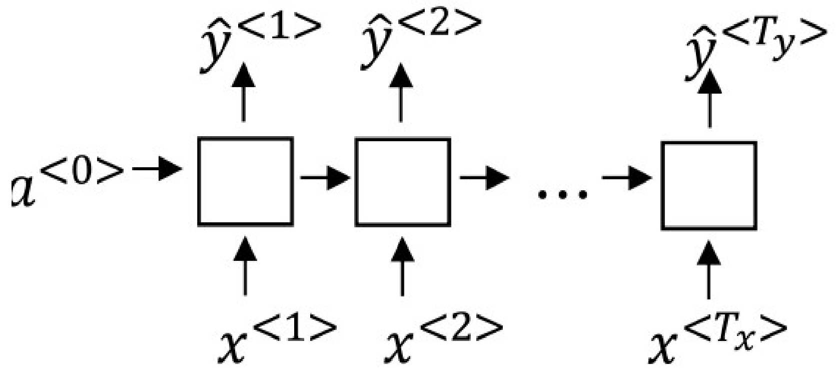
The input X and Y can be of many types and they do not have to be of the same lengths.



One to many (e.g. Music generation)

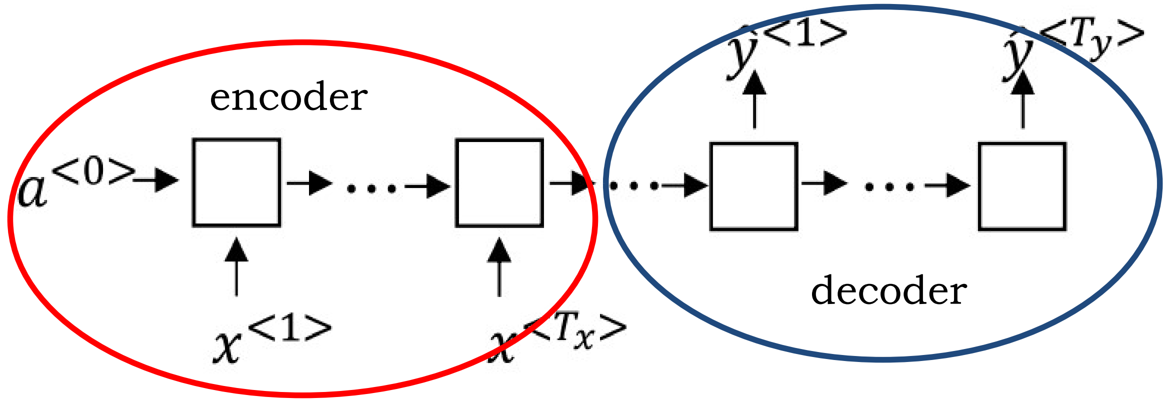


Many to one (e.g. Sentiment analysis)



Many to many

Tx=Ty (e.g. Name entity rec.)



Many to many

Tx different from Ty (e.g. machine translation)

Note: Time series forecasting (many to one, or many to many)

Q21. The following animal data set has 4 Features (Give Birth, Can Fly, Live in Water, Have legs) and 2 Classes (mammals and non-mammals) . Apply Naïve Bayes classifier to decide the class of the new example?

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

New example

Give birth	Can Fly	Live in water	Have legs	Class
No	Yes	No	Yes	?

Q22: K-means optimization objective (distortion = average distance)

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

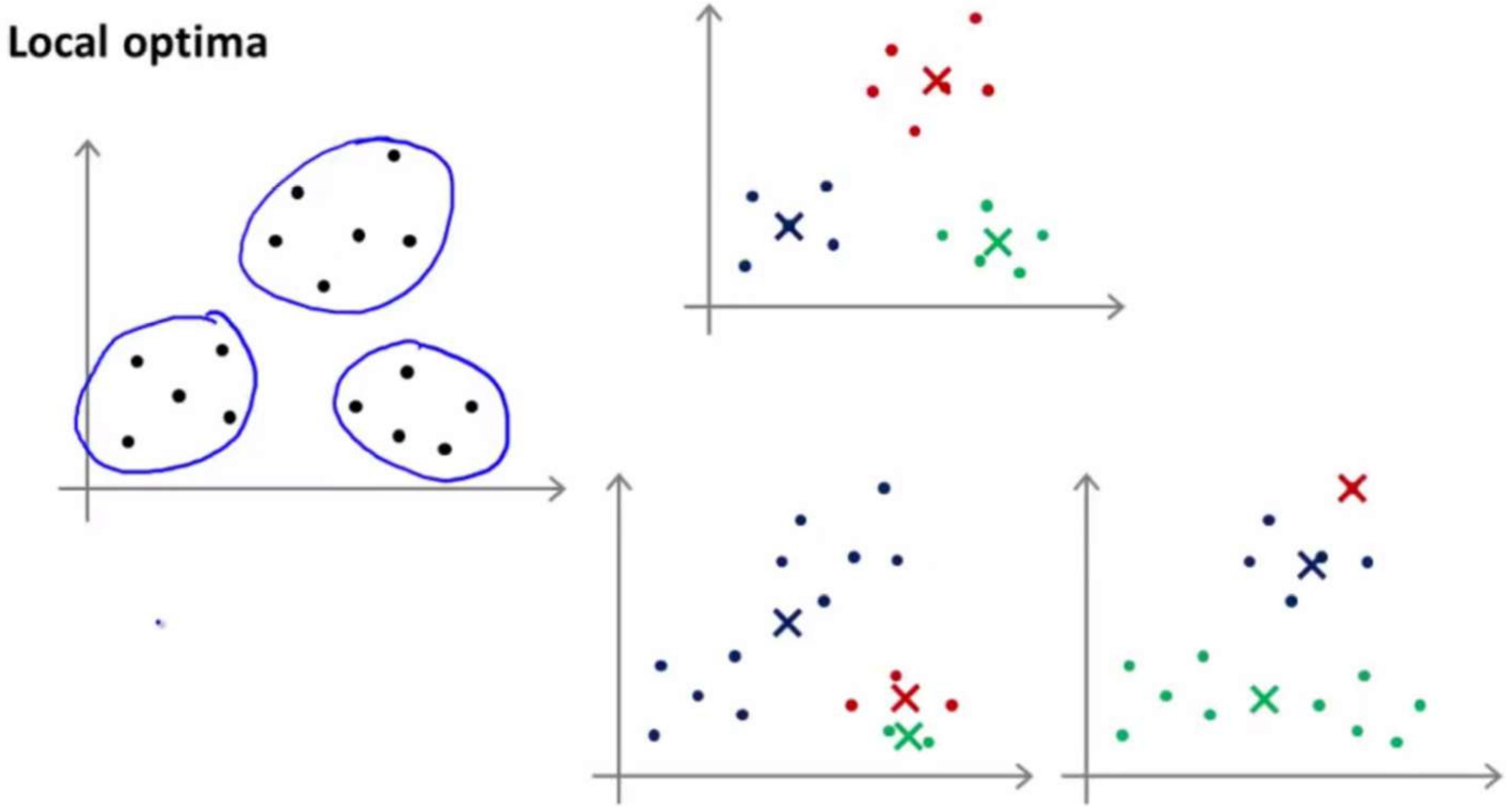
$$\min_{\substack{c^{(1)},\dots,c^{(m)},\\ \mu_1,\dots,\mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

Stop K-means learning (different criteria):

- Achieved Max number of iterations
- J < some threshold
- No improvement of *J* between subsequent iterations

Q22: Single (Random) Initialization

Choose number of clusters K
Inicialize K cluster centroids = randomly picked K training examples



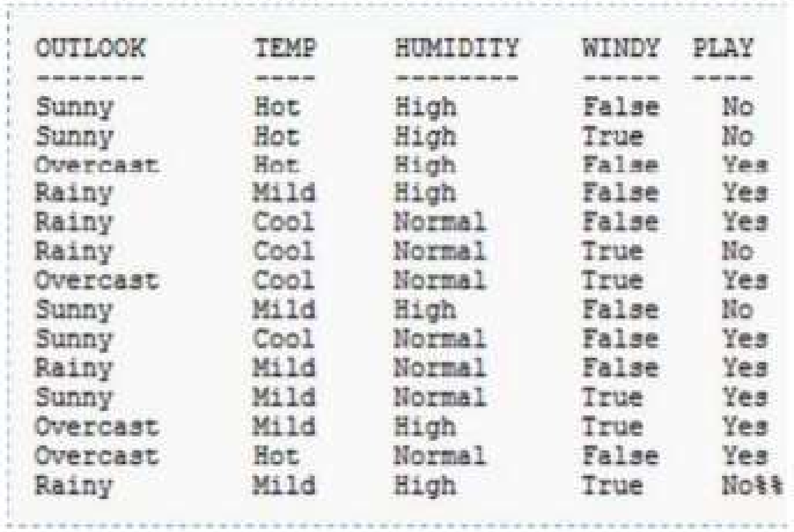
Q22: Repeat Random Initializations

```
For i = 1 to 100 {

        Randomly initialize K-means.
        Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.
        Compute cost function (distortion)
            $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$
        }
```

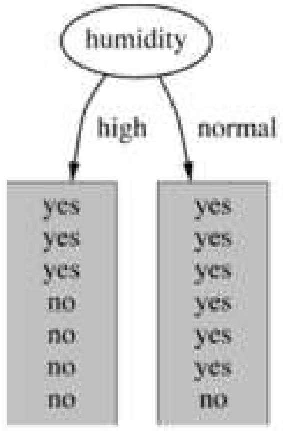
Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

Q27&26

Q27. You want to build a decision tree for the weather data based on which to decide Play golf or Not. Compute the information gain for the split at the node for humidity.



OUTLOOK	TEMP	HUMIDITY	WINDY	PLAY
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No%%



Bibliografy

•Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. Aurélien Géron. O'Reilly, 2019

•François Chollet. Deep Learning with Python, Manning, 2018.

•Andrew Ng, Machine Learning Yearning, 2017.

MOOC (Massive Open Online Courses)

• https://www.coursera.org/