Model Evaluation

Dr. Nassim Sohaee

1

Performance Measure

- Evaluating a machine learning model
- Classification is trickier than Regression

2

Cross-Validation

- *Cross-validation* is a statistical method of evaluating generalization performance that is more stable and thorough than using a split into a training and a test set.

3

Cross-Validation

- `cross_val_score:` returns accuracy values
- `cv` parameter: by default 3, but you can set it to any other value.
- Average accuracy

4

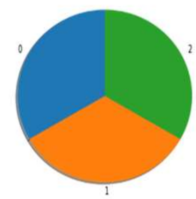
Benefits of Cross-Validation

- Average accuracy is more accurate than single split accuracy.
- Use data for effectively
- Control the structure of the folds

5

Stratified Cross-Validation

- Iris dataset – 3 fold cross validation



6

Leave-One-Out Cross-Validation

7

Grid Search

- Evaluate how well a model generalize:
  - Cross validation
- Improve the model's generalization performance by tuning its parameters.
  - Grid Search

8

Simple Grid Search

- For loop
  - Example: SVC - γ and C
  - γ∈{0.001, 0.01, 0.1, 1, 10, 100}
  - C∈{0.001, 0.01, 0.1, 1, 10, 100}

9

The Danger of Overfitting the Parameters and the Validation Set

- Try different parameters and selected the one with best accuracy on the test set.
- Because we used the test data to adjust the parameters, we can no longer use it to assess how good the model is.

10

Note

- The best_score_ stores the mean cross-validation accuracy, with cross-validation performed on the training set.
- This is different from the score for evaluating the generalization of the model.

11

Evaluation Metrics

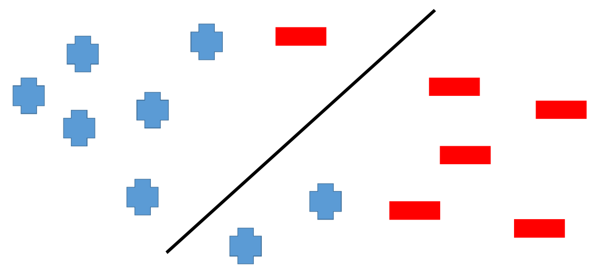
12

Goal

- Before picking a machine learning metric, you should think about the high-level goal of the application, often called the *business metric*.
- The consequences of choosing a particular algorithm for a machine learning application are called the *business impact*.

13

Binary Classification



14

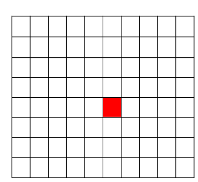
Errors

- False Positive
  - Type I error
- False Negative
  - Type II error

15

Imbalanced Dataset

- Fraud Prediction



16

Dummy classifiers completely ignore the input data!

- Dummy classifiers serve as a sanity check on your classifier's performance.
- They provide a null metric(e.g. null accuracy) baseline.
- Dummy classifiers should not be used for real problems.

17

Dummy classifiers completely ignore the input data!

- Some commonly-used settings for the strategy parameter for DummyClassifierin scikit-learn:
  - most_frequent: predicts the most frequent label in the training set.
  - stratified: random predictions based on training set class distribution.
  - uniform: generates predictions uniformly at random.
  - constant: always predicts a constant label provided by the user.
    - A major motivation of this method is F1-scoring, when the positive class is in the minority.

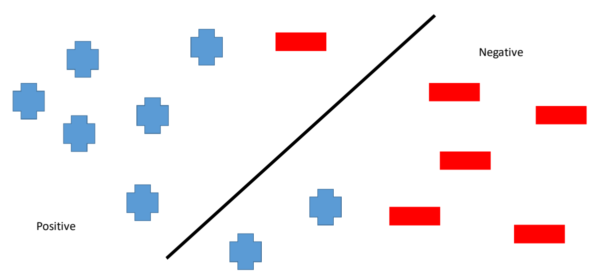
18

Confusion Matrix

True negative	TN	FP	False positive
False negative	FN	TP	True Positive

19

Confusion Matrix



20

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

5	1
2	6

Accuracy = (5 + 6) / (5 + 1 + 2 + 6) = 0.7857

21

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

5	1
2	6

Accuracy = (5 + 6) / (5 + 1 + 2 + 6) = 0.7857

22

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

5	1
2	6

Precision= 6 / 7 = 0.8571

23

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

5	1
2	6

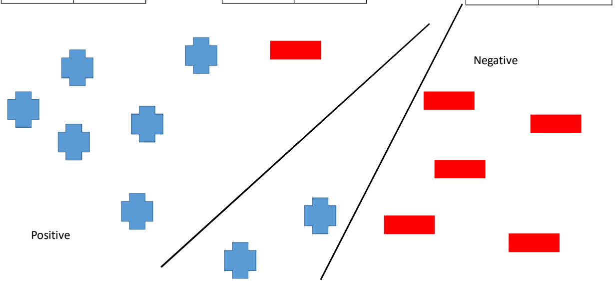
Recall= 6 / (6 + 2) = 0.75

24

Accuracy	0.7857
Precision	0.8571
Recall	0.75

Accuracy	0.8571
Precision	1
Recall	0.75

Accuracy	0.9286
Precision	0.8889
Recall	1



25

Precision-Recall trade off

- Recall-oriented machine learning tasks:
  - Search and information extraction in legal discovery
  - Tumor detection
  - Often paired with a human expert to filter out false positives
- Precision-oriented machine learning tasks:
  - Search engine ranking, query suggestion
  - Document classification
  - Many customer-facing tasks (users remember failures!)

26

F1-Score

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

27

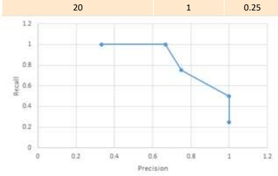
Taking Uncertainty into Account

- `decision_function`
- `predict_proba`

28

Precision-Recall Curves

classifier score threshold	Precision	Recall
-20	0.33333333	1
-10	0.66666667	1
0	0.75	0.75
10	1	0.5
20	1	0.25

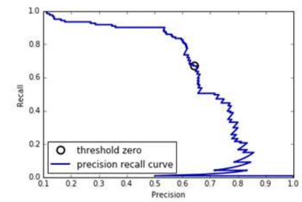


True Label	Classifier score
0	-27.6457
0	-25.8386
0	-25.1011
0	-24.1511
0	-23.1765
0	-22.575
0	-21.8271
0	-21.7226
0	-19.7361
0	-19.5768
0	-19.3071
0	-18.9077
0	-13.5411
0	-12.8594
1	-3.9198
0	-1.9798
1	1.824
0	4.74931
1	15.234624
1	21.20597

29

Precision-Recall Curve

- The `precision_recall_curve` function returns a list of precision and recall values for all possible thresholds



30

Receiver Operating Characteristics (ROC)

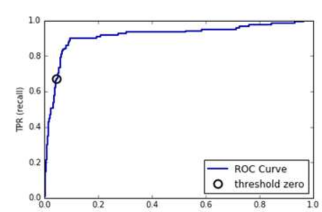
- ROC is a tool that is commonly used to analyze the behavior of the classifier at different threshold.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

31

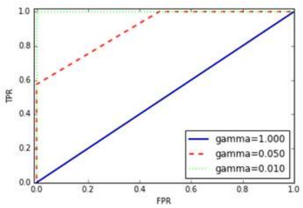
ROC



32

Area Under the Curve (AUC)

- We can compute the area under the ROC curve using the `roc_auc_score` function.



33

AUC

- Use AUC when evaluating models on imbalanced data.
- Adjusting the decision threshold might be necessary to obtain useful classification results from a model with a high AUC.

34

Metrics for Multiclass Classification

- Macro-Average
- Micro-Average

35

Macro Average

- Each class has equal weight.
  1. Compute metric within each class
  2. Average resulting metrics across classes

36

Example

Class	Predicted Class	Correct
Orange	Lemon	0
Orange	Lemon	0
Orange	Apple	0
Orange	Orange	1
Orange	Apple	0
Lemon	Lemon	1
Lemon	Apple	0
Apple	Apple	1
Apple	Apple	1

Class	Predicted Class	Correct
Orange	Not Orange	0
Orange	Not Orange	0
Orange	Not Orange	0
Orange	Orange	1
Orange	Not Orange	0
Not Orange	Not Orange	1
Not Orange	Not Orange	0
Not Orange	Not Orange	1
Not Orange	Not Orange	1

Class	Predicted Class	Correct
1	0	0
1	0	0
1	0	0
1	1	1
1	0	0
0	0	1
0	0	0
0	0	1
0	0	1

TP: 1
FP: 0
Precision = 1

37

Example

Class	Precision
Orange	1
Lemon	0.33
Apple	0.4

Macro Average Precision = $\frac{1 + 0.33 + 0.4}{3} = 0.58$

38

Micro Average

- Each instance has equal weight.
- Largest classes have most influence
  1. Aggregate outcomes across all classes
  2. Compute metric with aggregate outcomes

39

Example

Class	Predicted Class	Correct
Orange	Lemon	0
Orange	Lemon	0
Orange	Apple	0
Orange	Orange	1
Orange	Apple	0
Lemon	Lemon	1
Lemon	Apple	0
Apple	Apple	1
Apple	Apple	1

Micro-average Precision

$$= \frac{TP_{\text{orange}} + TP_{\text{Lemon}} + TP_{\text{Apple}}}{TP_{\text{orange}} + TP_{\text{Lemon}} + TP_{\text{Apple}} + FP_{\text{orange}} + FP_{\text{Lemon}} + FP_{\text{Apple}}}$$

= (1 + 1 + 2) / (1 + 1 + 2 + 0 + 2 + 3) = 0.44

40

Regression Metrics

- Analyzing overpredicting the target versus underpredicting the target.

41

Using Evaluation Metrics in Model Selection

- `scoring` argument

42

Ensemble Learning

Dr. Nassim Sohaee

43

Wisdom of the crowd

- Ask a question to thousands of random people, aggregate the answers. The aggregated answer is better than expert answer.
- Aggregate the predictions of a group of predictors, often get better predictions than individual predictor.

44

Ensemble

- A group of predictors is called an *ensemble.*
- Example:
  - train a group of Decision Tree classifiers, each on a different random subset of the training set.
  - To make predictions, obtain the predictions of all individual trees, then predict the class that gets the most votes.
  - ensemble of Decision Trees is called a Random Forest
- Use Ensemble methods near the end of a project, once you have already built a few good predictors, to combine them into an even better predictor.
  - the winning solutions in Machine Learning competitions often involve several Ensemble methods (most famously in the Netflix Prize competition).

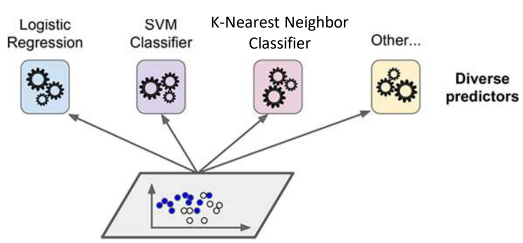
45

Ensemble

- Diverse set of algorithms
- Same algorithm on the diverse set of features
- Same algorithm on different set of instances

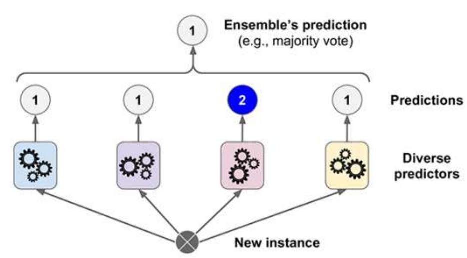
46

Voting Classifiers



47

Hard Voting Classifier



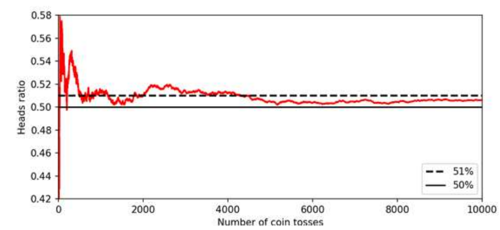
48

Hard Voting Classifier

- The ensemble of weak classifiers is often a strong classifier.
  - Weak classifier: only slightly better than random guessing
  - Strong classifier: high accuracy

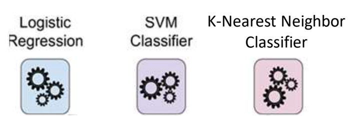
49

How is this possible?



50

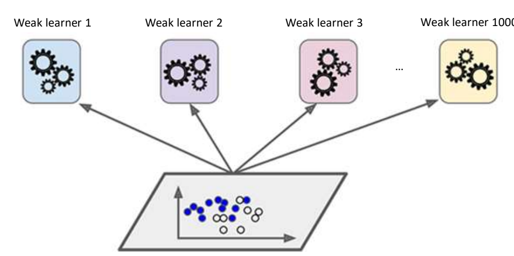
Voting Classifier



$$\binom{3}{2} \times 0.51^2 \times 0.49^1 + \binom{3}{3} \times 0.51^3 \times 0.49^0 = 0.523$$

51

Law of Large Numbers



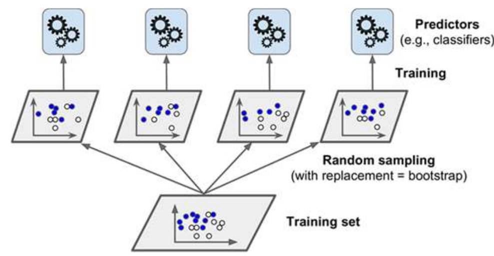
52

Soft Voting

- Predict the class with the highest class probability, averaged over all the individual classifiers.

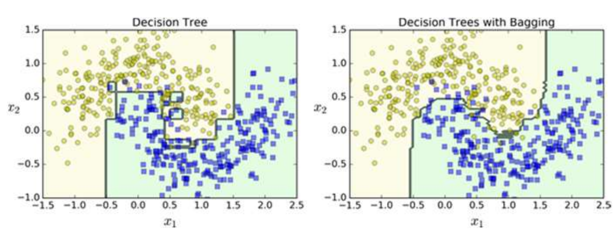
53

Bagging and Pasting



54

Decision Tree with Bagging and Pasting



55

Out-Of-Bag Evaluation

- On average only 63% of the training instances are sampled in average.
- The remaining samples are called out-of-bag (oob) objects.

56

Random Patches and Random Subspaces

- Random subspace: random sunset of feature set
- Random patch: random subset of instance set

57

Random Forest

- An ensemble of decision trees, training via bagging (or pasting)

58

Feature Importance

- Measure the relative importance of each feature.
  - how much the tree nodes that use that feature reduce impurity on average

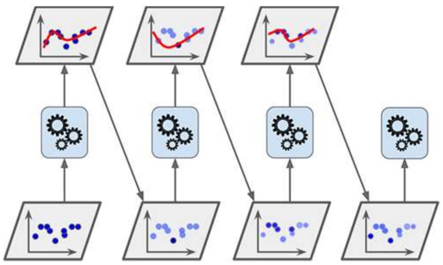
59

Boosting

- AdaBoost
- Gradient Boosting

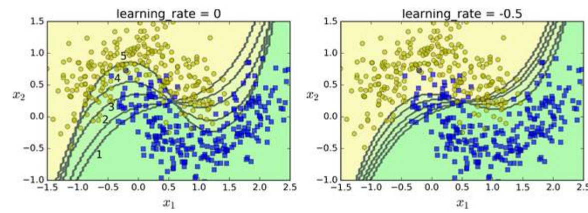
60

Adaboost



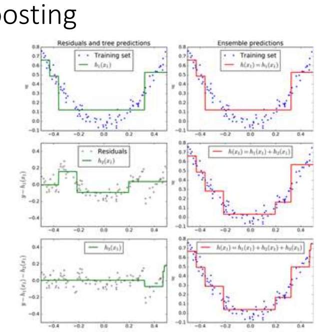
61

Learning Rate

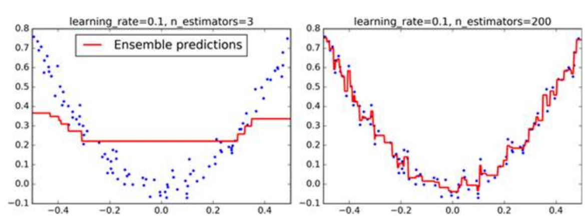


62

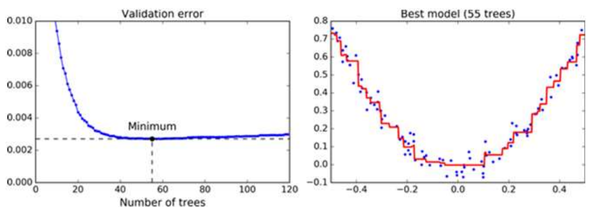
Gradient Boosting



63

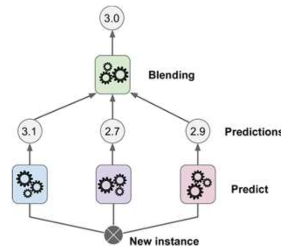


64



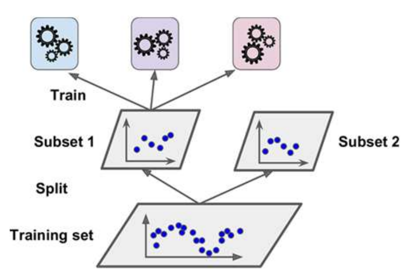
65

Stacking

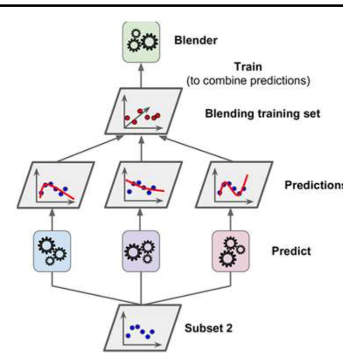


66

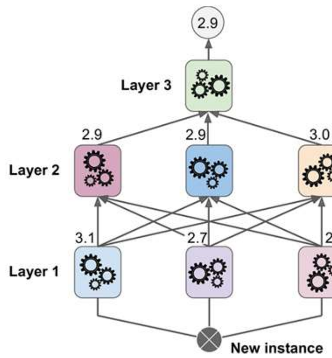
Stacking



67



68



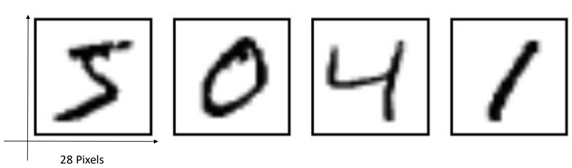
69

Dimensionality Reduction

Dr. Nassim Sohaee

70

Example: MNIST



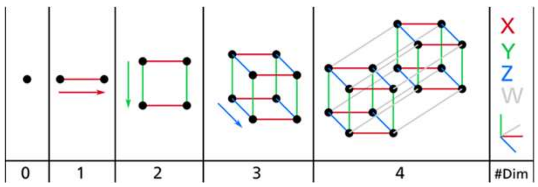
71

Pros and Cons

Pros	Cons
speed up training	system perform slightly worse
data visualization	lose some information

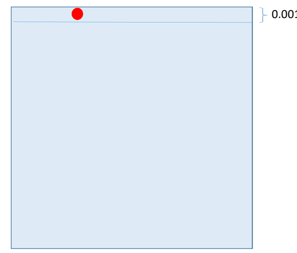
72

The Curse of Dimensionality



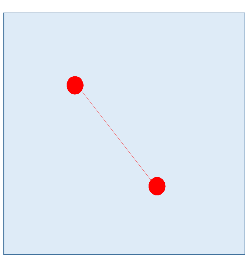
73

Unit square/cube/hypercube



74

Distance of two points



2 – dimension	0.52
3 – dimension	0.66
1,000,000 – dimension	408.25

75

Distance of two instances

- Sparse training sets
- Less reliable predictions in higher dimension
  - overfitting

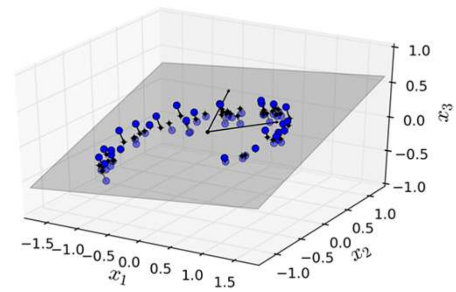
76

Approaches in Dimensionality Reduction

- Projection
- Manifold Learning

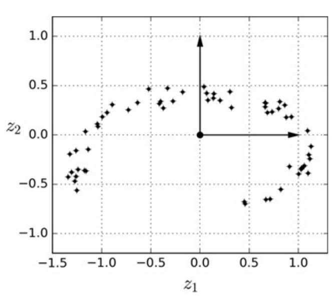
77

Projection



78

Projection



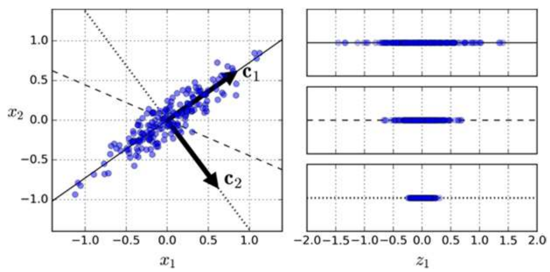
79

PCA

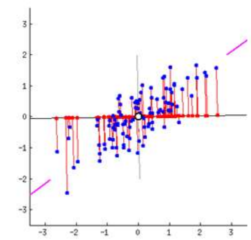
- *Principal Component Analysis* (PCA) is by far the most popular dimensionality reduction algorithm.
- First it identifies the hyperplane that lies closest to the data, and then it projects the data onto it.

80

Preserving variance



81



82

Principal components

- PCA identifies the axis that accounts for the largest amount of variance in the training set.
- The unit vector that defines the i^{th} axis is called the i^{th} *principal component* (PC).
- The direction of the principal components is not stable.

83

The eigenvectors are called *principal axes* or *principal directions* of the data.

$$X_{train} = U.\Sigma.V^T$$

$$\mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{c_1} & \mathbf{c_2} & \cdots & \mathbf{c_n} \\ | & | & & | \end{pmatrix}$$

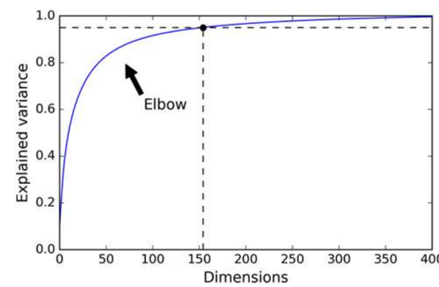
84

Projecting down to d dimensions

$$\mathbf{X}_{d\text{-proj}} = \mathbf{X} \cdot \mathbf{W}_d$$

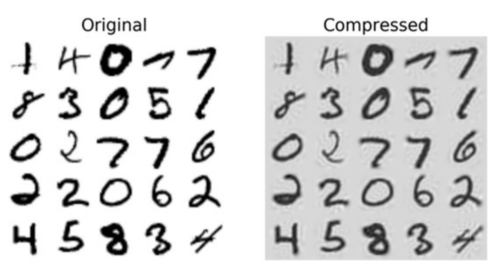
85

Explained Variance Ratio



86

PCA for compression



87

PCA for compression

- Reconstruction Error

$$\mathbf{X}_{\text{recovered}} = \mathbf{X}_{d\text{-proj}} \cdot \mathbf{W}_d^{\ T}$$

88

Incremental PCA

- Problem: PCA requires the whole training set to fit in memory in order for the SVD algorithm to run.
- Solution: Incremental PCA
- Running time: $O(m \times n^2) + O(n^3)$

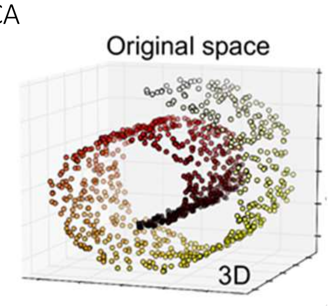
89

Stochastic PCA

- Finds an approximation of the first *d* principal components.
- Running time: $O(m \times d^2) + O(d^3)$

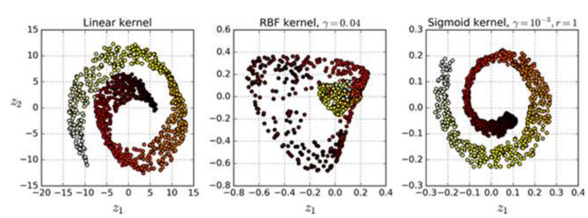
90

Kernel PCA



91

Kernel PCA



92

Reconstruction

- $X_{m\times n}$: Training set – Centered
- m : Number of instances
- n : Number of features

93

Reconstruction

Covariance Matrix : $\mathrm{C} = \frac{X \times X^T}{n-1}$

C is a symmetric matrix and so it can be diagonalized:

$C = V \times L \times V^T$

V: is eigenvector Matrix
L: is diagonal matrix with eigen values λ_i in increasing order on the diagonal.

94

Reconstruction

$$L = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix}$$

V is the principal axes.

Projecting training set X means finding $X \times V$

95

Reconstruction

$X = U \times S \times V^T$
U: is a unitary matrix, $U^T = U^{-1}$
S: is a diagonal matrix with singular values s_i
V: we prove that this is the same as before

$$C = \frac{X^T \times X}{n-1} = \frac{(U \times S \times V^T)^T \times (U \times S \times V^T)}{n-1}$$
$$= \frac{V \times S^T \times U^T \times U \times S \times V^T}{n-1} = V \times \frac{S^2}{n-1} \times V^T$$

96

Reconstruction

$$X = U \times S \times V^T$$

$$= \begin{bmatrix} | & | & | & | & | & | \\ U_1 & \cdots & U_k & U_{k+1} & \cdots & U_n \\ | & | & | & | & | & | \end{bmatrix} \times \begin{bmatrix} \begin{bmatrix} s_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_k \end{bmatrix} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_n \end{bmatrix} \times \begin{bmatrix} V_{k\times k} & \cdots \\ \cdots & \cdots \end{bmatrix}^T$$

97

Reconstruction

$$\underbrace{X_k}_{m \times k} = \underbrace{U_k}_{m \times k} \times \underbrace{S_k}_{k \times k} \times \underbrace{V_k^T}_{k \times k}$$

98

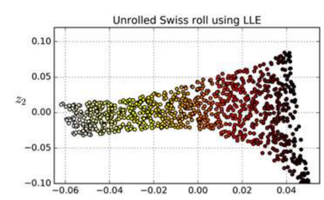
Approaches in Dimensionality Reduction

- Projection
- Manifold Learning

99

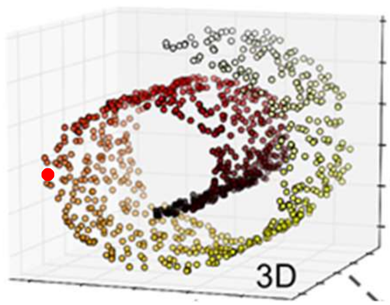
LLE

- Locally Linear Embedding



100

How LLE works:



101

How LLE works:

$$\sum_{j=1}^{m} w_{i,j} x^{(j)}$$

When $x^{(j)}$ is not one of the close neighbors of instance x_i, $w_{i,j} = 0$.

$$W = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{m} \left\| x^{(i)} - \sum_{j=1}^{m} w_{i,j} x^{(j)} \right\|^2$$

102

First Step

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{m} \| \mathbf{x}^{(i)} - \sum_{j=1}^{m} w_{i,j} \mathbf{x}^{(j)} \|^2$$

$$\text{subject to } \begin{cases} w_{i,j} = 0 & \text{if } \mathbf{x}^{(j)} \text{ is not one of the } k \text{ c.n. of } \mathbf{x}^{(i)} \\ \sum_{j=1}^{m} w_{i,j} = 1 & \text{for } i = 1, 2, \cdots, m \end{cases}$$

103

Second Step:

$$\hat{\mathbf{Z}} = \underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{m} \| \mathbf{z}^{(i)} - \sum_{j=1}^{m} \hat{w}_{i,j} \mathbf{z}^{(j)} \|^2$$

$\hat{Z}$ is the image of X in the lower dimension.

104

LLE

- Time complexity:
- $O(m \log(m) n \log(k))$ for finding the k nearest neighbors
- $O(mnk^3)$ for optimizing the weights
- $O(dm^2)$ for constructing the low-dimensional representations

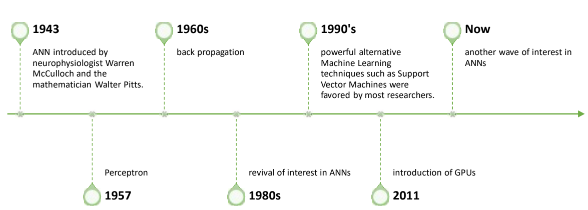
105

Neural Networks and Deep Learning

Dr. Nassim Sohaee

106

Timeline



107

Logical Computations with Neurons

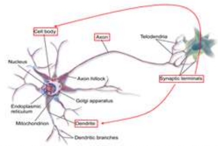
- Warren McCulloch and Walter Pitts proposed a very simple model of the biological neuron, which later became known as an artificial neuron.
    - An artificial neuron has one or more binary (on/off) inputs and one binary output.
    - The artificial neuron simply activates its output when more than a certain number of its inputs are active.
- They proved this simplified model can build a network of artificial neurons that computes any logical proposition.

108

Background

- Several interconnected neurons, organized in layers, which exchange messages when certain conditions satisfied.

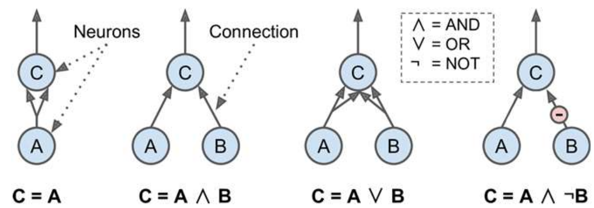
Biological Neurons



Biological neurons receive short electrical impulses called signals from other neurons via these synapses.

109

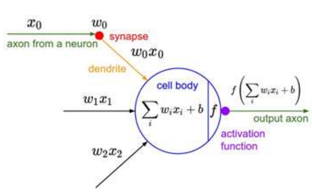
Logical Computations with Neurons



110

Perceptron

- The *Perceptron* is one of the simplest ANN architectures, invented in 1957 by Frank Rosenblatt.



111

Perceptron

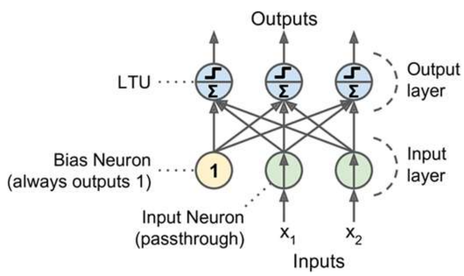
- Common step function:

$$\text{heaviside}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

- Linear threshold unit can be used for simple binary classification.
- It computes a linear combination of the inputs and if the results exceeds a threshold, it outputs the positive class or else outputs the negative class.

112

Multi-Class Classification



113

Perceptron training

- Hebb's rule: Cells that fire together, wire together.

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

connection weight between the i^{th} input neuron and the j^{th} output neuron.

Learning rate

i^{th} input value of the current training instance.

114

Perceptron

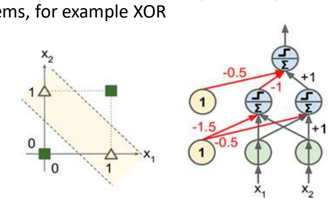
- A simple perceptron can be famulated as

$$f(x) = \begin{cases} 1 & wx + b > 0 \\ 0 & otherwise \end{cases}$$

115

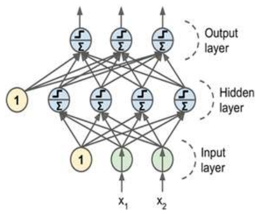
Single Layer Perceptron

- Perceptron does not output a class probability, they just make predictions based on a hard threshold.
- Like any other linear classification they are incapable of solving some trivial problems, for example XOR



116

Multi-Layer Perceptron



117

Dense Layer

- The network is dense, meaning that each neuron in a layer is connected to all neurons located in the previous layer and to all the neurons in the following layer.

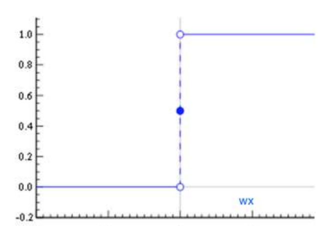
118

Back Propagation

1. make a prediction (forward pass)
2. measure the error
3. go through each layer in reverse to measure the error contribution from each connection (reverse pass)
4. tweak the weights to reduce the error

119

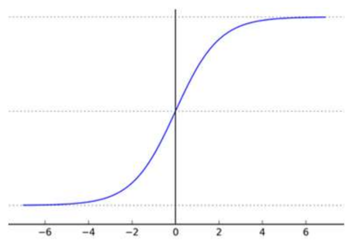
Step Function?



120

Sigmoid

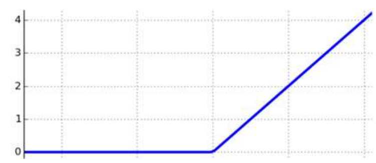
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



121

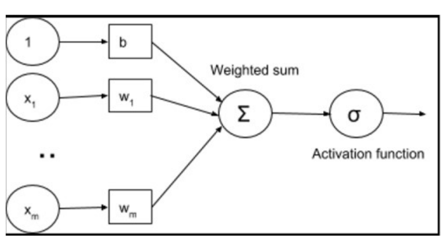
ReLU

- *f(x)* = max(*0*,*x*)



122

Activation Function



123

Number of Hidden Layers

- begin with a single hidden layer
- deep networks have a much higher *parameter efficiency* than shallow ones
  - DNN converges faster
  - Generalize better

124

Number of Neurons per Hidden Layers

- Input and output layer: input dataset and project description.
- Hidden layers: form a funnel

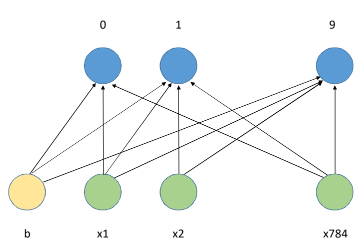
125

Activation Function

- Input layer
  - ReLU or Sigmoid
- Hidden layer
  - ReLU
- Output layer
  - Classification: Softmax, Logistic
  - Regression: None

126

Defining a simple neural net in Keras



127

Compile Model

- Optimizer
- Objective Function/Loss Function
- Evaluation method

128

Optimizer

- Speed up training process:
  1. Good initialization strategy
  2. Good activation function
  3. Faster optimizer
- Popular optimizers:
  - Momentum optimization
  - Nesterov Accelerated Gradient
  - AdaGrad
  - RMSProp
  - Adam optimization.

129

Adam Optimizer

- Adam: adaptive moment estimation
- Adam configuration parameter:
  - Alpha
  - Beta1
  - Beta2
  - Epsilon
- Adam is a popular algorithm in the field of deep learning because it achieves good results fast.

130

Objective Function/Loss Function

1. MSE
2. Binary cross-entropy: $-t\log(p) - (1-t)log(1-p)$
3. Categorical cross-entropy: $L_i = -\Sigma_j t_{i,j} \log(p_{i,j})$

131

Fit the model

- epochs: number of times the model is exposed to the training set. At each iteration, the optimizer tries to adjust the weights so that the objective function is minimized.
- batch_size: number of training instances observed before the optimizer performs a weight update.

132