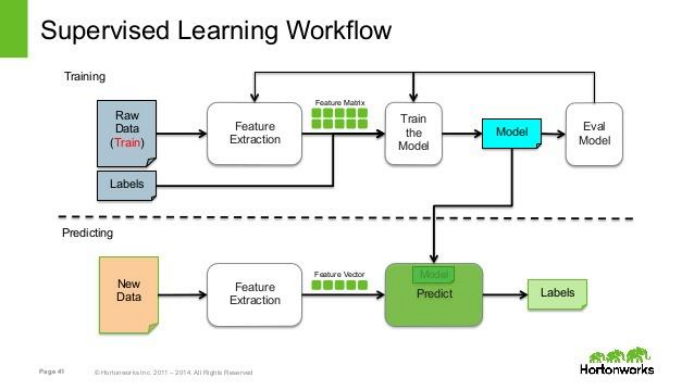
**Supervised Classification**

**Contents**

● kNN classifier ● Confusion Matrix ● Conclusions

**Types of Machine Learning**



Machinerning

Supervised Unsupervised Re

**Instance Based Classifiers**

● First Example of Supervised Classification ● Examples:

– Rote-learner

● Memorizes entire training data and performs classification only if attributes of record match one of the training

examples exactly

– Nearest neighbor

● Uses k “closest” points (nearest neighbors) for performing classification

**Instance-Based Classifiers**

Set of Stored Cases

Atr1 ……... AtrN Class A

B

B

C

A

C

B

• Store the training records

• Use training records to predict the class label of unseen cases

Unseen Case

Atr1 ……... AtrN

**Nearest Neighbor Classifiers**

● Basic idea:

– If it walks like a duck, quacks like a duck, then it’s probably a duck

Compute

Distance

Test

Record

Training Records

Choose k of the “nearest” records

**Nearest-Neighbor Classifiers**

**Unknown record**

****Requires three things

– The set of stored records – Distance Metric to compute distance between records

– The value of *k*, the number of nearest neighbors to retrieve

To classify an unknown record: – Compute distance to other training records

– Identify *k* nearest neighbors – Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

**Definition of Nearest Neighbor**

X X X

(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

**Nearest Neighbor Classification**

● Compute distance between two points: – Euclidean distance

( , ) ( ) =∑ −

= ∑ −

*d p q piqi*2 *i*

*d p q abs piqi*

( , ) ( )*i*

● Determine the class from nearest neighbor list – take the majority vote of class labels among the k nearest neighbors

– Weigh the vote according to distance

● weight factor, w = 1/d2

**Example (NN Classifier)**

| **F1** | **F2** | **Class** |
| --- | --- | --- |
| **1** | **5** | **0** |
| **0** | **8** | **0** |
| **0** | **6** | **1** |
| **1** | **2** | **1** |

| ***1*** | ***3*** | ***?*** |
| --- | --- | --- |
| ***1*** | ***4*** | ***?*** |
| ***0*** | ***3*** | ***?*** |
| ***0*** | ***4*** | ***?*** |

**Training Data Test Data**

**Example (NN Classifier)**

Step 1: Computer Distance from Test Sample 1 to Training Data Step 2:

| Distance from Test Sample 1 to All Training Samples  1 |1-1|+|3-5| = 0 + 2 = 2 | | **Class**  0 |
| --- | --- | --- |
| 2 | |1-0|+|3-8| = 1 + 5 = 6 | 0 |
| 3 | |1-0|+|3-6| = 1 + 3 = 4 | 1 |
| 4 | |1-1|+|3-2| = 0 + 1 = 1 | 1 |

Step 3: Assign the Test Sample to Class with minimum Distance, Here is Class 1. So Test Sample 1 belongs to Class 1

**Example (NN Classifier)** Exercise: Calculate for other 3 Test Samples

| **ID**  1 | **Actual**  0 | **Predicted**  1 |
| --- | --- | --- |
| 2 | 0 | 0 |
| 3 | 1 | 1 |
| 4 | 1 | 0 or 1 |

**Nearest Neighbor Classification…**

● Choosing the value of k:

– If k is too small, sensitive to noise points – If k is too large, neighborhood may include points from other classes

**X**

**Nearest Neighbor Classification…**

● Scaling issues

– Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes

– Example:

● height of a person may vary from 1.5m to 1.8m

● weight of a person may vary from 90lb to 300lb

● income of a person may vary from $10K to $1M

**Example (NN Classifier)**

Normalize Data from 0 to 1

| **F1** | **F2** | **Class** |
| --- | --- | --- |
| **1** | **0.5** | **0** |
| **0** | **1** | **0** |
| **0** | **0.667** | **1** |
| **1** | **0** | **1** |

| ***1*** | ***0.167*** | ***?*** |
| --- | --- | --- |
| ***1*** | ***0.334*** | ***?*** |
| ***0*** | ***0.167*** | ***?*** |
| ***0*** | ***0.334*** | ***?*** |

**Training Data Test Data**

**Example (NN Classifier)**

After Normalization

| **ID**  1 | **Actual**  0 | **Predicted**  1 |
| --- | --- | --- |
| 2 | 0 | 0 |
| 3 | 1 | 1 |
| 4 | 1 | 1 |

**Confusion Matrix**

● In the field of machine learning, a **confusion matrix** is a specific table layout that allows visualization of the performance of an algorithm

|  | Predicted Negative | Predicted Positive |
| --- | --- | --- |
| Actual Negative | True Negative | False Positive |
| Actual Positive | False Negative | True Positive |

**Confusion Matrix**

● *TN* is the number of correct predictions that an instance is negative

● *FP* is the number of incorrect predictions that an instance is positive

● *FN* is the number of incorrect predictions that an instance is negative

● *TP* is the number of correct predictions that an instance is positive

**Confusion Matrix**

● Confusion Matrix from the example of Lecture 2 (without Normalization)

| **ID**  1 | **Actual**  1 | **Predicted**  1 |
| --- | --- | --- |
| 2 | 0 | 0 |
| 3 | 1 | 1 |
| 4 | 1 | 0 |

| Negative | **Negative**  TN = 1 | **Positive**  FP = 0 |
| --- | --- | --- |
| Positive | FN = 1 | TP = 2 |

**Confusion Matrix**

● Several standard terms have been defined for the 2 class matrix

● The *accuracy* (*AC*) is the proportion of the total number of predictions that were correct

*TN TP Accuracy*+ + +

+

=

*TN FN TP FP*

● Accuracy = 3 / 4 = 75%

**Confusion Matrix**

● The *recall* or *true positive rat*e (*TPR*) is the proportion of positive cases that were correctly

identified

*TP TPR*+

=

*TP FN*

● The *false positive rate* (*FPR*) is the proportion of negatives cases that were incorrectly classified as

positive

*FP FPR*+

=

*FP TN*

● TPR or recall = 2 / 3 = 66.7% ● FPR = 0 / 1 = 0 %

**Confusion Matrix**

● The *true negative rate* (*TNR*) is defined as the proportion of negatives cases that were classified

correctly,

*TN TNR*+

=

*FP TN*

● The *false negative rate* (*FNR*) is the proportion of positives cases that were incorrectly classified as

negative

*FN FNR*+

=

● TNR = 1 / 1 = 100% ● FNR = 1 / 3 = 33.3%

*FN TP*

**Confusion Matrix**

● *precision* (*P*) is the proportion of the predicted positive cases that were correct,

*tp precision*+

=

● precision = 2/2 = 100%

*tp fp*

● F measure is harmonic mean of precision and recall



● F1 = (2 \* 1 \* 0.667)/(1+0.667) = 0.8

**Exercise**

**Predicted**

|  | **Actual** | |
| --- | --- | --- |
| Negative | **Negative** 9760 | **Positive**  40 |
| Positive | 140 | 60 |

**References**

● Introduction to Data Mining by Tan, Steinbach, Kumar (Lecture Slides)

● http://robotics.stanford.edu/~ronnyk/glossary.html ● http://www.cs.tufts.edu/comp/135/Handouts/introduction lecture-12-handout.pdf

Questions!