Review: Numeric Estimation

- *Numeric estimation* is like classification learning.
  - it involves learning a model that works like this:
    
    
    input attributes  \(\rightarrow\) model  \(\rightarrow\) output attribute/class

    
    - the model is learned from a set of training examples that include the output attribute
    - In numeric estimation, the output attribute is numeric.
      - we want to be able to *estimate* its value
Example Problem: CPU Performance

• We want to predict how well a CPU will perform on some task, given the following info. about the CPU and the task:
  • CTIME: the processor's cycle time (in nanosec)
  • MMIN: minimum amount of main memory used (in KB)
  • MMAX: maximum amount of main memory used (in KB)
  • CACHE: cache size (in KB)
  • CHMIN: minimum number of CPU channels used
  • CHMAX: maximum number of CPU channels used

• We need a model that will estimate a performance score for a given combination of values for these attributes.

CTIME MMIN, MMAX CACHE CHMIN, CHMAX ➔ model ➔ performance (PERF)

Example Problem: CPU Performance (cont.)

• The data was originally published in a 1987 article in the Communications of the ACM by Phillip Ein-Dor and Jacob Feldmesser of Tel-Aviv University.

• There are 209 training examples. Here are five of them:

```
input attributes:

CTIME  MMIN  MMAX  CACHE  CHMIN  CHMAX  PERF
125    256    6000   256     16     128     198
29     8000   32000  32      8      32      269
29     8000   32000  32      8      32      172
125    2000   8000   0       2      14      52
480    512    8000   32      0      0     67
```
Linear Regression

• The classic approach to numeric estimation is linear regression.

• It produces a model that is a linear function (i.e., a weighted sum) of the input attributes.
  • example for the CPU data:
    \[
    \text{PERF} = 0.066 \text{CTIME} + 0.0143 \text{MMIN} + 0.0066 \text{MMAX} + 0.4945 \text{CACHE} - 0.1723 \text{CHMIN} + 1.2012 \text{CHMAX} - 66.48
    \]
  • this type of model is known as a regression equation

• The general format of a linear regression equation is:
  \[y = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + c\]
  where
  \[y\] is the output attribute
  \[x_1, \ldots, x_n\] are the input attributes
  \[w_1, \ldots, w_n\] are numeric weights
  \[c\] is an additional numeric constant

Linear Regression (cont.)

• Once the regression equation is learned, it can estimate the output attribute for previously unseen instances.
  • example: to estimate CPU performance for the instance
    \[
    \begin{array}{cccccccc}
    \text{CTIME} & \text{MMIN} & \text{MMAX} & \text{CACHE} & \text{CHMIN} & \text{CHMAX} & \text{PERF} \\
    480 & 1000 & 4000 & 0 & 0 & 0 & ?
    \end{array}
    \]
    we plug the attribute values into the regression equation:
    \[
    \begin{align*}
    \text{PERF} &= 0.066 \times 480 + 0.0143 \times 1000 + 0.0066 \times 4000 + 0.4945 \times 0 - 0.1723 \times 0 + 1.2012 \times 0 - 66.48 \\
    &= 0.066 \times 480 + 0.0143 \times 1000 + 0.0066 \times 4000 + 0.4945 \times 0 - 0.1723 \times 0 + 1.2012 \times 0 - 66.48 \\
    &= 5.9
    \end{align*}
    \]
Linear Regression with One Input Attribute

- Linear regression is easier to understand when there’s only one input attribute, $x_1$.

- In that case:
  - the training examples are ordered pairs of the form $(x_1, y)$
    - shown as points in the graph above
  - the regression equation has the form $y = w_1 x_1 + c$
    - shown as the line in the graph above
    - $w_1$ is the slope of the line; $c$ is the y-intercept

- Linear regression finds the line that "best fits" the training examples.

---

Linear Regression with One Input Attribute (cont.)

- The dotted vertical bars show the differences between:
  - the actual $y$ values (the ones from the training examples)
  - the estimated $y$ values (the ones given by the equation)

Why do these differences exist?

- Linear regression finds the parameter values ($w_1$ and $c$) that minimize the sum of the squares of these differences.
Linear Regression with Multiple Input Attributes

- When there are \( k \) input attributes, linear regression finds the equation of a line in \((k+1)\) dimensions.
  - Here again, it is the line that "best fits" the training examples

- The equation has the form we mentioned earlier:
  \[ y = w_1x_1 + w_2x_2 + \ldots + w_nx_n + c \]

- Here again, linear regression finds the parameter values (for the weights \( w_1, \ldots, w_n \) and constant \( c \)) that minimize the sum of the squares of the differences between the actual and predicted \( y \) values.

Linear Regression in Weka

- Use the *Classify* tab in the Weka Explorer.

- Click the *Choose* button to change the algorithm.
  - Linear regression is in the folder labelled *functions*

- By default, Weka employs *attribute selection*, which means it may not include all of the attributes in the regression equation.
Linear Regression in Weka (cont.)

- On the CPU dataset with M5 attribute selection, Weka learns the following equation:
  \[ \text{PERF} = 0.0661 \text{CTIME} + 0.0142 \text{MMIN} + 0.0066 \text{MMAX} + 0.4871 \text{CACHE} + 1.1868 \text{CHMAX} - 66.60 \]
  - it does not include the 

- To eliminate attribute selection, you can click on the name of the algorithm and change the attributeSelectionMethod parameter to "No attribute selection".
  - doing so produces our earlier equation:
    \[ \text{PERF} = 0.066 \text{CTIME} + 0.0143 \text{MMIN} + 0.0066 \text{MMAX} + 0.4945 \text{CACHE} - 0.1723 \text{CHMIN} + 1.2012 \text{CHMAX} - 66.48 \]

- Notes about the coefficients:
  - what do the signs of the coefficients mean?
  - what about their magnitudes?

Evaluating a Regression Equation

- To evaluate the goodness of a regression equation, we again set aside some of the examples for testing.
  - do not use these examples when learning the equation
  - use the equation on the test examples and see how well it does

- Weka provides a variety of error measures, which are based on the differences between the actual and estimated y values.
  - we want to minimize them

- The correlation coefficient measures the degree of correlation between the input attributes and the output attribute.
  - its absolute value is between 0.0 and 1.0
  - we want to maximize its absolute value
Simple Linear Regression

- This algorithm in Weka creates a regression equation that uses only one of the input attributes.
  - even when there are multiple inputs

- Like 1R, simple linear regression can serve as a baseline.
  - compare the models from more complex algorithms to the model it produces

- It also gives insight into which of the inputs has the largest impact on the output.

Handling Non-Numeric Input Attributes

- We employ numeric estimation when the output attribute is numeric.

- Some algorithms for numeric estimation also require that the input attributes be numeric.

- If we have a non-numeric input attribute, it may be possible to convert it to a numeric one.
  - example: if we have a binary attribute (yes/no or true/false), we can convert the two values to 0 and 1

- In Weka, many algorithms – including linear regression – will automatically adapt to non-numeric inputs.
Handling Non-Numeric Input Attributes (cont.)

- There are algorithms for numeric estimation that are specifically designed to handle both numeric and nominal attributes.

- One option:
  - build a decision tree
  - have each classification be a numeric value that is the average of the values for the training examples in that subgroup
  - the result is called a regression tree

- Another option is to have a separate regression equation for each classification in the tree – based on the training examples in that subgroup.
  - this is called a model tree
Regression and Model Trees in Weka

- To build a tree for estimation in Weka, select the M5P algorithm in the trees folder.
  - by default, it builds model trees
  - you can click on the name of the algorithm and tell it to build regression trees