Data Mining

Regression
Outline

1. What is Regression?
2. KNN for Regression
3. Model Evaluation
4. Regression Trees
5. Linear Regression
6. Polynominal Regression
7. Local Regression
8. ANNs for Regression
9. Time Series Forecasting
1. What is Regression?

- Goal: Predict the value of a **scalar variable** based on the values of other variables, assuming a linear or nonlinear model of dependency.
  - The predicted variable is called **dependent** and is denoted $\hat{y}$
  - The other variables are called **explanatory variables** or independent variables, denoted $X = x_1, x_2, \ldots, x_n$

- Approach: Given training examples $(X_i, y_i)$ learn a model $f$ to predict $\hat{y}$ from $X$.

- Difference to classification: The predicted attribute is continuous, while classification is used to predict nominal class attributes.
Regression Model Learning and Application

- **Explanatory variables**: Tid, Attrib1, Attrib2, Attrib3
- **Dependent variable**: y

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>194</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>78</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>89</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>131</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>244</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>669</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
Regression Examples

- Weather Forecasting
  - dependent: wind speed
  - explanatory variables: temperature, humidity, air pressure change

- Gasoline Consumption
  - dependent: MPG (miles per gallon)
  - explanatory variables: weight of car, horse power, type of engine

- House Market
  - dependent: price of a house
  - explanatory variables: rooms, distance to public transport, size of garden

- Stock Market
  - dependent: price of a stock
  - explanatory variables: company profit, sector outlook, month of year, weather forecast

Source: KDnuggets online poll, 732 votes
Regression Techniques

1. Linear Regression
2. Polynomial Regression
3. Local Regression
4. K-Nearest-Neighbors Regression
5. Regression Trees
6. Support Vector Machines
7. Artificial Neural Networks
8. Deep Neural Networks
9. Component Models of Time Series
10. Many others …
2. K-Nearest-Neighbors Regression

Problem
- predict the temperature in a certain place
- where there is no weather station
- how could you do that?
Recap: K-Nearest-Neighbors Classification

- Idea: use the average of the nearest stations

- Example:
  - 3x sunny
  - 2x cloudy
  - Result: sunny

- Approach is called
  - “k nearest neighbors”
  - where k is the number of neighbors to consider
  - in the example: k=5
  - in the example: “near” denotes geographical proximity
K-Nearest-Neighbors Regression

- Idea: use the *numeric average* of the nearest stations

- Example:
  - 18°C, 20°C, 21°C, 22°C, 21°C

- Compute the average
  - again: k=5
  - average = \( \frac{18+20+21+22+21}{5} \)
  - prediction: \( \hat{y} = 20.4°C \)
Choosing a Good Value for K

- All considerations from KNN classification also apply to KNN regression
  - If k is too small, the result is sensitive to noise points
  - If k is too large, the neighborhood may include points from other classes
- Rule of thumb: Test k values between 1 and 10.
K-Nearest-Neighbor Regression in RapidMiner
Numeric Predictions are Added to the Dataset
3. Model Evaluation

Central Question:

How good is a model at predicting the dependent variable?

3.1 Methods for Model Evaluation

- How to obtain reliable estimates?

3.2 Metrics for Model Evaluation

- How to measure the performance of a regression model?
3.1 Methods for Model Evaluation

- The same considerations apply as for classification
  - X-Validation: 10-fold (90% for training, 10% for testing in each iteration)
  - Split Validation: 80% random share for training, 20% for testing

- Estimating performance metrics in RapidMiner
  - X-Validation Operator + Regression Performance Operator
3.2 Metrics for Model Evaluation

Recap: Which metrics did we use for classification?

Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Correct \ predictions}{All \ predictions}

\begin{array}{|c|c|c|}
\hline
& \text{PREDICTED CLASS} & \\
\hline
\text{ACTUAL CLASS} & \text{Class=Yes} & \text{Class=No} \\
\hline
\text{Class=Yes} & TP & FN \\
& 25 & 4 \\
\hline
\text{Class=No} & FP & TN \\
& 6 & 15 \\
\hline
\end{array}

Acc = \frac{25 + 15}{25 + 15 + 6 + 4} = 0.80
Metrics for Regression Model Evaluation

- **Mean Absolute Error (MAE)** computes the average deviation between predicted value $p_i$ and the actual value $r_i$

  \[
  MAE = \frac{1}{n} \sum_{i=1}^{n} |p_i - r_i|
  \]

- **Mean Square Error (MSE)** is similar to MAE, but places more emphasis on larger deviation

  \[
  MSE = \frac{1}{n} \sum_{i=1}^{n} (p_i - r_i)^2
  \]

- **Root Mean Square Error (RMSE)** has similar scale as MAE and places more emphasis on larger deviation

  \[
  RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (p_i - r_i)^2}
  \]
Metrics for Regression Model Evaluation

- **Pearson's Correlation Coefficient (PCC)**
  - Scores well if
    - high actual values get high predictions
    - low actual values get low predictions
  
  \[
PCC = \frac{\sum_{\text{all examples}} (\text{pred} - \overline{\text{pred}}) \times (\text{act} - \overline{\text{act}})}{\sqrt{\sum_{\text{all examples}} (\text{pred} - \overline{\text{pred}})^2} \times \sqrt{\sum_{\text{all examples}} (\text{act} - \overline{\text{act}})^2}}
\]

- **R Squared: Coefficient of Determination**
  - measures the part of the variation in the dependent variable y that is predictable from the explanatory variables X.
  - \( R^2 = 1 \) : Perfect model as y can be completely explained from X.
  - called Squared Correlation in RapidMiner

\[
R^2 = \frac{\sum_{i=1}^{n} (\hat{y}_i - \overline{y})^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2}
\]
4. Regression Trees

- The basic idea of how to learn and apply decision trees can also be used for regression.

- Differences:
  1. Splits are selected by maximizing the MSE reduction (not GINI or InfoGain)
  2. Predict the average value of the training examples in a specific leaf.
Pre-pruning parameters determine how closely the tree fits the training data.

- e.g. max_depth parameter
Overfitted Regression Tree

The learning algorithm uses available depth to cover strongest outliers.
5. Linear Regression

Assumption of Linear Regression: The target variable $y$ is (approximately) linearly dependent on explanatory variables $X$

- for visualization: we use one variable $x$ (simple linear regression)
- in reality: vector $X = x_1...x_n$ (multiple linear regression)

$$\hat{y}(X) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$
Fitting a Regression Function

Least-Squares Approach: Find the weight vector $W = (w_0, w_1, \ldots, w_n)$ that minimizes the sum of squared error (SSE) for all training examples.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}(X_i, W))^2$$

Errors: Differences between estimated value and value in training data, also called residuals.
Ridge Regularization

- Variation of least squares approach which tries to avoid overfitting by keeping the weights $W$ small.
- Ridge regression cost function to minimize

$$C(W) = \text{MSE}(W) + \alpha \frac{1}{2} \sum_{i=1}^{n} w_i^2$$

- $\alpha = 0$ : Normal least squares regression
- $\alpha = 100$ : Strongly regularized flat curve
- Example of overfitting due to biased training data
Feature Selection

- Question: Do all explanatory variables $X$ help to explain $y$ or is only a subset of the variables useful?
- Problem 1: **Highly correlated variables** (e.g. height in cm and inch)
  - Weights are meaningless and one variable should be removed for the better interpretability of the weights.
- Problem 2: **Insignificant variables** (e.g. the weather for stock prices)
  - Uncorrelated variables get $w=0$ or relatively small weights assigned.
  - Question for variables having small weights: Is the variable still useful or did it get the weight by chance due to biased training data?
  - Answer: Statistical test with null-hypothesis “$w=0$ as variable is insignificant”
    - **t-stat**: number of standard deviations that $w$ is away from 0.
      - High $t$-stat $\Rightarrow$ Variable is significant as it is unlikely that weight is assigned by chance.
    - **p-value**: Probability of wrongly rejecting the null-hypothesis
      - $p$-value close to zero $\Rightarrow$ Variable is significant.
  - See: James, Witten, et al.: An Introduction to Statistical Learning. Chapter 3.1.2
Linear Regression in RapidMiner

- Choose feature selection method
- Correlated features
- Ridge (>0) or least squares (0) regression

### Feature Quality Code

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Coefficient</th>
<th>Std. Error</th>
<th>Std. Coeff.</th>
<th>Tolerance</th>
<th>t-Stat</th>
<th>p-Value</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>85.440</td>
<td>5.692</td>
<td>0.463</td>
<td>1.000</td>
<td>15.009</td>
<td>0</td>
<td>****</td>
</tr>
<tr>
<td>a2</td>
<td>127.001</td>
<td>5.701</td>
<td>0.689</td>
<td>0.998</td>
<td>22.276</td>
<td>0</td>
<td>****</td>
</tr>
<tr>
<td>a3</td>
<td>68.118</td>
<td>5.730</td>
<td>0.369</td>
<td>1.000</td>
<td>11.888</td>
<td>0</td>
<td>****</td>
</tr>
<tr>
<td>a4</td>
<td>10.302</td>
<td>5.728</td>
<td>0.056</td>
<td>1.000</td>
<td>1.799</td>
<td>0.074</td>
<td>*</td>
</tr>
<tr>
<td>a5</td>
<td>-12.452</td>
<td>5.723</td>
<td>-0.068</td>
<td>0.999</td>
<td>-2.176</td>
<td>0.031</td>
<td>**</td>
</tr>
<tr>
<td>(Intercept)</td>
<td>180.418</td>
<td>5.671</td>
<td>?</td>
<td>?</td>
<td>31.814</td>
<td>0</td>
<td>****</td>
</tr>
</tbody>
</table>
**Interpolation vs. Extrapolation**

- **Training data:**
  - weather observations for current day
  - e.g., temperature, wind speed, humidity, …
  - target: temperature on the next day
  - training values between -15°C and 32°C

- **Interpolating regression**
  - only predicts values from the interval [-15°C, 32°C]

- **Extrapolating regression**
  - may also predict values outside of this interval
Interpolation vs. Extrapolation

- Interpolating regression is regarded as “safe”
  - i.e., only reasonable/realistic values are predicted

http://xkcd.com/605/
Interpolation vs. Extrapolation

- Sometimes, however, only extrapolation is interesting
  - how far will the sea level have risen by 2050?
  - how much will the temperature rise in my nuclear power plant?
Linear Regression vs. K-NN Regression

- Linear regression extrapolates
- K-NN and regression trees interpolate

we want a prediction for that $x$
Linear Regression Examples
…but what about non-linear Problems?

- One possibility is to apply transformations to the explanatory variables $X$ within the regression function.
  - e.g. log, exp, square root, square, etc.
    - example: $\hat{y} = \omega_0 + \omega_1 x_1^2 + \omega_2 x_2^2$
  - polynomial transformation
    - example: $\hat{y} = \omega_0 + \omega_1 x + \omega_2 x^2 + \omega_3 x^3$
- This allows use of linear regression techniques to fit much more complicated non-linear datasets.
6. Polynomial Regression

\[ \hat{y}(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]

- widely used extension of linear regression
- can also be fitted using the least squares method
- has tendency to over-fit training data for large degrees M
- Workarounds:
  1. decrease M
  2. increase amount of training data
Polynomial Regression in RapidMiner

Y = 100.000 \times \text{att1}^{3.000} - 16.644 \times \text{att1}^{2.000} - 31.516 \times \text{att2}^{2.000} + 58.019 \times \text{att2}^{1.000} + 18.014 \times \text{att3}^{1.000} - 9.555 \times \text{att3}^{2.000} - 7.056

How often may each explanatory variable appear in the function.

Maximal degree to be used in the function

Boundaries for weights W

Example of a function learned using the parameters shown above.
Overfitting often happens in sparse regions.

- left and right side of green line
- workaround: Local Regression
Local Regression

- Assumption: non-linear problems are approximately linear in local areas
- Idea: use linear regression locally for the data point at hand (lazy learning)
- A combination of
  - k nearest neighbors
  - linear regression
- Given a data point
  1. retrieve the k nearest neighbors
  2. learn a regression model using those neighbors
  3. use the learned model to predict y value
Local Regression in Rapidminer

- **Parameters**
  - **degree**: 2
  - **ridge factor**: 1.0E-9
  - **use robust estimation**: unchecked
  - **use weights**: checked
  - **numerical measure**: EuclideanDistance
  - **neighborhood type**: Fixed Number
  - **k**: 5
  - **smoothing kernel**: Triweight

- **Local Polynomial Regression**

  - **Maximal degree to be used**: Set to 1 for local linear regression
  - **Distance function for determining neighborhood**
  - **Neighborhood size**
Discussion of Local Regression

- **Advantage**: fits non-linear models well
  - good local approximation
  - often better than pure k-NN

- **Disadvantage**
  - slow at runtime
  - for each test example:
    - find k nearest neighbors
    - compute a local model
Recap: How did we use ANNs for classification?

\[
Y = I(0.3 X_1 + 0.3 X_2 + 0.3 X_3 - 0.4 > 0)
\]

where \( I(z) = \begin{cases} 
1 & \text{if } z \text{ is true} \\
0 & \text{otherwise}
\end{cases} \)
**ANNs for Regression**

- The function $I(z)$ was used to separate the two classes:

\[ Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0) \]

where \( I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases} \)

- However, we may simply use the inner formula to predict a numerical value (between 0 and 1):

\[ \hat{Y} = 0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 \]

- What has changed:
  - we do not use a cutoff for 0/1 predictions, but leave the numbers as they are

- Training examples:
  - pairs \((y_i, X_i)\), \(\hat{y}_i\) = scalar dependent variable, \(X_i\) = vector of explanatory variables
Artificial Neural Networks for Regression

- Given that our formula is of the form
  \[ \hat{Y} = 0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 \]
- we can learn only linear models
  - i.e., the target variable is a linear combination the input variables.
- More complex regression problems can be approximated
  - by using multiple hidden layers
  - this allows for arbitrary functions
- Deep ANNs take this idea further by
  - employing millions of neurons
  - arranging them into specific network topologies
  - more on Deep Learning in Data Mining II
- With all ANNs be cautious about overfitting!
9. Time Series Forecasting

- **A time series** is a series of data points indexed in time order.
  - Examples: Stock market prices, ocean tides, birth rates, temperature

- **Forecasting**: Given a time series, predict data points that continue the series into the future.
  - Explicitly deals with time, which is not explicitly considered by other regression techniques.
  - Aims to predict future values of the same variable

- **Approaches:**
  1. Data-driven: Smoothing
  2. Model-Driven:
     1. Component models of time series
     2. Other regression techniques
**Smoothing**

**Simple Moving Average (SMA)**
- average of the last $n$ values, as more recent values might matter more

$$m_{\text{MA}}^{(n)}(t) = \frac{1}{n} \sum_{i=0}^{n-1} x(t - i)$$

**Exponential Moving Average (EMA)**
- exponentially decrease weight of older values

$$m_{\text{EMA}}^{(n)}(t) = \alpha \cdot x(t) + (1 - \alpha) \cdot m_{\text{EMA}}^{(n)}(t - 1)$$

DAX: red = SMA(38 days), yellow = SMA(200 days)
Component Models of Time Series

Assume time series to consist of four components:

1. Long-term trend ($T_t$)
2. Cyclical effect ($C_t$)
3. Seasonal effect ($S_t$)
4. Random variation ($R_t$)

Series = $T_t + C_t + S_t + R_t$

Details in Data Mining II
Windowing

  - either classification or regression
  - by only taking the last k days into account

- Example: Weather forecasting
  - using the weather from the three previous days
  - Possible model:
    - sunny, sunny, sunny → sunny
    - sunny, cloudy, rainy → rainy
    - sunny, cloudy, cloudy → rainy

- Assumption:
  - only the last days matter for the forecast
  - the older past is irrelevant

Details in Data Mining II
### Windowing in RapidMiner

#### RuleModel

if Temperature-0 ≤ 21 and Temperature-2 > 20.500 then sunny (0 / 0 / 62)
if Temperature-1 > 21 and Temperature-1 ≤ 27 then cloudy (81 / 0 / 0)
if Weather-2 = cloudy then rainy (0 / 62 / 0)
else sunny (0 / 0 / 18)

Correct: 223 out of 223 training examples.
Summary

- Regression
  - predict numerical values instead of classes

- Performance measuring
  - absolute or squared error, Pearson correlation, R squared, …

- Methods
  - k Nearest Neighbors
  - Regression trees
  - Artificial neural networks
  - Linear regression
  - Polynomial regression
  - Local regression
  - Time Series Prediction
  - …
Final Remarks: Learning Classification and Regression Models

- We want to learn regression as well as classification models that generalize well to **unseen data**.

- The generalization error of any model can be understood as a sum of three errors:
  
  1. **Bias**: Part of the generalization error due to wrong model complexity.
     - simple model (e.g. linear regression) used for complex real world phenomena. Model thus underfits the training and test data.
  
  2. **Variance**: Part of the generalization error due to a model’s excessive sensitivity to small variations in the training data.
     - Models with high degree of freedom/flexibility (like polynomial regression models or deep trees) are likely to overfit the training data.
  
  3. **Irreducible Error**: Error due to noisiness of the data itself.
     - To reduce this part of the error the training data needs to be cleansed (by removing outliers, fixing broken sensors)
The Bias/Variance-Tradeoff

- Left: Three models with different flexibility trying to fit a function
  - Orange: Linear regression. Green, blue: Polynomials of different degrees
- Right: Training Error (gray) and test error (red) in relation to model flexibility.
Method Selection and Parameter Tuning

We try to find the ideal flexibility (bias/variance-tradeoff) by

1. Testing different methods
   - Linear regression, polynomial regression, …
   - Decision Trees, ANNs, Naïve Bayes, …

2. Testing different hyperparameters
   - degree of polynomial, ridge
   - depth of tree, min examples branch
   - number of hidden layers

But we have two more options:

1. Increase the amount of training data
2. Cleanse the training data
Learning Curves

- Visualize the training error and test error for different training set sizes.

Underfitting Model (e.g. Linear regression)

- For overfitting models, the gap between training and test error can often be narrowed by adding more training data.

Overfitting Model (e.g. high-degree polynomial)

- Thus, having more training data also allows us to use more flexible models, e.g. Deep Learning (Details in Data Mining II)

High training and test errors due to bias

Smaller errors and gap between training and test error
Literature

- Solving practical regression tasks using:
  • RapidMiner: Kotu: Predictive Analytics - Chapter 5, 10
  • Python: Geron: Hands-on Machine Learning - Chapter 4

- Deeper coverage including theoretical background
  • James, Witten, et al.: An Introduction to Statistical Learning
    Chapters 3, 7, 8