Regression

- Classification
  - covered in Data Mining I
  - predict a label from a finite collection
  - e.g., true/false, low/medium/high, ...

- Regression
  - predict a \textit{numerical} value
  - from a possibly infinite set of possible values

- Examples
  - temperature
  - sales figures
  - stock market prices
  - ...
Contents

• A closer look at the problem
  – e.g., interpolation vs. extrapolation
  – measuring regression performance

• Revisiting classifiers we already know
  – which can also be used for regression

• Adoption of classifiers for regression
  – model trees
  – support vector machines
  – artificial neural networks

• Other methods of regression
  – linear regression
  – isotonic regression
  – local regression
The Regression Problem

• Classification
  – algorithm “knows” all possible labels, e.g. yes/no, low/medium/high
  – all labels appear in the training data
  – the prediction is always one of those labels

• Regression
  – algorithm “knows” some possible values, e.g., 18°C and 21°C
  – prediction may also be a value not in the training data, e.g., 20°C
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

![Diagram of interpolation vs. extrapolation]

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

- Sometimes, however, only extrapolation is interesting
  - how far will the sea level have risen by 2050?
  - will there be a nuclear meltdown in my power plant?

http://i1.ytimg.com/vi/FVfiujbGLfM/hqdefault.jpg
Baseline Prediction

- For classification: predict most frequent label
- For regression: predict average value
  - or median
  - or mode
  - in any case: only interpolating regression
- often a strong baseline

http://xkcd.com/937/
k Nearest Neighbors Revisited

• Problem
  – find out what the weather is in a certain place
  – where there is no weather station
  – how could you do that?
k Nearest Neighbors Revisited

• 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 for 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: $k=5$
  – $(18+20+21+22+21)/5$
  – prediction: 20.4°C

• Only interpolating regression!
k Nearest Neighbor Regression in RapidMiner

ExampleSet (100 examples, 1 special attribute, 5 regular attributes)

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Type</th>
<th>Statistics</th>
<th>Range</th>
<th>Missings</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>label</td>
<td>real</td>
<td>(\text{avg} = 74.105 +/− 42)</td>
<td>([-873.112; 968.407])</td>
<td>0</td>
</tr>
<tr>
<td>att1</td>
<td>regular</td>
<td>real</td>
<td>(\text{avg} = 0.650 +/− 5.8)</td>
<td>([-9.645; 9.843])</td>
<td>0</td>
</tr>
<tr>
<td>att2</td>
<td>regular</td>
<td>real</td>
<td>(\text{avg} = 0.460 +/− 6.3)</td>
<td>([-9.939; 9.798])</td>
<td>0</td>
</tr>
<tr>
<td>att3</td>
<td>regular</td>
<td>real</td>
<td>(\text{avg} = 0.079 +/− 6.0)</td>
<td>([-9.661; 9.795])</td>
<td>0</td>
</tr>
<tr>
<td>att4</td>
<td>regular</td>
<td>real</td>
<td>(\text{avg} = -1.355 +/− 5.2)</td>
<td>([-9.837; 9.612])</td>
<td>0</td>
</tr>
<tr>
<td>att5</td>
<td>regular</td>
<td>real</td>
<td>(\text{avg} = -0.075 +/− 5.7)</td>
<td>([-9.968; 9.973])</td>
<td>0</td>
</tr>
</tbody>
</table>
Recap: measuring performance for classification:

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

If we use the numbers 0 and 1 for class labels, we can reformulate this as

\[
\sum_{\text{all examples}} |\text{predicted} - \text{actual}| \quad \text{Accuracy} = 1 - \frac{\sum_{\text{all examples}} |\text{predicted} - \text{actual}|}{N}
\]

Why?

- the nominator is the sum of all correctly classified examples
  - i.e., the difference of the prediction and the actual label is 0
- the denominator is the total number of examples
Mean Absolute Error

- We have
  \[
  \text{Accuracy} = 1 - \frac{\sum_{\text{all examples}} |\text{predicted} - \text{actual}|}{N}
  \]

- For an arbitrary numerical target, we can define
  \[
  \text{MAE} = \frac{\sum_{\text{all examples}} |\text{predicted} - \text{actual}|}{N}
  \]

- Mean Absolute Error
  - intuition: how much does the prediction differ from the actual value on average?
(Root) Mean Squared Error

• Mean Squared Error:

\[
MSE = \frac{\sum_{\text{all examples}} |\text{predicted} - \text{actual}|^2}{N}
\]

• Root Mean Squared Error:

\[
RMSE = \sqrt{\frac{\sum_{\text{all examples}} |\text{predicted} - \text{actual}|^2}{N}}
\]

• More severe errors are weighted higher by MSE and RMSE
Correlation

- Pearson's correlation coefficient
- Scores well if
  - high actual values get high predictions
  - low actual values get low predictions
- Caution: PCC is scale-invariant!
  - actual income: $1, $2, $3
  - predicted income: $1,000, $2,000, $3,000
    → PCC = 1

\[
PCC = \frac{\sum_{\text{all examples}} (\text{pred} - \overline{\text{pred}}) \times (\text{act} - \overline{\text{act}})}{\sqrt{\sum_{\text{all examples}} (\text{pred} - \overline{\text{pred}})^2} \times \sqrt{\sum_{\text{all examples}} (\text{act} - \overline{\text{act}})^2}}
\]
Linear Regression

• Assumption: target variable $y$ is (approximately) linearly dependent on attributes
  – for visualization: one attribute $x$
  – in reality: $x_1...x_n$
Linear Regression

- Target: find a linear function \( f(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n \)
  - so that the error is minimized
  - i.e., for all examples \((x_1, \ldots, x_n, y)\), \( f(x) \) should be a correct prediction for \( y \)
  - given a performance measure
Linear Regression

• Typical performance measure used: Mean Squared Error
• Task: find $w_0,\ldots,w_n$ so that $\sum_{\text{all examples}} \left( w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n - y \right)^2$ is minimized
• note: we omit the denominator $N$
FIGURE 3.1. Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of $X$ that minimizes the sum of squared residuals from $Y$. 
Linear Regression vs. k-NN Regression

- Recap: Linear regression extrapolates, k-NN interpolates

We want a prediction for that $x$.

Prediction of linear regression

Prediction of 3-NN

Three nearest neighbors

We want a prediction for that $x$. 
Linear Regression Examples
...but what about Non-linear Problems?
Isotonic Regression

• Special case:
  – Target function is *monotonous*
    • i.e., \(f(x_1) \leq f(x_2)\) for \(x_1 < x_2\)
  – For that class of problem, efficient algorithms exist

• Simplest: Pool Adjacent Violators Algorithm (PAVA)
Isotonic Regression

- Identify adjacent violators, i.e., \( f(x_i) > f(x_{i+1}) \)
- Replace them with new values \( f'(x_i) = f'(x_{i+1}) \) so that sum of squared errors is minimized
  - ...and *pool* them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left
Isotonic Regression

- Identify adjacent violators, i.e., \( f(x_i) > f(x_{i+1}) \)
- Replace them with new values \( f'(x_i) = f'(x_{i+1}) \) so that sum of squared errors is minimized
  - ...and *pool* them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left
Isotonic Regression

- Identify adjacent violators, i.e., \( f(x_i) > (x_{i+1}) \)
  - Replace them with new values \( f'(x_i) = f'(x_{i+1}) \)
    - so that sum of squared errors is minimized
      - ...and pool them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left
Isotonic Regression

- Identify adjacent violators, i.e., $f(x_i) > (x_{i+1})$
- Replace them with new values $f'(x_i) = f'(x_{i+1})$
  so that sum of squared errors is minimized
  - ...and *pool* them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left
Isotonic Regression

- After all points are reordered so that $f'(x_i)=f'(x_{i+1})$ holds for every $i$
  - Connect the points with a piecewise linear function
Isotonic Regression

• Comparison to the original points
  – Plateaus exist where the points are not monotonous
  – Overall, the mean squared error is minimized

• Operator in RapidMiner: from the Weka Extension
...but what about non-linear, *non-monotonous* Problems?
The attributes $X$ for linear regression can be:

- Original attributes $X$
- Transformation of original attributes, e.g. log, exp, square root, square, etc.
- Polynomial transformation
  - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
- Basis expansions
- Interactions between variables
  - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques to fit much more complicated non-linear datasets.
Example with Polynomially Transformed Attributes

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

- Find hyperplane **maximizes** the margin => B1 is better than B2
Linear Regression and SVM

- Linear Regression
  - find a linear function that *minimizes* the distance to data points w.r.t. the attribute to predict

- Support Vector Machine
  - find a linear function that *maximizes* the distance to data points (from different classes)

- Both problems are similar
  - hence, many SVMs also support regression
Support Vector Regression

• Maximum margin hyperplane only applies to classification
• However, idea of support vectors and kernel functions can be used for regression
• Basic method same as in linear regression: want to minimize error
  − Difference A: ignore errors smaller than $\varepsilon$ and use absolute error instead of squared error
  − Difference B: simultaneously aim to maximize flatness of function
• User-specified parameter $\varepsilon$ defines “tube”
Examples

\( \varepsilon = 1 \)

\( \varepsilon = 2 \)

\( \varepsilon = 0.5 \)
Local Regression

- Assumption: non-linear problems are approximately linear in local areas
  - idea: use linear regression locally
  - only for the data point at hand (lazy learning)
Local Regression

• A combination of
  – k nearest neighbors
  – local regression

• Given a data point
  – retrieve the k nearest neighbors
  – compute a regression model using those neighbors
  – locally weighted regression:
    uses distance as weight for error computation
Local Regression

• Advantage: fits non-linear models well
  – good local approximation
  – often more exact than pure k-NN

• Disadvantage
  – runtime
  – for each test example:
    • find k nearest neighbors
    • compute a local model
Combining Decision Trees and Regression

- Idea: split data first so that it becomes “more linear”
- example: fuel consumption by car weight
Combining Decision Trees and Regression

- Idea: split data first so that it becomes “more linear”
- example: fuel consumption by car weight

![Graph of fuel consumption by car weight with two lines for benzine and diesel](image)
Combining Decision Trees and Regression

- Observation:
  - by cleverly splitting the data, we get more accurate linear models

- Regression trees:
  - decision tree for splitting data
  - constants as leaves

- Model trees:
  - more advanced
  - linear functions as leaves

\[
y = 0.005x + 1 \quad \text{if} \quad \text{fuel type} = \text{diesel} \\
y = 0.01x + 2 \quad \text{if} \quad \text{fuel type} = \text{benzine}
\]
Regression Trees

- Differences to classification decision trees:
  - Splitting criterion: minimize intra-subset variation
  - Termination criterion: standard deviation becomes small
  - Pruning criterion: based on numeric error measure
  - Prediction: Leaf predicts average class values of instances

- Easy to interpret

- Resulting model: piecewise constant function
Model Trees

• Build a regression tree
  – For each leaf ⇒ learn linear regression function
• Need linear regression function at each node
• Prediction: go down tree, then apply function
• Resulting model: piecewise linear function
Local Regression and Regression/Model Trees

- Assumption: non-linear problems are approximately linear in local areas
  - idea: use linear regression locally
  - only for the data point at hand (lazy learning)
Building the Tree

• Splitting: standard deviation reduction
  \[ SDR = sd(T) - \sum_i |\frac{T_i}{T}| \times sd(T_i) \]

• Termination:
  – Standard deviation < 5% of its value on full training set
  – Too few instances remain (e.g. < 4)

• Pruning:
  – Proceed bottom up:
    • Compute LR model at internal node
    • Compare LR model error to error of subtree
    • Prune if the subtree's error is not significantly smaller
  – Heavy pruning: single model may replace whole subtree
Model Tree Learning Illustrated

- Standard deviation of complete value set: 3.08
- Standard deviation of two subsets after split $x > 9$: 1.22
  - Standard deviation reduction: 1.86
  - This is the best split
Model Tree Learning Illustrated

- Assume that we have split further (min. 4 instances per leaf)
  - Standard deviation reduction for the new splits is still 0.57
- Resulting model tree:

  - The error of the inner nodes is the same as for the root nodes → prune
• Assume that we have split further (min. 4 instances per leaf)
  – Standard deviation reduction for the new splits is still 0.57
• Resulting model tree:
  \[ x < 9 \]
  \[ y = 0.5x \]
  \[ y = 0.59x - 0.29 \]
  \[ y = 0.5x + 1 \]

• The error of the root node is larger than that of the leaf nodes → keep leaf nodes
Model Tree Learning Illustrated

\[ x < 9 \]

- \[ y = 0.5x \]
- \[ y = 0.5x + 1 \]
Rules from Model Trees

• Recap: PART algorithm generates classification rules by building partial decision trees
• M5Rules uses the same method to build rule sets for regression
  – Use model trees instead of decision trees
  – Use variance instead of entropy to choose node to expand when building partial tree
• Rules will have linear models on right-hand side
Artificial Neural Networks Revisited

Output $Y$ is 1 if at least two of the three inputs are equal to 1.
Artificial Neural Networks Revisited

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

where \( I (z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases} \)
• This final function was used to separate two classes:

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

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

• However, we may simply use it to predict a numerical value (between 0 and 1) by changing it to:

\[ Y = 0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 \]
Artificial Neural Networks for Regression

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

• Training examples:
  – attribute vectors – not with a class label, but numerical target

• Error measure:
  – Not classification error, but mean squared error
Artificial Neural Networks for Regression

\[ Y = 0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 \]
Artificial Neural Networks for Regression

• Given that our target formula is of the form

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

• we can learn only linear problems
  – i.e., the target variable is a linear combination the input variables

• More complex regression problems can be approximated
  – by combining several perceptrons
  – this allows for arbitrary functions
Summary

- Regression
  - predict numerical values instead of classes
- Performance measuring
  - absolute or relative error, correlation, …
- Methods
  - k nearest neighbors
  - linear regression
  - isotonic regression
  - SVMs
  - model trees
  - artificial neural networks
Questions?