Regression

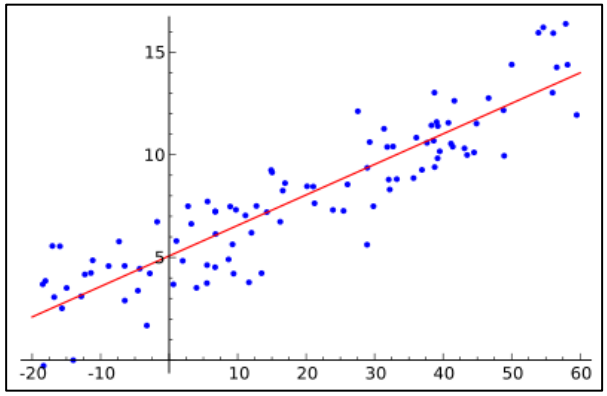
IE500 Data Mining

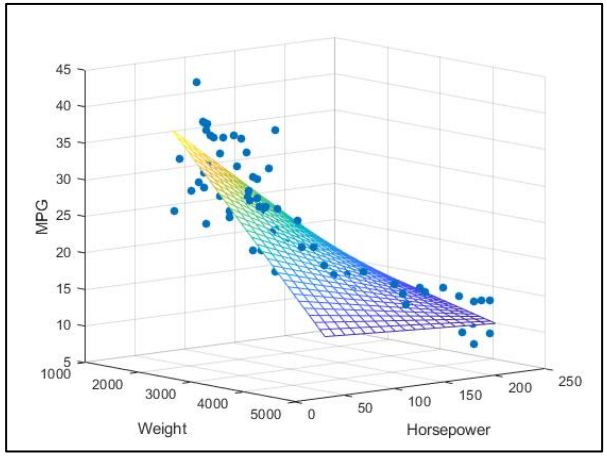
UNIVERSITY OF MANNHEIM
Data and Web Science Group

What is Regression?

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Regression
  - Goal: **predict a numerical value**
  - From a possibly infinite set of possible values
  - The predicted variable is called **dependent and is denoted** $\hat{y}$
  - The other variables are called **explanatory variables or independent variables** denoted $X = x_1, x_2, \dots, x_n$





The Regression Problem

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Examples
  - Weather Forecasting
    - Dependent: wind speed
    - Explanatory variables: temperature, humidity, air pressure change
  - House Market
    - Dependent: price of a house
    - Explanatory variables: rooms, distance to public transport, size of garden
- Regression vs. Classification
  - 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

Switching from Classification to Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Many classification approaches
  can also be used for regression (with modifications)

- In the following slides each approach will be discussed
  - K-Nearest-Neighbors -> K-Nearest-Neighbors Regression
  - Decision Trees -> Regression Tree, Model Tree
  - Artificial Neural Networks (ANNs) -> ANNs for Regression

Outline

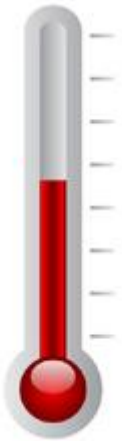
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- KNN for Regression
- Evaluation Metrics
- Regression Trees, Model Trees
- Linear Regression, Ridge / Lasso Regression
- Isotonic Regression
- Polynominal Regression
- Local Regression
- ANNs for Regression
- The Bias/Variance-Tradeoff

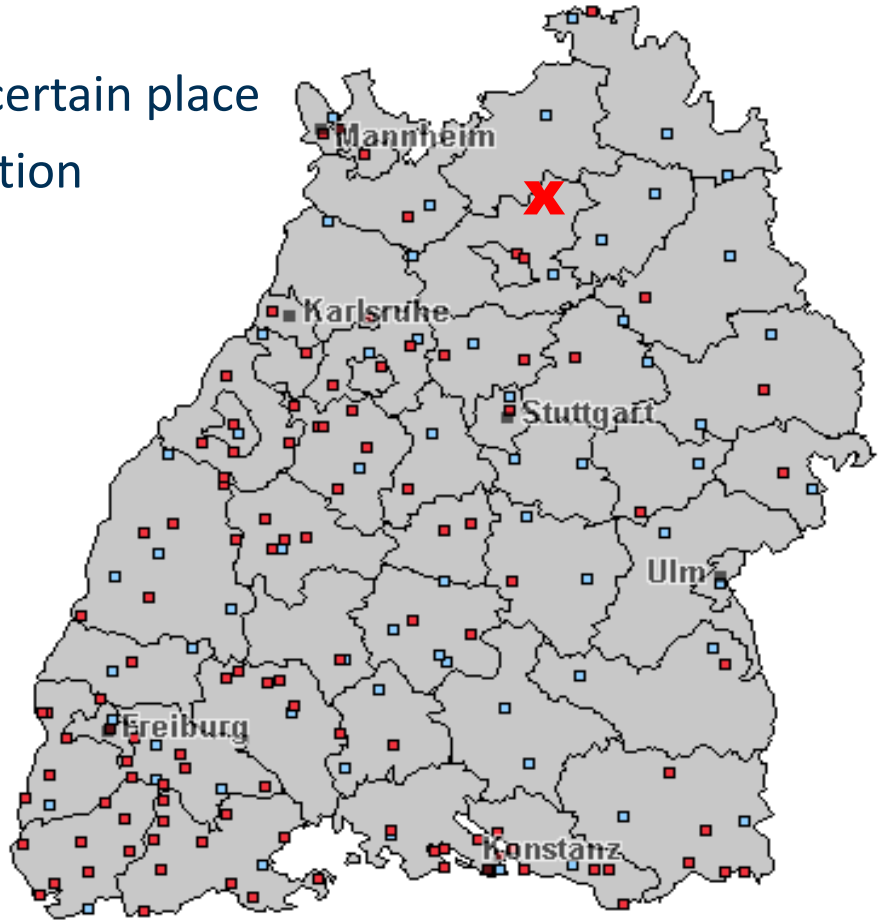
K-Nearest-Neighbors Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Problem
  - predict the **temperature** in a certain place
  - where there is no weather station
  - how could you do that?

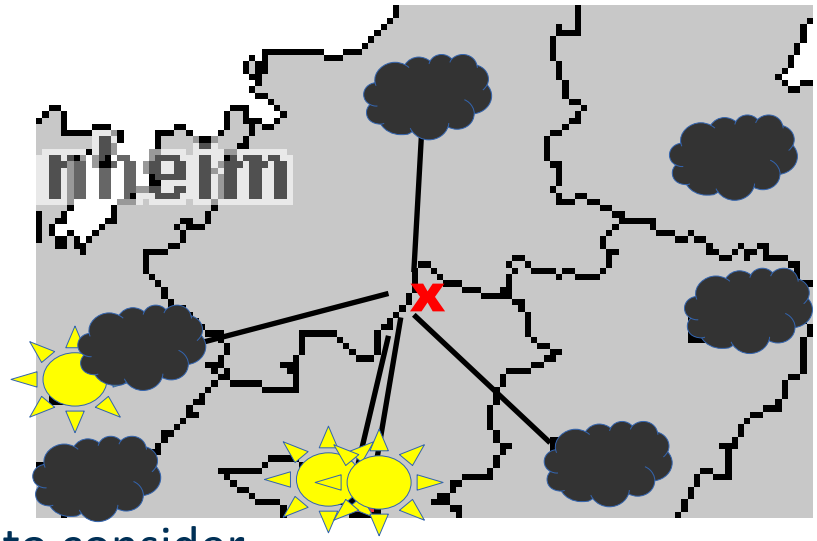


? °C



Recap: K-Nearest-Neighbors Classification

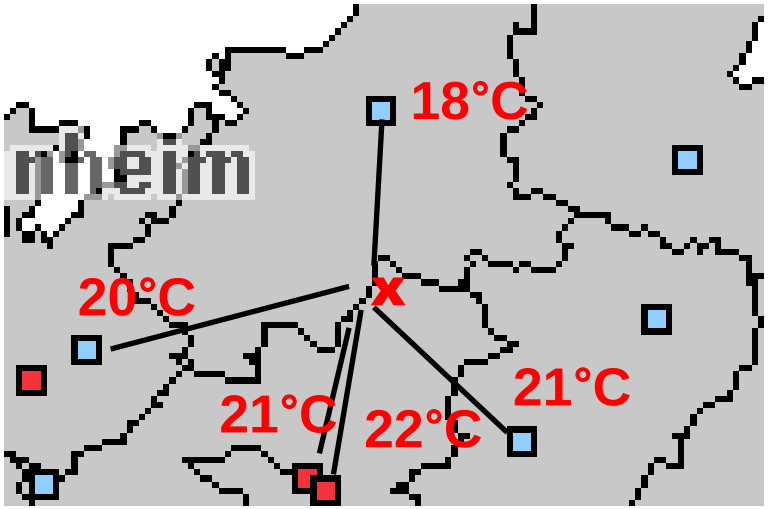
- Idea: **Vote of the nearest stations**
- Example:
  - 3x cloudy
  - 2x sunny
  - Result: cloudy
- Approach is called
  - “k nearest neighbors”
  - where k is the number of neighbors to consider
  - in the example: k=5
  - in the example: “near” denotes geographical proximity



K-Nearest-Neighbors Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- 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 = (18+20+21+22+21) / 5
  - prediction: $\hat{y}$ = 20.4°C



- Can also be weighted by the distance to the nearest neighbors

K-NN Regression in Python

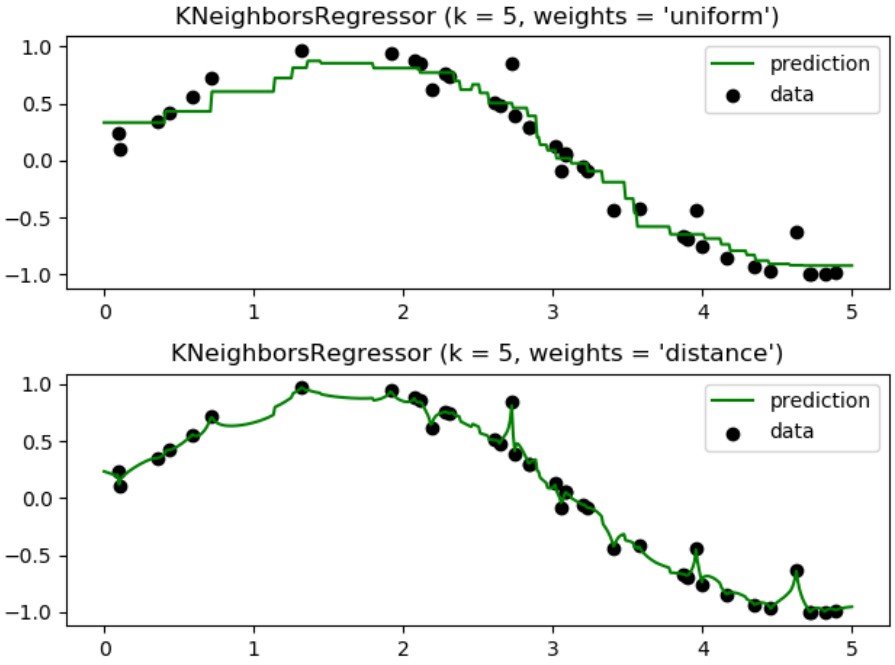
UNIVERSITY OF MANNHEIM
Data and Web Science Group

Python

```python
from sklearn.neighbors import KNeighborsRegressor

# Create and fit a KNN regressor
estimator = KNeighborsRegressor(n_neighbors=15)
estimator.fit(training_set_X, training_dependent_y)

# Make predictions for unseen examples
y_hat = estimator.predict(test_set_X)
```



Evaluation Metrics

UNIVERSITY OF MANNHEIM
Data and Web Science Group

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

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - a_i|$$

Same scale as the domain e.g. temperature

- **Mean Squared Error (MSE)** places more emphasis on larger deviations

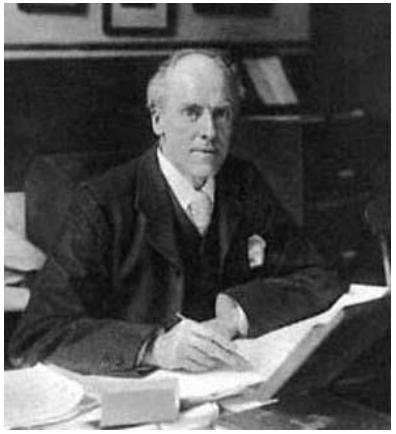
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2$$

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

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2}$$

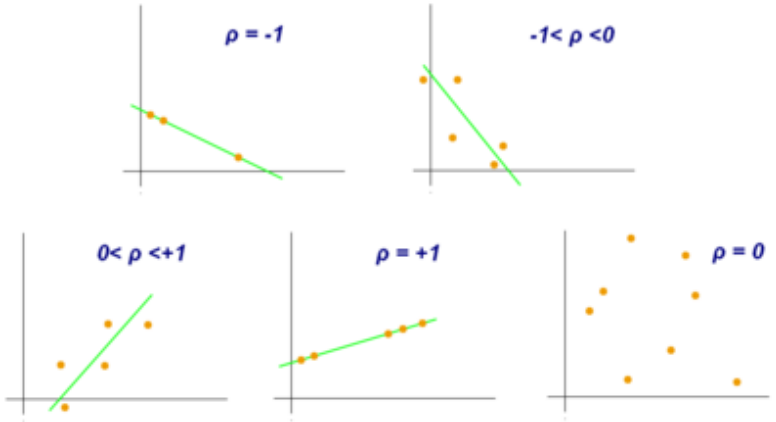
Same scale as the domain e.g. temperature

Evaluation Metrics



- **Pearson's correlation coefficient**
  - Normalized value [-1, 1]
- 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

$$\mathrm{PCC} = \frac{\sum_{i=1}^{n}(p_i - \bar{p}) * (a_i - \bar{a})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2} * \sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2}}$$

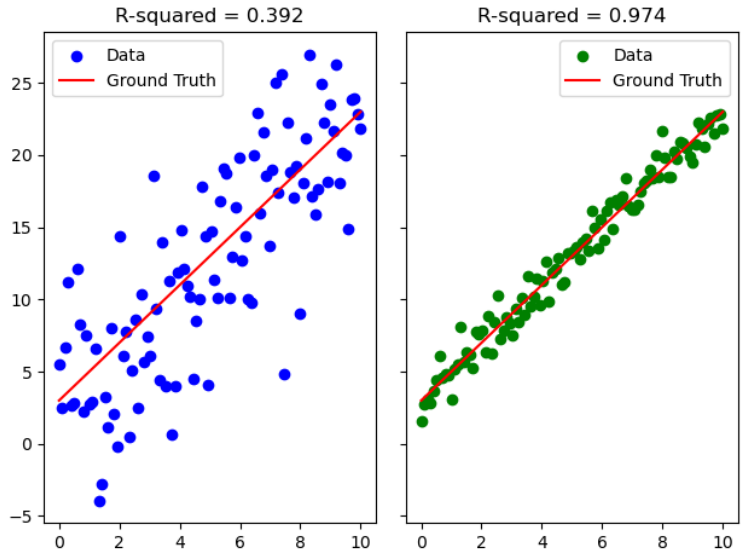


Evaluation Metrics

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- **R Squared** (also called **Coefficient of Determination**)
  - Normalized value [0, 1]
  - Measures the part of the total variation in the dependent variable y that is predictable (explainable) from the explanatory variables X
  - $R^2 = 1$: Perfect model as total variation of y can be completely explained from X

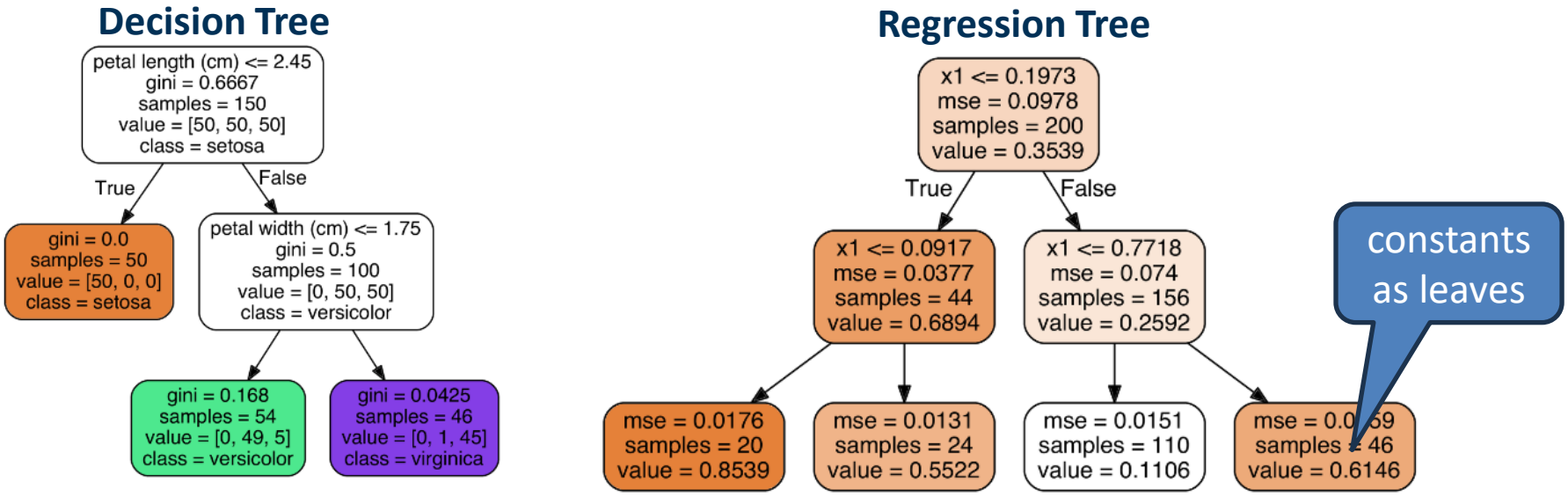
$$R^2 = \frac{\overbrace{\sum_{i=1}^{n}(p_i - \bar{a})^2}^{\text{explained sum of squares}}}{\underbrace{\sum_{i=1}^{n}(a_i - \bar{a})^2}_{\text{total sum of squares}}}$$



Regression Trees

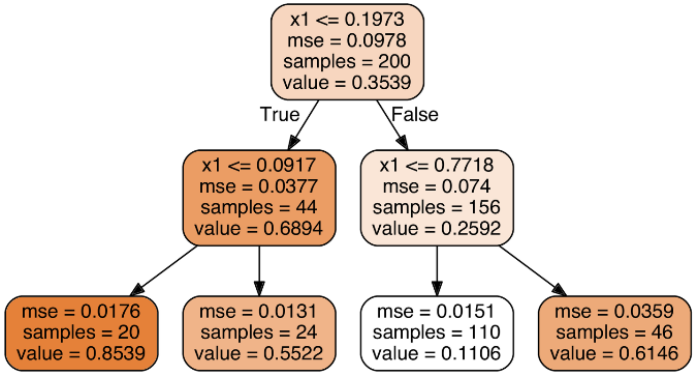
UNIVERSITY OF MANNHEIM
Data and Web Science Group

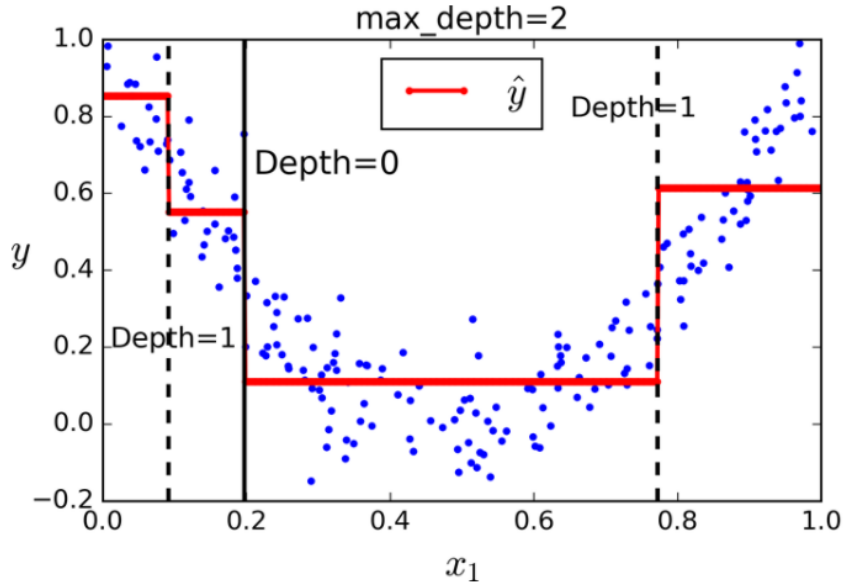
- The basic idea of how to learn and apply decision trees can also be used for regression
- Differences:
  1. Splits are selected by maximizing the MSE reduction (not GINI)
  2. Prediction is average value of the training examples in a specific leaf

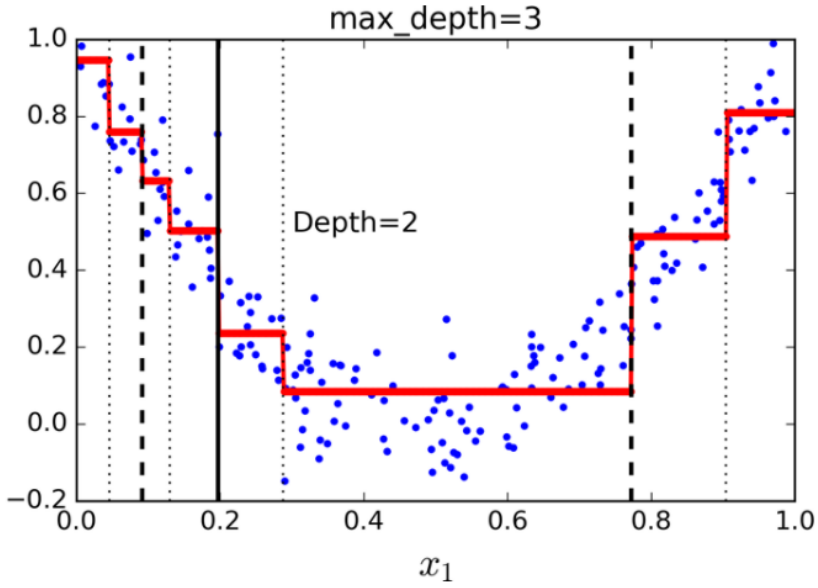


Regression Trees

- Pre-pruning parameters determine how closely the tree fits the training data
  - E.g. max_depth parameter
- Resulting model: piecewise constant function

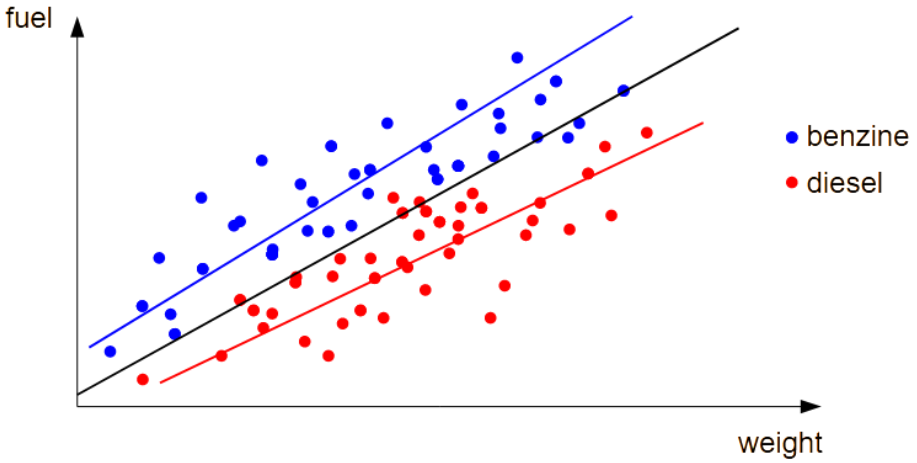


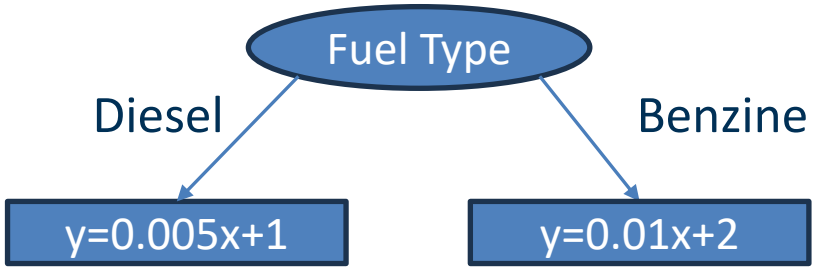




Model Trees

- Idea: split data first so that it becomes “more linear”
  - Example: fuel consumption by car weight
- Instead of a constant in each leave, learn a **linear regression function**
- Prediction: go down tree, then apply function
- Resulting model: piecewise **linear** function

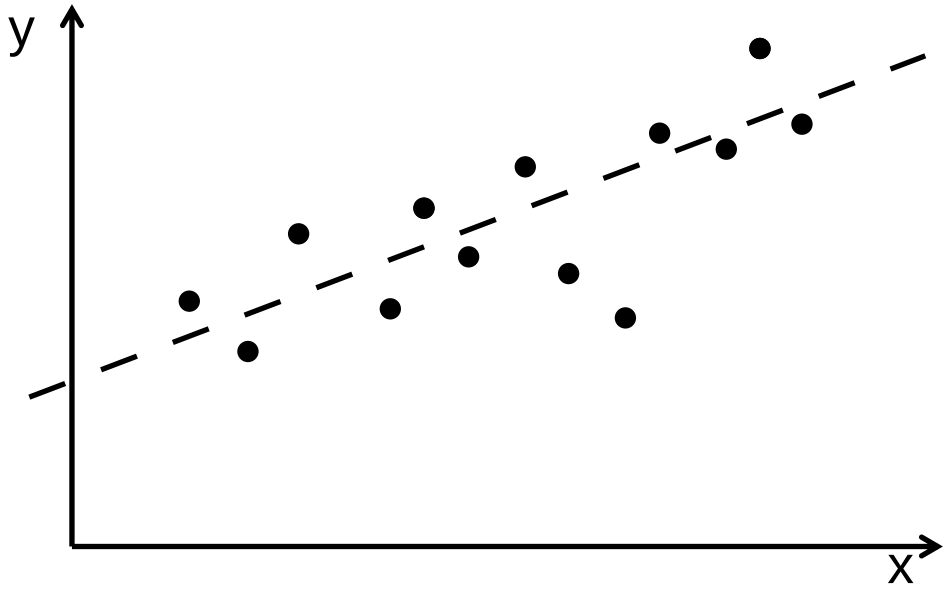




Linear Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

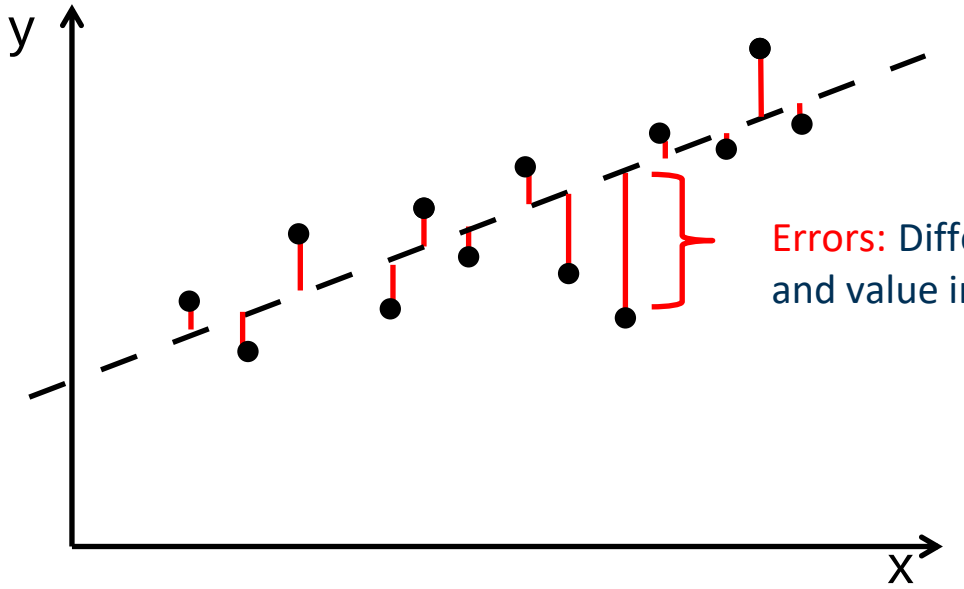
- How to learn a linear regression function?
- Assumption: The target variable y is (approximately) linearly dependent on explanatory variables X
  - For visualization: we use one variable x (simple linear regression)
  - In reality: vector $X = x_1, x_2, \ldots, x_n$ (multiple linear regression)



Fitting a Regression Function

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Least-Squares Approach (simple linear regression):
  - Find the weight vector $W = w_0, w_1$
    of a linear function $f(x) = w_0 + w_1 x_1$
    that minimizes the sum of squared error $SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2$



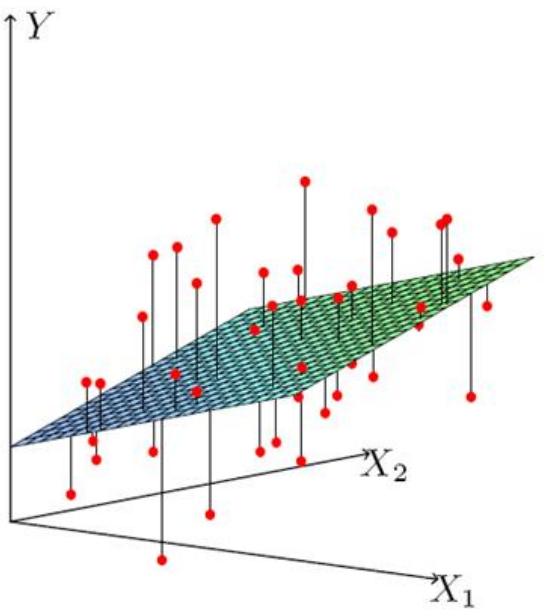
Errors: Differences between estimated value and value in training data, also called residuals.

Fitting a Regression Function

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Least-Squares Approach (multiple linear regression):
  - Find the weight vector $W = w_0, w_1, \ldots, w_n$ of a linear function $\hat{y}(X) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$ that minimizes the sum of squared error

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



Linear Regression and Overfitting

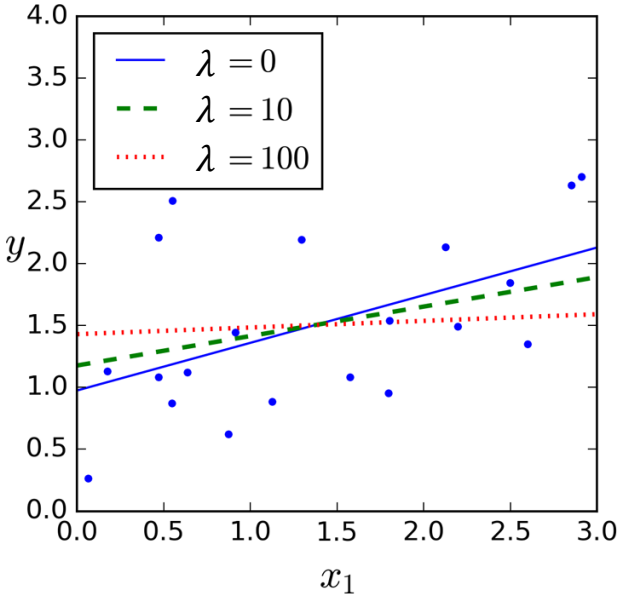
- Given two regression models
  - One using five variables to explain a phenomenon
  - Another one using 100 variables
- Which one do you prefer?

- Recap: Occam's Razor
  - Out of two theories explaining the same phenomenon, prefer the smaller one

Ridge Regression

- Variation of least squares approach which tries to avoid overfitting by keeping the weights W small
- Ridge Regression:
  - introduces *regularization*
  - create a simpler model by favoring larger factors, minimize

$$\underbrace{\sum_{i=1}^{n}(y_i - \hat{y}(X_i, W))^2}_{\text{Linear regression}} + \underbrace{\lambda \sum_{i=1}^{n} w_i^2}_{\text{Regularization}}$$



Lasso Regression

- Ridge Regression yields small, but non-zero coefficients
- Lasso Regression tends to yield zero coefficients
- $\lambda = 0$: no normalization (i.e., ordinary linear regression) → overfitting
- $\lambda \to \infty$: all weights will ultimately vanish → underfitting

- Lasso Regression optimizes

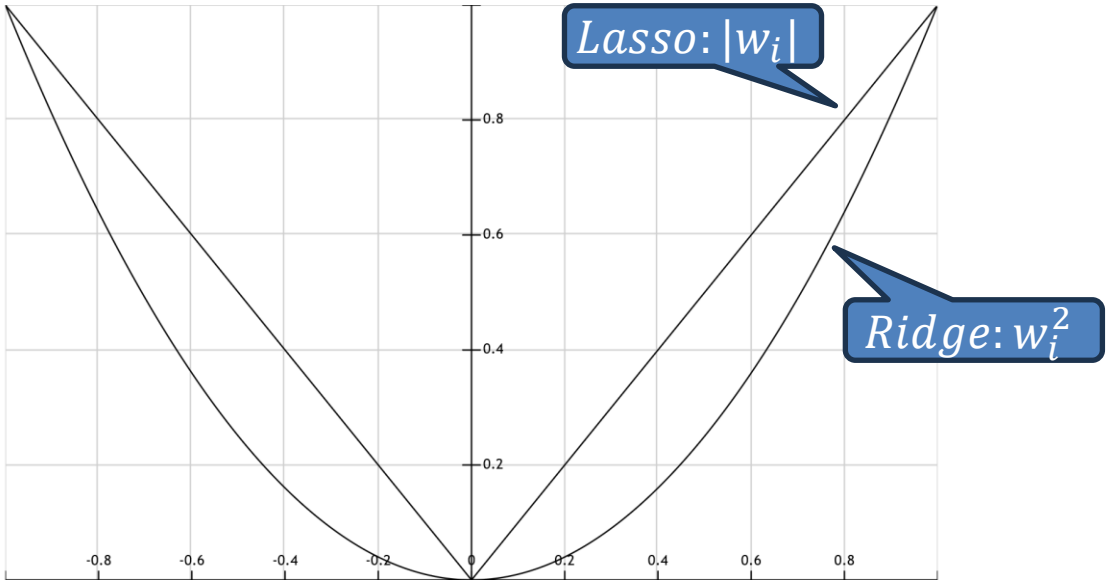
$$\underbrace{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \hat{y}(X_i, W))^2}_{\text{Linear regression}} + \underbrace{\lambda \sum_{\mathrm{i}=1}^{\mathrm{n}} |w_i|}_{\text{Regularization}}$$

Lasso: $|w_i|$ vs. Ridge: w_i^2

Lasso vs. Ridge Regression

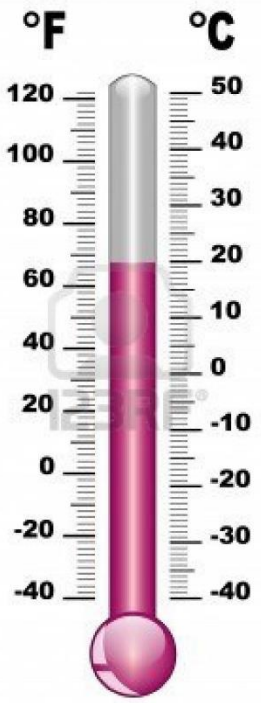
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Lasso: for $|w_i|$ close to 0, the contribution to the squared error is often smaller than the contribution to the regularization
- Hence, minimization pushes small weights down to 0



Interpolation vs. Extrapolation

UNIVERSITY OF MANNHEIM
Data and Web Science Group

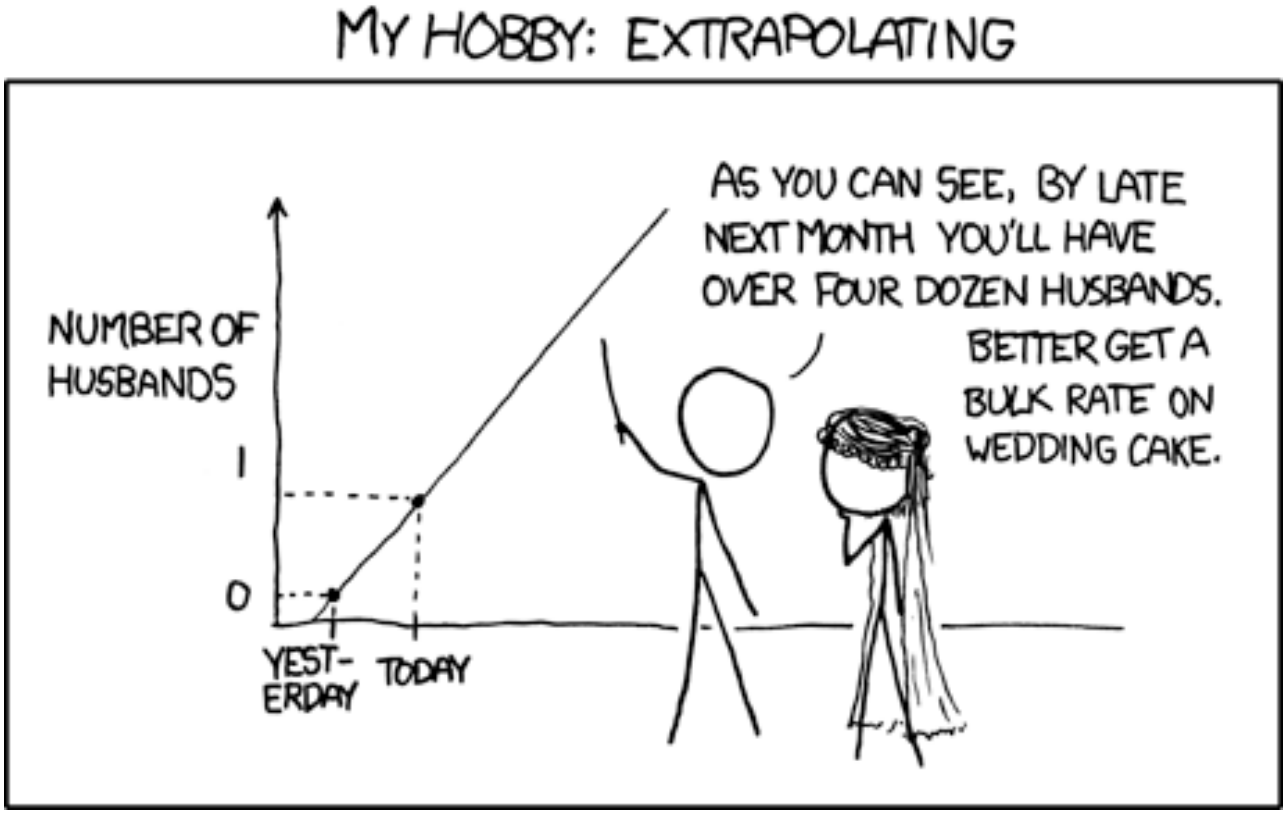


- Training data:
  - Weather observations for current day
  - E.g., temperature, wind speed, humidity, …
  - Target: temperature on the next day
  - Training values between -15°C and 32°C
- Interpolating regression
  - Only predicts values **from the training interval** [-15°C,32°C]
- Extrapolating regression
  - May also predict values ***outside* of this interval**

Interpolation vs. Extrapolation

UNIVERSITY OF MANNHEIM
Data and Web Science Group

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



http://xkcd.com/605/

Interpolation vs. Extrapolation

UNIVERSITY OF MANNHEIM
Data and Web Science Group

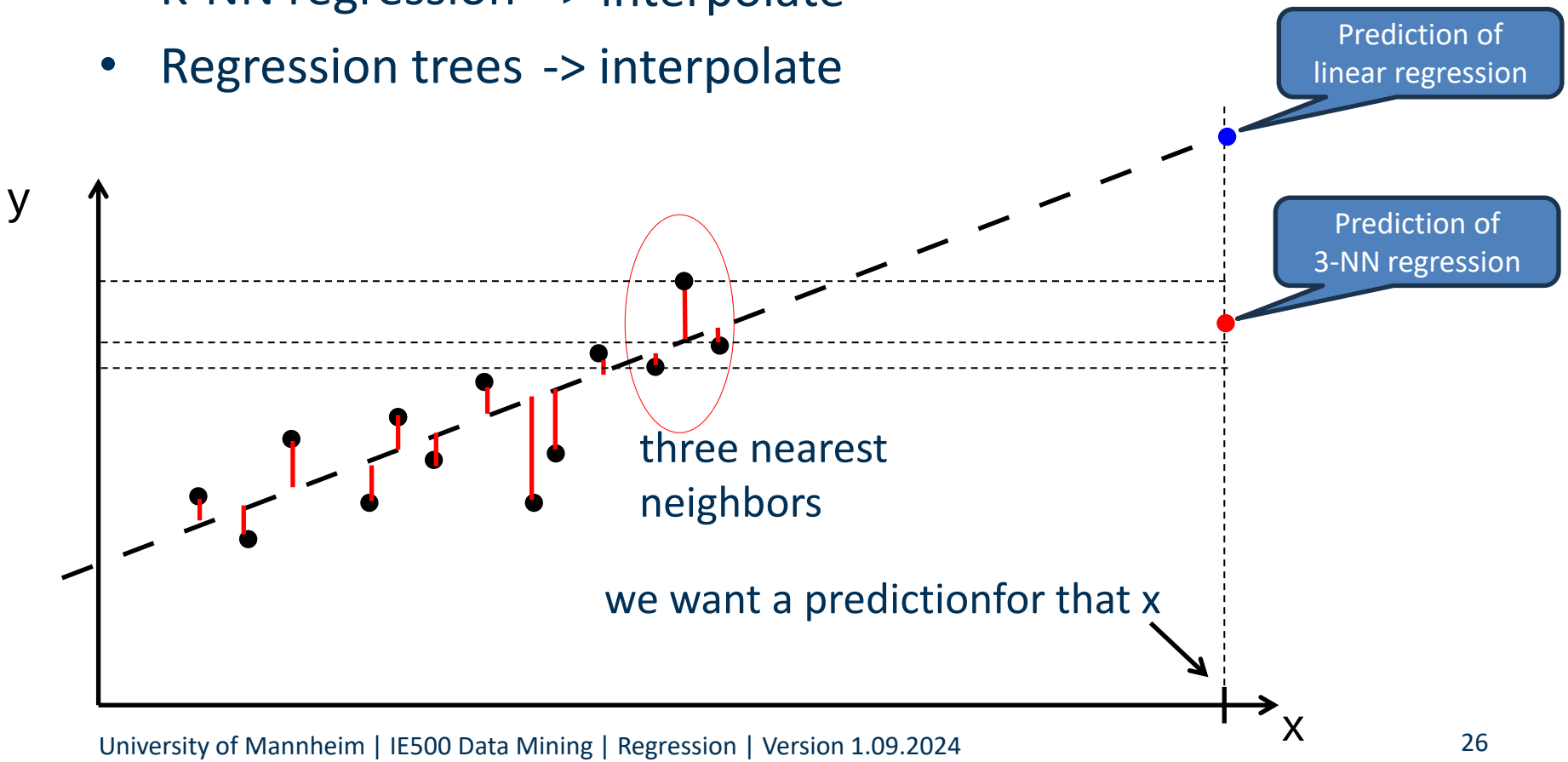
- Sometimes, however, only extrapolation is interesting
  - how far will the sea level have risen by 2050?
  - how much will the temperature rise in my nuclear power plant?

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

Linear Regression vs. K-NN Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Linear regression -> extrapolates
- K-NN regression -> interpolate
- Regression trees -> interpolate



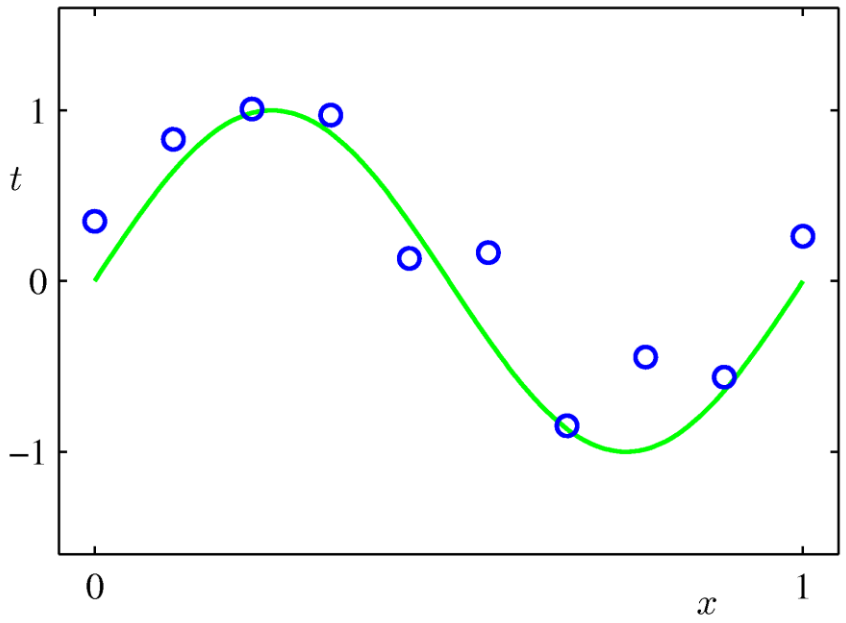
Baseline Prediction

- 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

THE PROBLEM WITH AVERAGING STAR RATINGS

http://xkcd.com/937/

..but what about Non-linear Problems?



- Possible Solutions:
  - Isotonic Regression
  - Polynomial Regression

Isotonic Regression

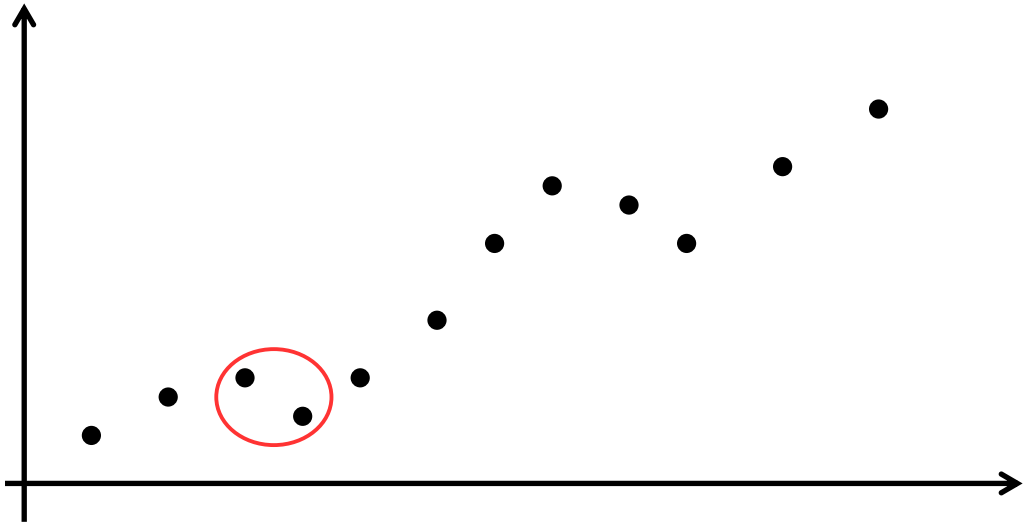
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- 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

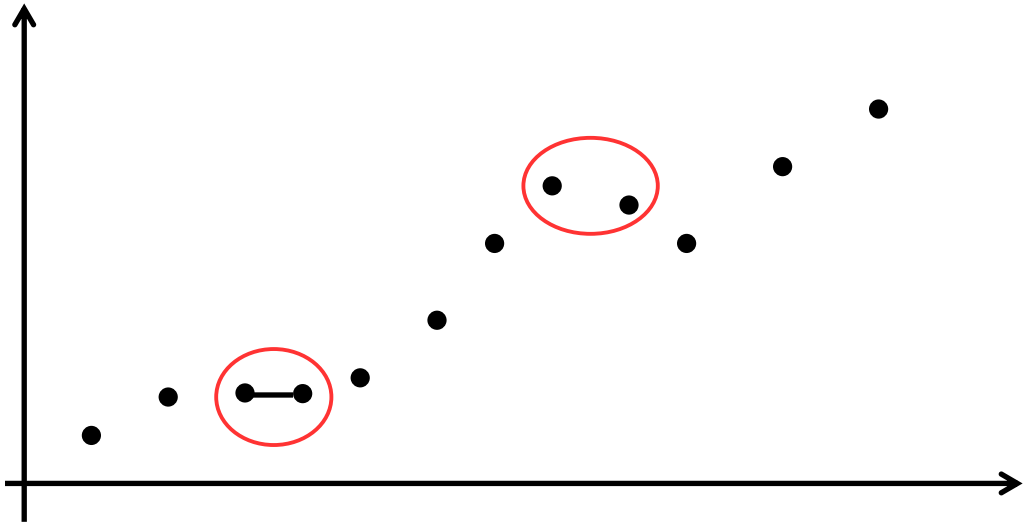
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- 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

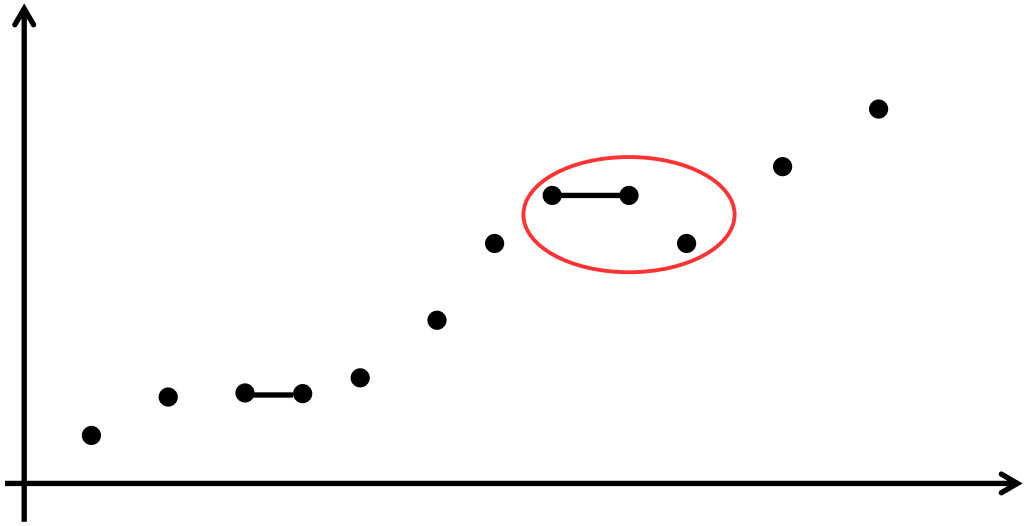
- Identify adjacent violators, i.e., $f(x_i) > f(x_{i+1})$
- Replace them with new values $f'(x_i) = f'(x_{i+1})$ so that sum of squared errors is minimized
  - ...and pool them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left



Isotonic Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

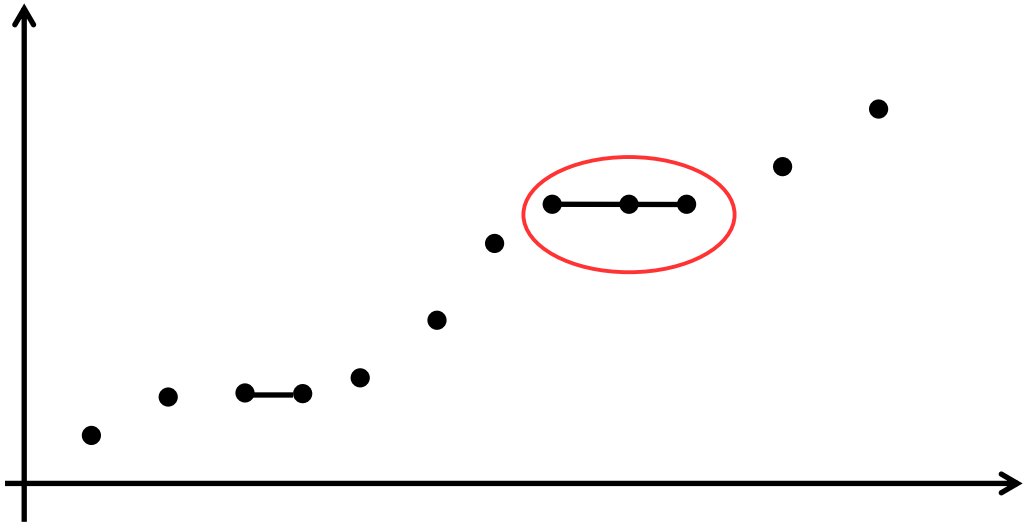
- Identify adjacent violators, i.e., $f(x_i) > f(x_{i+1})$
- Replace them with new values $f'(x_i) = f'(x_{i+1})$ so that sum of squared errors is minimized
  - ...and pool them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left



Isotonic Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

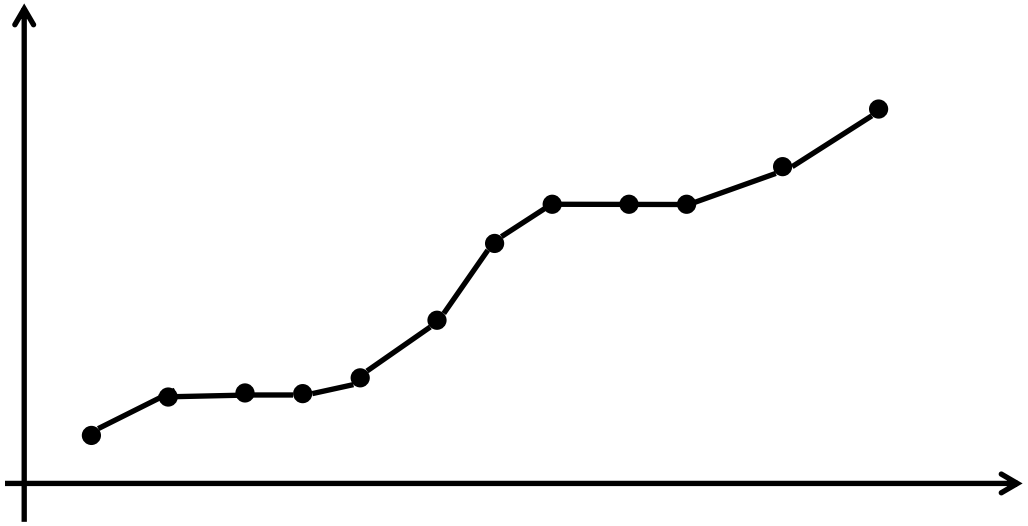
- Identify adjacent violators, i.e., $f(x_i) > f(x_{i+1})$
- Replace them with new values $f'(x_i) = f'(x_{i+1})$ so that sum of squared errors is minimized
  - ...and pool them, i.e., they are going to be handled as one point
- Repeat until no more adjacent violators are left



Isotonic Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

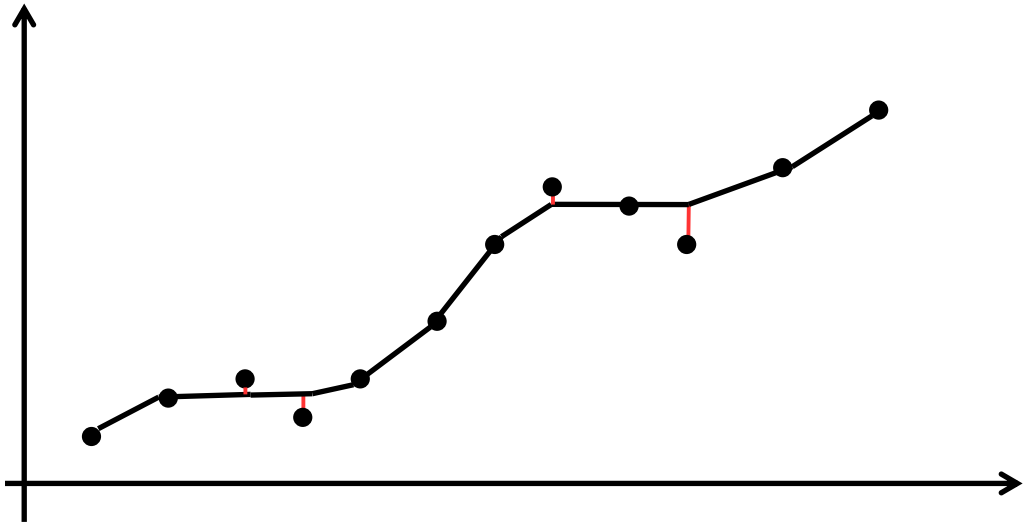
- After all points are reordered so that $f'(x_i) = f'(x_{i+1})$ holds for every i
  - Connect the points with a piecewise linear function



Isotonic Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

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

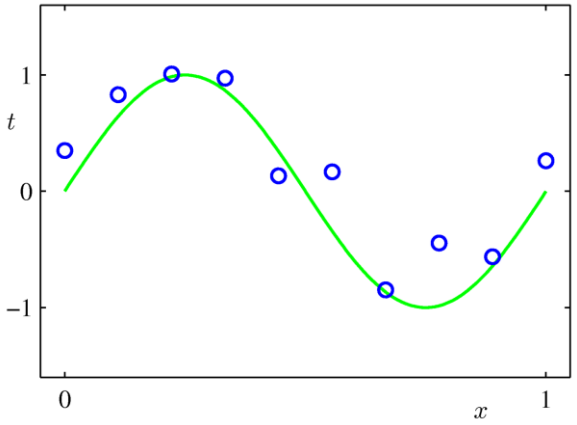


Isotonic Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Python:
  - `IsotonicRegression`
- Parameter *increasing*
  - Can be constrained to increasing (true) or decreasing (false)
  - *auto* finds the best setting
- Parameter *out_of_bounds* (how to handle unseen x values)
  - *clip* uses the smallest/largest y value in the training data
  - *nan* assigns NaN
  - *raise* creates an error

…but what about non-linear, *non-monotonous* Problems?

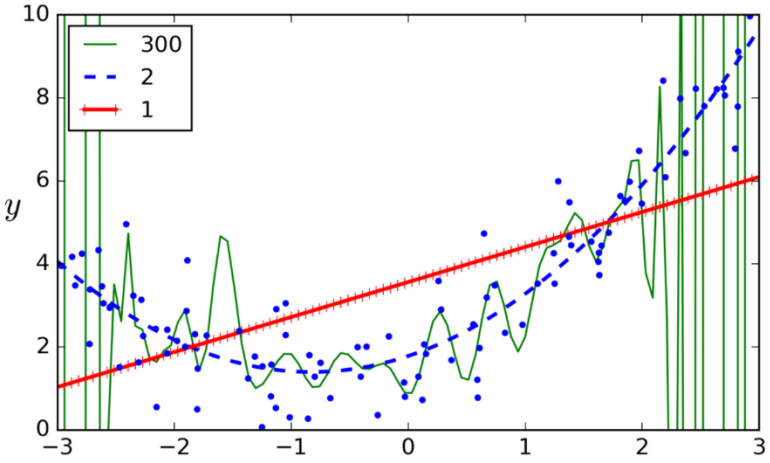


- One possibility is to apply transformations to the explanatory variables X within the regression function
  - e.g. log, exp, square root, square, etc.
  - polynomial transformation
    - example: $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

Polynomial Regression

$$\hat{y}(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d = \sum_{j=0}^{d} w_j x^j$$

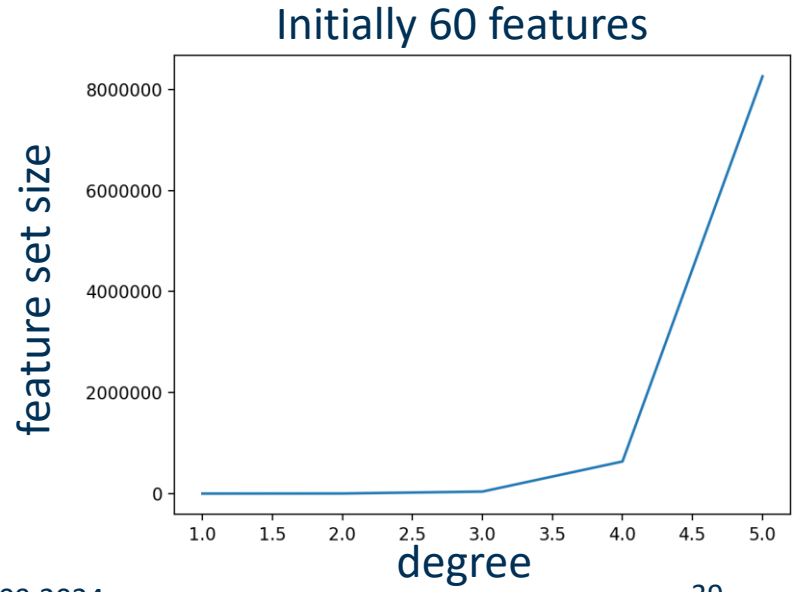
- Widely used extension of linear regression
- Can also be fitted using the least squares method
- has tendency to over-fit training data for large degrees d
- Workarounds:
  - Decrease d
  - Increase amount of training data



Polynomial Regression – Multiple Features

- Number of polynomial features of degree d for a dataset with f features: $O(f^d)$ features
- Consider: three features x,y,z, d=3
  - 6 new features d=2: x^2, y^2, z^2, xy, xz, yz
  - 10 new features d=3: x^3, y^3, z^3, x^2y, x^2z, y^2x, y^2z, z^2x, z^2y, xyz

- With higher values for f and d, we are likely to generate a very large number of additional features



Polynomial Regression & Overfitting

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Why are larger feature sets dangerous?
  - Think of overfitting as “memorizing”
- With 1k variables and 1k examples
  - we can probably identify each example by a unique feature combination
  - but with 2 variables for 1k examples, the model is forced to abstract
- Rule of thumb
  - Datasets should never be wider than long!

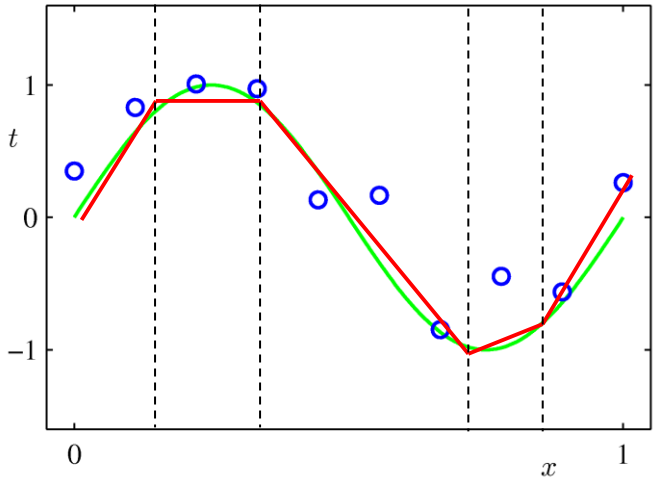
Python

```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

poly_features = PolynomialFeatures(degree=2, include_bias=False).fit_transform(X, y)
estimator = LinearRegression()
estimator.fit(poly_features, y)
```

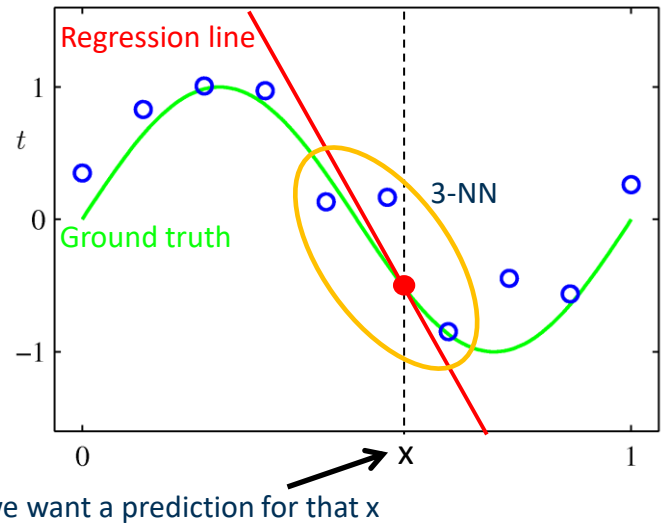
Local Regression

- 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** and **linear regression**
- Given a data point for prediction
  1. retrieve the k nearest neighbors
  2. learn a regression model using those neighbors
  3. use the learned model to predict y value



Discussion of Local Regression

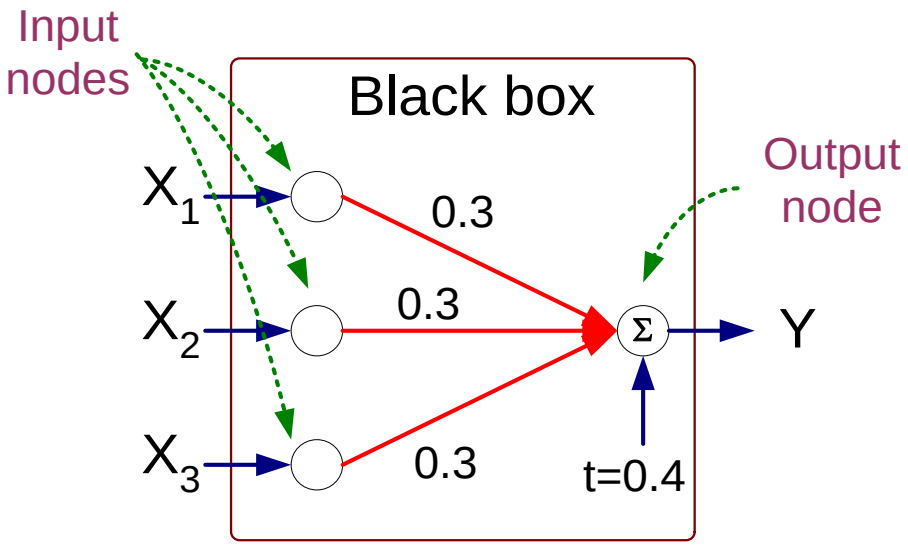
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- 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

Artificial Neural Networks (ANNs) for Regression

- **Recap**: How did we use ANNs for classification?

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

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

Artificial Neural Networks (ANNs) for Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

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

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

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

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

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

Artificial Neural Networks (ANNs) for Regression

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- 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 e.g. mean squared error

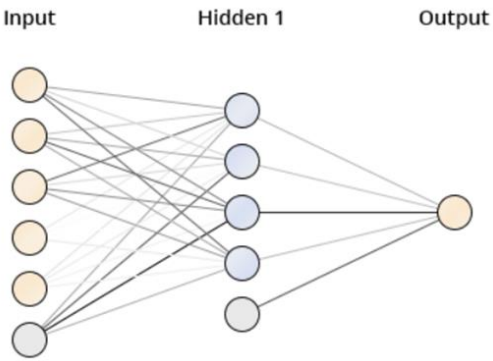
Artificial Neural Networks (ANNs) for Regression

- 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**



The Bias/Variance-Tradeoff

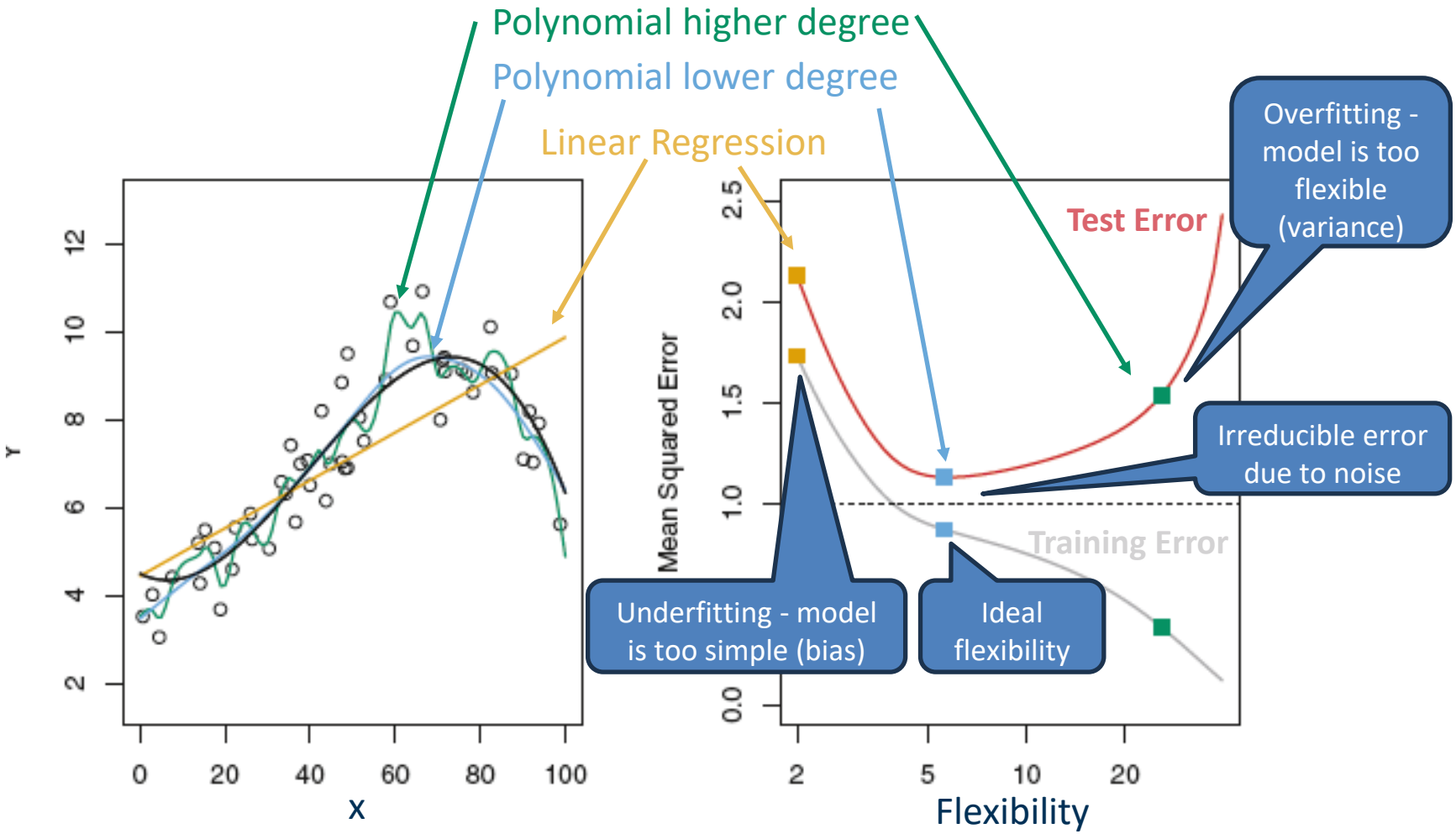
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- 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:
  - **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
  - **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
  - **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 – only in training, fixing broken sensors)

The Bias/Variance-Tradeoff

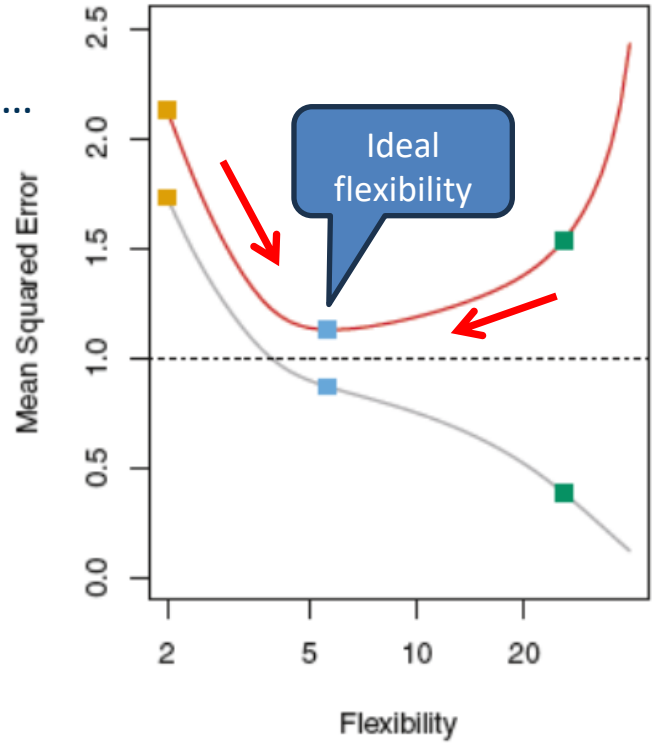
UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Three models with different flexibility trying to fit a function



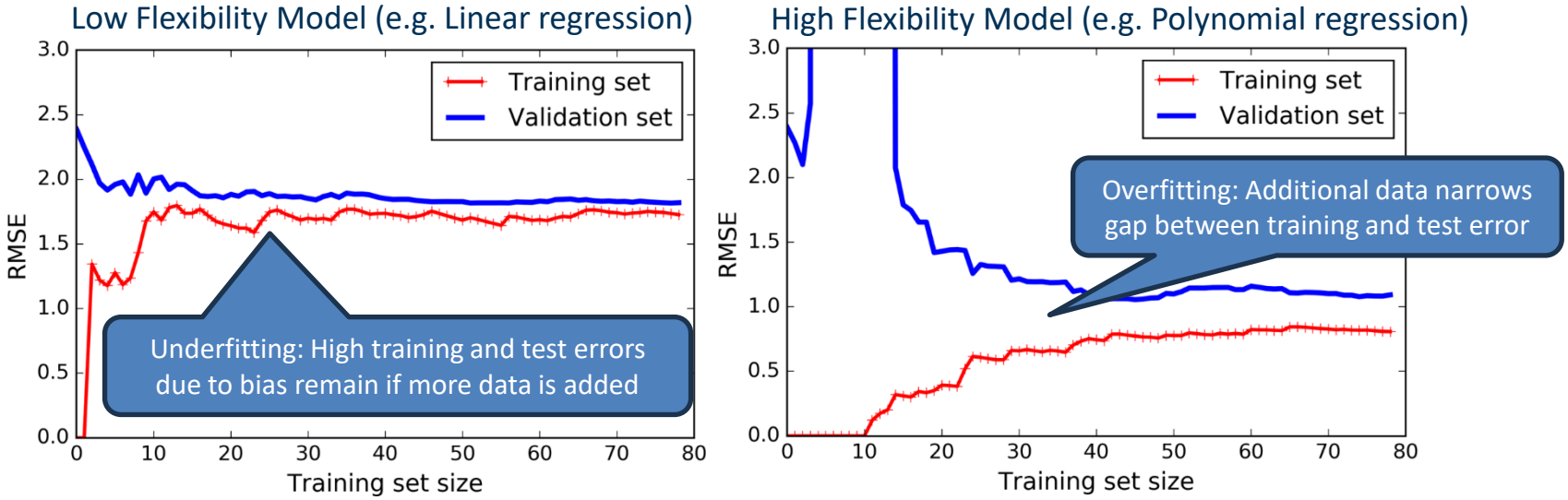
Learning Method and Hyperparameter Selection

- We try to find the **ideal flexibility** (bias/variance-tradeoff) by
  - Testing different learning methods
    - Linear regression, polynomial regression, …
    - Decision Trees, ANNs, Naïve Bayes, …
  - Testing different hyperparameters
    - degree of polynomial, ridge
    - max depth of tree, min examples branch
    - number of hidden layers of ANN
- But we have **three more options**:
  - Increase the amount of training data
  - Increase the interestingness of the data by including more corner cases
  - Cleanse the training data



Learning Curves for Under- and Overfitting Models

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



  - For overfitting models, the gap between training and test error can often be narrowed by adding more training data
  - Thus, having more training data also allows us to use models having a higher flexibility, e.g. Deep Learning

From Classification to Regression and Back

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- We have got to known classification first
  - And asked: how can we get from classification to regression?

- Turning the question upside down:
  - Can we use regression algorithms for classification?

- Transformation:
  - For binary classification: encode true as 1.0 and false as 0.0
    - Learn regression model
    - Predict false for (-∞,0.5] and true for (0.5,∞)
  - Similarly for ordinal (e.g., good, medium, bad)
  - Non-ordinal multi-class problems are trickier

Summary

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Regression
  - Predict numerical values instead of classes
- Model evaluation
  - Metrics: (root) mean squared error, R squared, …
- Methods
  - K nearest neighbors, regression trees, model trees
  - Linear regression (ridge, lasso)
  - Isotonic regression, polynomial regression, local regression
  - Artificial neural networks for regression
- For good performance on unseen data
  - Choose learning method having the right flexibility (bias/variance-tradeoff)
  - Use large quantities of interesting training data

Additional material

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- This week additional material is about
  - Time Series

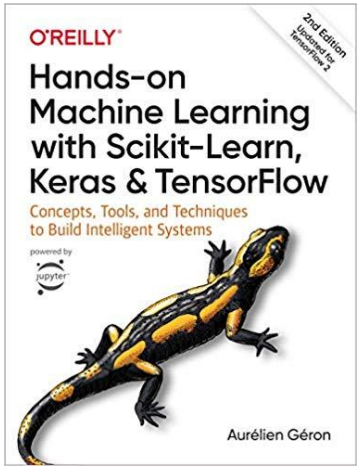
- Additional material is exercise and exam relevant

Questions?

Literature for this Slideset

UNIVERSITY OF MANNHEIM
Data and Web Science Group

- Solving practical regression tasks using Python:
  - Geron: Hands-on Machine Learning - Chapter 4



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