

Data Mining Regression



Regression

- Classification
 - covered in the previous lectures
 - predict a label from a finite collection
 - e.g., true/false, low/medium/high, ...
- Regression
 - predict a *numerical* value
 - from a possibly infinite set of possible values
- Examples
 - temperature
 - sales figures
 - stock market prices
 - ...

Contents

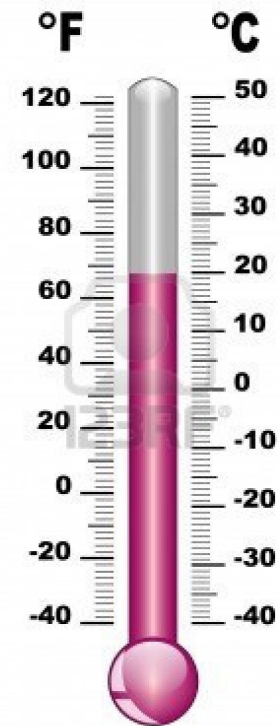
- A closer look at the problem
 - e.g., interpolation vs. extrapolation
 - measuring regression performance
- Revisiting classifiers we already know
 - which can also be used for regression
- Adoption of classifiers for regression
 - model trees
 - support vector machines
 - artificial neural networks
- Other methods of regression
 - linear regression and its regularized variants
 - 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

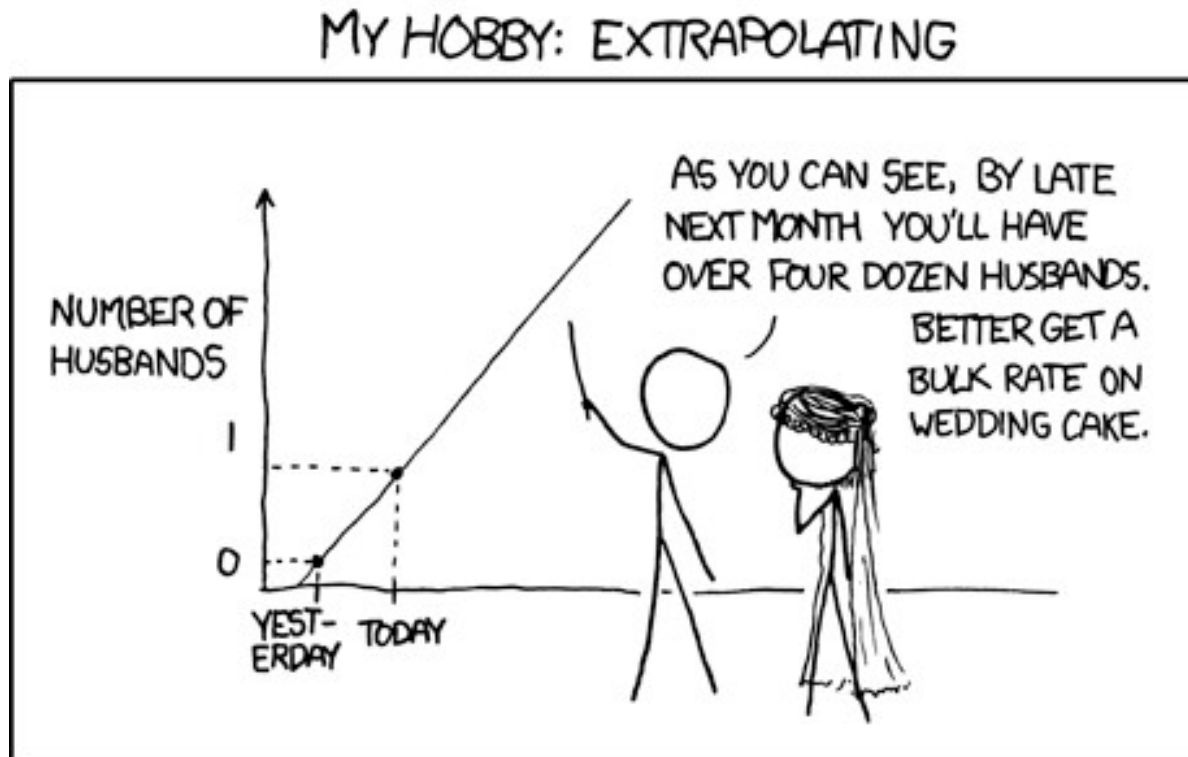
Interpolation vs. Extrapolation

- 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 $[-15^{\circ}\text{C}, 32^{\circ}\text{C}]$
- 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?
 - 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

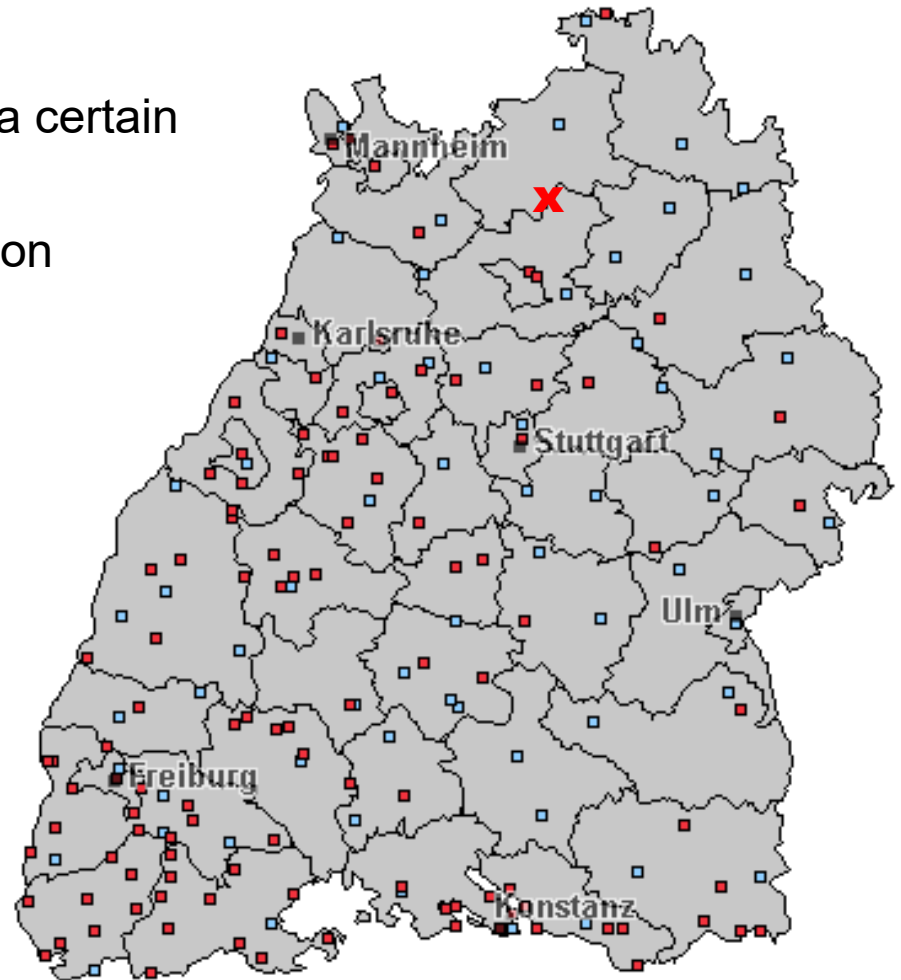
<http://xkcd.com/937/>



THE PROBLEM WITH
AVERAGING STAR RATINGS

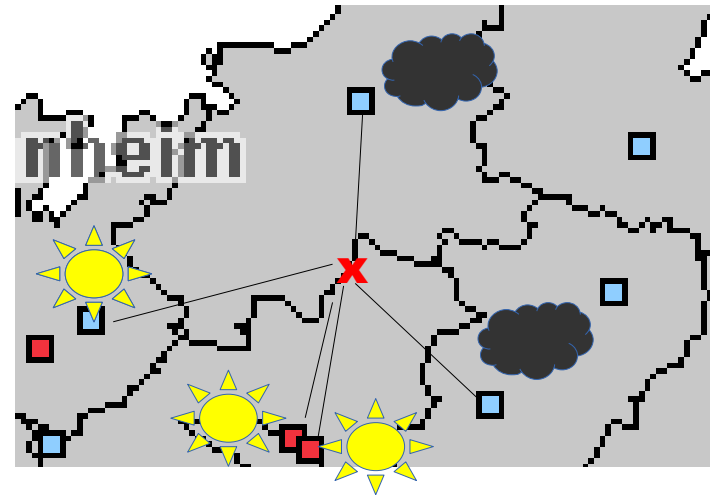
k Nearest Neighbors Revisited

- Problem
 - find out what the weather is in a certain place
 - where there is no weather station
 - how could you do that?



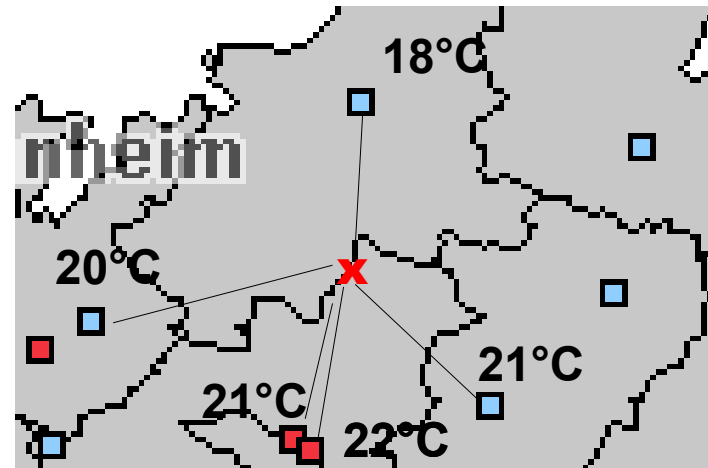
k Nearest Neighbors Revisited

- 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



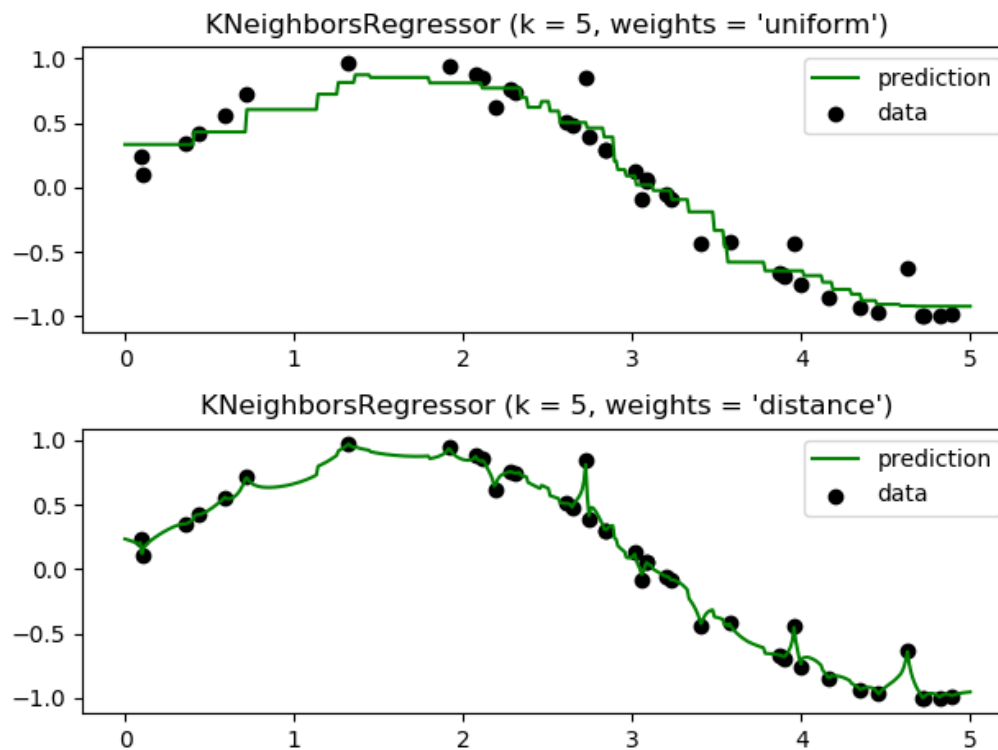
k Nearest Neighbors for Regression

- 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 Python

```
model = KNeighborsRegressor(k=number,  
                             weights='uniform'/'distance')  
  
model.fit(X,Y)
```



Performance Measures

- 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

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

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

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

- For an arbitrary numerical target, we can define

$$\text{MAE} = \frac{\sum_{\text{all examples}} |predicted - actual|}{N}$$

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

(Root) Mean Squared Error

- Mean Squared Error:

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

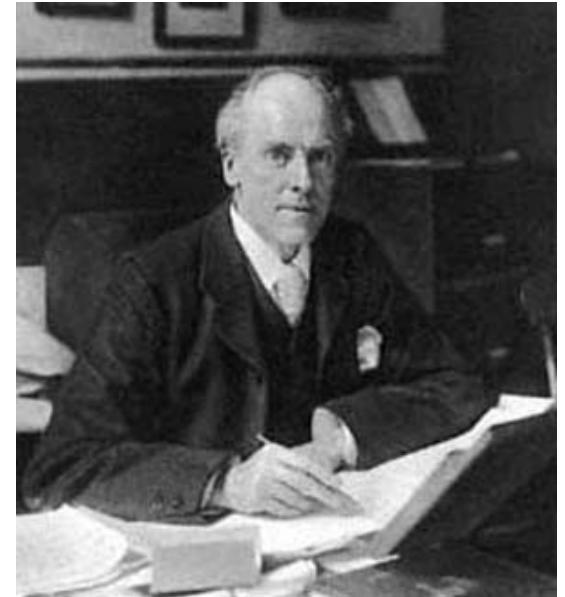
- Root Mean Squared Error:

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

- More severe errors are weighted higher by MSE and RMSE

Correlation

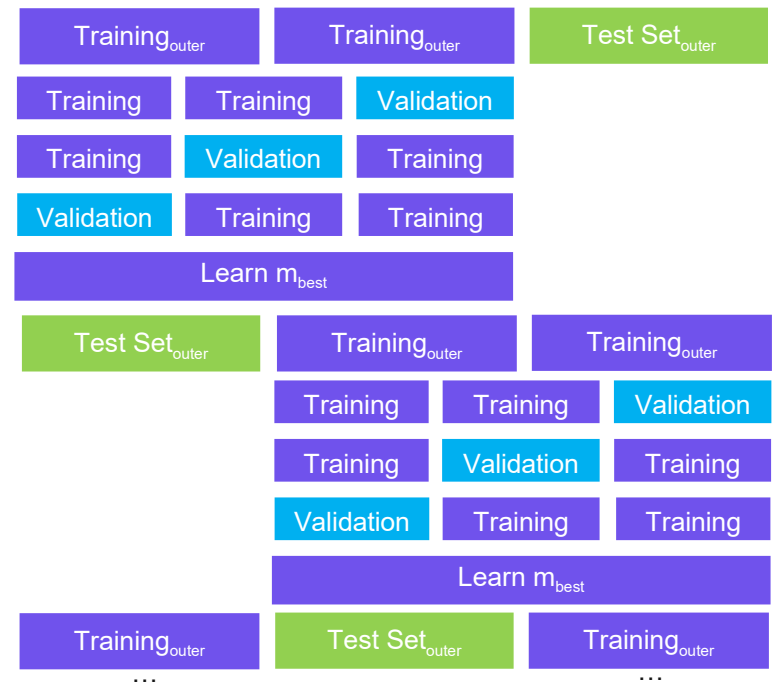
- Pearson's correlation coefficient
- Scores well if
 - high actual values get high predictions
 - low actual values get low predictions
- Caution: PCC is scale-invariant!
 - actual income: \$1, \$2, \$3
 - predicted income: \$1,000, \$2,000, \$3,000
 - PCC = 1



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

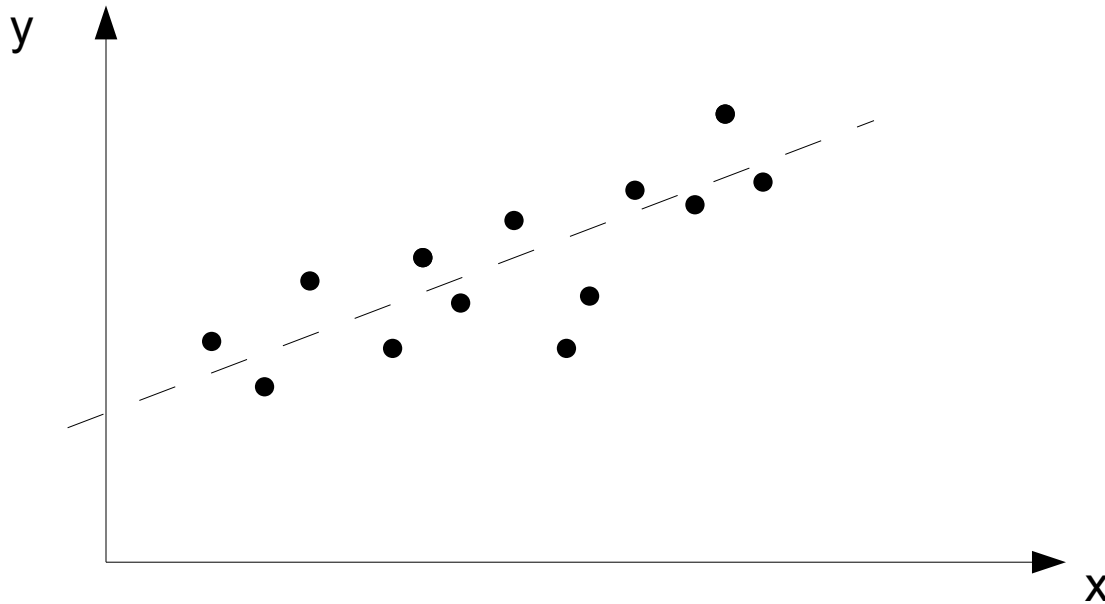
Evaluation Protocols

- Just as for classification
 - we are interested in performance on *unseen* data
 - we want to avoid *overfitting*
- Recap
 - split/cross validation
 - hyperparameter tuning in nested cross validation



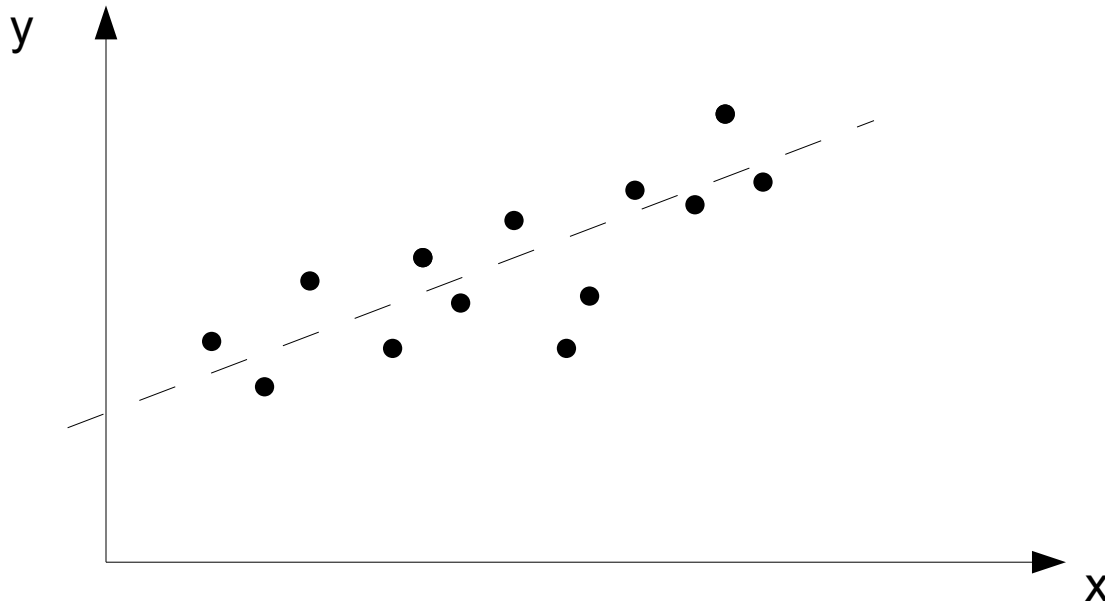
Linear Regression

- Assumption: target variable y is (approximately) linearly dependent on attributes
 - for visualization: one attribute x
 - in reality: $x_1 \dots x_n$



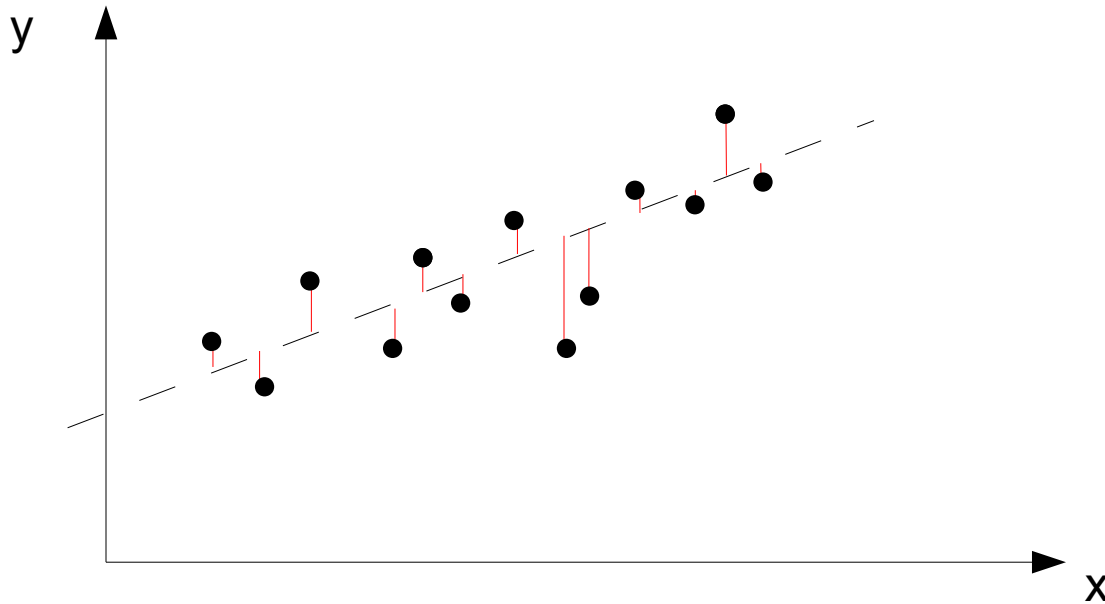
Linear Regression

- Target: find a linear function $f: f(x) = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$
 - so that the error is minimized
 - i.e., for all examples (x_1, \dots, 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, \dots, w_n so that $\sum_{\text{all examples}} (w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n - y)^2$ is minimized
- note: we omit the denominator N



Linear Regression: Multi Dimensional Example

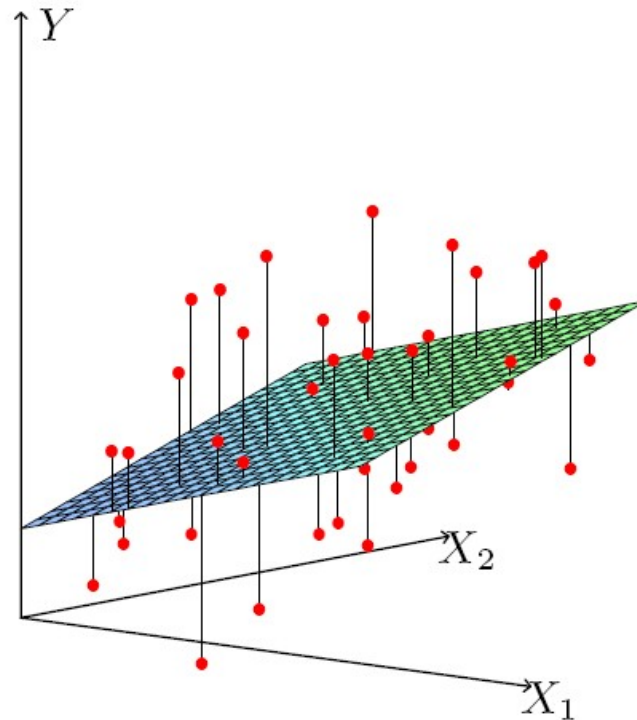
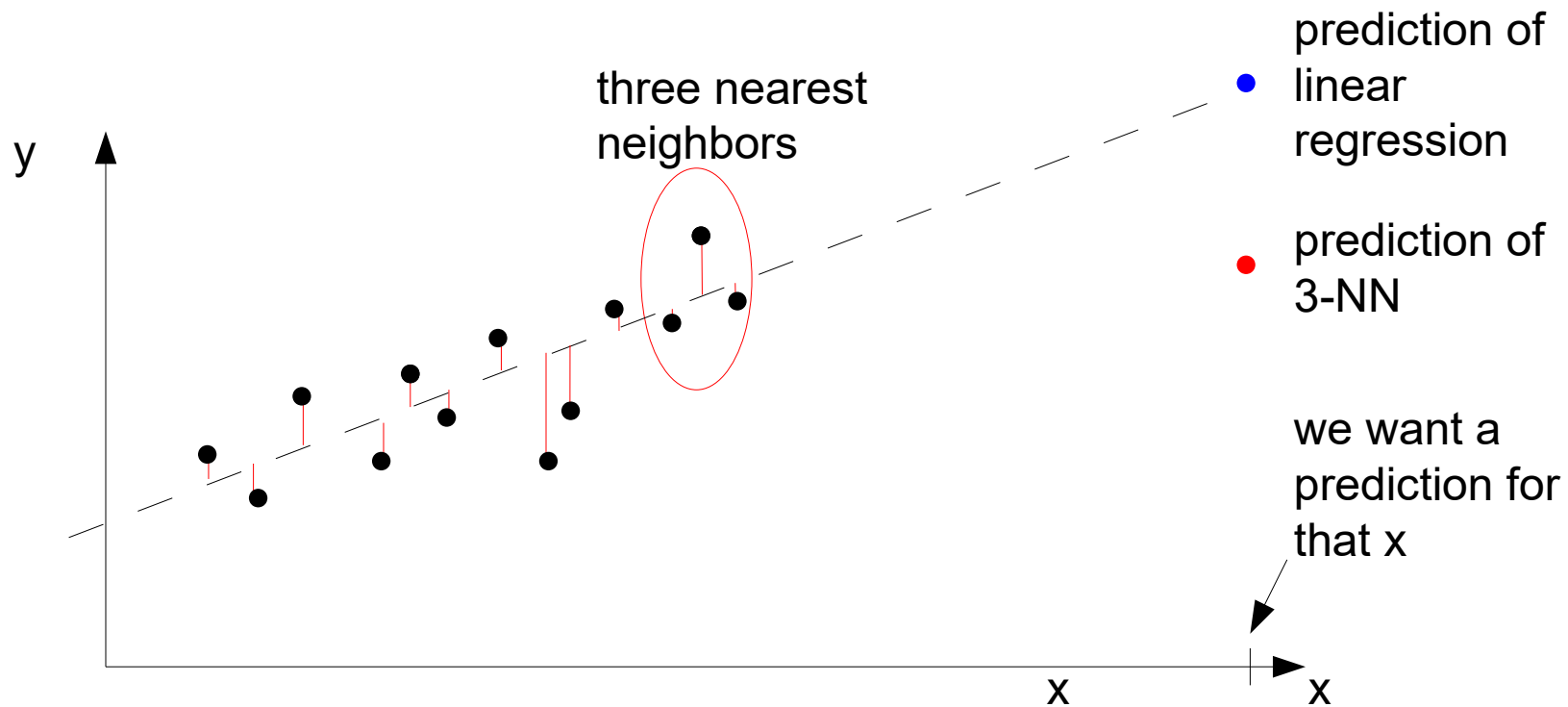


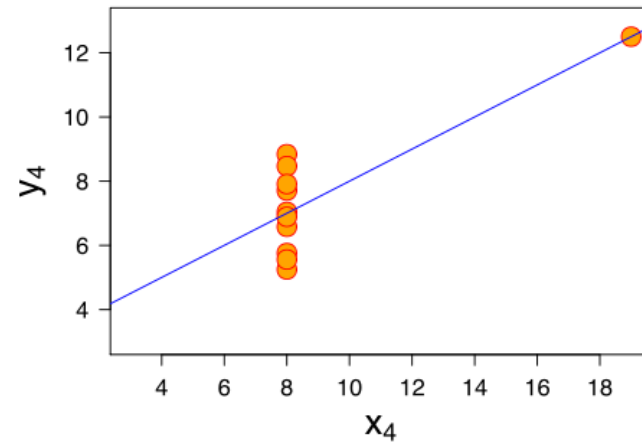
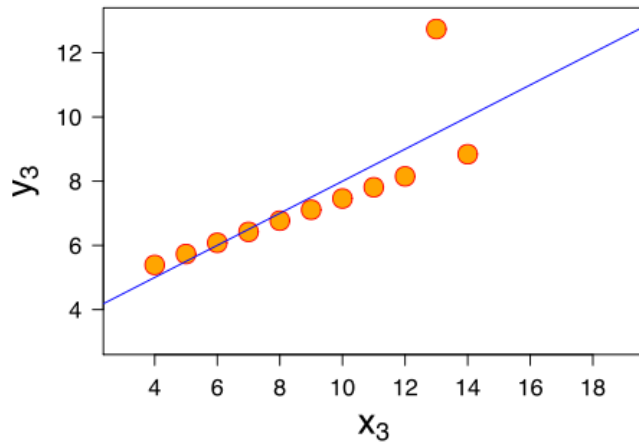
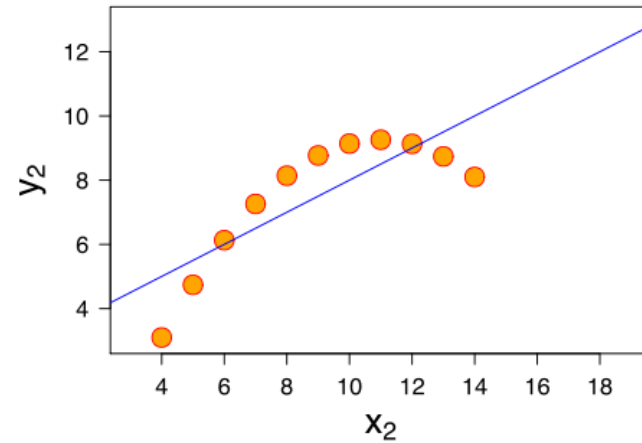
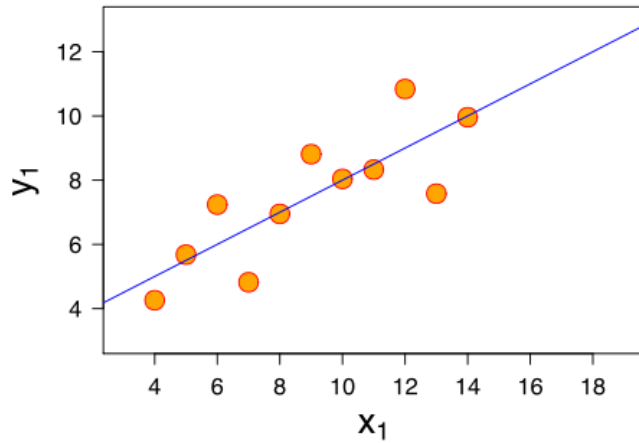
FIGURE 3.1. *Linear least squares fitting with $X \in \mathbb{R}^2$. We seek the linear function of X that minimizes the sum of squared residuals from Y .*

Linear Regression vs. k-NN Regression

- Recap: Linear regression extrapolates, k-NN interpolates



Linear Regression Examples



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

- Linear regression only minimizes the errors on the training data
 - i.e., $\sum_{\text{all examples}} \left(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n - y \right)^2$
- With *many* variables, we can have a large set of very small w_i
 - this might be a sign of overfitting!
- Ridge Regression:
 - introduces *regularization*
 - create a simpler model by favoring larger factors, minimize

$$\sum_{\text{all examples}} \left(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n - y \right)^2 + \lambda \sum_{\text{all variables}} w_i^2$$

Ridge & Lasso Regression

- Ridge Regression optimizes

$$\sum_{\text{all examples}} \left(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n - y \right)^2 + \lambda \sum_{\text{all variables}} w_i^2$$

- Lasso Regression optimizes

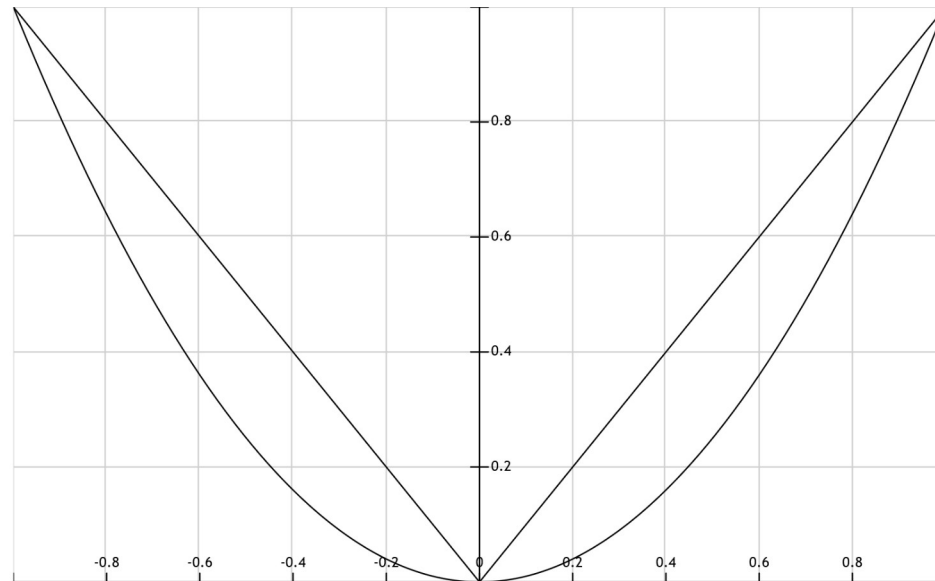
$$\sum_{\text{all examples}} \left(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n - y \right)^2 + \lambda \sum_{\text{all variables}} |w_i|$$

- Observations

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

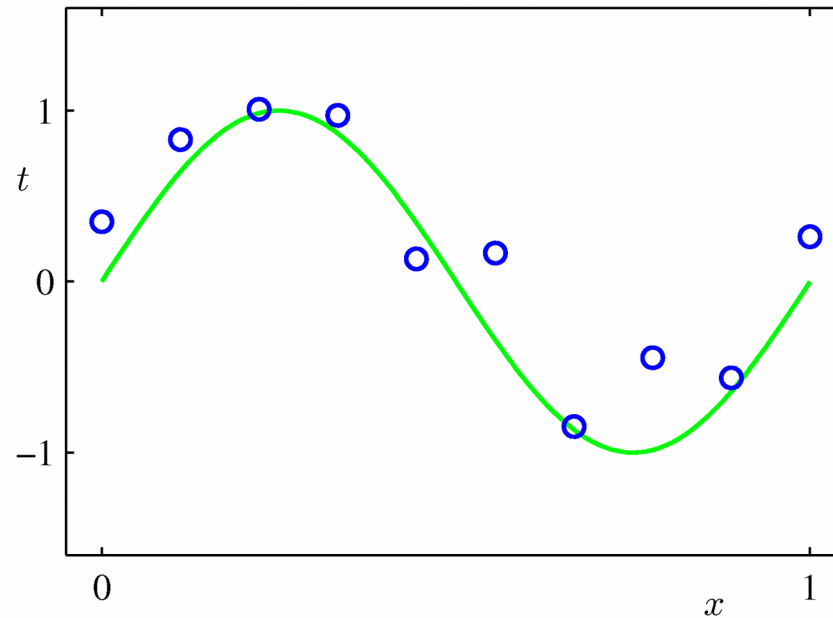
Lasso vs. Ridge Regression

- 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



$$\sum_{\text{all examples}} \left(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n - y \right)^2 + \lambda \sum_{\text{all variables}} |w_i|$$

...but what about Non-linear Problems?

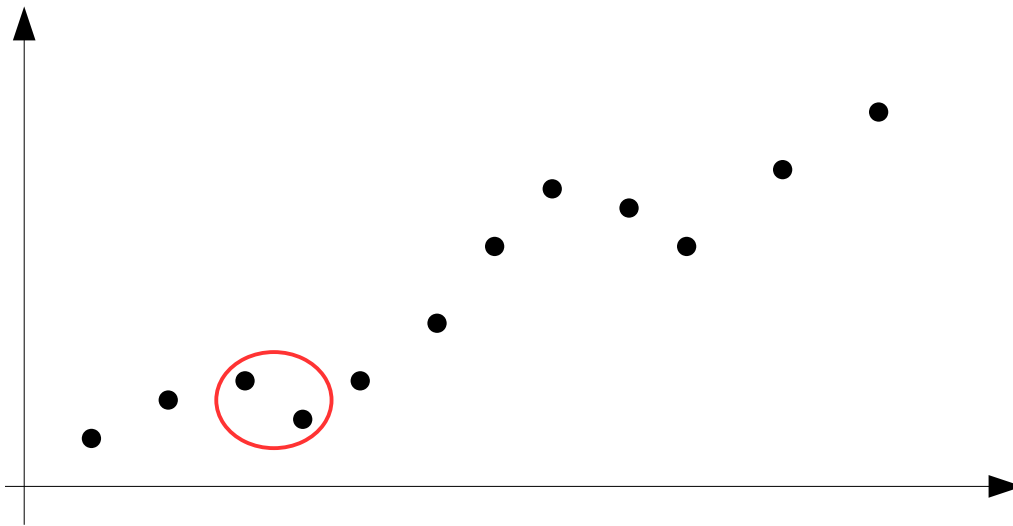


Isotonic Regression

- 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)

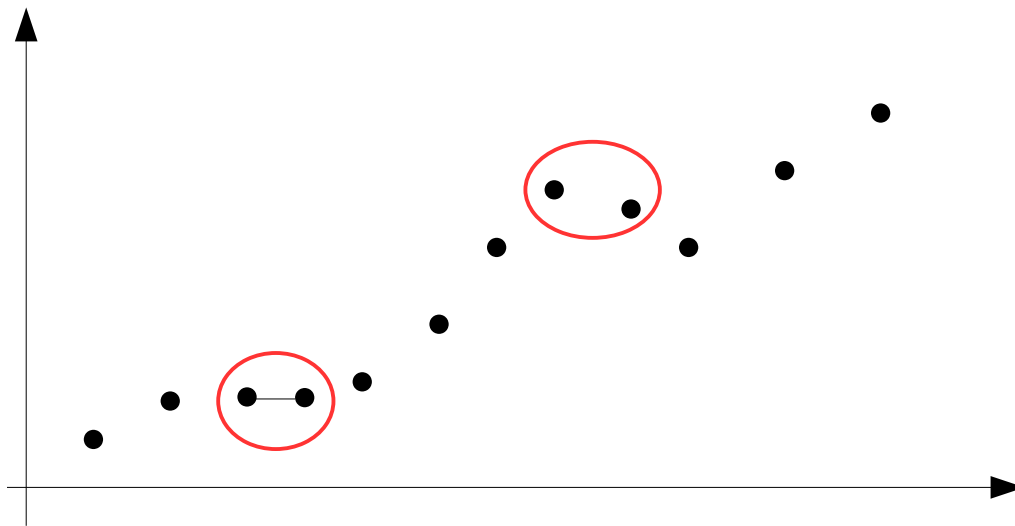
Isotonic Regression

- 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



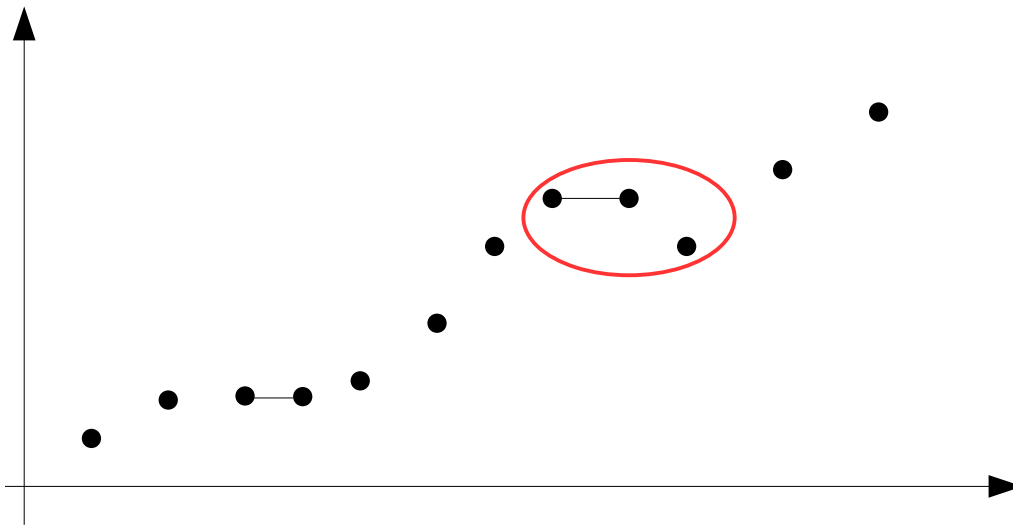
Isotonic Regression

- 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



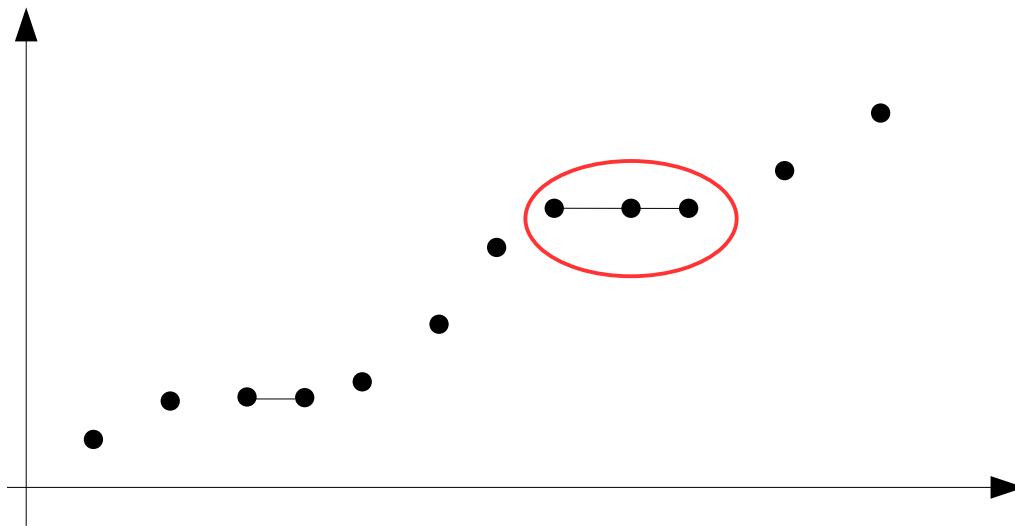
Isotonic Regression

- Identify adjacent violators, i.e., $f(x_i) > f(x_{i+1})$
- 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



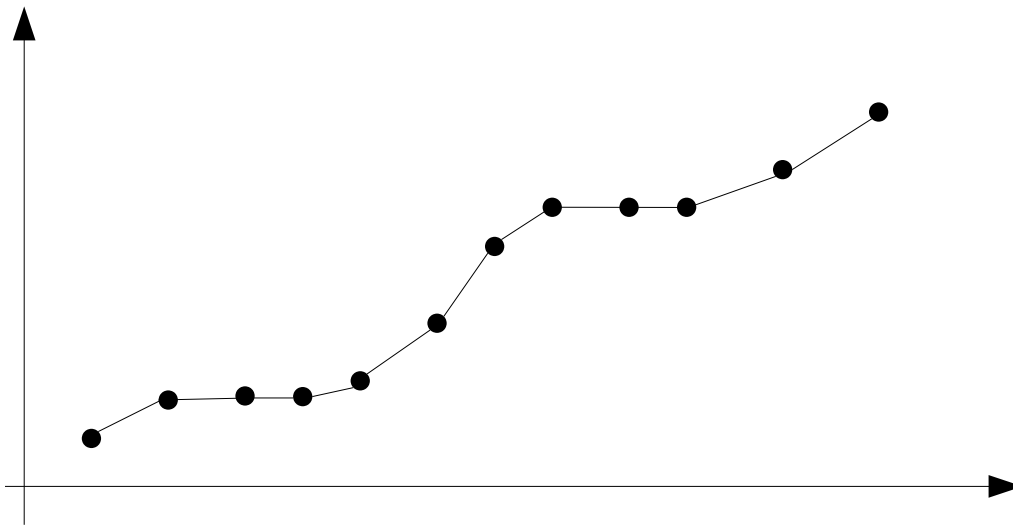
Isotonic Regression

- 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



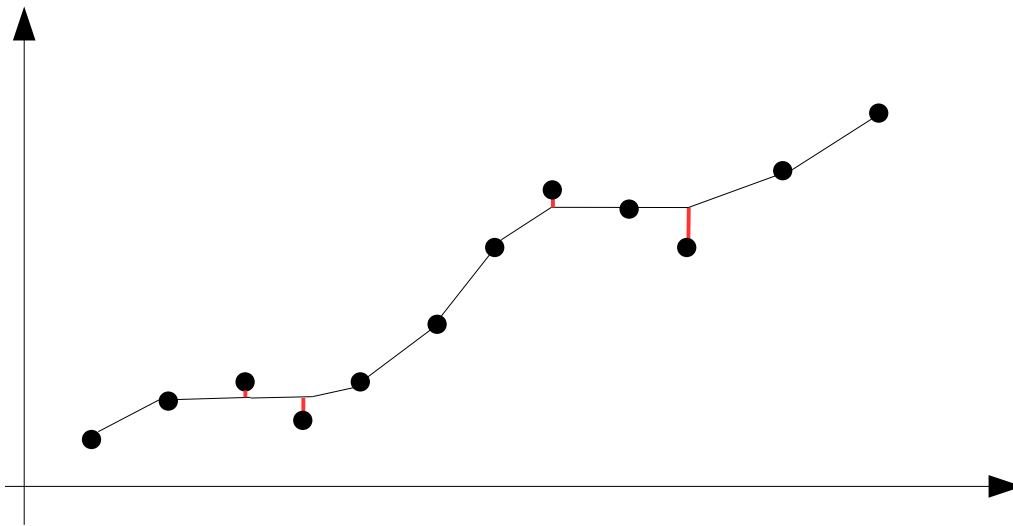
Isotonic Regression

- 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



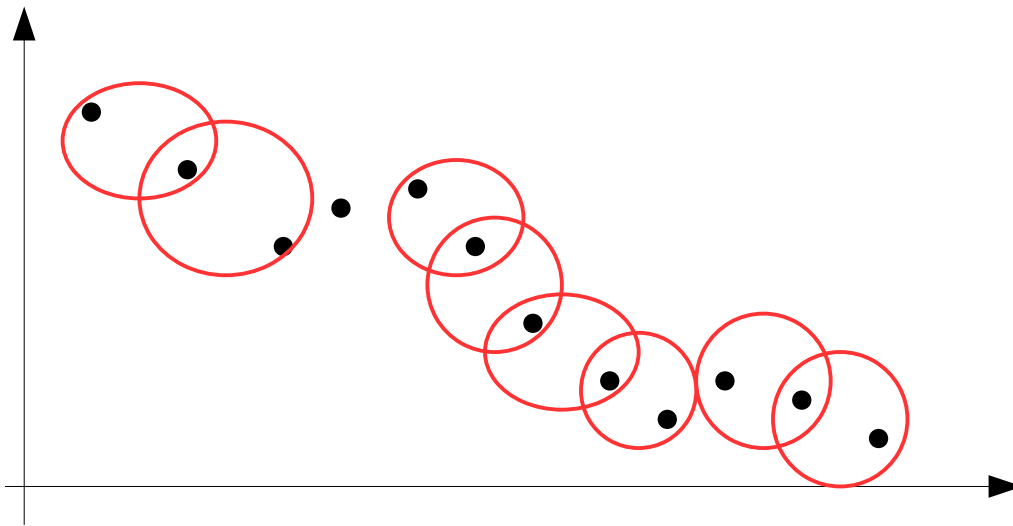
Isotonic Regression

- Comparison to the original points
 - Plateaus exist where the points are not monotonous
 - Overall, the mean squared error is minimized
- Python: `IsotonicRegression` (caution: only increasing case)



Isotonic Regression: Mind the Direction!

- Thought experiment:
 - What happens if we assume a monotonously increasing function here?
 - We will identify quite a few adjacent violations!

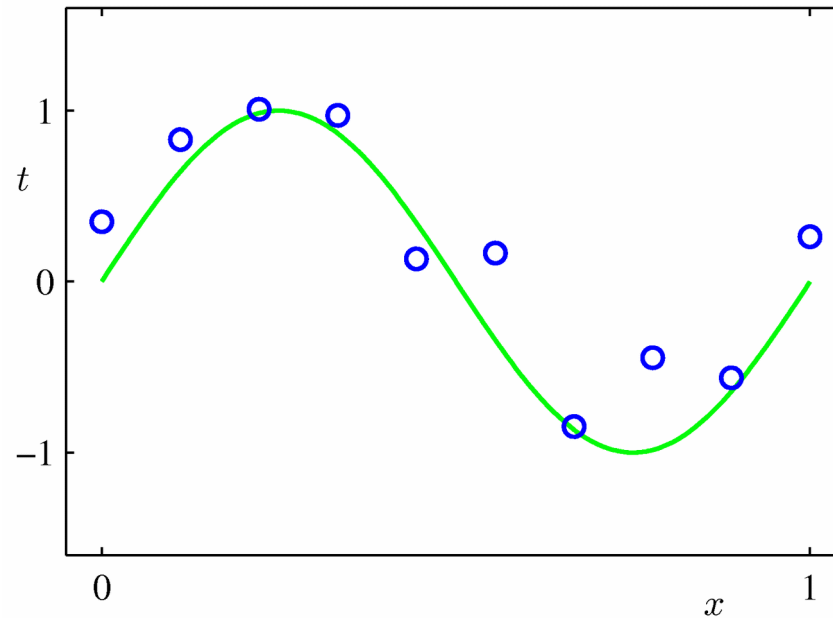


Isotonic Regression: Mind the Direction!

- Thought experiment:
 - What happens if we assume a monotonously increasing function here?
 - We will identify quite a few adjacent violations!
 - Ultimately, they will all be pooled!



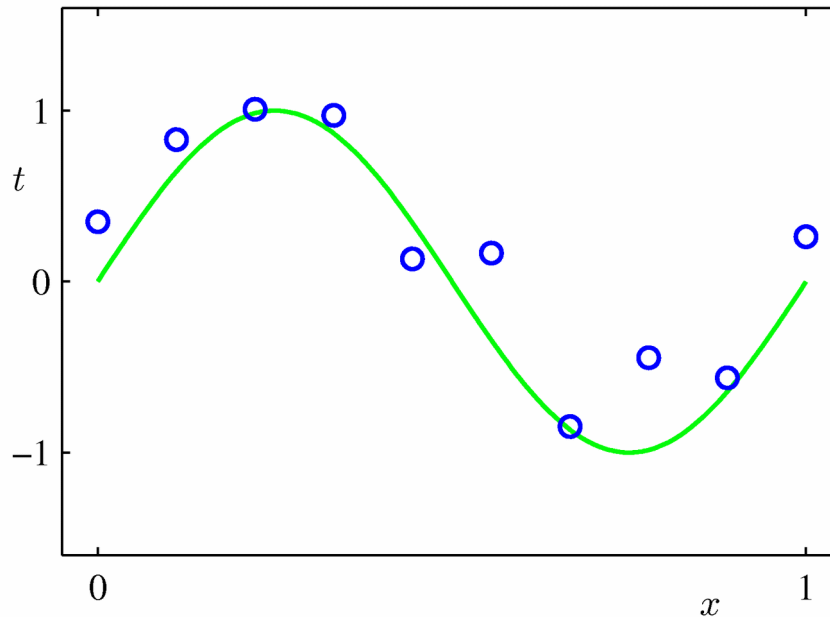
...but what about non-linear, *non-monotonous* Problems?



Possible Option: new Attributes

- 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



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

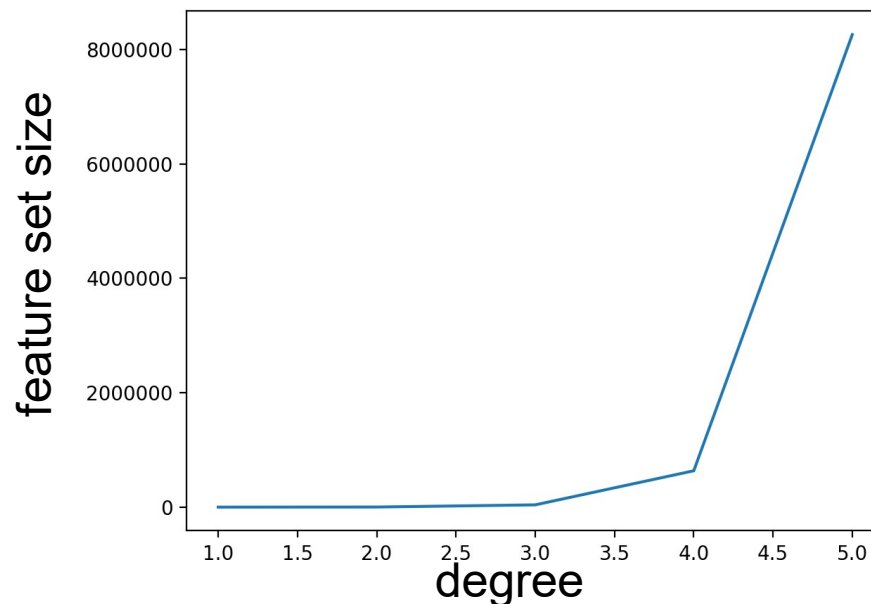
```
Xp = PolynomialFeatures(degree=M).fit_transform(X)
```

Polynomial Regression & Overfitting

- Creating 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

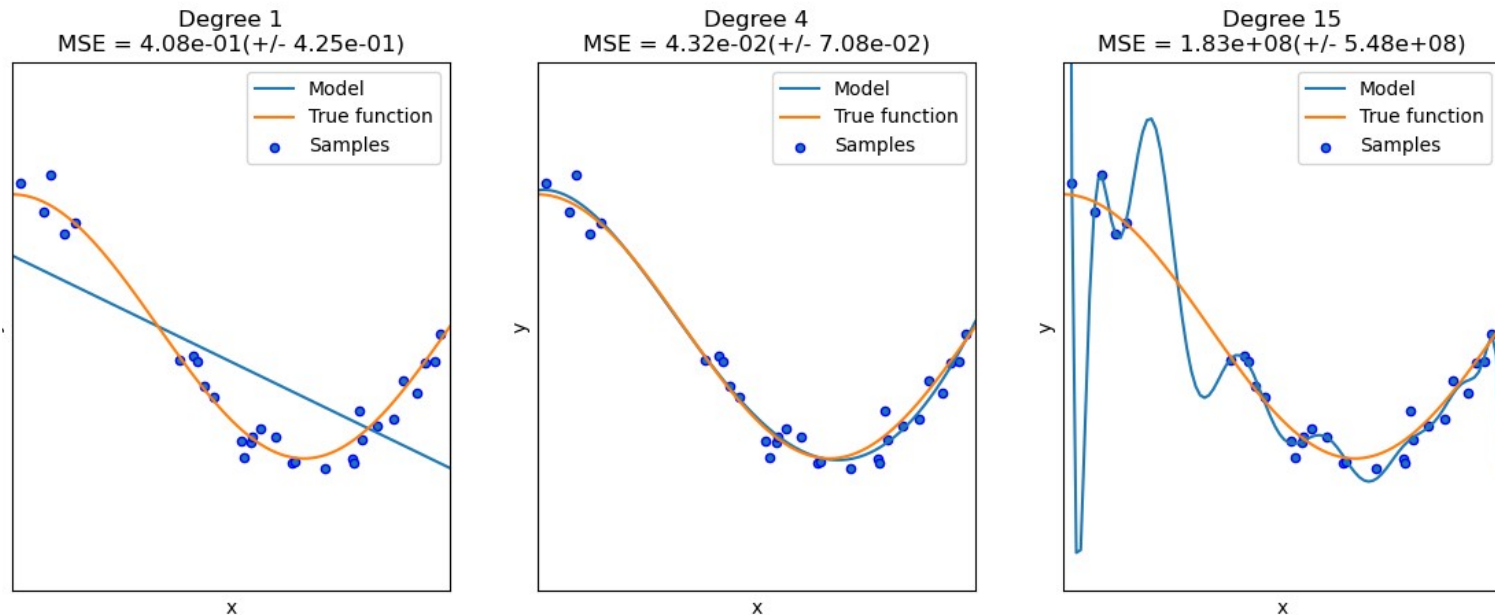
- Creating polynomial features of degree d for a dataset with f features: $O(f^d)$ features
- Example: Sonar dataset (60 features)



<https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/>

Polynomial Regression & Overfitting

- Larger feature sets lead to higher degrees of overfitting

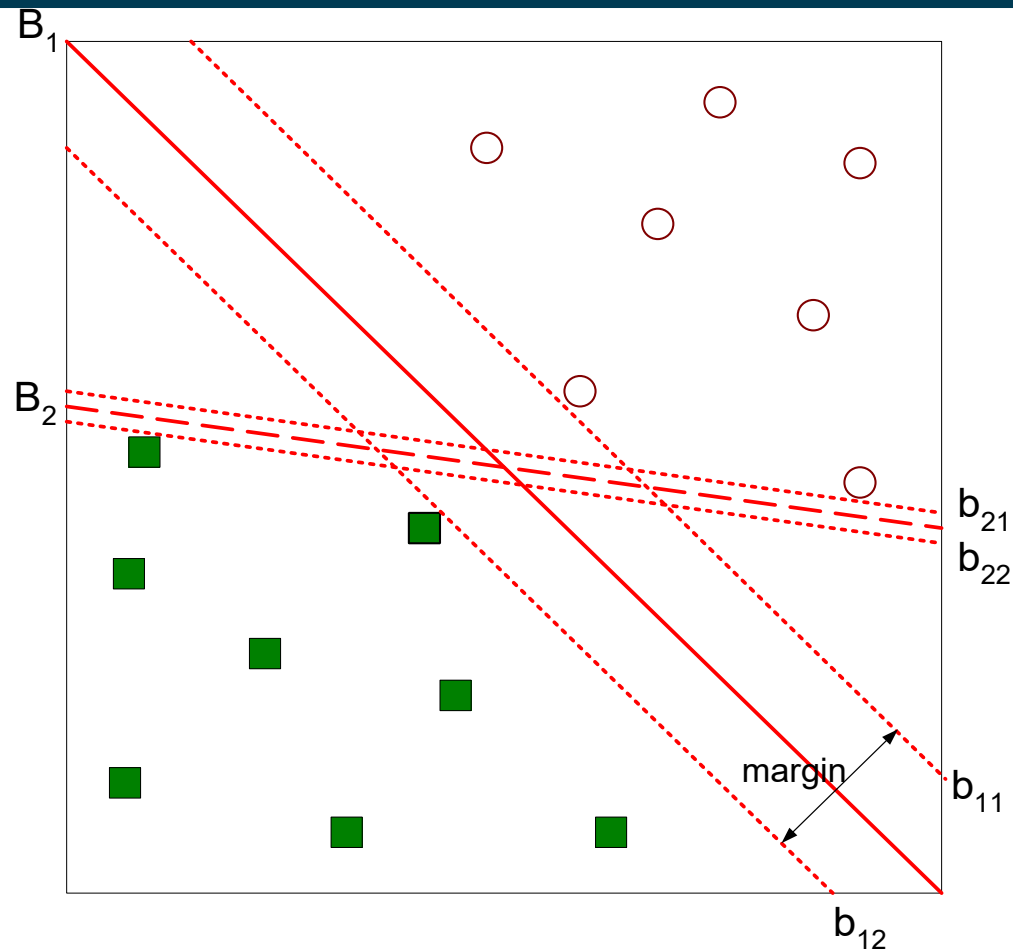


Polynomial Regression & Overfitting

- 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!



Support Vector Machines Revisited



- Find hyperplane **maximizes** the margin \Rightarrow B1 is better than B2

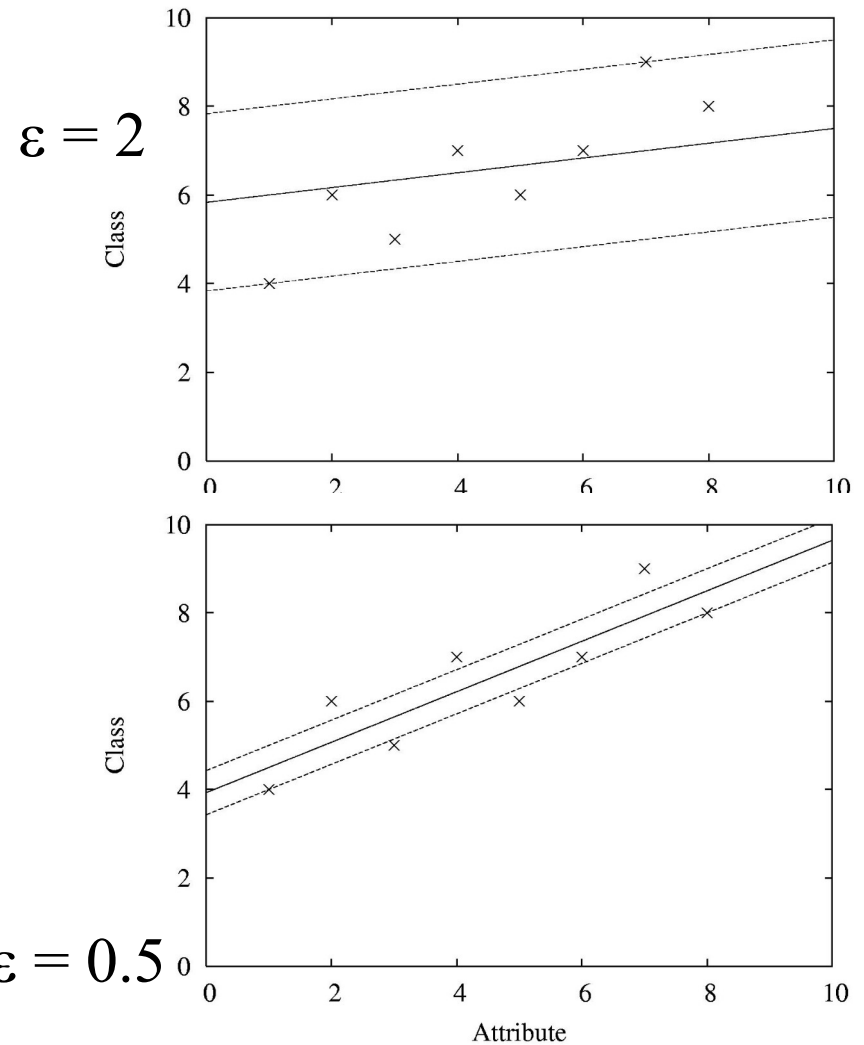
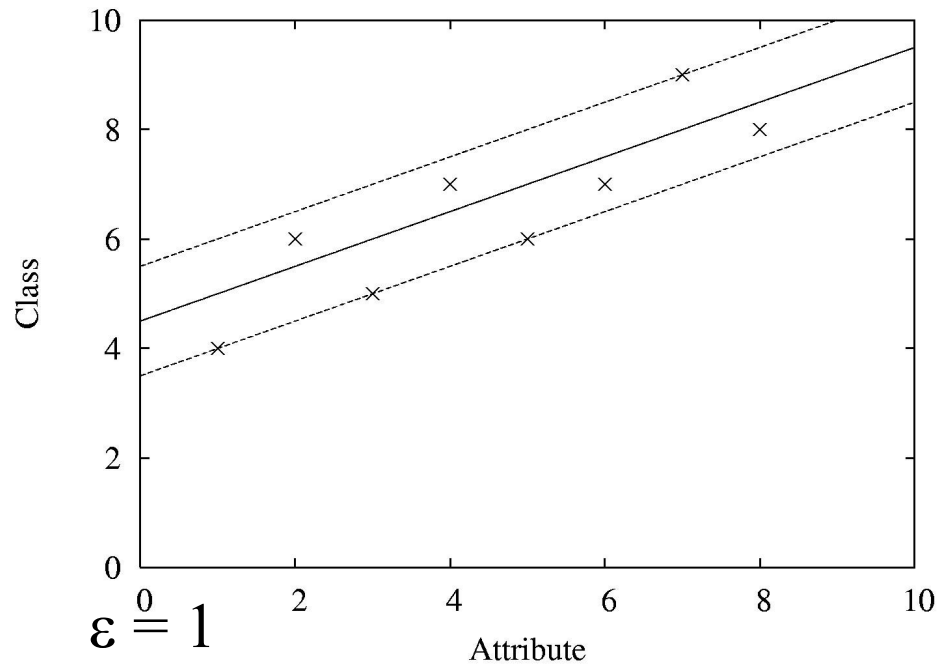
Linear Regression and SVM

- 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 ϵ and use absolute error instead of squared error
 - Difference B: simultaneously aim to maximize flatness of function
- User-specified parameter ϵ defines “tube”

Examples

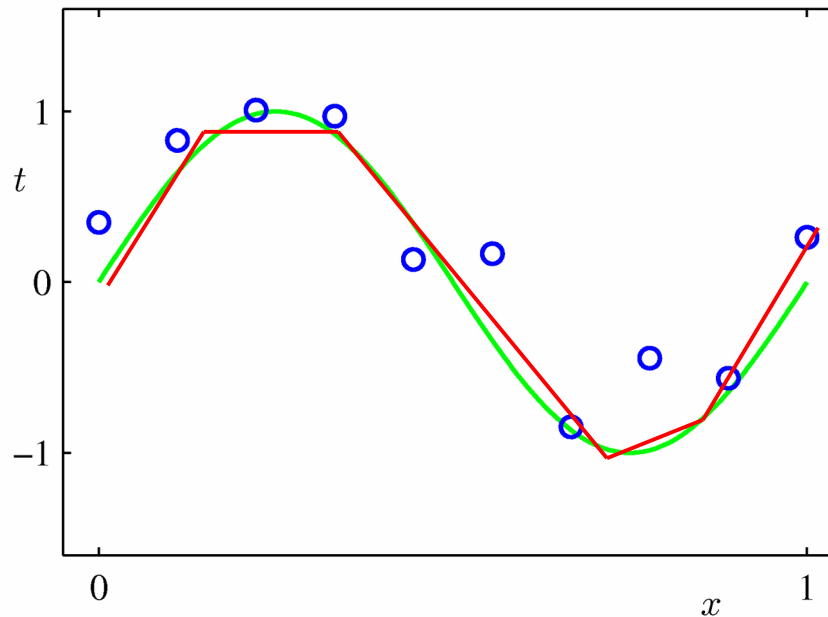


SVMs vs. Polynomial Regression

- Polynomial regression:
 - Create polynomial recombinations as additional features
 - Perform linear regression
- Support vector machine
 - Transform vector space with kernel function (e.g., polynomial kernel)
 - Find linear hyperplane

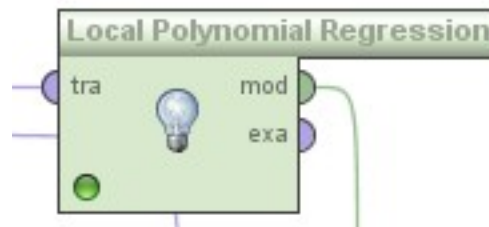
Local Regression

- 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

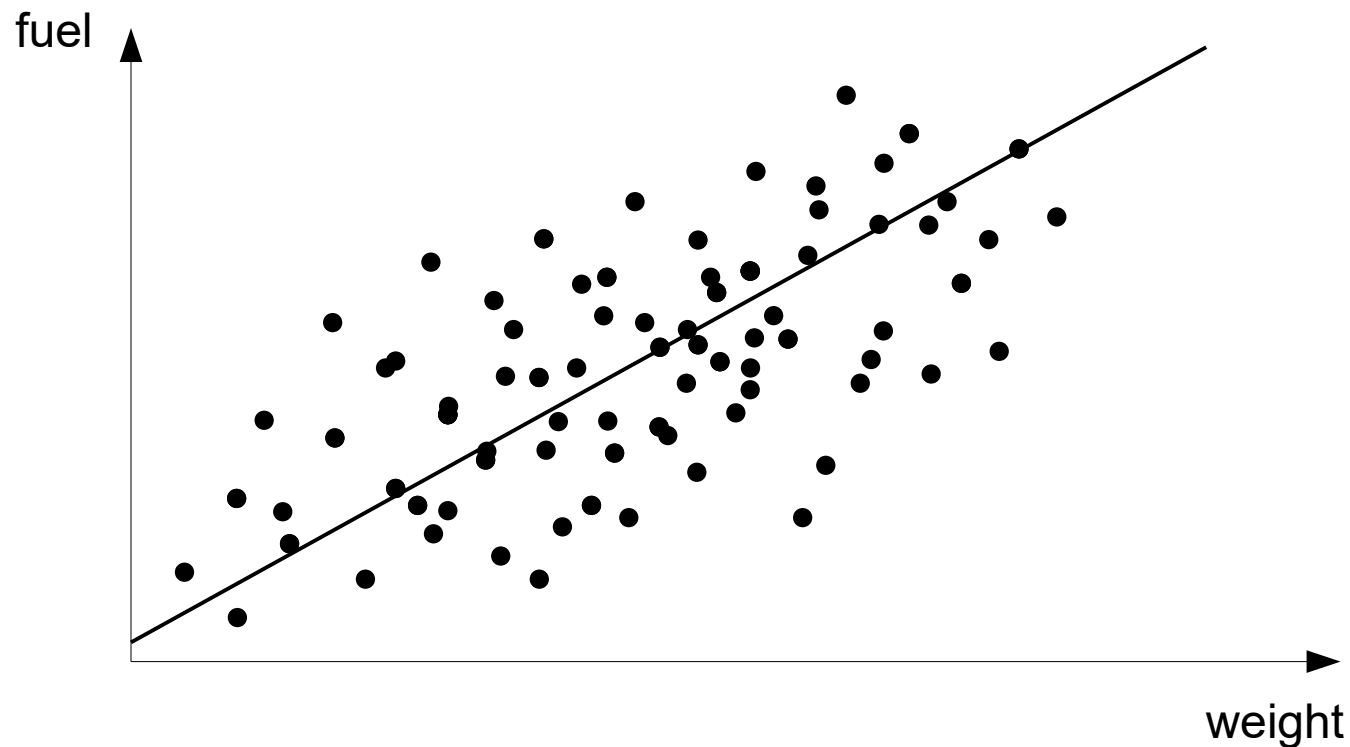


Local Regression

- 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

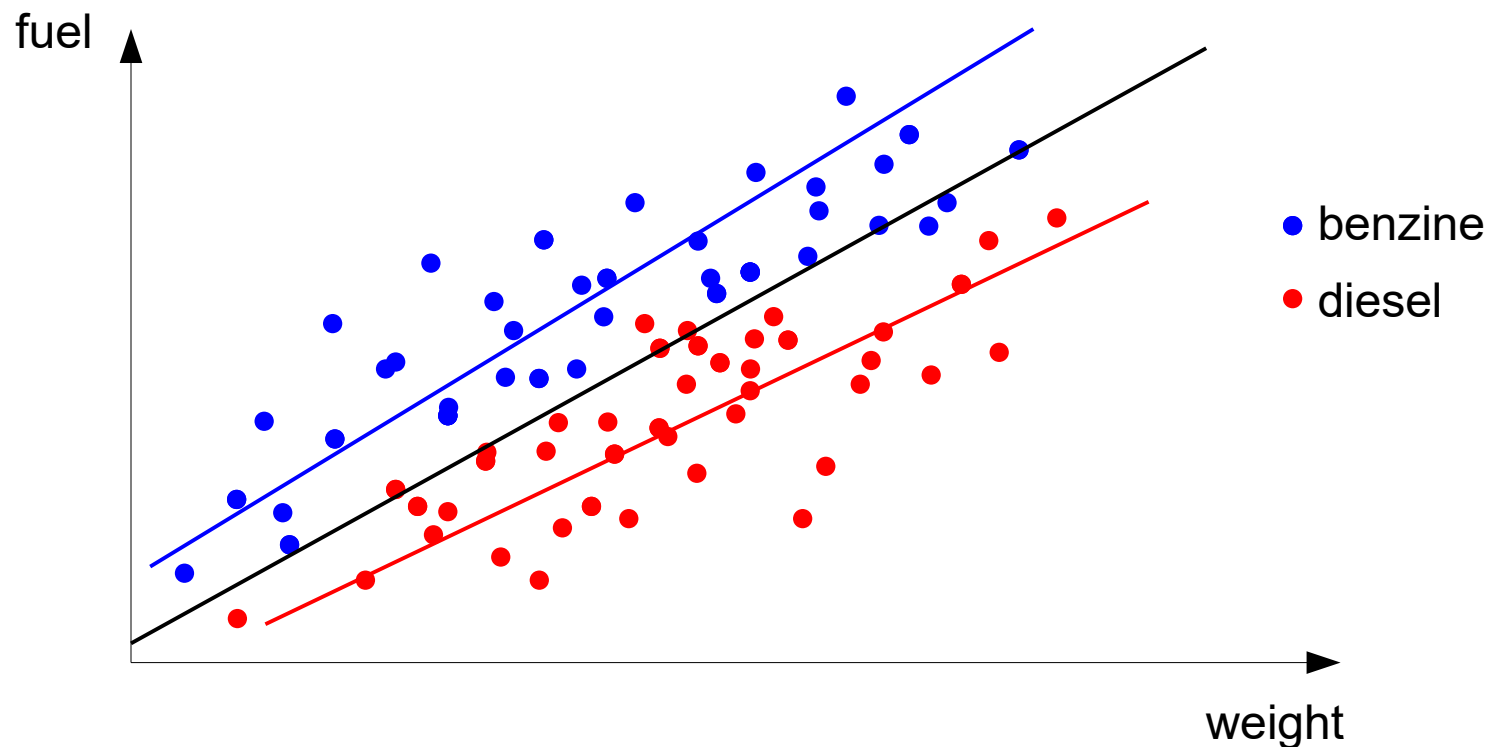
Combining Decision Trees and Regression

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



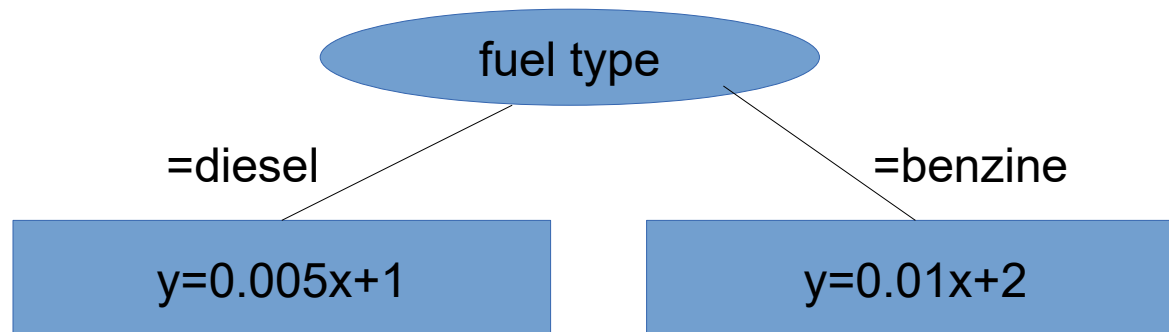
Combining Decision Trees and Regression

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



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



Regression Trees

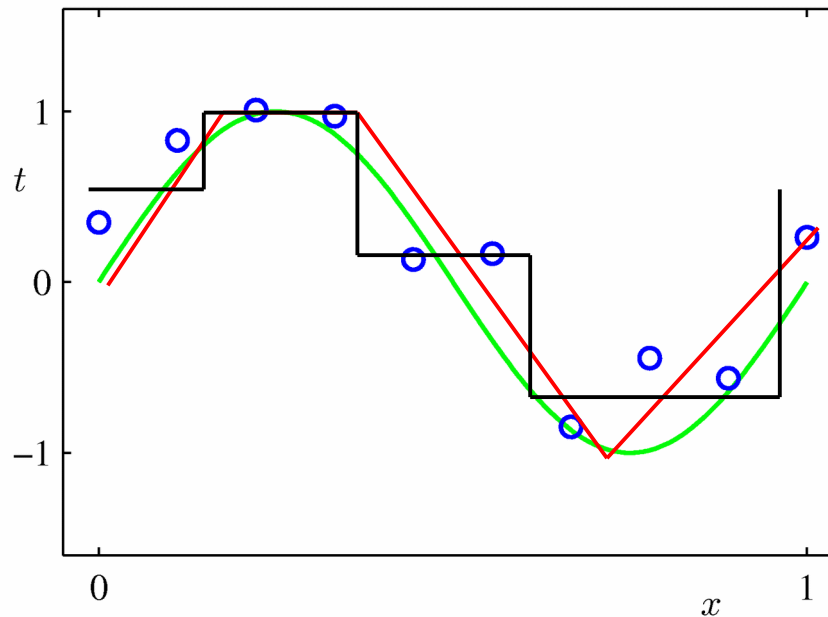
- Differences to classification decision trees:
 - Splitting criterion: minimize intra-subset variation
 - Termination criterion: standard deviation becomes small
 - Pruning criterion: based on numeric error measure
 - Prediction: Leaf predicts average class values of instances
- Easy to interpret
-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

Model Trees

- Build a regression tree
 - For each leaf \Rightarrow 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)



—
piecewise linear
(model tree)

—
piecewise constant
(regression tree)

Building the Tree

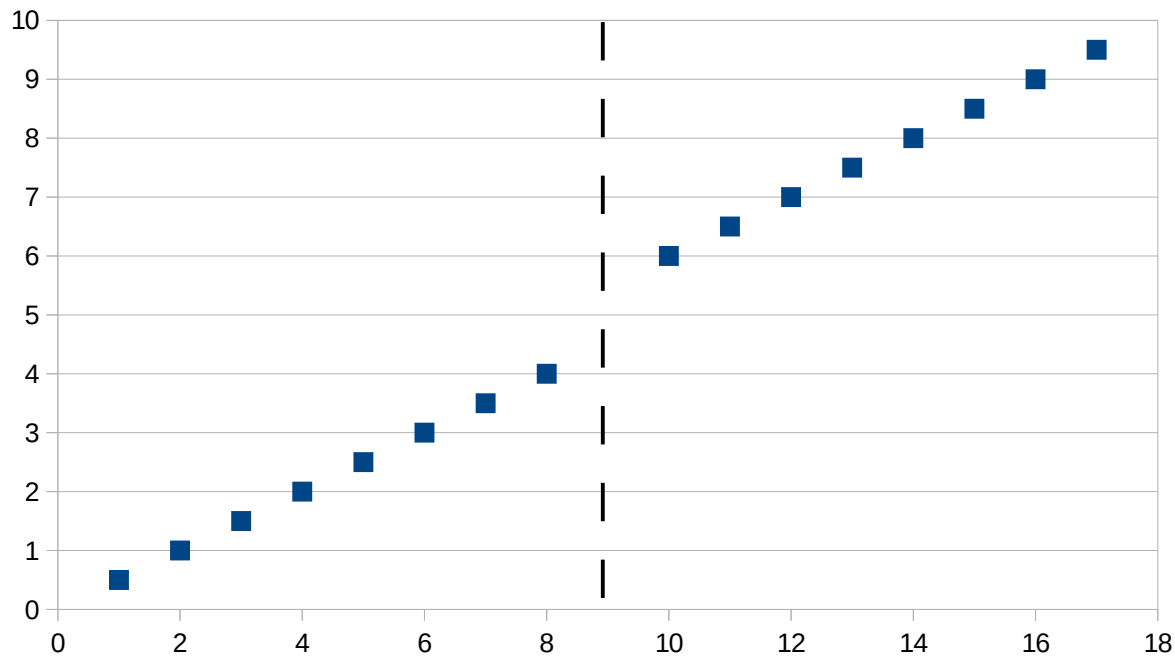
- Splitting: standard deviation reduction

$$SDR = sd(T) - \sum_i \left| \frac{T_i}{T} \right| \times sd(T_i)$$

- 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

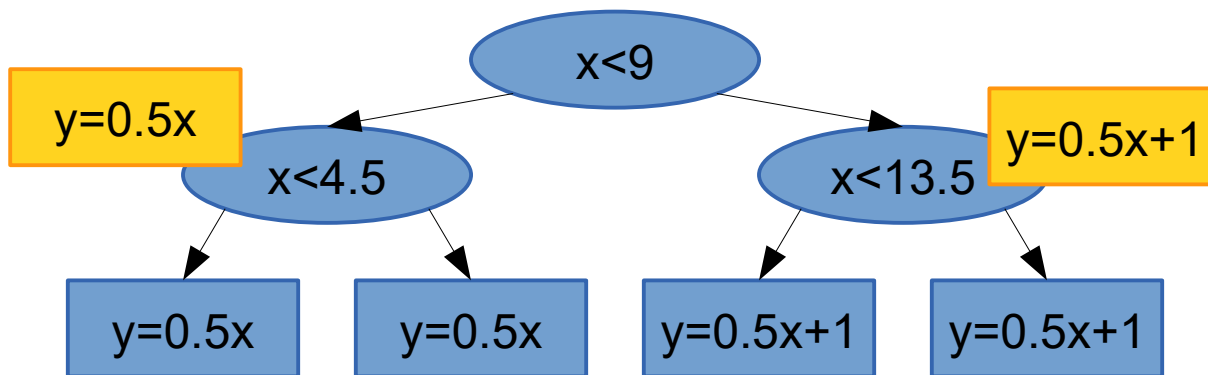
Model Tree Learning Illustrated

- 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

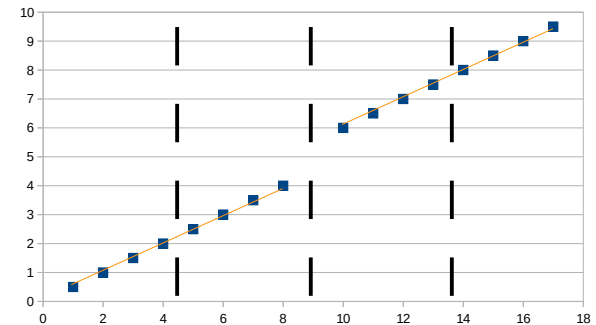


Model Tree Learning Illustrated

- 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:

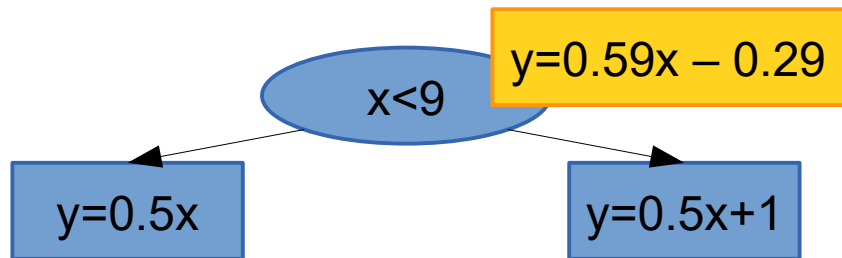


- The error of the inner nodes is the same as for the root nodes → prune

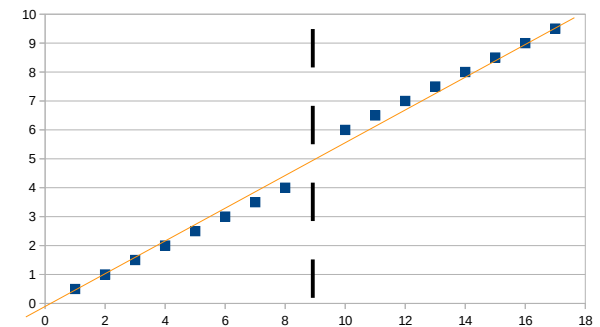


Model Tree Learning Illustrated

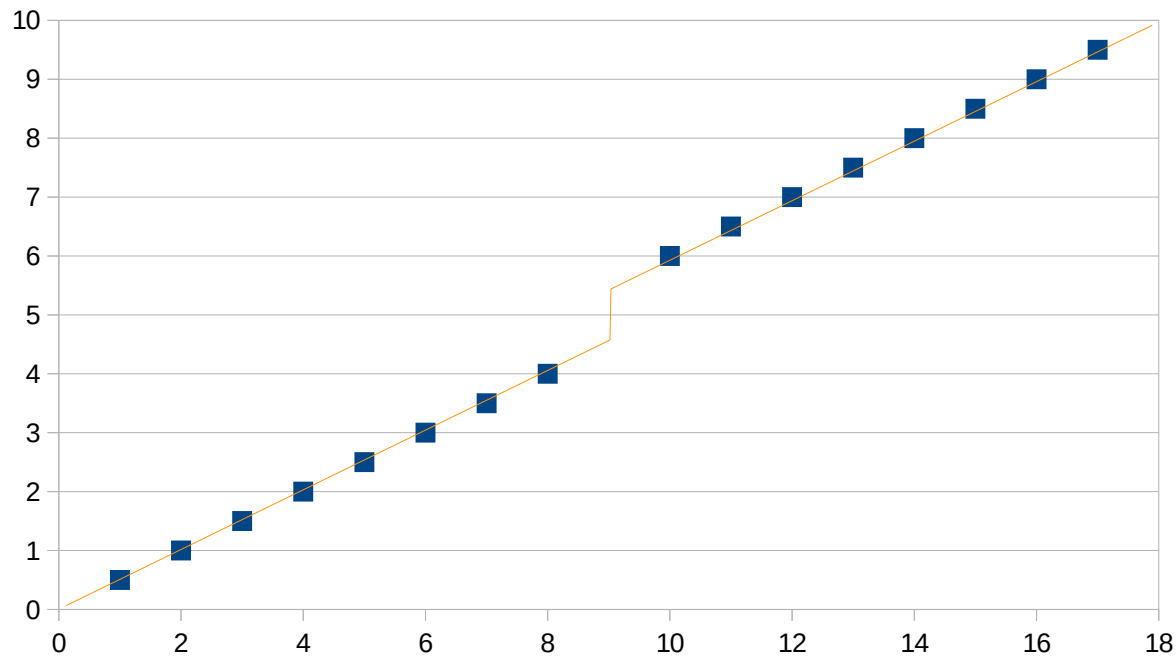
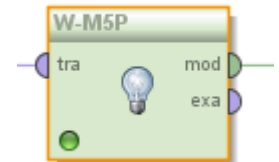
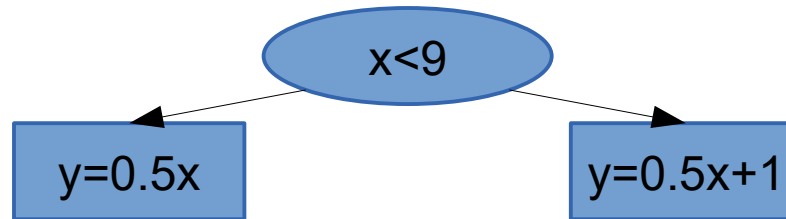
- Assume that we have split further (min. 4 instances per leaf)
 - Standard deviation reduction for the new splits is still 0.57
- Resulting model tree:



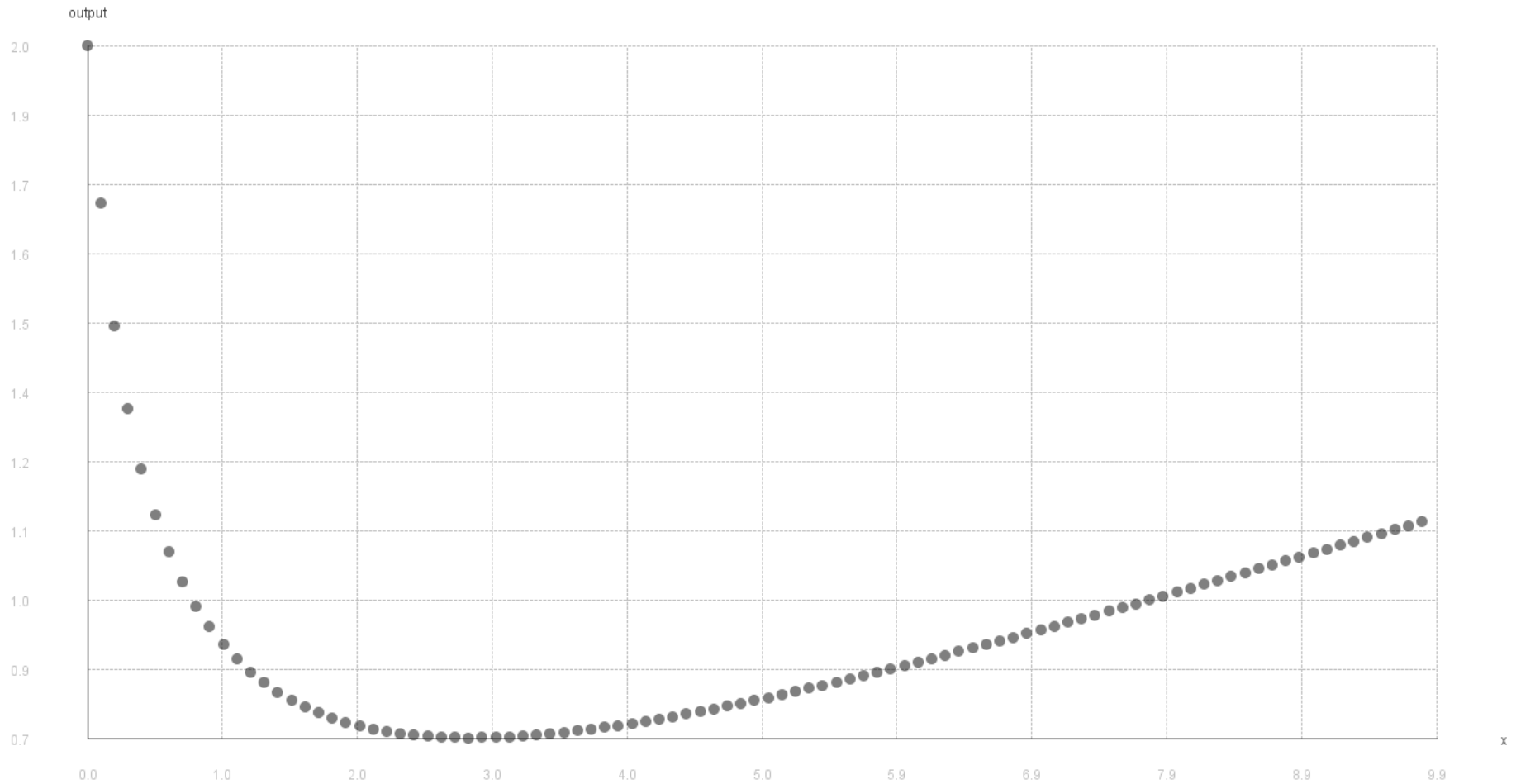
- The error of the root node is larger than that of the leaf nodes \rightarrow keep leaf nodes



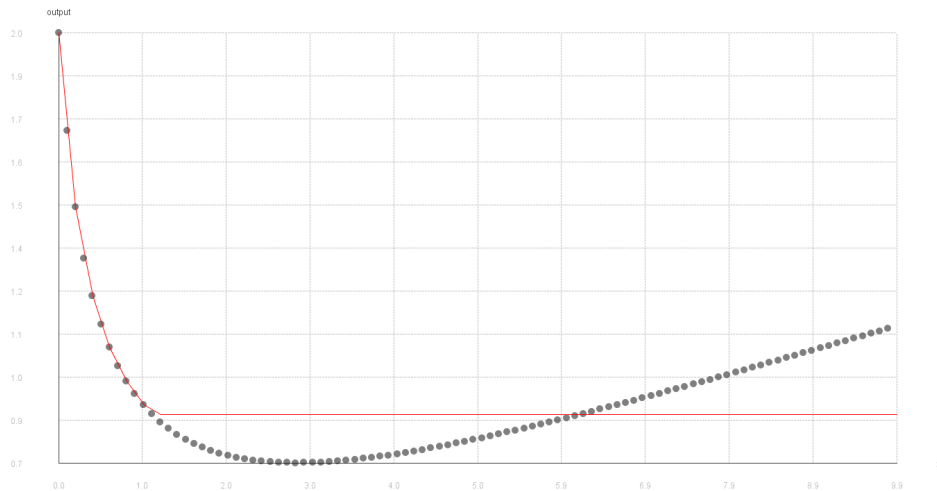
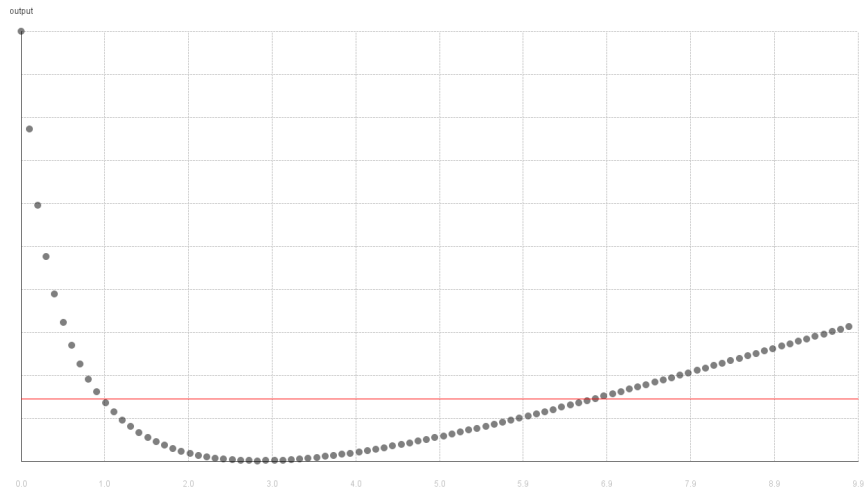
Model Tree Learning Illustrated



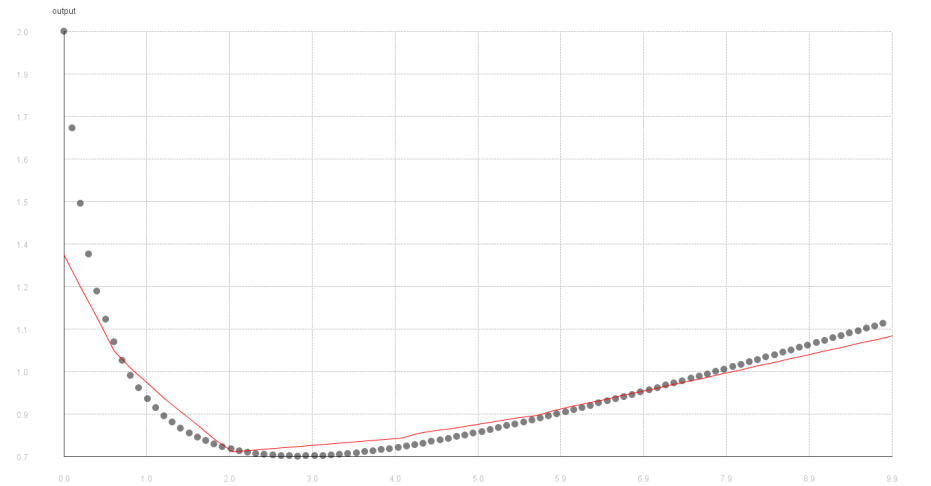
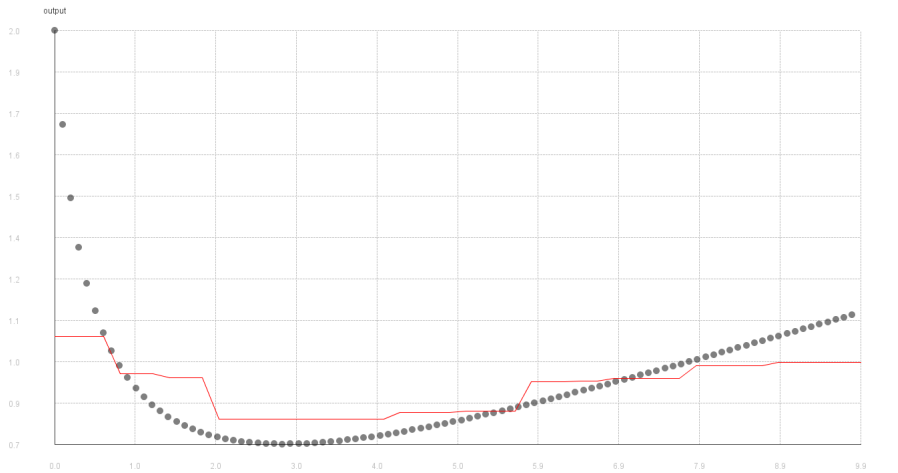
Comparison



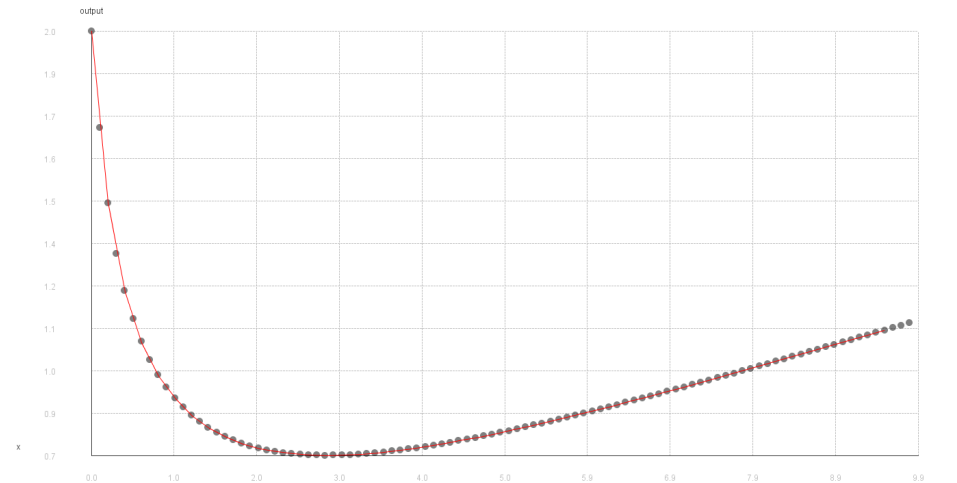
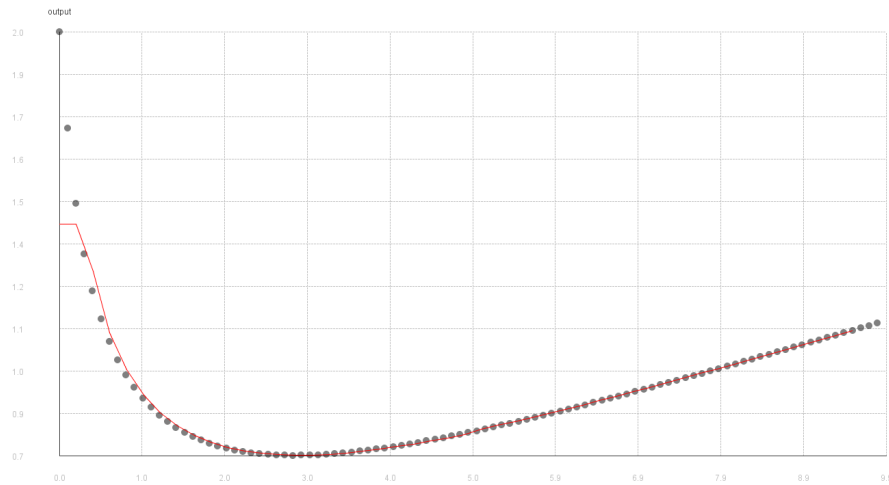
Comparison – Linear and Isotonic Regression



Comparison – M5' Regression and Model Tree

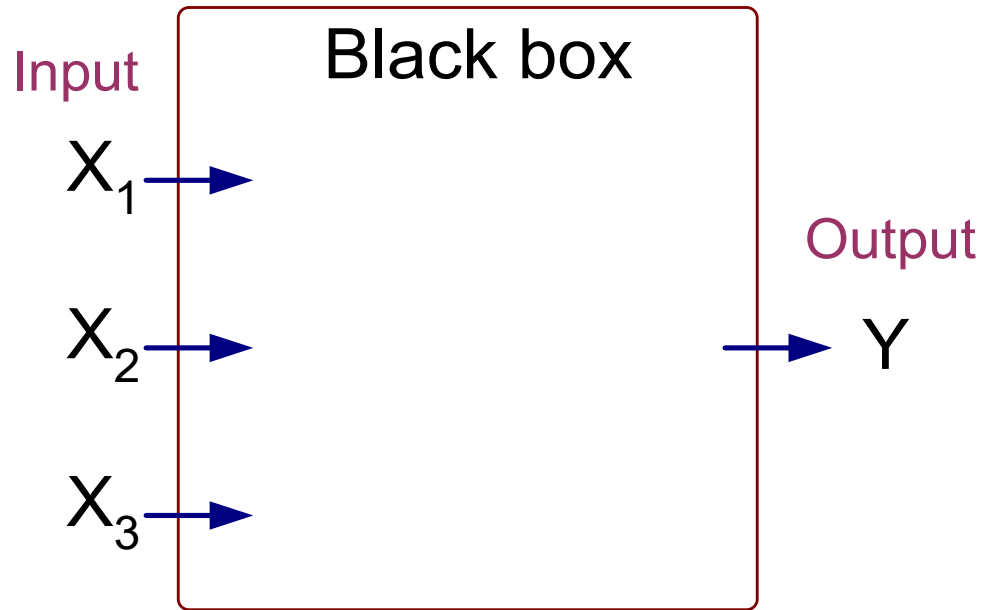


k-NN and Local Polynomial Regression (k=7)



Artificial Neural Networks Revisited

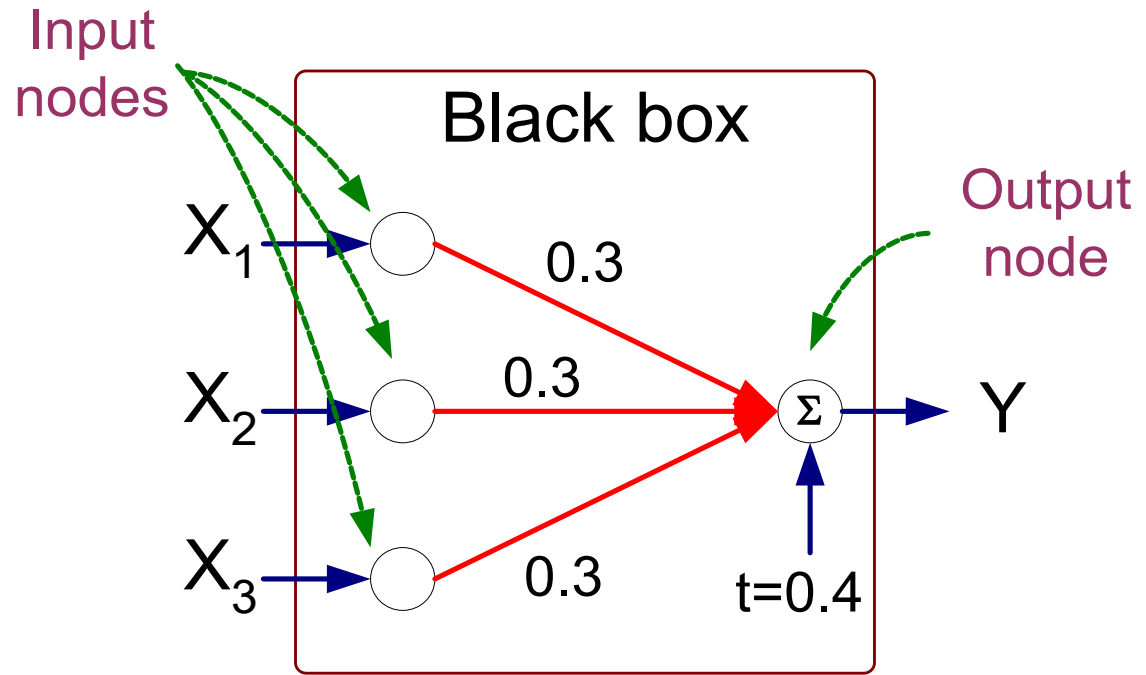
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks Revisited

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 Revisited

- This final function was used to separate two classes:

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

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

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

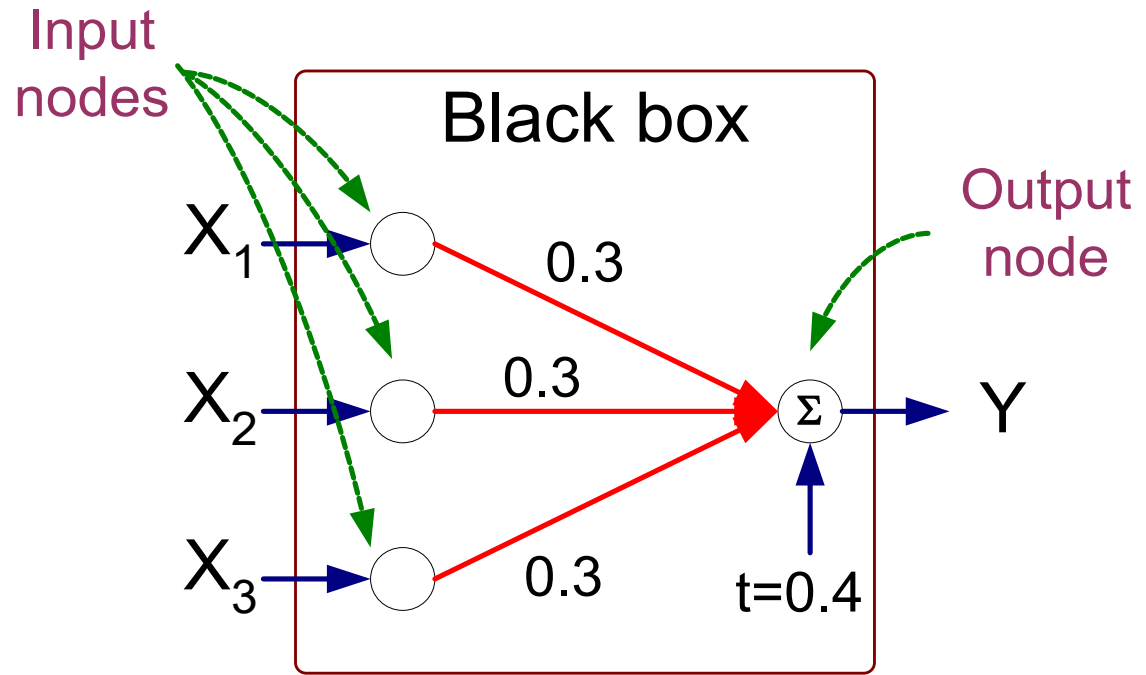
$$Y = 0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4$$

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

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



$$Y = 0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4$$

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
 - in neural networks: hidden layers
 - with non-linear activation functions!
 - this allows for arbitrary functions
- Hear more about ANNs in Data Mining 2!

From Classification to Regression and Back

- We have got to know 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 $(-\infty, 0.5]$ and true for $(0.5, \infty)$
 - similarly for ordinal (e.g., good, medium, bad)
 - non-ordinal multi-class problems are trickier

Summary

- Regression
 - predict numerical values instead of classes
- Performance measuring
 - absolute or relative error, correlation, ...
- Methods
 - k nearest neighbors
 - linear regression and regularized variants
 - polynomial regression
 - isotonic regression
 - model trees
 - artificial neural networks
 - ...

Questions?

