Project 2

Least squares regression and nearest neighbor classifiers

Lukas Drexler, Leif Van Holland, Reza Jahangiri, Mark Springer, Maximilian Thiessen

Rheinische Friedrich-Wilhelms-Universität

December 21, 2017

Least squares regression for missing value prediction

Height and weight data:

$$\mathbf{x} = [x_1, ..., x_n]^T \text{ and } \mathbf{y} = [y_1, ..., y_n]^T$$

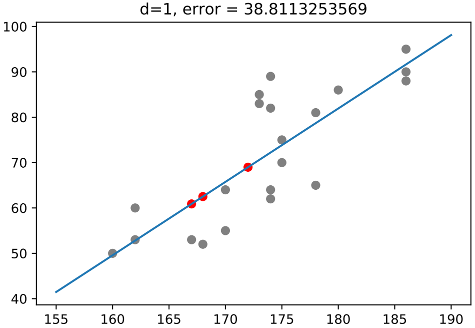
Fit polynomials:

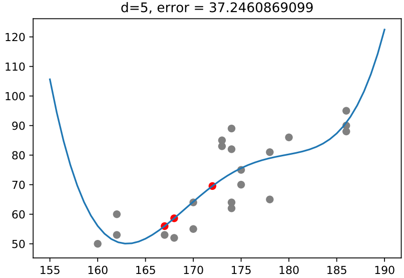
$$y(x) = \sum_{j=0}^{d} w_j x^j$$

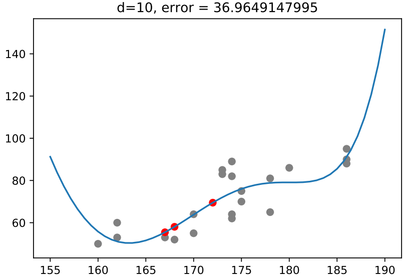
Use least squares method with Vandermonde matrix:

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ 1 & x_2 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^d \end{pmatrix}$$

Used pseudo-inverse for numerical stability


d=1, error = 38.8113253569


d=5, error = 37.2460869099


d=10, error = 36.9649147995

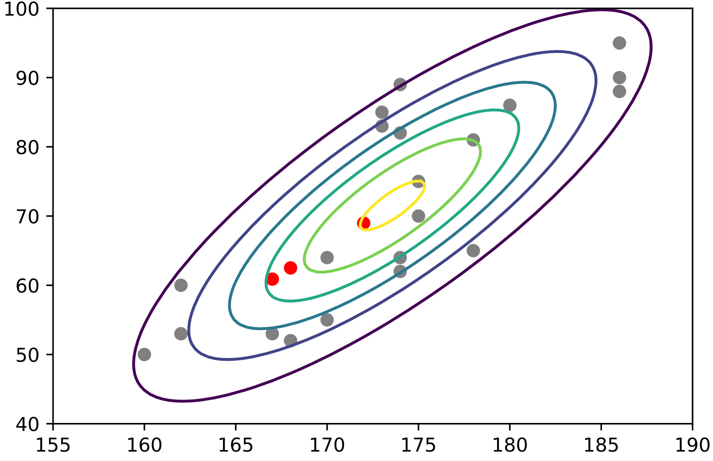
Conditional expectation for missing value prediction

Fit bi-variate Gaussian and use conditional expectation for missing value prediction:

$$
\begin{aligned}
\mathbb{E}[w|h_0] &= \int wp(w|h_0)dw \\
&= \mu_w + \rho\frac{\sigma_w}{\sigma_h}(h_0 - \mu_h)
\end{aligned}
$$

Numeric Results

The red points have the predicted weight.



Bayesian regression for missing value prediction

Compare fifth degree polynomial

$$y(x) = \sum_{j=0}^{5} w_j x^j$$

to a Bayesian regression assuming a Gaussian prior

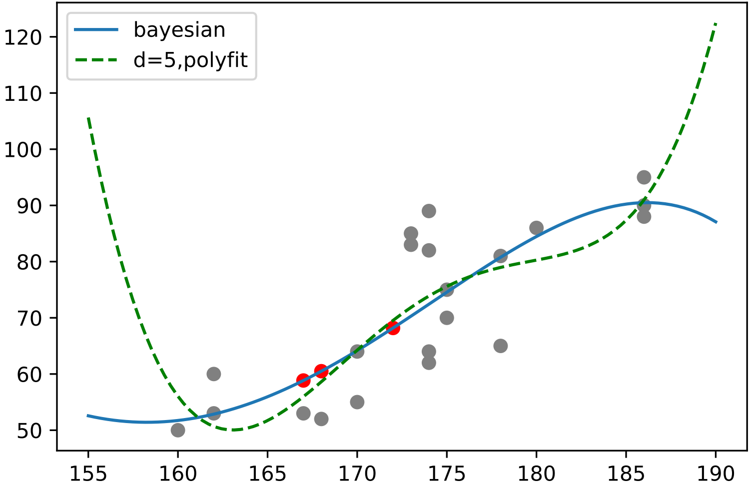
$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{w}|\mu_0, \sigma_0^2 \mathbf{I})$$

with $\mu_0 = \mathbf{0}$ and $\sigma_0^2 = 3$

Use:

$$w = (\mathbf{X}^T \mathbf{X} + \frac{\sigma^2}{\sigma_0^2})^{-1} \mathbf{X}^T \mathbf{y}$$

Results


bayesian
d=5,polyfit
120
110
100
90
80
70
60
50
155
160
165
170
175
180
185
190

Boolean functions and the Boolean Fourier transform

Subtask 1.
Rule 110 and 126 are given by

$$y_{110} = (-1, 1, 1, 1, -1, 1, 1 - 1)^T$$
$$y_{126} = (-1, 1, 1, 1, 1, 1, 1 - 1)^T$$

$$w^* = \text{argmin}||\mathbf{X}w - y||^2$$

yields for $\hat{y} = \mathbf{X}w^*$:

$$\hat{y}_{110} = (-0.25, 0.25, 0.25, 0.75, -0.75, -0.25, -0.25, 0.25)^T$$
$$\hat{y}_{126} = (0, 0, 0, 0, 0, 0, 0, 0)^T$$

Subtask 2. There are 2^m different basis functions $\phi_{S_i} : \{-1,1\}^m \to \{-1,1\}, i \in \{1,...,2^m\}$, and we have 2^m different input vectors $x_1, x_2, ..., x_{2^m} \in \{-1,1\}^m$.
Thus:

$$\begin{pmatrix} f(x_1) \\ \vdots \\ f(x_{2^m}) \end{pmatrix} = \begin{pmatrix} \phi_{S_1}(x_1) \dots \phi_{S_{2^m}}(x_1) \\ \vdots \qquad \ddots \qquad \vdots \\ \phi_{S_1}(x_{2^m}) \dots \phi_{S_{2^m}}(x_{2^m}) \end{pmatrix} \cdot \begin{pmatrix} w_{S_1} \\ \vdots \\ w_{S_{2^m}} \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_{2^m})^T \end{pmatrix}}_{=:\mathbf{\Phi}} \cdot \underbrace{\begin{pmatrix} w_{S_1} \\ \vdots \\ w_{S_{2^m}} \end{pmatrix}}_{=:w} = \mathbf{\Phi} w$$

Subtask 3. Minimizing

$$w^* = \text{argmin}||\mathbf{\Phi} w - y||^2$$

yields for $\mathbf{\Phi} w^*$:

$$\hat{y}_{110} = (-1, 1, 1, 1, -1, 1, 1 - 1)^T$$
$$\hat{y}_{126} = (-1, 1, 1, 1, 1, 1, 1 - 1)^T$$

The reconstructed rules match the original ones.

(Naive) k nearest neighbor

Use the train data set as a prediction for the test data set

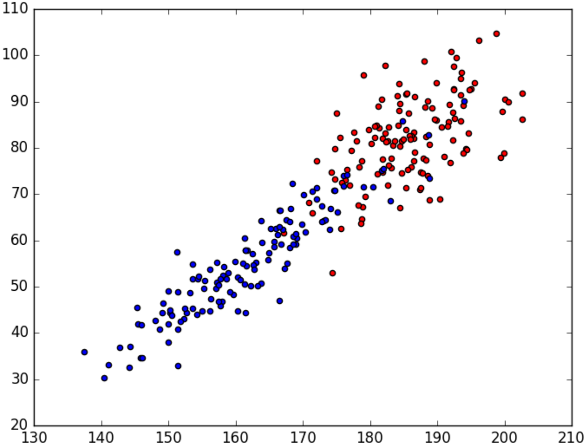
Compute the k nearest neighbors for each data point in the data set

And use the majorant of those k labels as a prediction

Experiment on `data2.dat` with $k \in \{1, 3, 5\}$

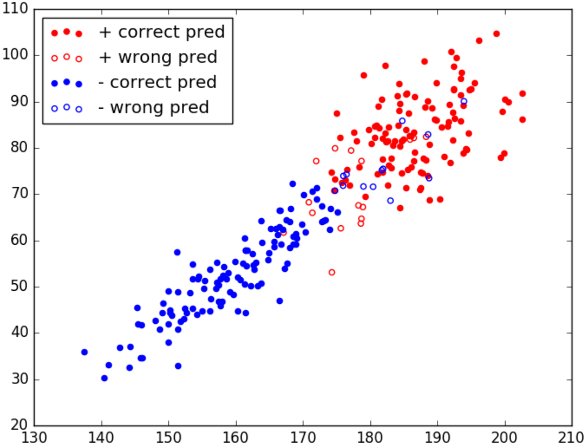
Results prediction

Correct labels of test data:



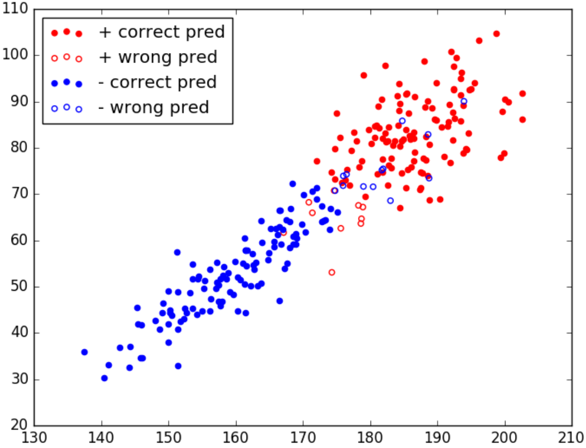
Results prediction

Prediction of 1NN:



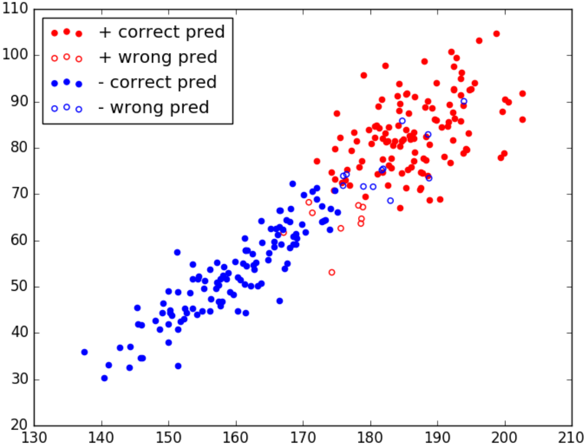
Results prediction

Prediction of 3NN:



Results prediction

Prediction of 5NN:



Accuracy and running time

Accuracies of kNN predictions:

- 0.77 for $k = 1$
- 0.82 for $k = 3$
- 0.83 for $k = 5$

Running times of our distance calculation vs `sklearn.metrics.pairwise`:

k	our implementation	`sklearn`
1	0.02s	0.003s
3	0.02s	0.003s
5	0.02s	0.003s

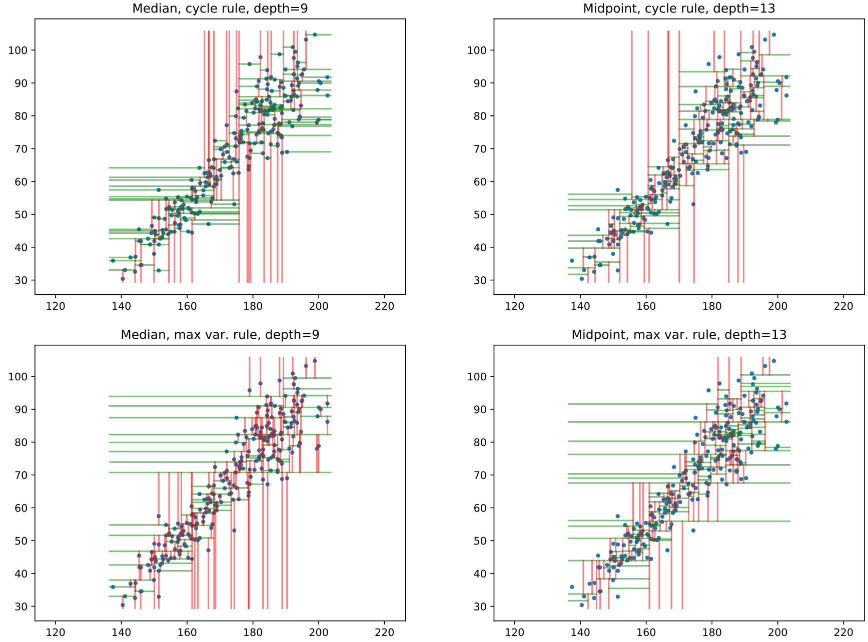
KDTrees

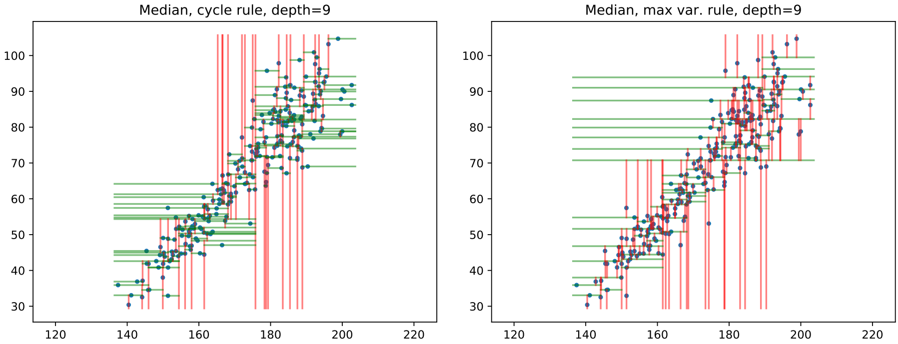
Plot four different KDTrees for combinations of axis-cycling rules:

- cycle through axes
- select axis with highest variance

and the split point rules w.r.t. the splitting axis:

- select the median point
- select the midpoint


Median, cycle rule, depth=9
Midpoint, cycle rule, depth=13
Median, max var. rule, depth=9
Midpoint, max var. rule, depth=13


Median, cycle rule, depth=9
Median, max var. rule, depth=9
100
90
80
70
60
50
40
30
120
140
160
180
200
220

Timings for 1-NN per combination

	Median	Midpoint
Cycle	0.01425s	0.00994s
Max. Var	0.01360s	0.01053s

Table: Mean running time in seconds, 100 runs

Thank you for your attention!