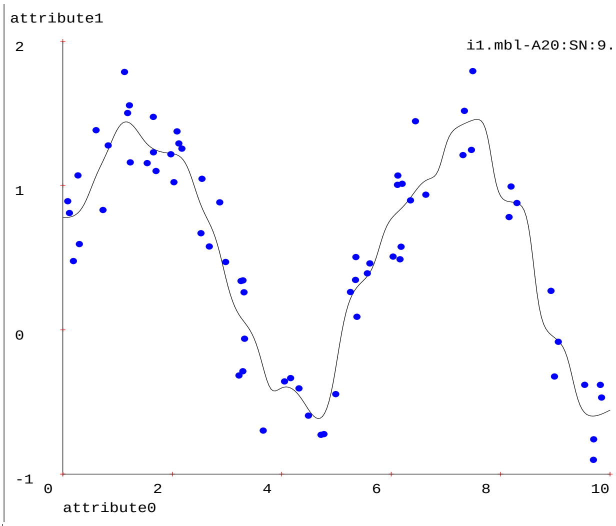
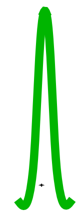
Kernel, Linear , Quadratic...

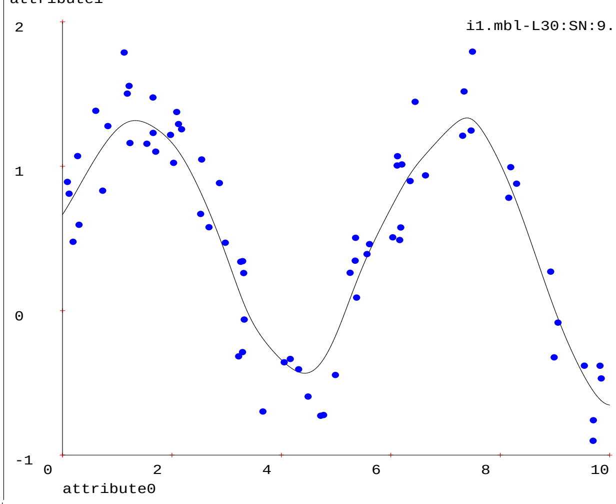


Kernel Regression (locally weighted average)

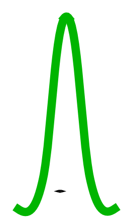


$K_W = 5$

Applying facode L30:SN:9 to file i1.mbl

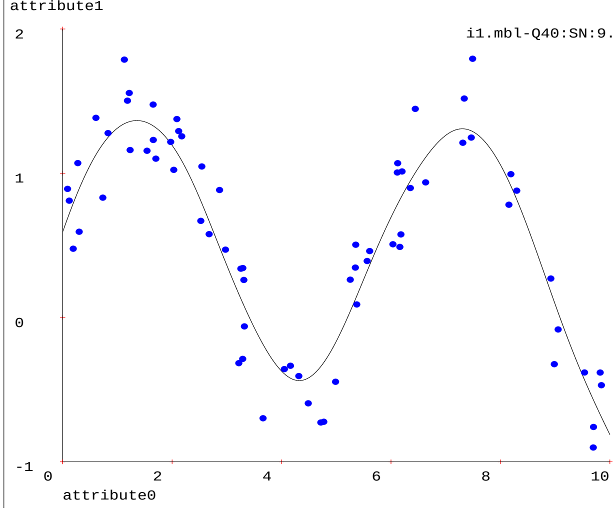


Locally weighted linear regression

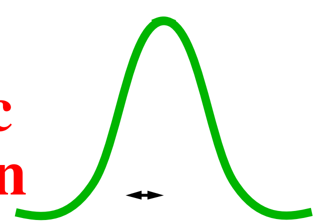


$K_W = 20$

Applying facode Q40:SN:9 to file i1.mbl



Locally weighted quadratic regression



$K_W = 80$

(Kernel widths optimized by cross validation)

Nice things about L   WPR

1. It naturally extends to multiple inputs / multiple outputs.

- 1 input, 1 parameter: T   o find β_1 to minimize weighted sum-squares residual $\sum_k w_k (y_k - \beta_1 x_k)^2$, choose

$$\beta_1 = \sum_k w_k^2 y_k x_k \,/\, \sum_k w_k^2 x_k^2.$$

- N inputs: T o find $\beta_1 \ldots \beta_N$ to minimize weighted sum-squares residual $\sum_k w_k (y_k - \beta_1 x_{k1} - \beta_2 x_{k2} - .. \beta_N x_{kN})^2$, choose

$$\beta = \left(\sum_k w_k^2 \mathbf{x}_k^{\mathrm{T}} \mathbf{x}_k\right)^{-1} \left(\sum_k w_k^2 \mathbf{x}_k^{\mathrm{T}} y_k \right).$$

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_N \end{bmatrix} = \begin{bmatrix} \sum w_k^2 x_{k1}^2 & \sum w_k^2 x_{k1} x_{k2} & .. & \sum w_k^2 x_{k1} x_{kN} \\ \sum w_k^2 x_{k1} x_{k2} & \sum w_k^2 x_{k2}^2 & .. & \sum w_k^2 x_{k2} x_{kN} \\ \vdots & \vdots & & \vdots \\ \sum w_k^2 x_{k1} x_{kN} & \sum w_k^2 x_{k2} x_{kN} & .. & \sum w_k^2 x_{kN}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum w_k^2 x_{k1} y_k \\ \sum w_k^2 x_{k2} y_k \\ \vdots \\ \sum w_k^2 x_{kN} y_k \end{bmatrix}$$

Nice things about L WPR

1. It naturally extends to multiple inputs / multiple outputs .

2. **There' s no interference: new data doesn't cause unlearning of old parts of the model.**

Nice things about L WPR

1. It naturally extends to multiple inputs / multiple outputs .
2. There’ s no interference: new data doesn’t cause unlearning of old parts of the model.
3. **Incorporating new data is very cheap!**

Nice things about L   WPR

1. It naturally extends to multiple inputs / multiple outputs   .

2. There' s no interference: new data doesn't cause unlearning of old parts of the model.

3. Incorporating new data is very c   heap!

4. **Nicely handles functions where the output is linear in some variables and non-linear in others.**

Suppose

$$y(x_1,x_2,x_3,x_4) = f(x_1)\,x_2 + g(x_1)\,x_3 + h(x_1)\,x_4$$

where $f(x_1)$, $g(x_1)$, $h(x_1)$ are complicated, non-linear .

- Then do L WR using a weighting function that only pa   ys attention to x_1.

Nice things about L   WPR

1. It naturally extends to multiple inputs / multiple outputs   .
2. There' s no interference: new data doesn't cause unlearning of old parts of the model.
3. Incorporating new data is very c   heap!
4. Handles functions where the output is linear in some variables and non-linear in others very naturally   .
5. **Leave-One-Out cross-validation is cheap**

Nice things about L WPR

1. It naturally extends to multiple inputs / multiple outputs .
2. There' s no interference: new data doesn't cause unlearning of old parts of the model.
3. Incorporating new data is very c heap!
4. Handles functions where the output is linear in some variables and non-linear in others very naturally .
5. Lea ve-One-Out cross-validation is c heap
6. **You don't just get a prediction. Y ou can get a full regression analysis!**

Bayesian locally weighted regression analysis

Plausibleish model of the meaning of the weights:

$$y_k = \beta_1 x_{k1} + \beta_2 x_{k2} + .. \beta_N x_{kN} + noise_k$$

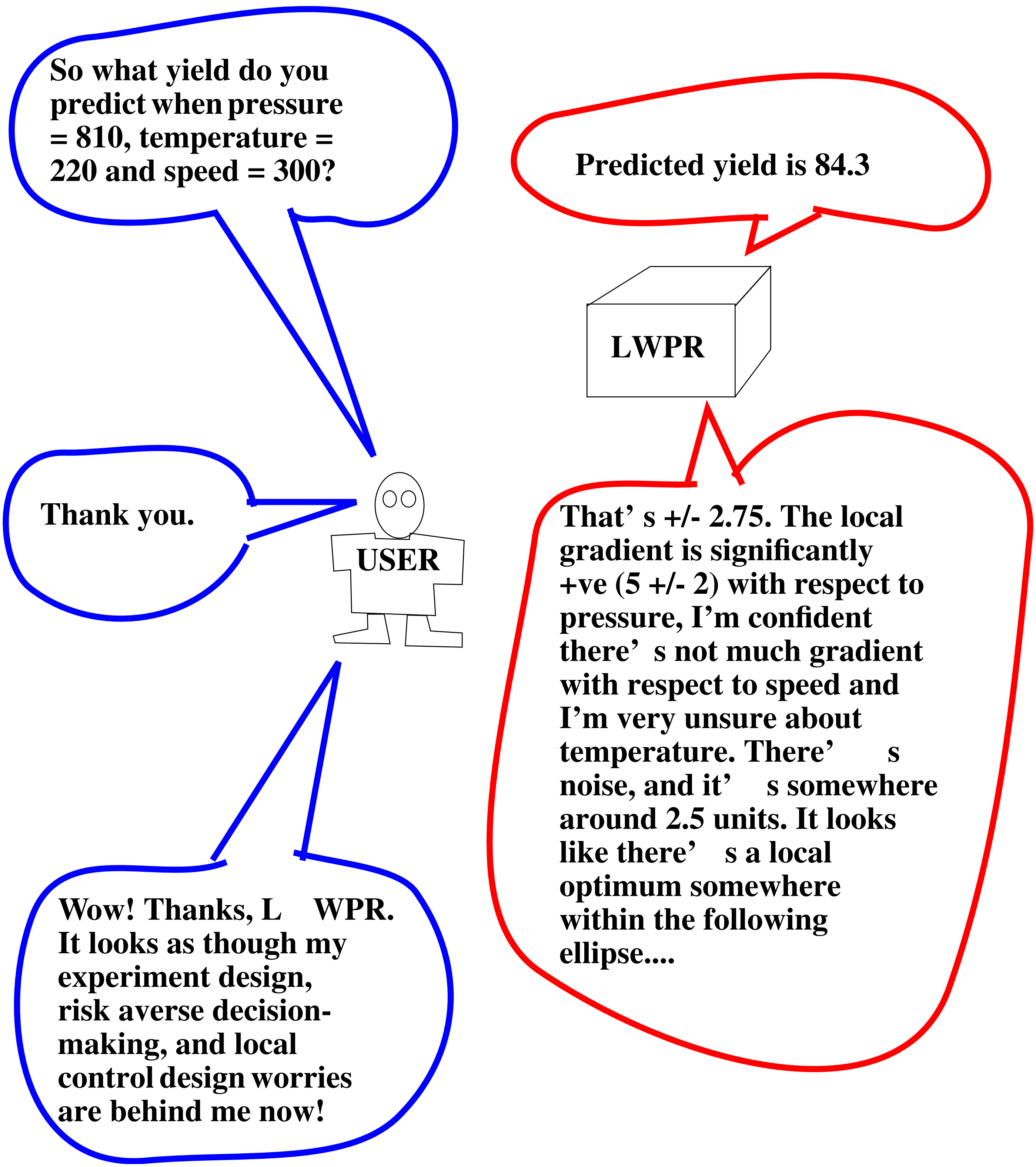
$$\text{where } noise_k \sim \text{Normal}(0, \sigma^2 / w_k).$$

Then you get all these goodies for free using conventional Ba yesian regression analysis

- You ma y (optionally) give prior values for $\beta_1, \beta_2 ... \beta_N$ and σ^2.
- You ma y state how confident you are in the priors .
- You can then get a posterior distribution on $\beta_1, \beta_2 ... \beta_N$ and σ^2 around your query

(See for example , [Gelman, Carlin, Stern and Rubin])

LWPR Regression Analysis



LWPR is good for making decisions

- We have used it for a number of industrial processes with very nonlinear models, including packaging control and cooling tunnel control.
- Useful for non-linear forward models in robotics (e.g. a pool robot [Moore, 1992])
- Confidence intervals on predictions can help exploration. E.g, Atkeson and Schaal's devil sticking robot. Cohn's kinematics explorer. Schneider's ball-thrower.
- Local Gradients and Hessians essential for learning optimal control.
- Confidence intervals on a learned LWPR model can be important for risk-averse reinforcement learning [See Schneider's Reinforcement Learning Workshop talk].

LWPR: The Dark Side

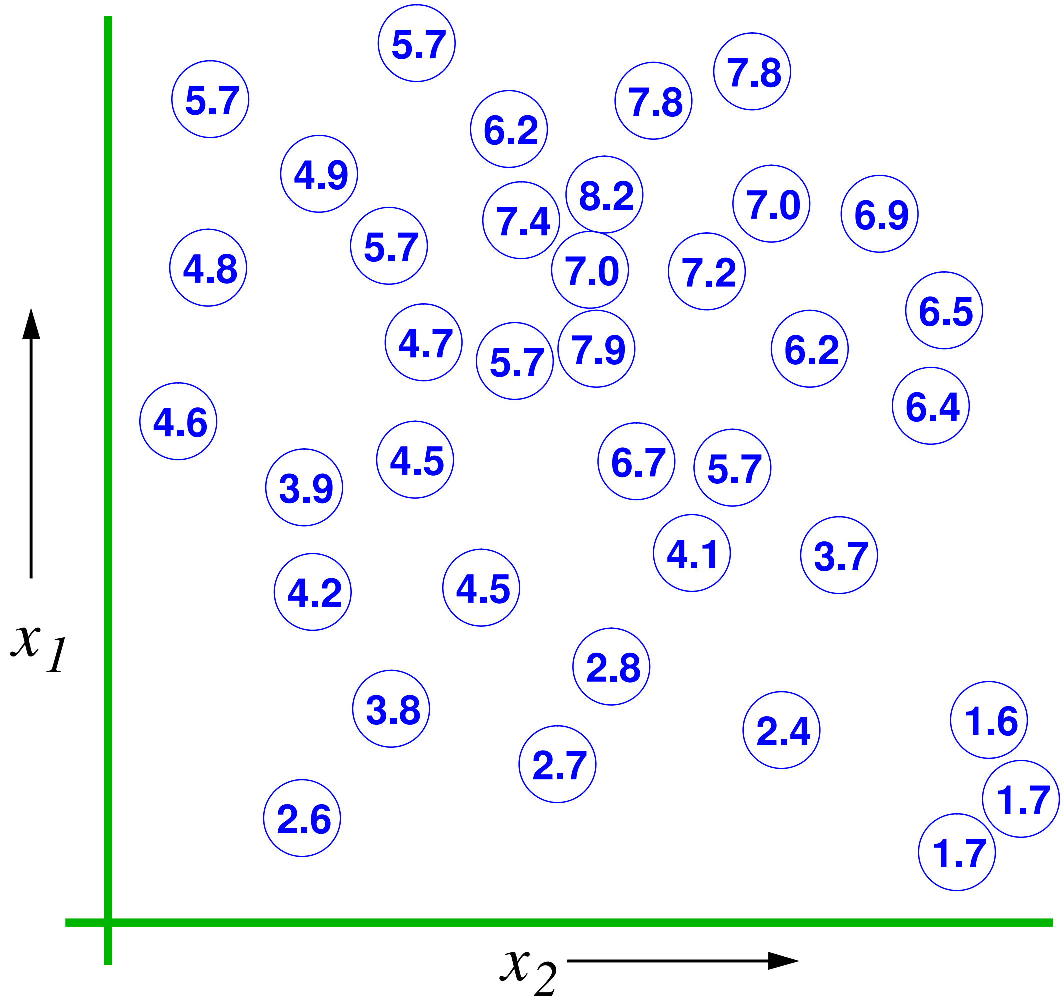
Cost of each prediction grows linearly with the number of datapoints.

With N datapoints , and with M polynomial terms ....

- training is FREE
- eac h individual prediction costs $O(NM^2 + M^3)$

What can we do????

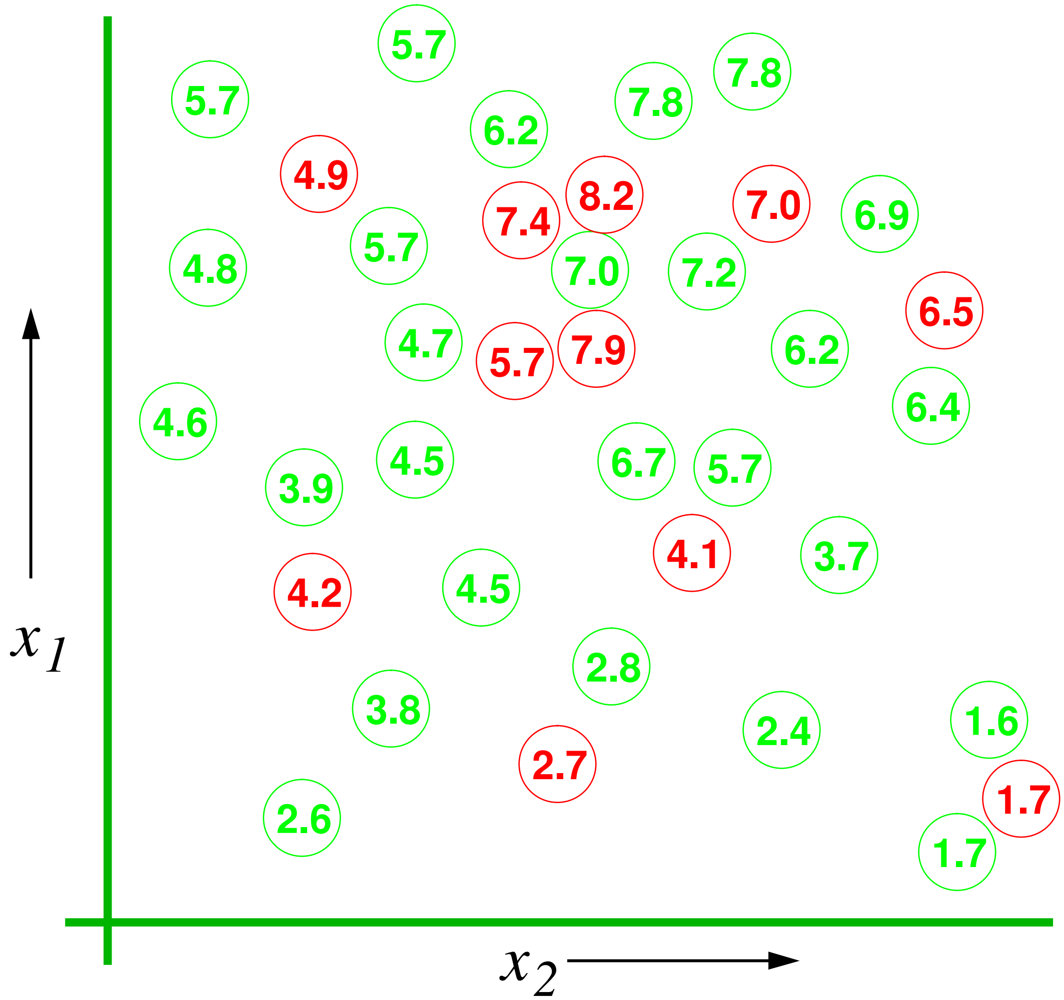
Idea One: Editing



Here we have $N=36$ examples, each with two inputs (x_1,x_2) and one output.

Instead of retaining and searching over all N, we could choose to keep only a few "representative" points called **prototypes**.

Idea One: Editing



- Examples: [Kibler and Aha, 1988; Skalak, 1994].
- Other editing tric ks: maintain a verages in the prototypes [Aha, 1992].

BUT : Lose accuracy . Lose ability to give different queries different kernel widths . Lose cheap L-O-O . Hard to estimate noise .