Robust Regression
Robust Regression
Robust Regression

This is the best fit that Quadratic Regression can manage
Robust Regression

...but this is what we’d probably prefer
LOESS-based Robust Regression

After the initial fit, score each datapoint according to how well it’s fitted...

You are a very good datapoint.
LOESS-based Robust Regression

After the initial fit, score each datapoint according to how well it’s fitted...

You are a very good datapoint.

You are not too shabby.
LOESS-based Robust Regression

After the initial fit, score each datapoint according to how well it’s fitted...

But you are pathetic.

You are not too shabby.

You are a very good datapoint.
For $k = 1$ to $R$...

- Let $(x_k, y_k)$ be the $k$th datapoint
- Let $y_{est}^k$ be predicted value of $y_k$
- Let $w_k$ be a weight for datapoint $k$ that is large if the datapoint fits well and small if it fits badly:
  \[ w_k = KernelFn([y_k - y_{est}^k]^2) \]
Robust Regression

For $k = 1$ to $R$...

- Let $(x_k, y_k)$ be the $k$th datapoint
- Let $y_{est_k}$ be predicted value of $y_k$
- Let $w_k$ be a weight for datapoint $k$ that is large if the datapoint fits well and small if it fits badly:
  \[ w_k = KernelFn((y_k - y_{est_k})^2) \]

Then redo the regression using weighted datapoints.

Weighted regression was described earlier in the “vary noise” section, and is also discussed in the “Memory-based Learning” Lecture.

Guess what happens next?
Robust Regression

For $k = 1$ to $R$...

- Let $(x_k, y_k)$ be the $k$th datapoint
- Let $y_{est_k}$ be predicted value of $y_k$
- Let $w_k$ be a weight for datapoint $k$ that is large if the datapoint fits well and small if it fits badly:
  \[ w_k = \text{KernelFn}(y_k - y_{est_k})^2 \]

Then redo the regression using weighted datapoints.

I taught you how to do this in the “Instance-based” lecture (only then the weights depended on distance in input-space)

Repeat whole thing until converged!
Robust Regression---what we’re doing

What regular regression does:

Assume $y_k$ was originally generated using the following recipe:

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + N(0, \sigma^2)$$

Computational task is to find the Maximum Likelihood $\beta_0, \beta_1$ and $\beta_2$. 
Robust Regression---what we’re doing

What LOESS robust regression does:

Assume $y_k$ was originally generated using the following recipe:

With probability $p$:

$$y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + \mathcal{N}(0, \sigma^2)$$

But otherwise

$$y_k \sim \mathcal{N}(\mu, \sigma_{\text{huge}}^2)$$

Computational task is to find the Maximum Likelihood $\beta_0$, $\beta_1$, $\beta_2$, $p$, $\mu$ and $\sigma_{\text{huge}}$. 
Robust Regression---what we’re doing

What LOESS robust regression does:
Assume $y_k$ was originally generated using the following recipe:

With probability $p$:
$$y_k = \beta_0 + \beta_1 x_k + \beta_2 x_k^2 + N(0, \sigma^2)$$

But otherwise
$$y_k \sim N(\mu, \sigma_{\text{huge}}^2)$$

Computational task is to find the Maximum Likelihood estimates for
$$\beta_1, \beta_2, p, \mu \text{ and } \sigma_{\text{huge}}$$

Mysteriously, the reweighting procedure does this computation for us.

Your first glimpse of two spectacular letters: E.M.
Citations

Radial Basis Functions

LOESS
W. S. Cleveland, Robust Locally Weighted Regression and Smoothing Scatterplots, Journal of the American Statistical Association, 74, 368, 829-836, December, 1979