WEEK 4

LINEAR REGRESSION

THE LINEAR MODEL

→ is shown by the equation:

$$f(x; w) = w_0 + w_1 x_1 + \cdots + w_D x_D$$

Input Data; Model Parameters

$$= \underline{\phi(x)}\,\underline{w}$$

Dot product of 2 vectors:

→ $\phi(x) = (1, x_1, \ldots, x_D) = (1, x^T)$

→

$$w = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$$

→ Let a design matrix be of size $n \times (D+1)$

$$\Phi = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1D} \\ 1 & x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nD} \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Then $\hat{y} = \Phi w$ is the model's predicted values on training inputs

SOLVING FOR MODEL PARAMETERS

→ We know y and Φ, so why can't we take $w = \Phi^{-1} y$

1) Φ is not square. It is $n \times (D+1)$.

2) The system is over-constrained. In other words...
   ↳ We have a lot more data pts. than we have dimensions. [n equations for D+1 parameters]

3) ... The data has noise

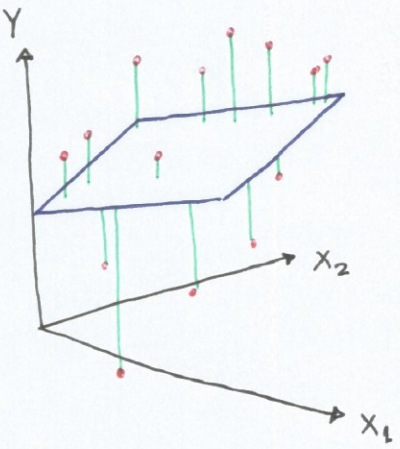
LOSS FN

→ So instead we'll define a loss/error fn of the weights $O(w)$ and we are going to try to minimize the fn w.r.t to the weights w

→ At that minimum, $\hat{y}$ looks like y

$\hat{y}$: what our model predicts

y: actual output



$$O(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2 = (y - \Phi w)^T (y - \Phi w)$$

($y_i - w^T x_i$: Residual; $\sum_{i=1}^{n}$: sum over the training patterns)

In the graph, this is the sum of squared length of green sticks. Each one is called a residual.

↳ To minimize, set partial derivatives to 0

This has an analytical soln:

$$\hat{w} = \underbrace{(\Phi^T \Phi)^{-1} \Phi^T}_{\text{Pseudo-inverse of } \Phi} y$$

↳ It isn't an inverse, but acts like one.

→ Assume tht. $y = w^T x + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$

↳ Gaussian noise

This implies that $y | x_i \sim N(w^T x_i, \sigma^2)$

i.e. $$-\log p(y_i | x_i) = \underbrace{\log \sqrt{2\pi} + \log \sigma}_{\text{constant}} + \frac{(y_i - w^T x_i)^2}{2\sigma^2}$$

So, minimising $O(w) \equiv$ maximising likelihood

→ Can view $w^T x$ as expected value of y given x $E[y|x]$

→ Can estimate variance from the squared residuals:

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

DIAGNOSTICS

→ Linear regression is sensitive to outliers, so it is useful to run graphical diagnostics → Visualize data & remove outliers before fitting!

- → Is the relationship non-linear?
- → Fit the model and look at the structure in residuals.
- → Are there obvious outliers?

MULTIPLE REGRESSION — Predict multiple output values

→ Suppose there are q diff. targets for each input x

→ We introduce a diff. weight vector w_i for each target dimension and do regression separately for each output variable

BASIS EXPANSION

→ Linear regression express the expected target as a linear fn of features

However, the true underlying fn will typically not be linear

→ Capturing this non-linearity in the model might yield more predictive power

→ HOW TO DO THIS? → We can augment the input features x w/ some nonlinear transformations into $\phi(x)$ and do linear regression to them

i.e. You can run dataset through polynomials, Gaussians, sigmoids, ...

Design matrix is now $n \times m$ — m: no. of diff. transforms we're going to try

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_m(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_m(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_m(x_n) \end{pmatrix}$$

→ You can now cope w/ problems where there isn't a linear relationship between input and outputs

i.e. Suppose I want to predict the CPU performance and one of the features is *manufacturer*.

$$x_1 = \begin{cases} 1 & \text{if Intel} \\ 2 & \text{if AMD} \\ 3 & \text{if Apple} \\ 4 & \text{if Motorola} \end{cases}$$

↳ Not the way you want; Do you believe AMD is twice faster than INTEL

↳ Instead, assign each category a new binary attribute:

$x_1 = 1$ if Intel, 0 otherwise

$x_2 = 1$ if AMD, 0 otherwise

⋮

RADIAL BASIS FUNCTION (RBF) MODELS

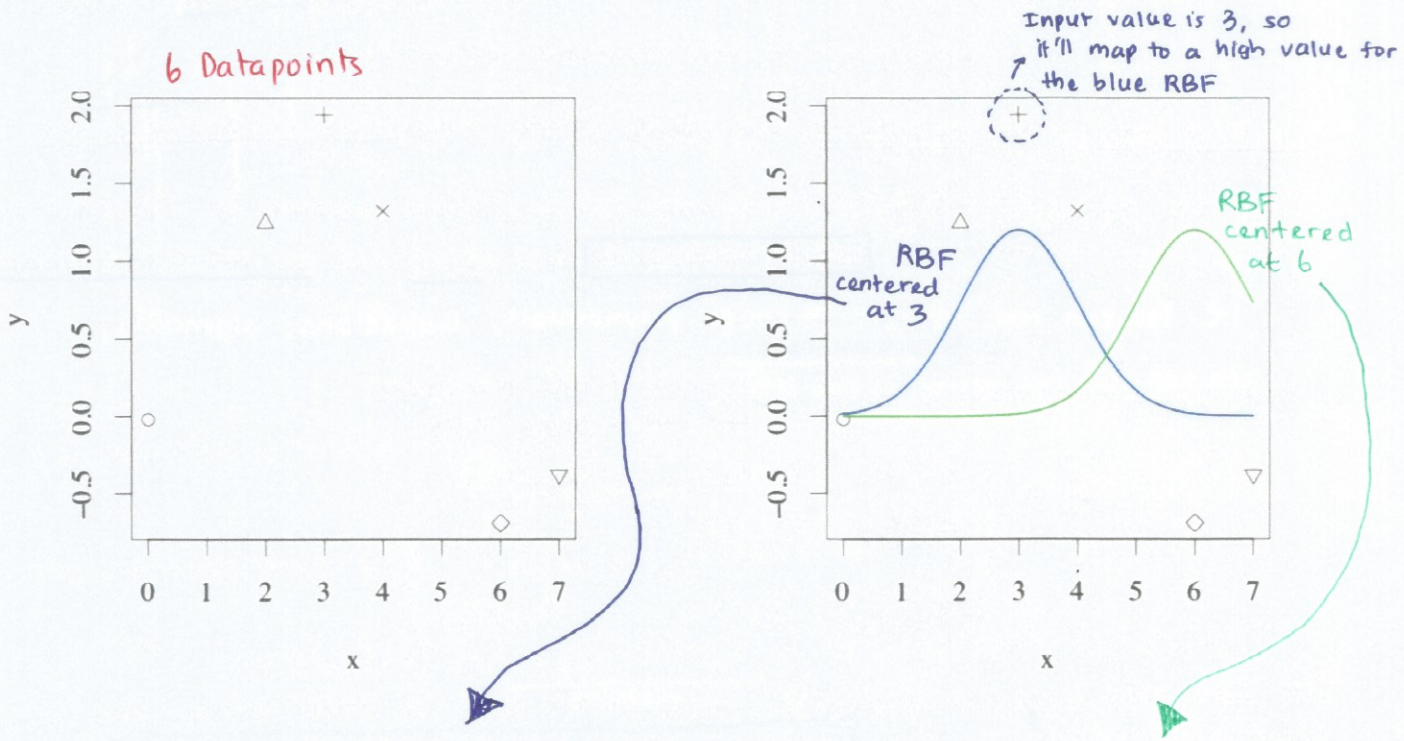
→ is a kind of transformation by setting:

$$\phi_i(x) = \exp\left(-\frac{1}{2}|x - c_i|^2 / \alpha^2\right)$$

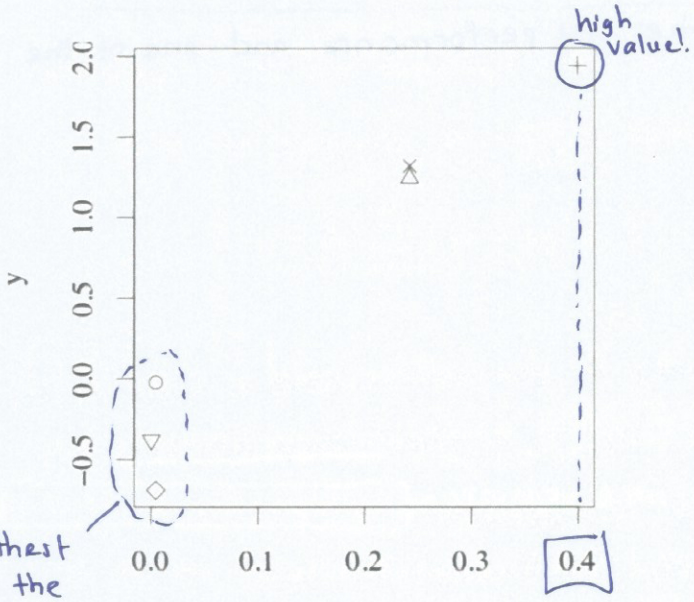
(2 parameters)

→ We pick a certain number for the centre point c_i.
Many ways to do this but choosing a subset of the datapts. as centres is an effective method

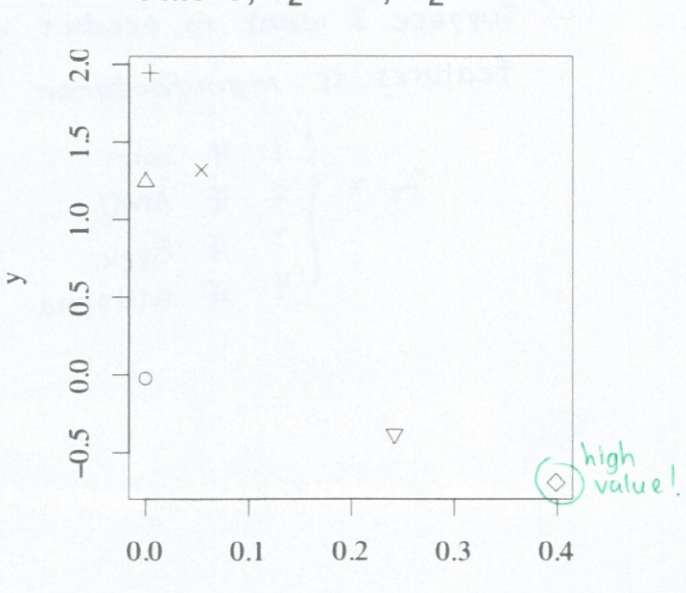
→ Finding the weights is the same → Pseudo-Inverse soln



RBF feature, $c_1 = 3$, $\alpha_1 = 1$



RBF feature, $c_2 = 6$, $\alpha_2 = 1$

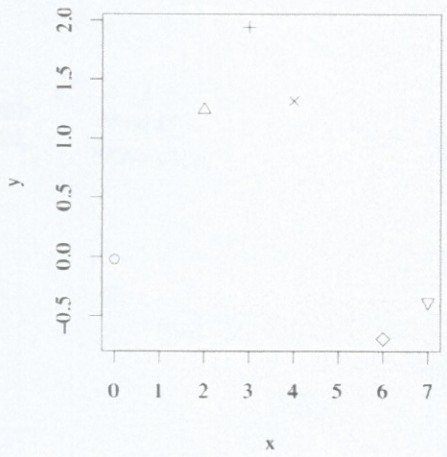


Run the RBF with both basis functions above, plot the residuals

→ we've turned our 1D problem to 2D problem bc. we've turned our single feature value into 2 values (one frm. each of the RBF)

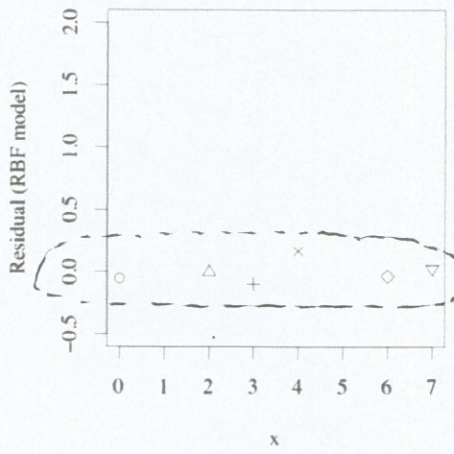
$$y_i - \phi(\mathbf{x}_i)^T \mathbf{w}$$

Original data



Residuals

→ We then fit a linear model to look at the residuals



They correspond to the difference between the actual output y_i and the linear model using the transformed features shown by ϕ

→ more/less around 0, so we can conclude tht. the linear model w/ the transformed features is a reasonable model

→ Why not use RBF for everything?

- You might need too many basis fns, especially true in high dimensions, hence you may end up overfitting the data