Machine Learning for Signal Processing

Regression and Prediction

Instructor: Bhiksha Raj

Topics

- Nearest neighbor regression and classification
- Linear regression
  - With an application to glitch elimination in sound
  - And its relation to nearest-neighbor regression
- Regression in kernel spaces
- Kernel regression
- Regularization..

Topics

- **Nearest neighbor regression and classification**
- Linear regression
  - With an application to glitch elimination in sound
  - And its relation to nearest-neighbor regression
- Regression in kernel spaces
- Kernel regression
- Regularization..

The problems of classification and regression

- Classification: Given a feature X, determine the class Y
  - Given image features, classify if this is a face
- Regression: Given an input X, estimate another feature Y
  - Given height, age, gender, etc. of a person, estimate weight
- In reality both are the same problem:
  - The class is simply a categorical feature

Example-based estimation

- Classification:
  - Have seen one or more people who are exactly 160cm, 50kg, and all are female
  - Get a new test instance of a person who is exactly 160cm, 50kg. Is this person..
    - Male?
    - Female?

- Regression:
  - Have seen one or more people who are exactly 160cm, female, and their weight is 50kg
  - Get a new test instance of a 160cm female person. What is your best guess for her weight?

Example-based estimation

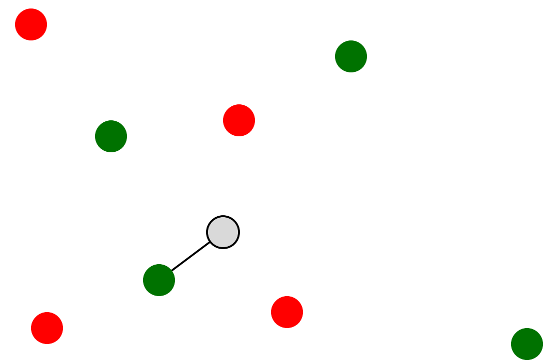
- Classification:
  - Have seen one or more people who are exactly 160cm, 50kg, and all are female
  - Get a new test instance of a person who is ex[illegible]m, 50kg. Is this person..
    - Male?
    - Female?

BUT WHAT IF THE WEIGHT OF THE TEST SUBJECT IS 49 KG?

- Regression:
  - Have seen one or more people who are exactly 160cm, female, and their weight is 50kg
  - Get a new test instance of a 160cm female person. What is your best guess for her weight?

Example based prediction

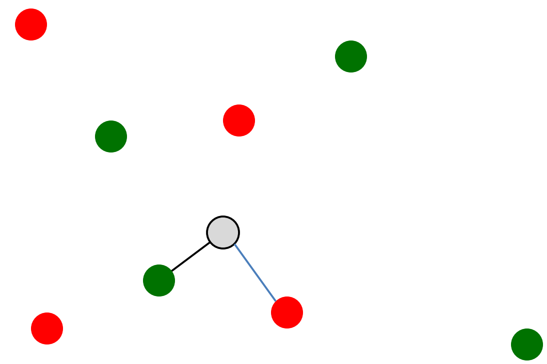
- Problem: the gray circle is missing its color attribute. Predict it



- Find the nearest training instance
  – Based on observed feature X
- Predict Y from it
  – Y may be a class value or a continuous valued estimator

Nearest-neighbor based prediction

- Problem: the gray circle is missing its color attribute. Predict it



- Find the nearest training instance
  - Based o
  - e a class value or a continuous valued estimator

Can you trust ONLY the nearest neighbor???

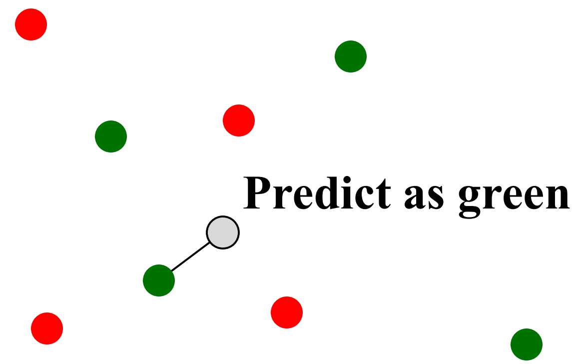
What if the next-nearest neighbor is almost as close, but gives you a different answer?

Nearest-neighbor prediction

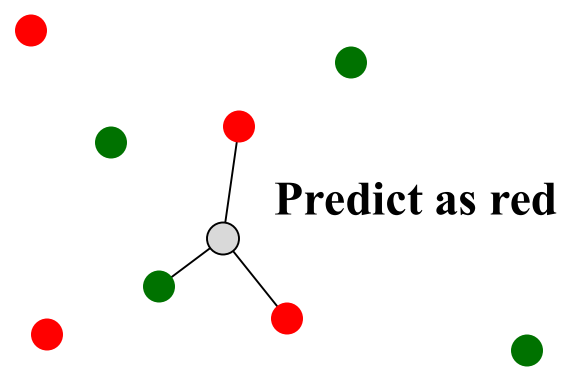
- Alternately, find the k closest training instances
  - Called the k-nearest-neighbor method
- Predict desired attribute based on these k closest neighbors

K-nearest neighbor prediction

- Problem: the gray circle is missing its color attribute. Predict it



- Nearest neighbor

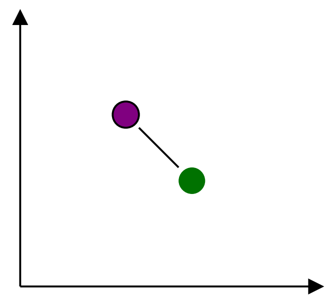


- K-nearest neighbor
  - Example for k=3

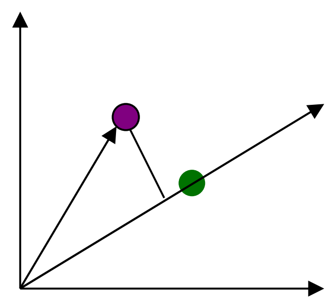
Distance functions

- How does one define the distance between two instances?
  - Some attributes may be numeric
  - Other attributes may nominal
- Numeric attributes: Usually the Euclidean distance between attribute values is used
- Nominal attributes: Usually a binary distance function – distance is set to 1 if attribute values are different, 0 if they are the same
- Will assume *numeric attributes* for our signals..

Distance on numeric features



$$d(x_1, x_2) = \|x_1 - x_2\|^2$$

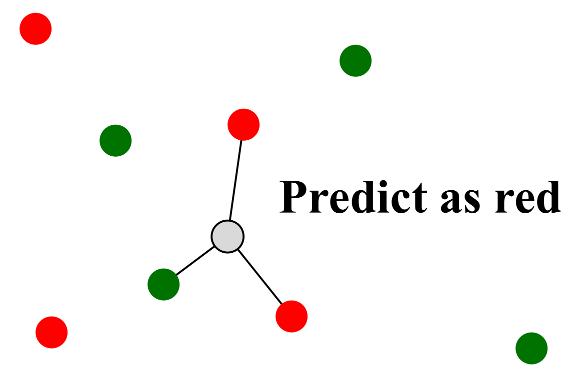


$$w(x_1, x_2) = x_1^T x_2$$

$$d(x_1, x_2) = \frac{1}{x_1^T x_2}$$

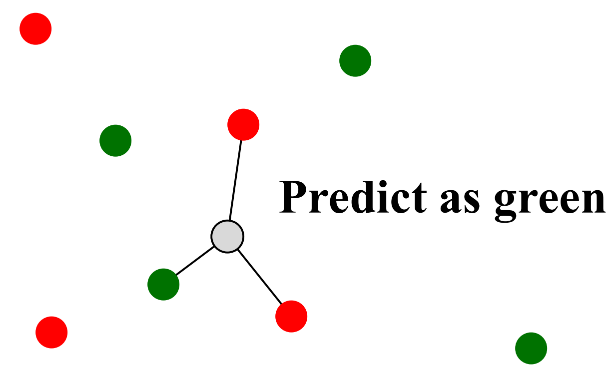
K-nearest neighbor prediction

- Find the K nearest neighbors
- Predict as the majority opinion
  - But should we also consider the actual distance
    - Is a farther neighbor as important as a closer one?
  - What about numeric prediction?
    - No notion of "majority"
      - No two neighbors may have the same value for Y



Weighted K-nearest neighbor prediction

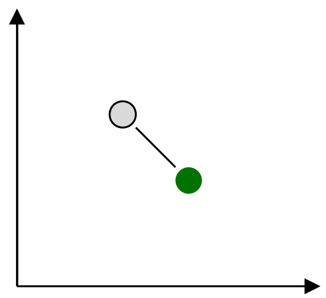
- Classification
  - $Score(class) = \sum_{i:(i \in KNN) \& \, class(i)=class} w(x, x_i)$
  - $class(x) = \underset{class}{\operatorname{argmax}} \, Score(class)$



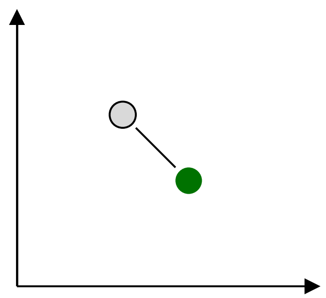
- Regression:
  - $Y(x) = \sum_{i \in KNN} w(x, x_i) Y_i$

- The weight $w(x, x_i)$ is inversely related to $d(x, x_i)$
  - If $d(x, x_i)$ increases, $w(x, x_i)$ decreases

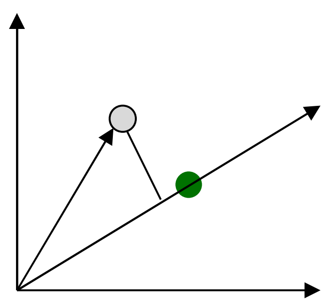
Weights of neighbors..



$$d(x_1, x_2) = \|x_1 - x_2\|^2 \qquad w(x, y) = \frac{1}{d(x, y)}$$



$$w(x_1, x_2) = exp(-\alpha d(x_1, x_2))$$

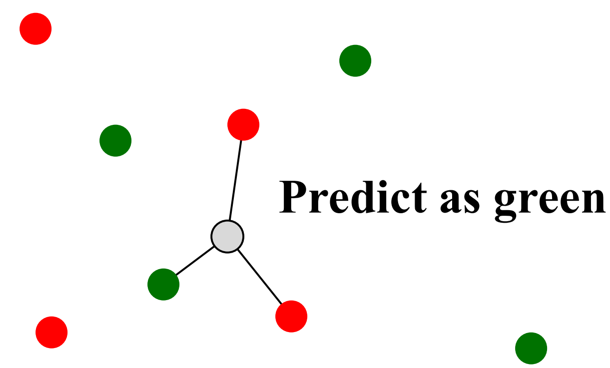


$$w(x_1, x_2) = x_1, x_2 = \ x_1^T x_2$$

Poll 1

Weighted K-nearest neighbor prediction

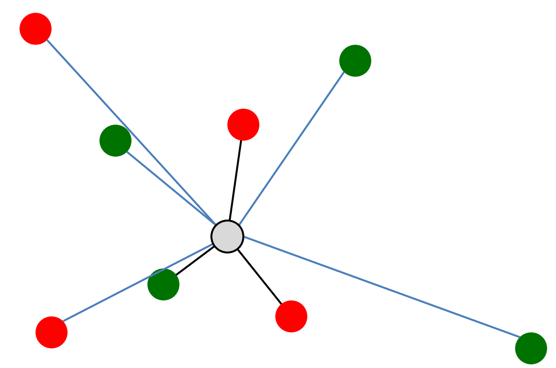
- Classification
  - $Score(class) = \sum_{i:(i \in KNN) \& \, class(i)=class} w(x, x_i)$
  - $class(x) = \underset{class}{\operatorname{argmax}} \, Score(class)$
- Regression:
  - $Y(x) = \sum_{i \in KNN} w(x, x_i) Y_i$
- The weight $w(x, x_i)$ is inversely related to $d(x, x_i)$
  - If $d(x, x_i)$ increases, $w(x, x_i)$ decreases



WHY RESTRICT TO K NEAREST NEIGHBORS? Considering that distant examples carry less weight

Weighted example-based prediction

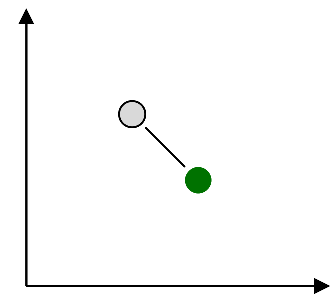
- Classification
  - $Score(class) = \sum_{i:class(i)=class} w(x, x_i)$
  - $class(x) = \underset{class}{\operatorname{argmax}}\, Score(class)$



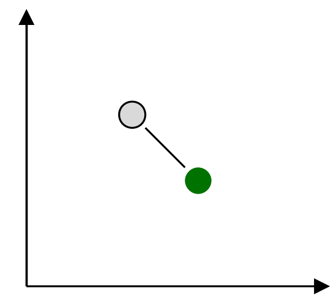
- Regression:
  - $Y(x) = \sum_i w(x, x_i) Y_i$

- All training instances invoked!

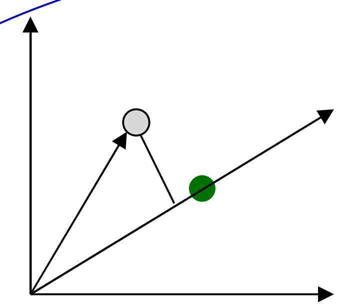
Weights of cohort



$$d(x_1, x_2) = \|x_1 - x_2\|^2 \qquad w(x, y) = \frac{1}{d(x, y)}$$

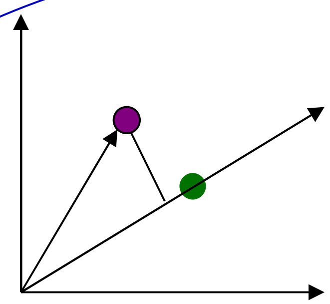


$$w(x_1, x_2) = exp(-\alpha d(x_1, x_2))$$



$$w(x_1, x_2) = x_1, x_2 = \ x_1^T x_2$$

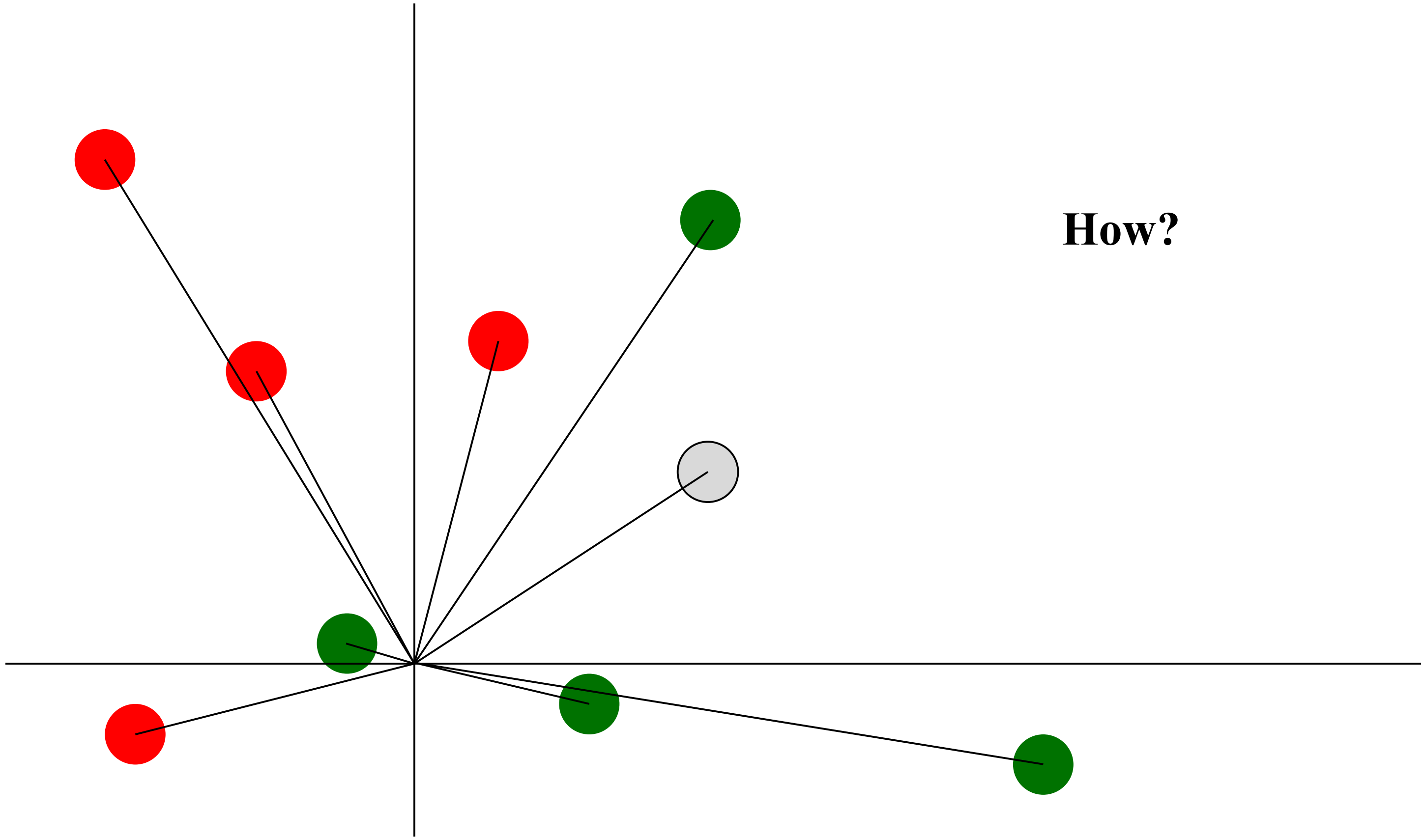
NN prediction with inner-product weights



$$w(x_1, x_2) = x_1, x_2 = \ x_1^T x_2$$

$$Y_{test} = \sum_{i \in training\ set} (x_{test}^T x_i) Y_i$$

Nearest Neighbor Classification

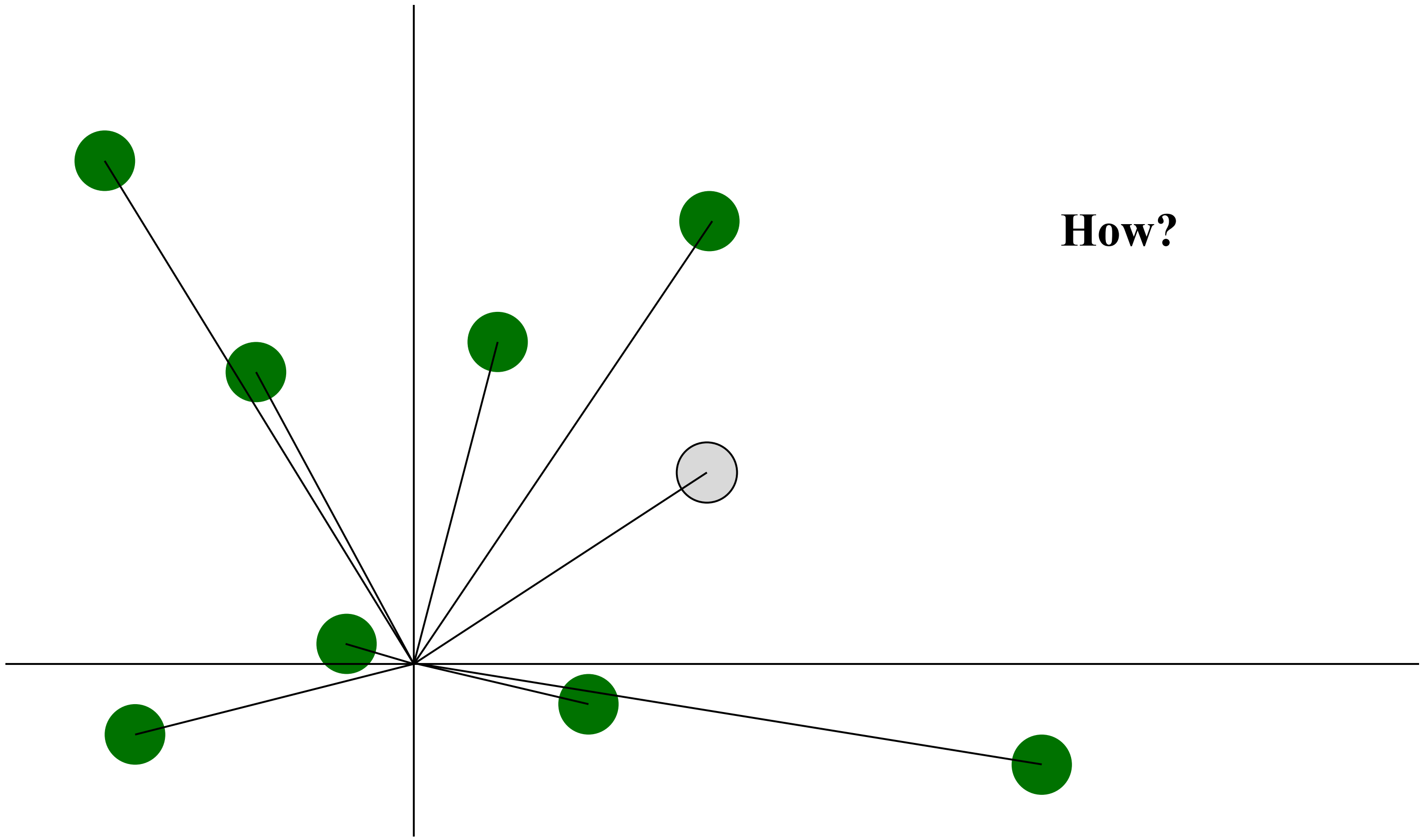


How?

$$Score_{green} = \sum_{i \in green} (x_{test}^T x_i) \qquad Score_{red} = \sum_{i \in red} (x_{test}^T x_i)$$

$$Y_{test} = Score_{green} > Score_{red}? \ Green, \ else\ Red$$

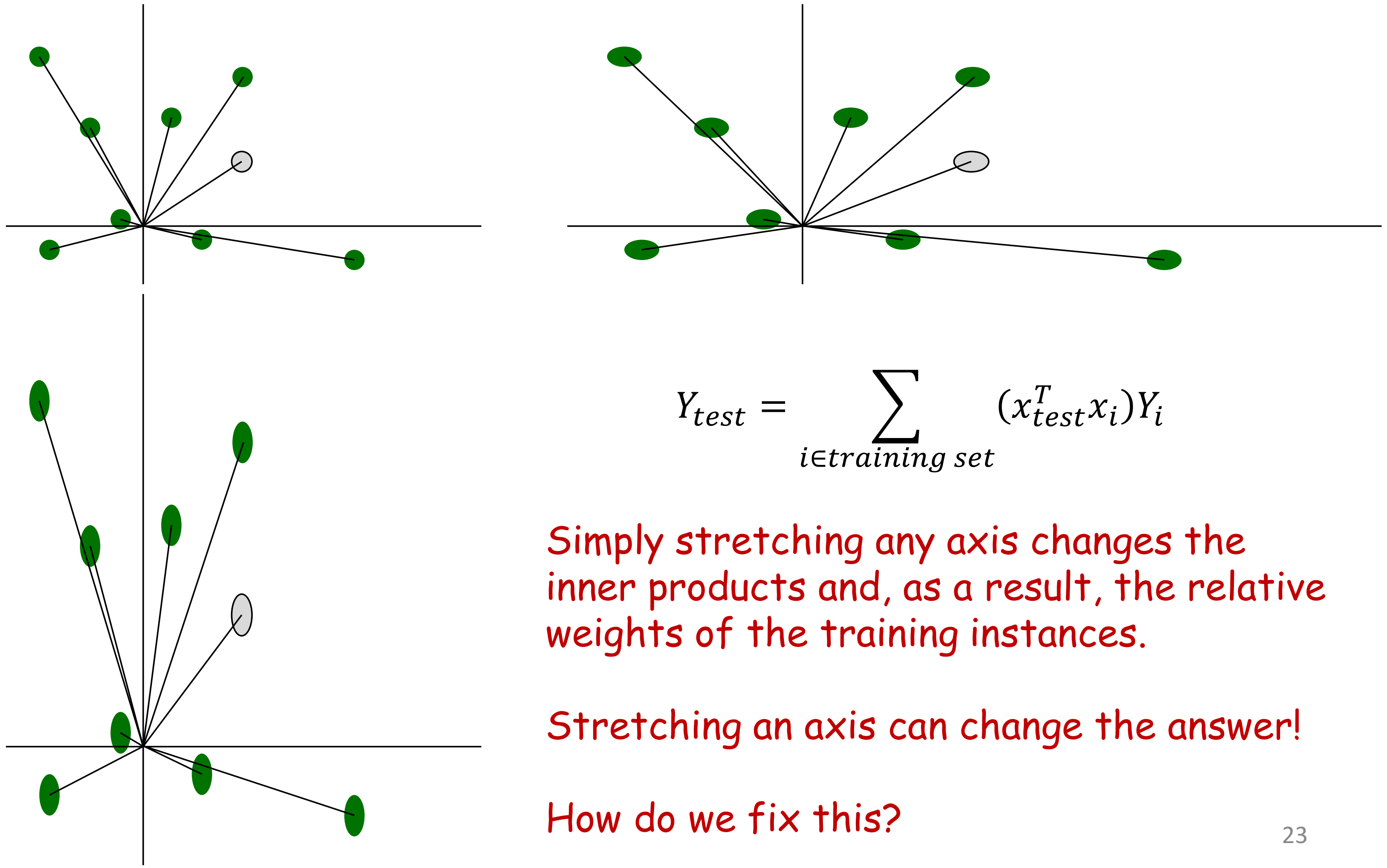
Nearest Neighbor *Regression*



How?

$$Y_{test} = \sum_{i \in training\ set} (x_{test}^T x_i) Y_i$$

Nearest Neighbor *Regression*



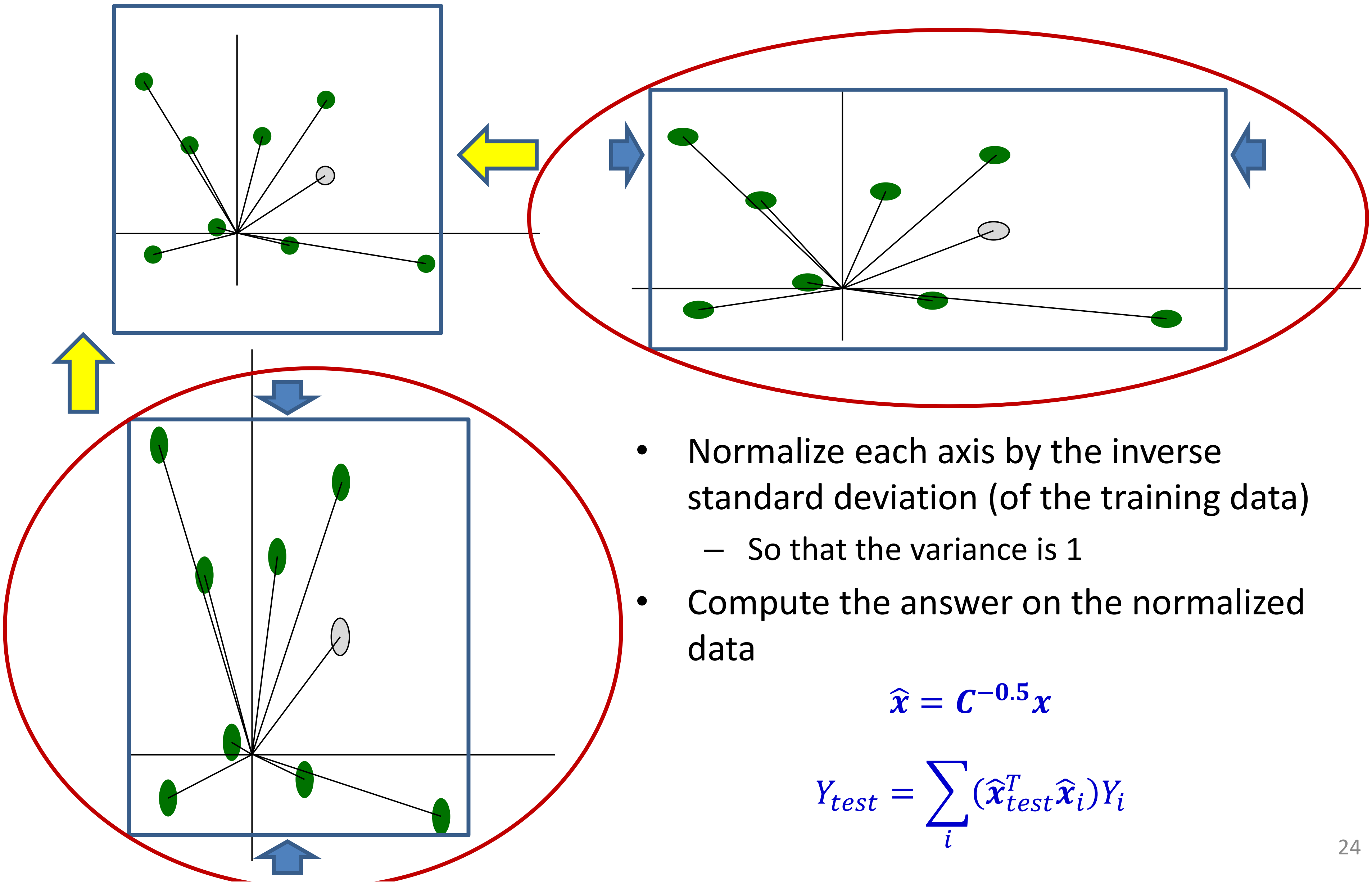
$$Y_{test} = \sum_{i \in training\ set} (x_{test}^T x_i) Y_i$$

Simply stretching any axis changes the inner products and, as a result, the relative weights of the training instances.

Stretching an axis can change the answer!

How do we fix this?

Normalizing the axes



- Normalize each axis by the inverse standard deviation (of the training data)
  - So that the variance is 1
- Compute the answer on the normalized data

$$\hat{\boldsymbol{x}} = \boldsymbol{C}^{-0.5}\boldsymbol{x}$$

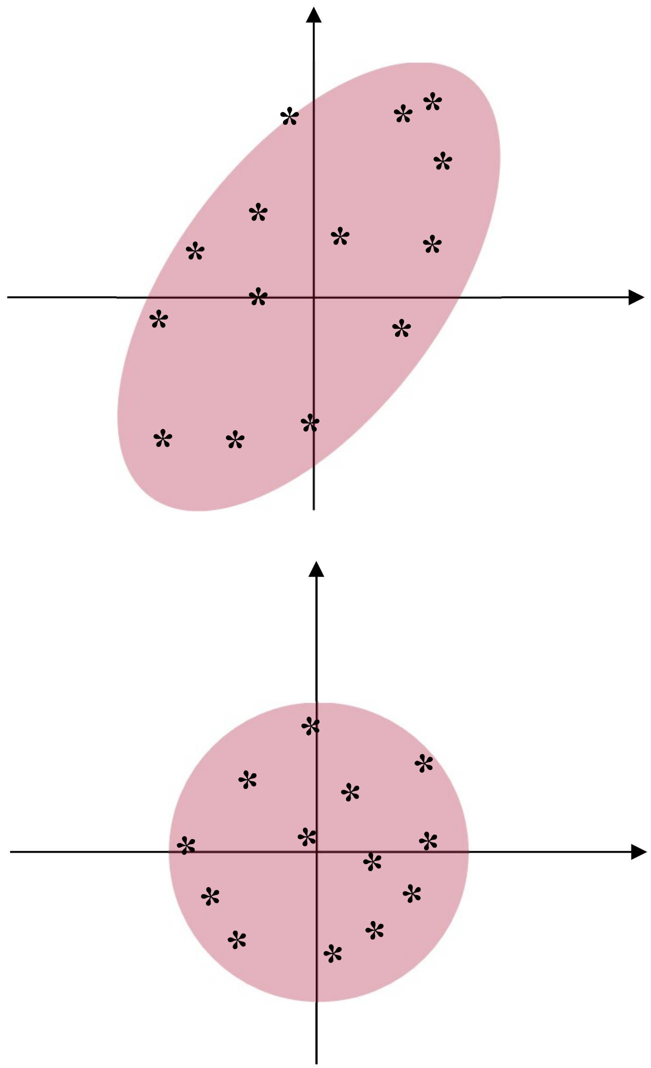
$$Y_{test} = \sum_i (\hat{\boldsymbol{x}}_{test}^T \hat{\boldsymbol{x}}_i) Y_i$$

The whitening matrix

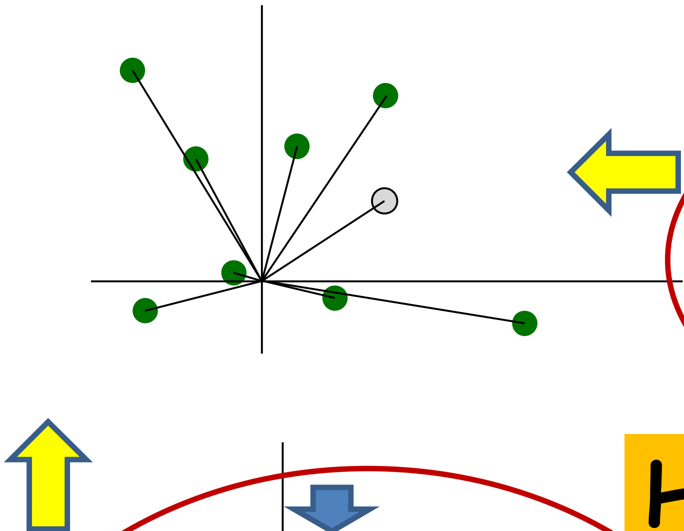
- Top: Skewed natural scatter of a data set
- Bottom: Scatter after whitening via

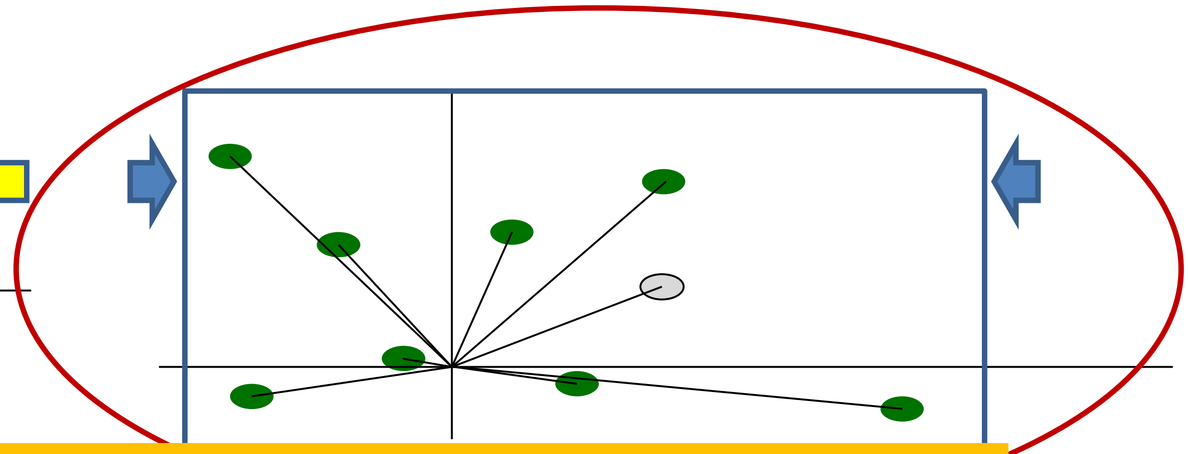
$$\hat{\mathbf{x}} = \mathbf{C}^{-\frac{1}{2}}\mathbf{x}$$

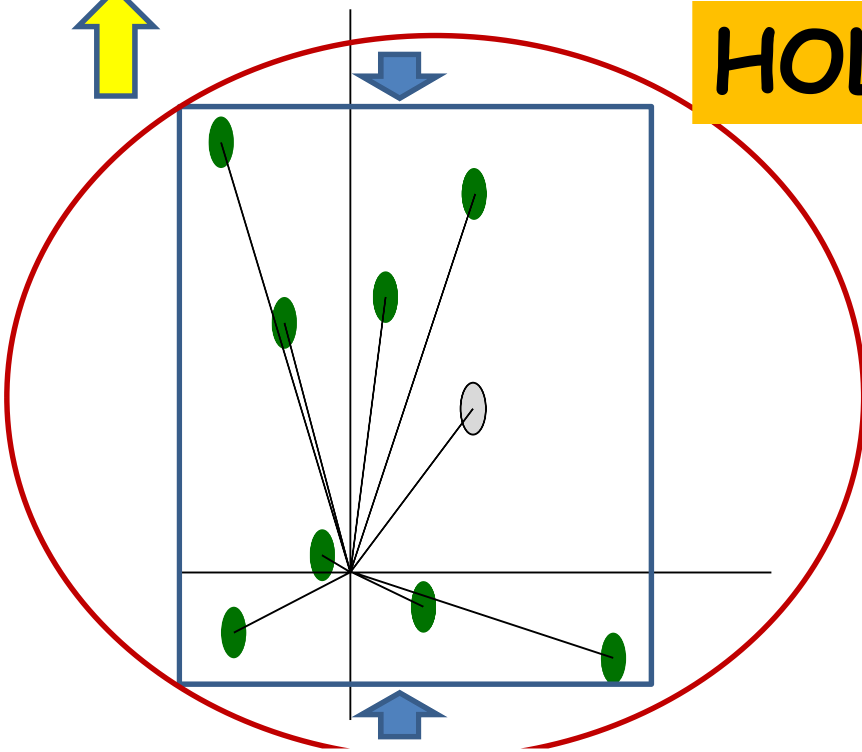
- Rotates and rescales the axes to make scatter circular (spherical)



Normalizing the axes







HOLD THIS THOUGHT

- Normalize each axis by the inverse standard deviation (of the training data)
  - So that the variance is 1
- Compute the answer on the normalized data

$$\boxed{\hat{\boldsymbol{x}} = \boldsymbol{C}^{-0.5}\boldsymbol{x}}$$

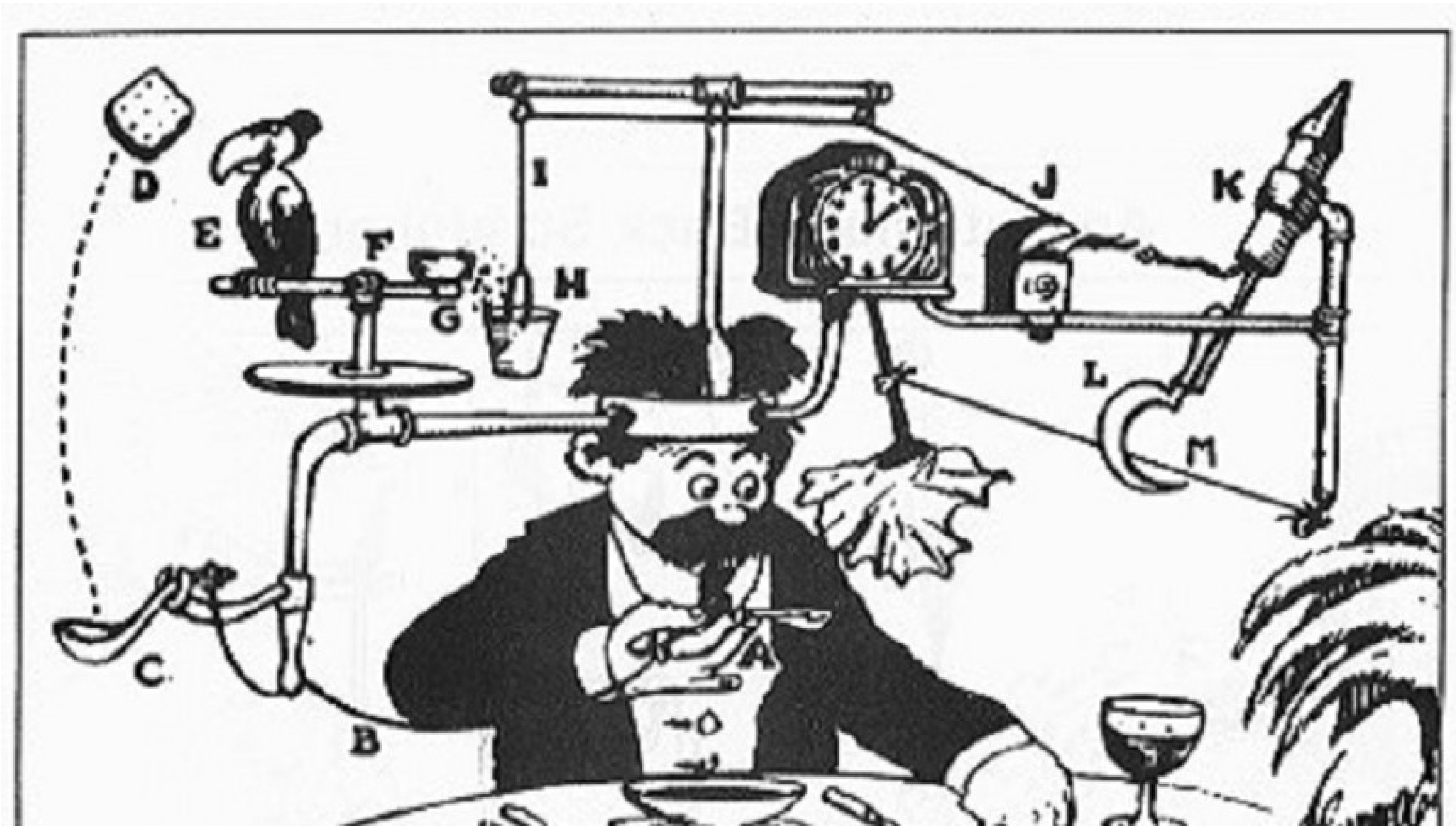
$$\boxed{Y_{test} = \sum_i (\hat{\boldsymbol{x}}_{test}^T \hat{\boldsymbol{x}}_i) Y_i}$$

Poll 2

Lessons

- Classification are regression are two versions of the same problem
  - Predicting an attribute of a data instance based on other attributes
- Nearest-neighbor based prediction: Predict the weighted average value of desired attribute from all the training instances
- Amazing fact they never told you: *Every form of prediction/classification/regression* is actually just a variant of weighted nearest-neighbor prediction

Changing Gears

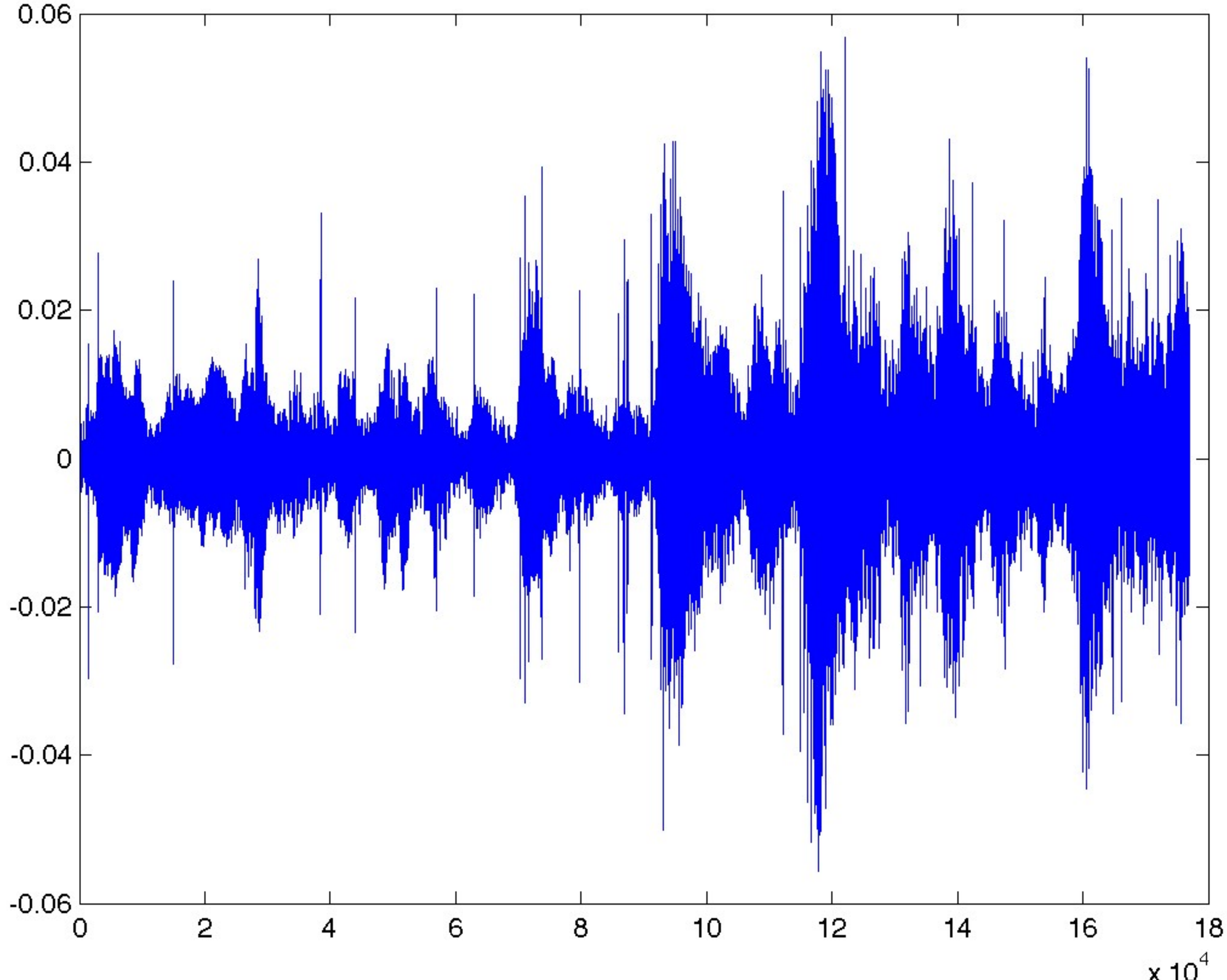


Rube Goldberg

Topics

- Nearest neighbor regression and classification
- **Linear regression**
  - With an application to glitch elimination in sound
  - And its relation to nearest-neighbor regression
- Regression in kernel spaces
- Kernel regression
- Regularization..

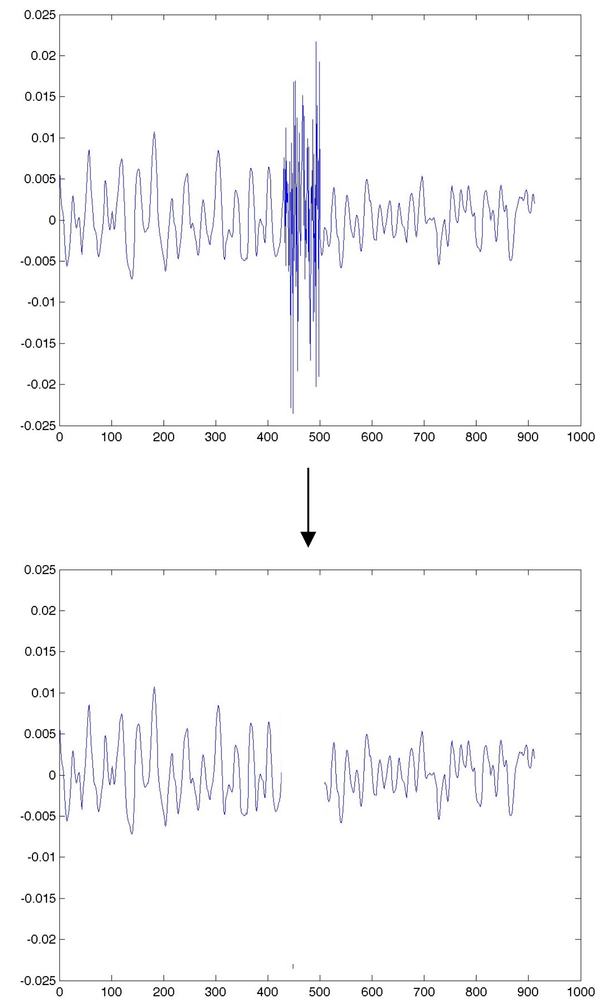
A Common Problem



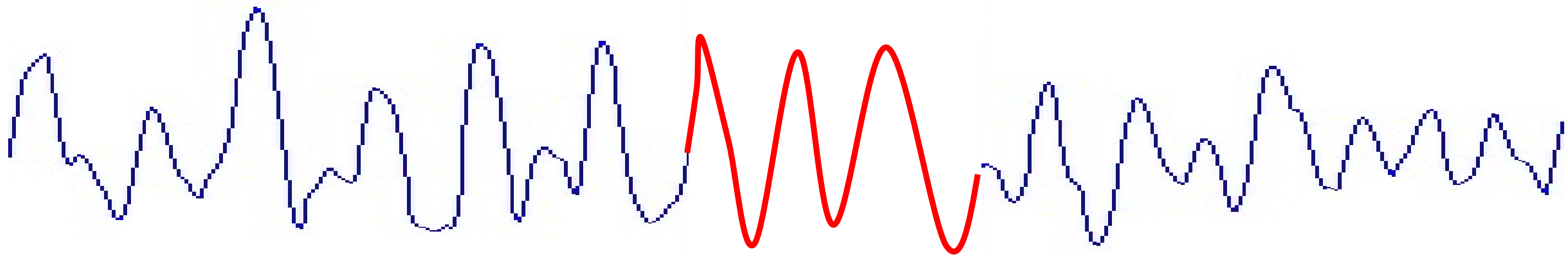
- Can you spot the glitches?

How to fix this problem?

- "Glitches" in audio
  - Must be detected
  - How?
- Then what?
- Glitches must be "fixed"
  - Delete the glitch
    - Results in a "hole"
  - Fill in the hole
  - How?

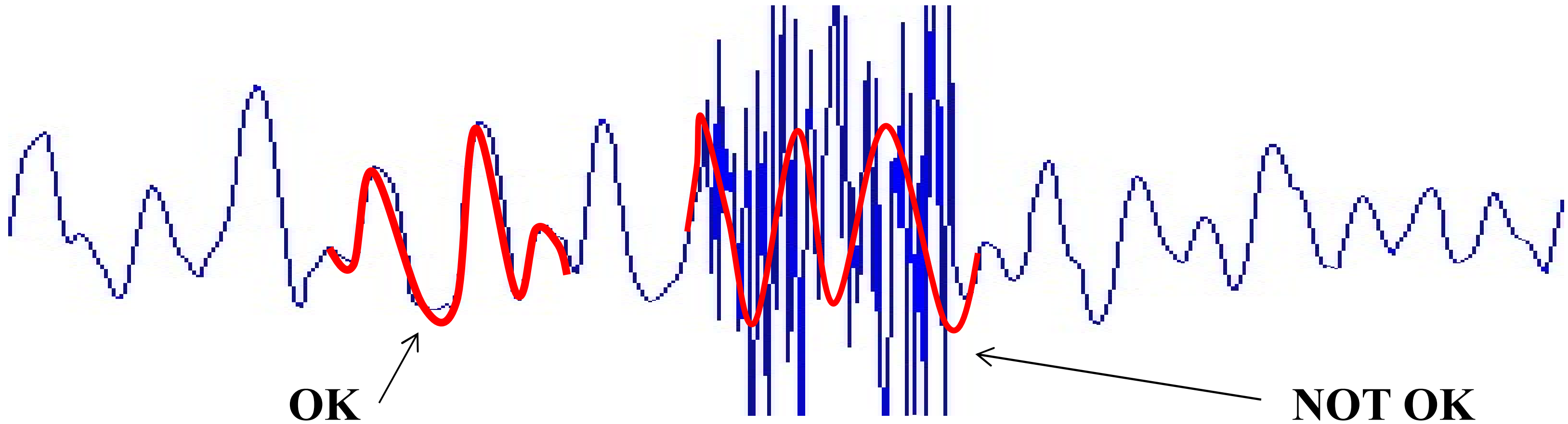


Interpolation..



- “Extend” the curve on the left to “predict” the values in the “blank” region
  - *Forward* prediction
- Extend the blue curve on the right leftwards to predict the blank region
  - *Backward* prediction
- How?
  - Regression analysis..

Detecting the Glitch



- Regression-based reconstruction can be done anywhere
- Reconstructed value will not match actual value
- Large error of reconstruction identifies glitches

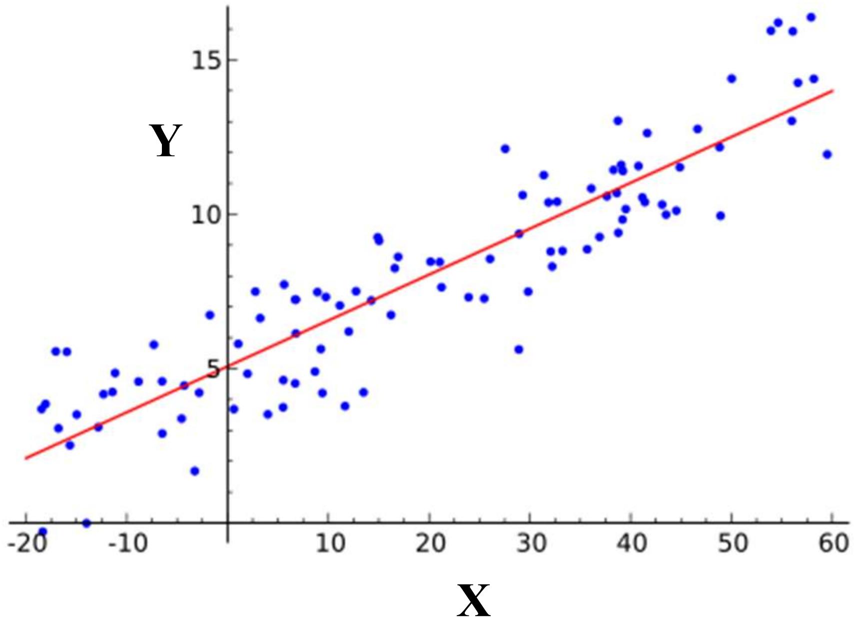
What is a regression

- Analyzing relationship between variables
- Expressed in many forms
- Wikipedia
  - Linear regression, Simple regression, Ordinary least squares, Polynomial regression, General linear model, Generalized linear model, Discrete choice, Logistic regression, Multinomial logit, Mixed logit, Probit, Multinomial probit, ….
- Generally a tool to ***predict*** variables

Regressions for prediction

- $\mathbf{y} = f(\mathbf{x}; \Theta) + e$
- Different possibilities
  - $\mathbf{y}$ is a scalar
    - $\mathbf{y}$ is real
    - $\mathbf{y}$ is categorical (classification)
  - $\mathbf{y}$ is a vector
  - $\mathbf{x}$ is a vector
    - $\mathbf{x}$ is a set of real valued variables
    - $\mathbf{x}$ is a set of categorical variables
    - $\mathbf{x}$ is a combination of the two
  - $f(\mathbf{.})$ is a linear or affine function
  - $f(\mathbf{.})$ is a non-linear function
  - $f(\mathbf{.})$ is a *time-series* model

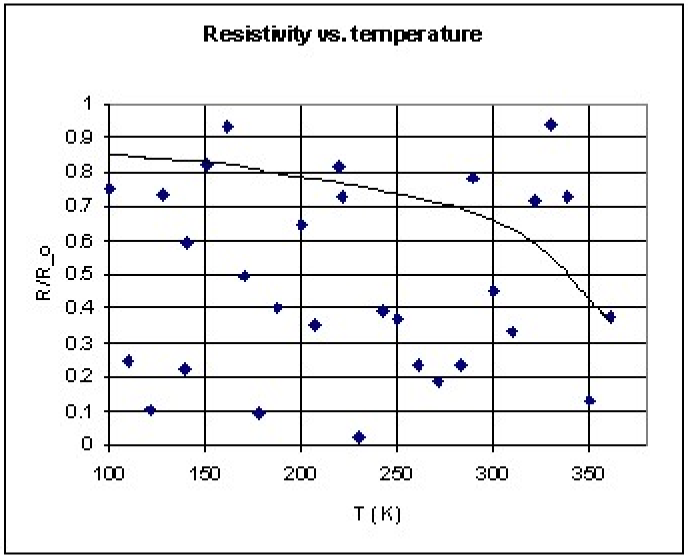
A *linear* regression



- Assumption: relationship between variables is linear
  - A linear *trend* may be found relating $\mathbf{x}$ and $\mathbf{y}$
  - $\mathbf{y}$ = *dependent* variable
  - $\mathbf{x}$ = *explanatory* variable
  - Given $\mathbf{x}$, $\mathbf{y}$ can be predicted as an affine function of $\mathbf{x}$

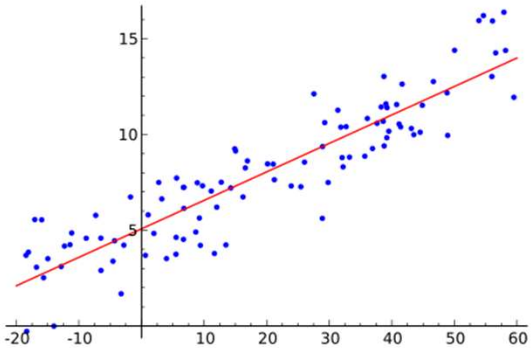
An imaginary regression..

- http://pages.cs.wisc.edu/~kovar/hall.html
- Check this shit out (Fig. 1).
  That's bonafide, 100%-real data, my friends. I took it myself over the course of two weeks. And this was not a leisurely two weeks, either; I busted my ass day and night in order to provide you with nothing but the best data possible. Now, let's look a bit more closely at this data, remembering that it is absolutely first-rate. Do you see the exponential dependence? I sure don't. I see a bunch of crap.
  Christ, this was such a waste of my time.
  Banking on my hopes that whoever grades this will just look at the pictures, I drew an exponential through my noise. I believe the apparent legitimacy is enhanced by the fact that I used a complicated computer program to make the fit. I understand this is the same process by which the top quark was discovered.



Linear Regressions

- $\mathbf{y} = \mathbf{a}^T\mathbf{x} + \mathbf{b} + \mathbf{e}$
  - $\mathbf{e}$ = prediction error



- Given a "training" set of $\{\mathbf{x}, \mathbf{y}\}$ values: estimate $\mathbf{a}$ and $\mathbf{b}$
  - $\mathbf{y}_1 = \mathbf{a}^T\mathbf{x}_1 + \mathbf{b} + \mathbf{e}_1$
  - $\mathbf{y}_2 = \mathbf{a}^T\mathbf{x}_2 + \mathbf{b} + \mathbf{e}_2$
  - $\mathbf{y}_3 = \mathbf{a}^T\mathbf{x}_3 + \mathbf{b} + \mathbf{e}_3$
  - …
- If $\mathbf{a}$ and $\mathbf{b}$ are well estimated, prediction error will be small

Linear Regression to a scalar

$$y_1 = \boldsymbol{a}^{\mathrm{T}}\mathbf{x_1} + \mathrm{b} + \mathrm{e}_1$$
$$y_2 = \boldsymbol{a}^{\mathrm{T}}\mathbf{x_2} + \mathrm{b} + \mathrm{e}_2$$
$$y_3 = \boldsymbol{a}^{\mathrm{T}}\mathbf{x_3} + \mathrm{b} + \mathrm{e}_3$$

- Define:

$$\mathbf{y} = [y_1\ y_2\ y_3 \ldots] \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \\ 1 & 1 & 1 & \ldots \end{bmatrix} \qquad \mathbf{a} = [\boldsymbol{a}^T \quad b]$$

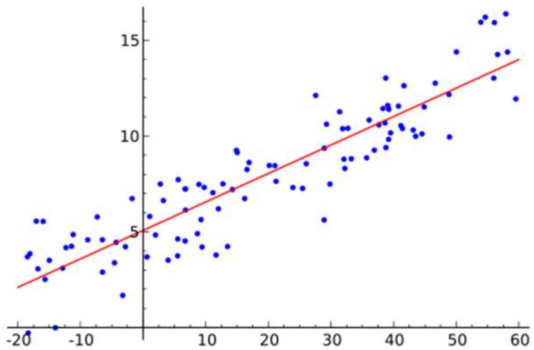
$$\mathbf{e} = [e_1\ e_2\ e_3 \ldots]$$

- Rewrite

$$\mathbf{y} = \mathbf{aX} + \boldsymbol{e}$$

Learning the parameters

$$\mathbf{y} = \mathbf{aX} + e$$

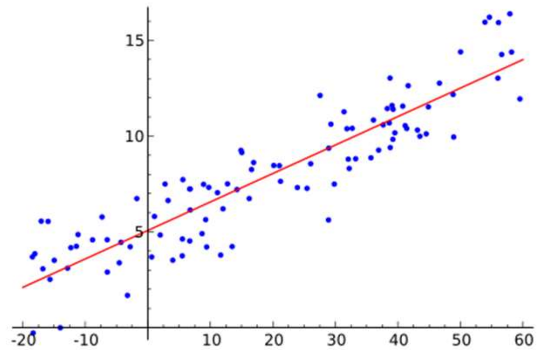


$$\hat{\mathbf{y}} = \mathbf{aX}$$ **Assuming no error**

- Given training data: several $\mathbf{x}, y$
- Can define a "divergence": $\mathrm{D}(\mathbf{y}, \hat{\mathbf{y}})$
  - Measures how much $\hat{\mathbf{y}}$ differs from $\mathbf{y}$
  - Ideally, if the model is accurate this should be small
- Estimate $\mathbf{a}$ to minimize $\mathrm{D}(\mathbf{y}, \hat{\mathbf{y}})$

The prediction error as divergence

$$y_1 = \boldsymbol{a}^{\mathrm{T}}\mathbf{x_1} + b + e_1$$
$$y_2 = \boldsymbol{a}^{\mathrm{T}}\mathbf{x_2} + b + e_2$$
$$y_3 = \boldsymbol{a}^{\mathrm{T}}\mathbf{x_3} + b + e_3$$



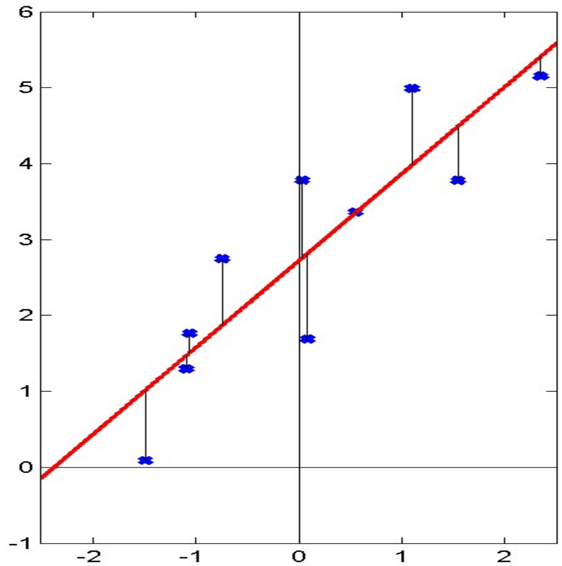
$$\mathbf{y} = \mathbf{aX} + \boldsymbol{e}$$

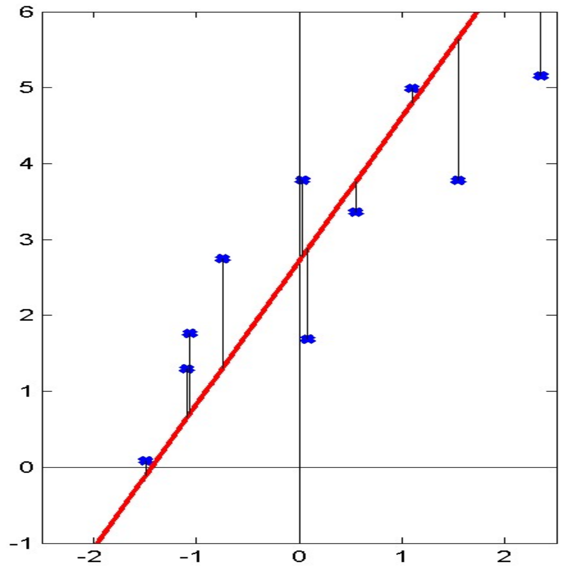
$$\mathbf{D}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbf{E} = e_1^2 + e_2^2 + e_3^2 + \ldots$$

$$\mathbf{E} = \|\mathbf{y} - \mathbf{aX}\|^2$$

- Define divergence as sum of the squared error in predicting $\mathbf{y}$

Prediction error as divergence





- $y = \mathbf{ax} + e$
  - e = prediction error
  - Find the "slope" $\mathbf{a}$ such that the total squared length of the error lines is minimized

Solving a linear regression

$$\mathbf{y} = \mathbf{aX} + \boldsymbol{e}$$

- Minimize squared error

$$\mathbf{E} = \|\mathbf{y} - \mathbf{aX}\|^2$$

$$\mathbf{a} = \mathbf{y}\,pinv(\mathbf{X})$$

More Explicitly

$$\mathbf{y} = [y_1\ y_2\ y_3 \ldots] \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \ldots \\ 1 & 1 & 1 & \end{bmatrix}$$

$$\mathbf{A} = \mathbf{y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}$$

$$\mathbf{a} = \mathbf{y}\,pinv(\mathbf{X})$$

- $\mathbf{X}$ is wider than it is tall

$$pinv(\mathbf{X}) = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$$

$$\mathbf{a} = \mathbf{y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{-1}$$

Regression in multiple dimensions

$$\mathbf{y}_1 = \mathbf{A}\mathbf{x}_1 + \mathbf{b} + \mathbf{e}_1$$
$$\mathbf{y}_2 = \mathbf{A}\mathbf{x}_2 + \mathbf{b} + \mathbf{e}_2$$
$$\mathbf{y}_3 = \mathbf{A}\mathbf{x}_3 + \mathbf{b} + \mathbf{e}_3$$

$\mathbf{y}_i$ is a vector

y_{ij} = j^{th} component of vector $\mathbf{y}_i$

$\mathbf{a}_i$ = i^{th} row of $\mathbf{A}$

b_j = j^{th} component of $\mathbf{b}$

- Also called *multiple regression*
- Equivalent of saying:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b} + \mathbf{e}_i \quad \Rightarrow \quad \begin{aligned} y_{i1} &= \mathbf{a}_1\mathbf{x}_i + b_1 + e_{i1} \\ y_{i2} &= \mathbf{a}_2\mathbf{x}_i + b_2 + e_{i2} \\ y_{i3} &= \mathbf{a}_3\mathbf{x}_i + b_3 + e_{i3} \end{aligned}$$

- Fundamentally no different from *N* separate single regressions
  - But we can use the relationship between $\mathbf{y}$s to our benefit

Multiple Regression

$$\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \mathbf{y}_3 \ldots] \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \ldots \end{bmatrix} \qquad \hat{\mathbf{A}} = [\mathbf{A} \quad \mathbf{b}]$$

$$\mathbf{E} = [\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3 \ldots]$$

$$\mathbf{Y} = \hat{\mathbf{A}}\mathbf{X} + \mathbf{E}$$

Frobenius norm

$$DIV = \sum_i \left\| \mathbf{y}_i - \hat{\mathbf{A}}\bar{\mathbf{x}}_i \right\|^2 = \left\| \mathbf{Y} - \hat{\mathbf{A}}\mathbf{X} \right\|_F^2$$

- Minimizing

$$\hat{\mathbf{A}} = \mathbf{Y}pinv(\mathbf{X}) = \mathbf{Y}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}^T\right)^{-1}$$

Aside: The Frobenius norm

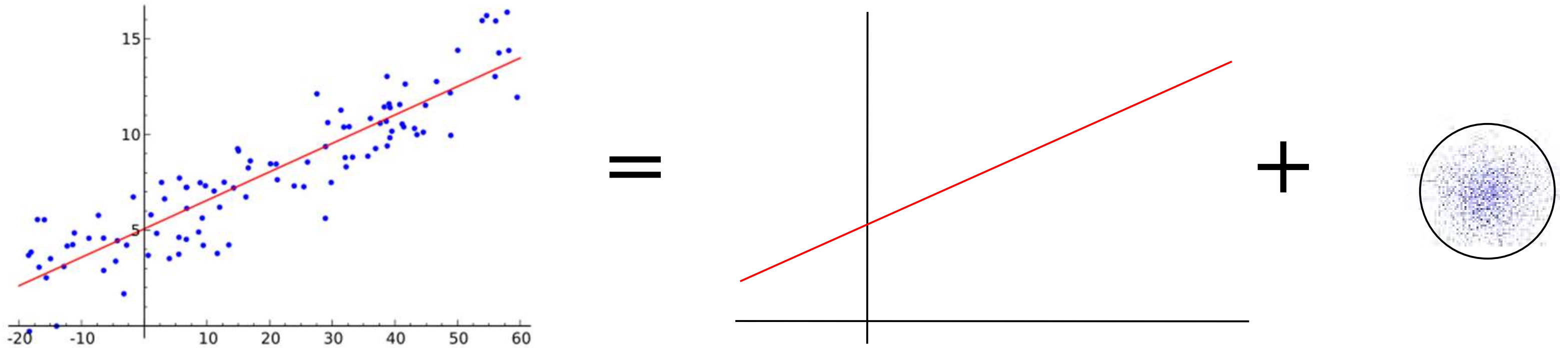
- The Frobenius norm is the square root of the sum of the squares of all the components of the matrix

$$\|\mathbf{E}\|_F = \sqrt{\sum_{i,j} e_{i,j}^2}$$

- The derivative of the squared Frobenius norm:

$$\nabla_A \|\mathbf{Y} - \mathbf{AX}\|_F^2 = 0 \Rightarrow \mathbf{A} = \mathbf{YX}(\mathbf{XX}^T)^{-1}$$

A Different Perspective



- $\mathbf{y}$ is a noisy reading of $\mathbf{Ax}$

$$\mathbf{y} = \mathbf{Ax} + \mathbf{e}$$

- Error $\mathbf{e}$ is Gaussian

$$\mathbf{e} \sim N(0, \sigma^2 \mathbf{I})$$

- Estimate $\mathbf{A}$ from $\mathbf{Y} = [\mathbf{y}_1 \;\; \mathbf{y}_2 \ldots \mathbf{y}_N] \quad \mathbf{X} = [\mathbf{x}_1 \;\; \mathbf{x}_2 \ldots \mathbf{x}_N]$

The *Likelihood* of the data

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e} \qquad \mathbf{e} \sim N(0, \sigma^2 \mathbf{I})$$

- Probability of observing a specific $\mathbf{y}$, given $\mathbf{x}$, for a particular matrix $\mathbf{A}$

$$P(\mathbf{y} \mid \mathbf{x}; \mathbf{A}) = N(\mathbf{y}; \mathbf{A}\mathbf{x}, \sigma^2 \mathbf{I})$$

- Probability of collection: $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \mathbf{y}_3 \ldots] \quad \mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3 \ldots$

$$P(\mathbf{Y} \mid \mathbf{X}; \mathbf{A}) = \prod_i N(\mathbf{y}_i; \mathbf{A}\mathbf{x}_i, \sigma^2 \mathbf{I})$$

- Assuming IID for convenience (not necessary)

A Maximum Likelihood Estimate

$$\mathbf{y} = \mathbf{A}^T\mathbf{x} + \mathbf{e} \qquad \mathbf{e} \sim N(0, \sigma^2\mathbf{I}) \qquad \mathbf{Y} = [\mathbf{y}_1 \ \ \mathbf{y}_2 \ldots \mathbf{y}_N] \quad \mathbf{X} = [\mathbf{x}_1 \ \ \mathbf{x}_2 \ldots \mathbf{x}_N]$$

$$P(\mathbf{Y} \mid \mathbf{X}) = \prod_i \frac{1}{\sqrt{(2\pi\sigma^2)^D}} \exp\left(\frac{-1}{2\sigma^2}\left\|\mathbf{y}_i - \mathbf{A}^T\mathbf{x}_i\right\|^2\right)$$

$$\log P(\mathbf{Y} \mid \mathbf{X}; \mathbf{A}) = C - \sum_i \frac{1}{2\sigma^2} \underline{\left\|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\right\|^2}$$

- Maximizing the log probability is identical to minimizing the error
  - Identical to the least squares solution

$$\boxed{\mathbf{A} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1} = \mathbf{Y}\,pinv(\mathbf{X})}$$

Returning to Multiple Regression

$$\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \mathbf{y}_3 \ldots] \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \ldots \end{bmatrix} \qquad \hat{\mathbf{A}} = [\mathbf{A} \quad \mathbf{b}]$$

$$\mathbf{E} = [\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3 \ldots]$$

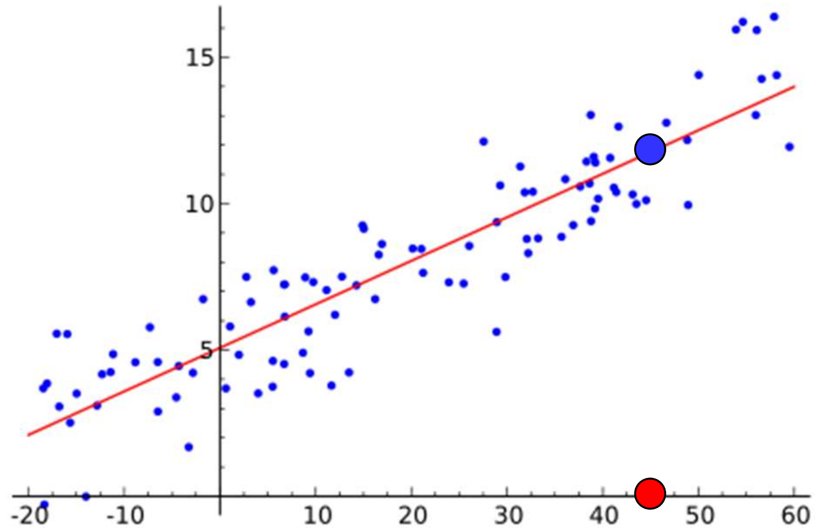
$$\mathbf{Y} = \hat{\mathbf{A}}\mathbf{X} + \mathbf{E}$$

$$DIV = \sum_i \left\| \mathbf{y}_i - \hat{\mathbf{A}}\bar{\mathbf{x}}_i \right\|^2 = \left\| \mathbf{Y} - \hat{\mathbf{A}}\mathbf{X} \right\|_F^2$$

- Minimizing

$$\hat{\mathbf{A}} = \mathbf{Y}pinv(\mathbf{X}) = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}$$

Predicting an output

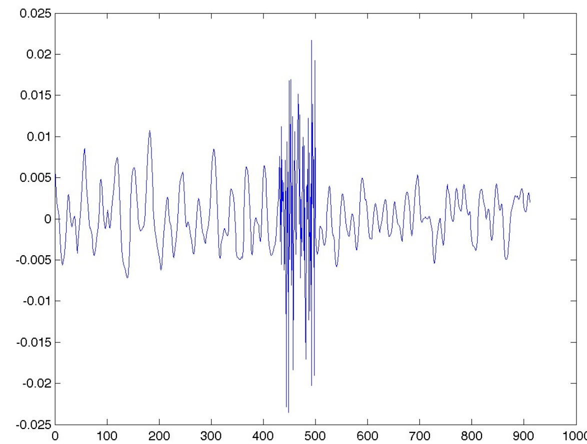


- From a collection of training data, have learned $\mathbf{A}$
- Given $\mathbf{x}$ for a new instance, but not $\mathbf{y}$, what is $\mathbf{y}$?
- Simple solution:

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} + \mathbf{b}$$

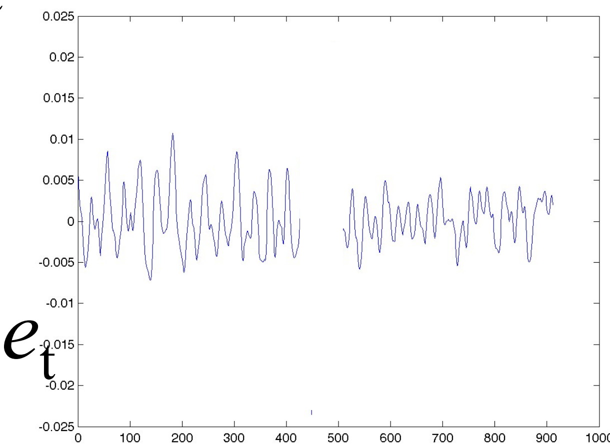
Applying it to our problem

- Prediction by regression



- Forward regression
- $x_t = a_1 x_{t-1} + a_2 x_{t-2} \ldots a_k x_{t-k} + e_t$

- Backward regression
- $x_t = b_1 x_{t+1} + b_2 x_{t+2} \ldots b_k x_{t+k} + e_t$



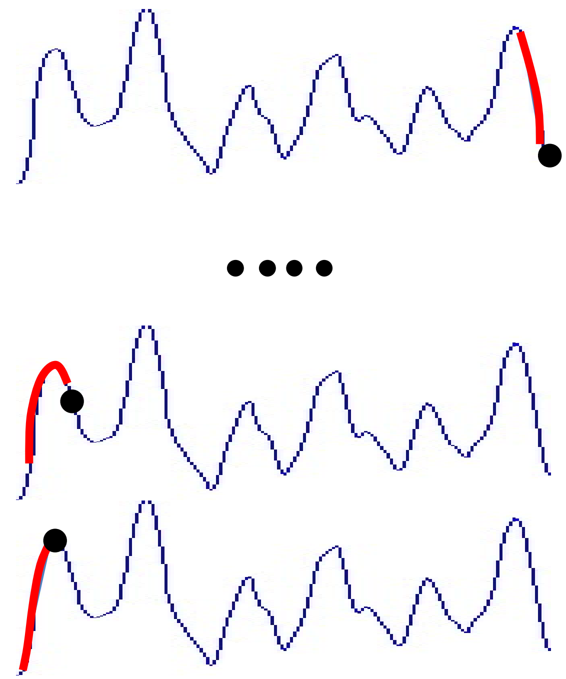
Applying it to our problem

- Forward prediction

$$\begin{bmatrix} x_t \\ x_{t-1} \\ .. \\ x_{K+1} \end{bmatrix} = \begin{bmatrix} x_{t-1} & x_{t-2} & .. & x_{t-K} \\ x_{t-2} & x_{t-3} & .. & x_{t-K-1} \\ .. & .. & .. & .. \\ x_K & x_{K-1} & .. & x_1 \end{bmatrix} \mathbf{a}_t + \begin{bmatrix} e_t \\ e_{t-1} \\ .. \\ e_{K+1} \end{bmatrix}$$

$$\mathbf{x} = \mathbf{X}\mathbf{a}_t + \mathbf{e}$$

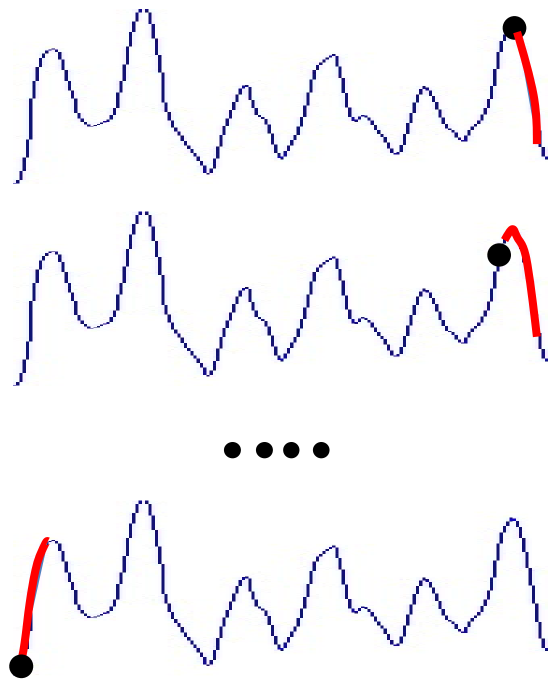
$$pinv(\mathbf{X})\mathbf{x} = \mathbf{a}_t$$



Applying it to our problem

- Backward prediction

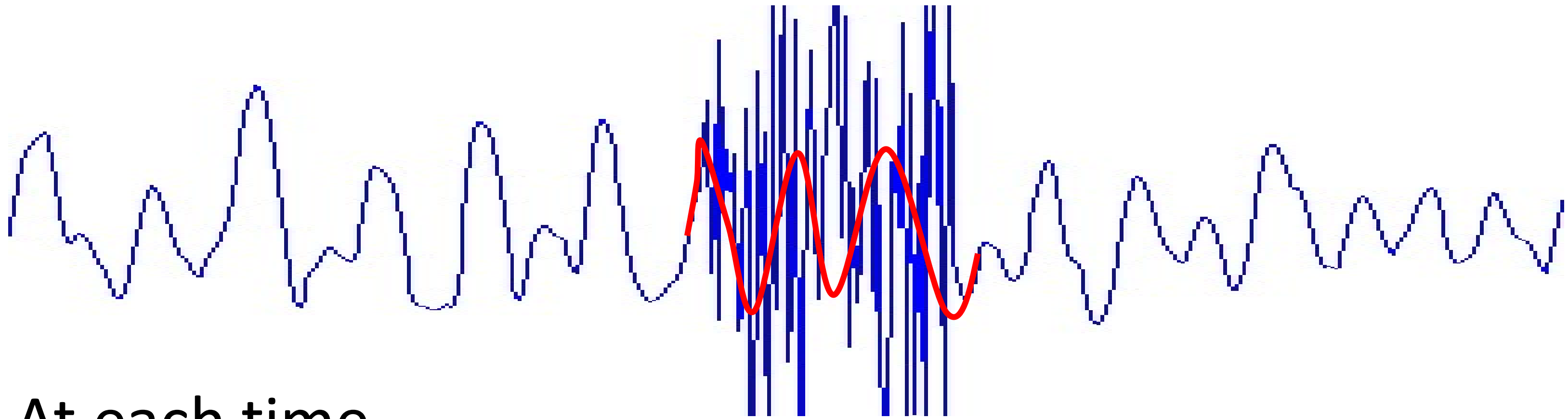
$$\begin{bmatrix} x_{t-K-1} \\ x_{t-K-2} \\ .. \\ x_1 \end{bmatrix} = \begin{bmatrix} x_t & x_{t-1} & .. & x_{t-K} \\ x_{t-1} & x_{t-2} & .. & x_{t-K-1} \\ .. & .. & .. & .. \\ x_{K+1} & x_K & .. & x_2 \end{bmatrix} \mathbf{b}_t + \begin{bmatrix} e_{t-K-1} \\ e_{t-K-2} \\ .. \\ e_1 \end{bmatrix}$$



$$\overline{\mathbf{x}} = \overline{\mathbf{X}}\mathbf{b}_t + \mathbf{e}$$

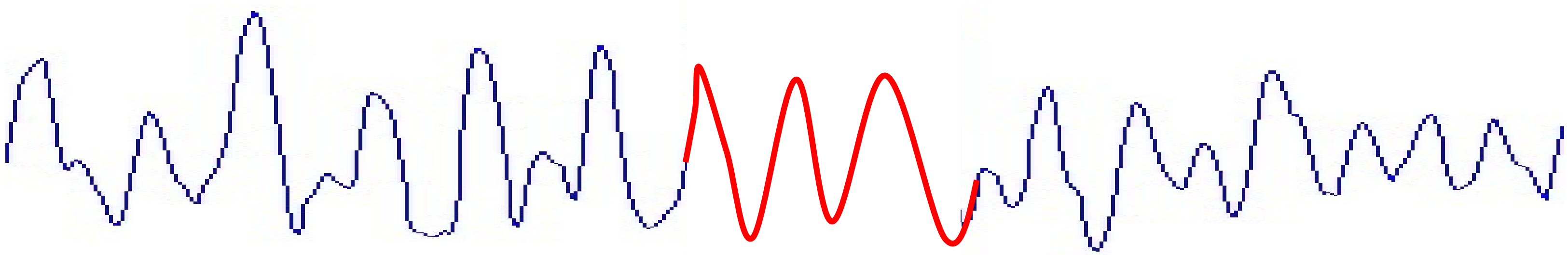
$$pinv(\overline{\mathbf{X}})\overline{\mathbf{x}} = \mathbf{b}_t$$

Finding the burst



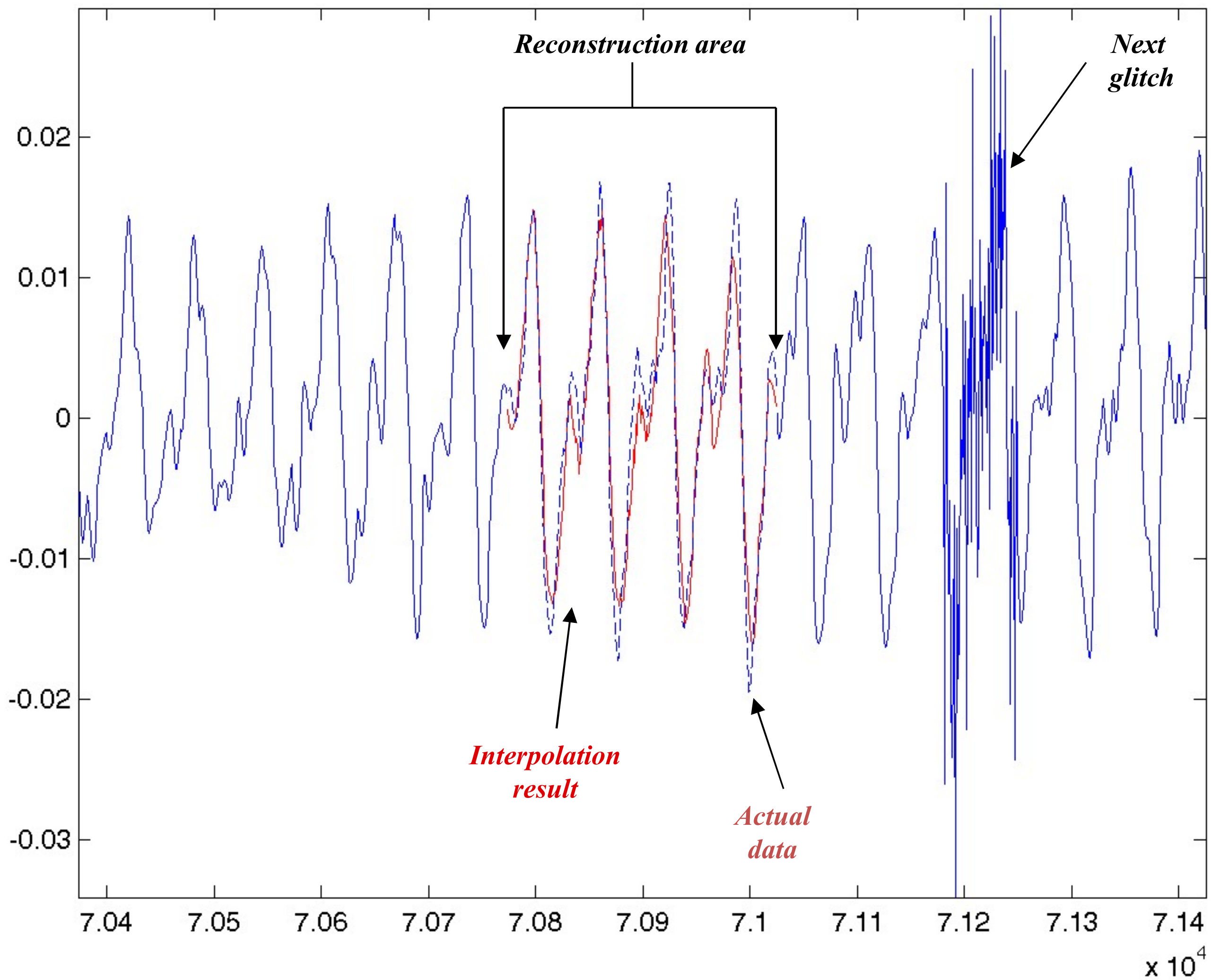
- At each time
  - Learn a "forward" predictor $\mathbf{a}_t$
  - At each time, predict next sample $x_t^{est} = \Sigma_i \, a_{t,k} x_{t-k}$
  - Compute error: $ferr_t = |x_t - x_t^{est}|^2$
  - Learn a "backward" predict and compute backward error
    - $berr_t$
  - Compute average prediction error over window, threshold
- If the error exceeds a threshold, identify burst

Filling the hole



- Learn "forward" predictor at left edge of "hole"
  - For each missing sample
  - At each time, predict next sample $x_t^{est} = \Sigma_i \, a_{t,k} x_{t-k}$
    - Use estimated samples if real samples are not available
- Learn "backward" predictor at left edge of "hole"
  - For each missing sample
  - At each time, predict next sample $x_t^{est} = \Sigma_i \, b_{t,k} x_{t+k}$
    - Use estimated samples if real samples are not available
- Average forward and backward predictions

Reconstruction zoom in



Distorted signal

Recovered signal

Incrementally learning the regression

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}$$

Requires knowledge of *all* (x,y) pairs

- Can we learn $\mathbf{A}$ incrementally instead?
  - As data comes in?

- The Widrow Hoff rule

Scalar prediction version

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \eta\underbrace{\left(y_t - \hat{y}_t\right)}_{\text{error}}\mathbf{x}_t \qquad \hat{y}_t = \left(\mathbf{a}^t\right)^T \mathbf{x}_t$$

- Note the structure
  - Can also be done in batch mode!

Predicting a value

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}$$

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{x}$$

- What are we doing exactly?
  - For the explanation we are assuming no "$\mathbf{b}$" ($\mathbf{X}$ is 0 mean)
  - Explanation generalizes easily even otherwise

$$\mathbf{C} = \mathbf{X}\mathbf{X}^T$$

- Let $\hat{\mathbf{x}} = \mathbf{C}^{-\frac{1}{2}}\mathbf{x}$ and $\hat{\mathbf{X}} = \mathbf{C}^{-\frac{1}{2}}\mathbf{X}$
  - Whitening $\mathbf{x}$
  - $N^{-0.5}\,\mathbf{C}^{-0.5}$ is the *whitening* matrix for $\mathbf{x}$

$$\hat{\mathbf{y}} = \mathbf{Y}\mathbf{X}^T\mathbf{C}^{-\frac{1}{2}}\mathbf{C}^{-\frac{1}{2}}\mathbf{x} = \mathbf{Y}\hat{\mathbf{X}}^T\hat{\mathbf{x}}_i$$

Predicting a value

$$\hat{\mathbf{y}} = \mathbf{Y}\hat{\mathbf{X}}^T\hat{\mathbf{x}} = \sum_i \mathbf{y}_i \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}$$

$$\hat{\mathbf{y}} = \mathbf{Y}\hat{\mathbf{X}}^T\hat{\mathbf{x}} = \begin{bmatrix}\mathbf{y}_1 & \dots & \mathbf{y}_N\end{bmatrix}\begin{bmatrix}\hat{\mathbf{x}}_1^T \\ \vdots \\ \hat{\mathbf{x}}_N^T\end{bmatrix}\hat{\mathbf{x}} = \sum_i \mathbf{y}_i \left(\hat{\mathbf{x}}_i^T \hat{\mathbf{x}}\right)$$

- What are we doing exactly?

Predicting a value

$$\hat{\mathbf{y}} = \sum_i \mathbf{y}_i \left(\hat{\mathbf{x}}_i^T \hat{\mathbf{x}} \right)$$

- Given training instances $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1..N$, estimate $\mathbf{y}$ for a new test instance of $\mathbf{x}$ with unknown $\mathbf{y}$:
- $\mathbf{y}$ is simply a *weighted sum of the* $\mathbf{y}_i$ *instances from the training data*
- The weight of any $\mathbf{y}_i$ is simply the inner product between its corresponding $\mathbf{x}_i$ and the new $\mathbf{x}$
  - With due whitening and scaling..

Poll 3

What are we doing: A different perspective

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{x}$$

- Assumes $\mathbf{X}\mathbf{X}^\mathrm{T}$ is invertible
- What if it is not
  - Dimensionality of $\mathbf{X}$ is greater than number of observations?
  - Underdetermined
- In this case $\mathbf{X}^\mathrm{T}\mathbf{X}$ will generally be invertible

$$\mathbf{A} = \mathbf{Y}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$$

$$\hat{\mathbf{y}} = \mathbf{Y}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{x}$$

High-dimensional regression

$$\hat{\mathbf{y}} = \mathbf{Y}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{x}$$

- $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is the "Gram Matrix"

$$\mathbf{G} = \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \cdots & \mathbf{x}_1^T\mathbf{x}_N \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \cdots & \mathbf{x}_2^T\mathbf{x}_N \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_N^T\mathbf{x}_1 & \mathbf{x}_N^T\mathbf{x}_2 & \cdots & \mathbf{x}_N^T\mathbf{x}_N \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{Y}\mathbf{G}^{-1}\mathbf{X}^T\mathbf{x}$$

High-dimensional regression

$$\hat{\mathbf{y}} = \mathbf{Y}\mathbf{G}^{-1}\mathbf{X}^T\mathbf{x}$$

- Normalize $\mathbf{Y}$ by the inverse of the gram matrix

$$\dddot{\mathbf{Y}} = \mathbf{Y}\mathbf{G}^{-1}$$

- Working our way down..

$$\hat{\mathbf{y}} = \dddot{\mathbf{Y}}\mathbf{X}^T\mathbf{x} \qquad \hat{\mathbf{y}} = \sum_i \dddot{\mathbf{y}}_i \mathbf{x}_i^T \mathbf{x}$$

Linear Regression in High-dimensional Spaces

$$\hat{\mathbf{y}} = \sum_i \dddot{\mathbf{y}}_i \mathbf{x}_i^T \mathbf{x}$$

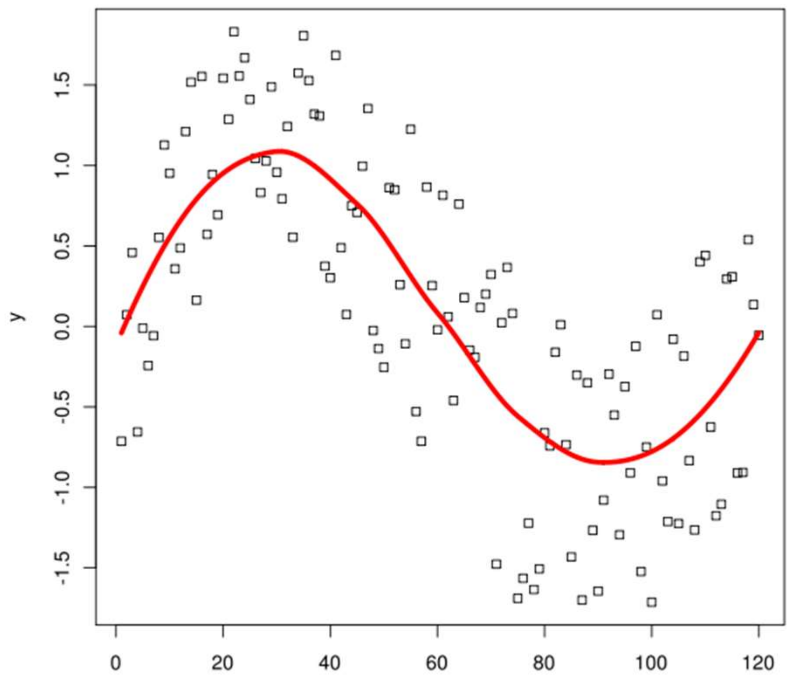
$$\dddot{\mathbf{Y}} = \mathbf{Y}\mathbf{G}^{-1}$$

- Given training instances $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1..N$, estimate $\mathbf{y}$ for a new test instance of $\mathbf{x}$ with unknown $\mathbf{y}$:
- $\mathbf{y}$ is simply a *weighted sum of the normalized* $\mathbf{y}_i$ *instances from the training data*
  - The normalization is done via the Gram Matrix
- The weight of any $\mathbf{y}_i$ is simply the inner product between its corresponding $\mathbf{x}_i$ and the new $\mathbf{x}$

Topics

- Nearest neighbor regression and classification
- Linear regression
  - With an application to glitch elimination in sound
  - And its relation to nearest-neighbor regression
- **Regression in kernel spaces**
- Kernel regression
- Regularization..

Relationships are not always linear



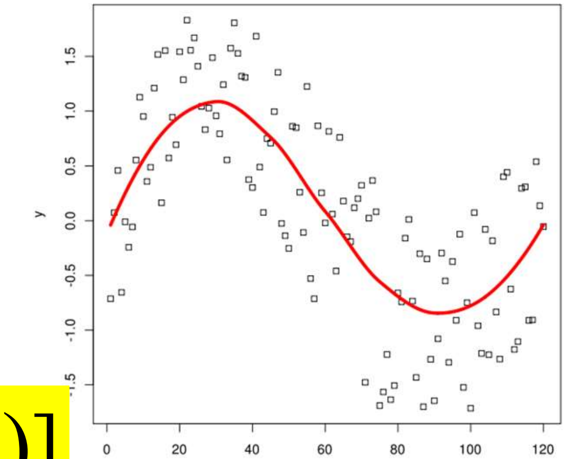
- How do we model these?
- Multiple solutions

Non-linear regression

- $y = \mathbf{A}\varphi(\mathbf{x}) + e$

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$$

$$\mathbf{X} \rightarrow \Phi(\mathbf{X}) = [\boldsymbol{\varphi}(\mathbf{x}_1)\ \ \boldsymbol{\varphi}(\mathbf{x}_2) \ldots \boldsymbol{\varphi}(\mathbf{x}_K)]$$

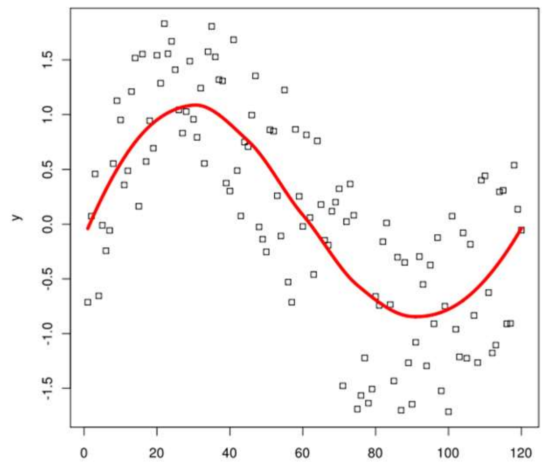


- $\mathbf{Y} = \mathbf{A}\Phi(\mathbf{X}) + \mathbf{e}$
- Replace $\mathbf{X}$ with $\Phi(\mathbf{X})$ in earlier equations for solution

$$\mathbf{A} = \mathbf{Y}\left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T\right)^{-1}\Phi(\mathbf{X})^T$$

Problem

- $\mathbf{Y} = \mathbf{A}\Phi(\mathbf{X}) + \mathbf{e}$
- Replace $\mathbf{X}$ with $\Phi(\mathbf{X})$ in earlier equations for solution



$$\mathbf{A} = \mathbf{Y}\left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T\right)^{-1}\Phi(\mathbf{X})^T$$

- $\Phi(\mathbf{X})$ may be in a very high-dimensional space
- The high-dimensional space (or the transform $\Phi(\mathbf{X})$) may be unknown..
  - Note: For any new instance **x**:

$$\hat{\mathbf{y}} = \mathbf{A}\Phi(\mathbf{x}) = \mathbf{Y}\left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T\right)^{-1}\Phi(\mathbf{X})^T\Phi(\mathbf{x}) = \mathbf{Y}\mathbf{G}^{-1}\Phi(\mathbf{X})^T\Phi(\mathbf{x})$$

The regression is in high dimensions

- *Linear regression*: $\hat{\mathbf{y}} = \sum_i \dddot{\mathbf{y}}_i \mathbf{x}_i^T \mathbf{x}$ $\quad \dddot{\mathbf{Y}} = \mathbf{Y}\mathbf{G}^{-1}$

- High-dimensional regression

$$\mathbf{G} = \begin{bmatrix} \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_2) & \cdots & \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_N) \\ \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_2) & \cdots & \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_N)^T \Phi(\mathbf{x}_2) & \cdots & \Phi(\mathbf{x}_N)^T \Phi(\mathbf{x}_N) \end{bmatrix}$$

$$\dddot{\mathbf{Y}} = \mathbf{Y}\mathbf{G}^{-1} \qquad \hat{\mathbf{y}} = \sum_i \dddot{\mathbf{y}}_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x})$$

Doing it with Kernels

- *High-dimensional regression with Kernels*:

$$K(\mathbf{x},\mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$$

$$\mathbf{G} = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_1) & \cdots & K(\mathbf{x}_1,\mathbf{x}_N) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & \cdots & K(\mathbf{x}_2,\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_N,\mathbf{x}_1) & K(\mathbf{x}_N,\mathbf{x}_2) & \cdots & K(\mathbf{x}_N,\mathbf{x}_N) \end{bmatrix}$$

- Regression in Kernel Hilbert Space..

$$\dddot{\mathbf{Y}} = \mathbf{Y}\mathbf{G}^{-1}$$

$$\hat{\mathbf{y}} = \sum_i \dddot{\mathbf{y}}_i K(\mathbf{x}_i, \mathbf{x})$$

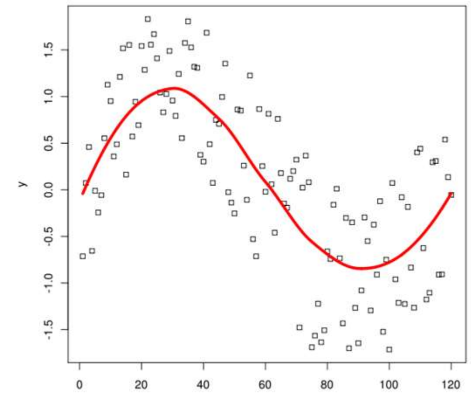
Poll 4

Topics

- Nearest neighbor regression and classification
- Linear regression
  – With an application to glitch elimination in sound
  – And its relation to nearest-neighbor regression
- Regression in kernel spaces
- **Kernel regression**
- Regularization..

A different way of finding nonlinear relationships: Locally linear regression

- Previous discussion: Regression parameters are optimized over the entire training set
- Minimize

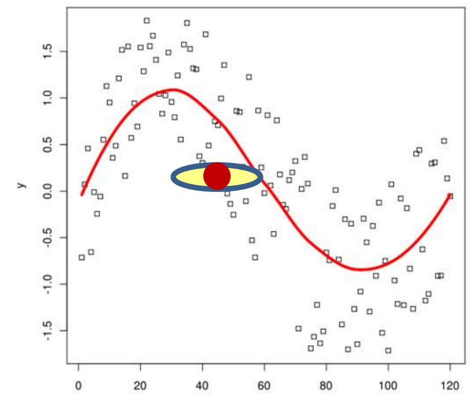


$$\mathbf{E} = \sum_{all\ i} \left\| \mathbf{y}_i - \mathbf{A}^T \mathbf{x}_i - \mathbf{b} \right\|^2$$

- Single global regression is estimated and applied to all future $\mathbf{x}$
- Alternative: *Local regression*
- *Learn a regression that is specific to* $\mathbf{x}$

Being non-committal: Local Regression

- Estimate the regression to be applied to any $\mathbf{x}$ using training instances near $\mathbf{x}$



$$\mathbf{E} = \sum_{\mathbf{x}_j \in neighborhood(\mathbf{x})} \left\| \mathbf{y}_i - \mathbf{A}^T \mathbf{x}_i - \mathbf{b} \right\|^2$$

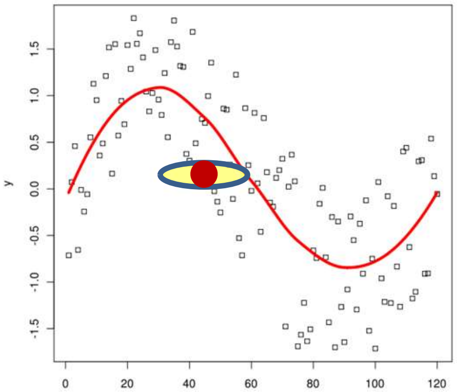
- The resultant regression has the form

$$\mathbf{y} = \sum_{\mathbf{x}_j \in neighborhood(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_j)\mathbf{y}_j + \mathbf{e}$$

  – Note : this regression is specific to $\mathbf{x}$
    - A separate regression must be learned for every $\mathbf{x}$

Local Regression

$$\mathbf{y} = \sum_{\mathbf{x}_j \in neighborhood(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_j)\mathbf{y}_j + \mathbf{e}$$

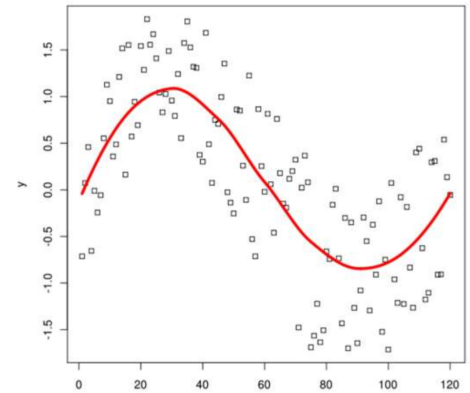


- But what is w()?
  - For linear regression d() is an inner product
- More generic form: Choose d() as a function of the *distance between* $\mathbf{x}$ *and* $\mathbf{x}_j$
- If w() falls off rapidly with $|\mathbf{x}$ *and* $\mathbf{x}_j|$ the "neighbhorhood" requirement can be relaxed

$$\mathbf{y} = \sum_{all} w(\mathbf{x}, \mathbf{x}_j)\mathbf{y}_j + \mathbf{e}$$

Kernel Regression: d() = K()

$$\hat{\mathbf{y}} = \frac{\sum_i K_h(\mathbf{x}-\mathbf{x}_i)\mathbf{y}_i}{\sum_i K_h(\mathbf{x}-\mathbf{x}_i)}$$



- Typical Kernel functions: Gaussian, Laplacian, other density functions
  - Must fall off rapidly with increasing distance between $\mathbf{x}$ and $\mathbf{x}_j$
- Regression is *local* to every $\mathbf{x}$: Local regression
- Actually a non-parametric MAP estimator of $\mathbf{y}$
  - But first.. MAP estimators..

Topics

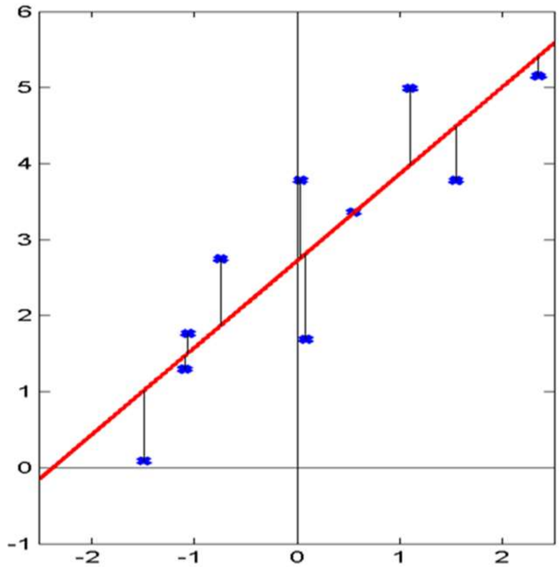
- Nearest neighbor regression and classification
- Linear regression
  - With an application to glitch elimination in sound
  - And its relation to nearest-neighbor regression
- Regression in kernel spaces
- Kernel regression
- **Regularization..**

Returning to Linear Regression

Model:

$$y = \widehat{A}x + \widehat{b}$$

$$\widehat{A}, \widehat{b} = \underset{A,b}{\operatorname{argmin}} (Y - (Ax + b))^2$$



Without outliers

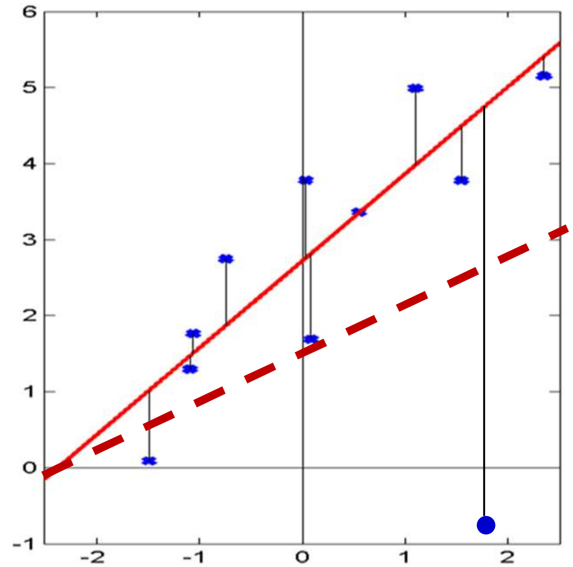
- The problem with fitting a linear model to minimize L2 error
  - Highly sensitive to outliers

Returning to Linear Regression

Model:

$$y = \widehat{A}x + \widehat{b}$$

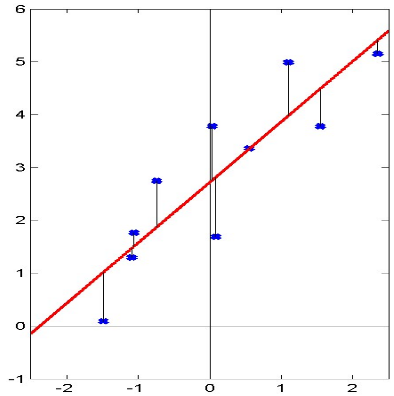
$$\widehat{A}, \widehat{b} = \underset{A,b}{\operatorname{argmin}} (Y - (Ax + b))^2$$

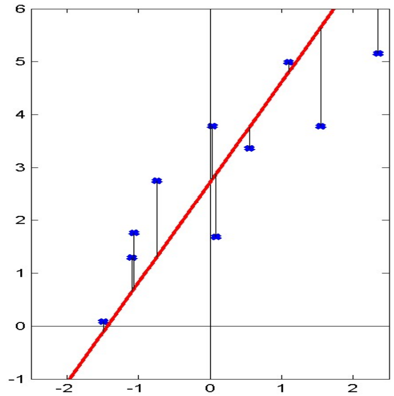


With a single outlier

- The problem with fitting a linear model to minimize L2 error
  - Highly sensitive to outliers

A problem with regressions





$$\mathbf{A} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}$$

- Least-squares fit is sensitive
  - Error is squared
  - Small variations in data → large variations in weights
  - Outliers affect it adversely
- Unstable
  - If dimension of $\mathbf{X}$ >= no. of instances
    - $(\mathbf{X}\mathbf{X}^T)$ is not invertible

Conservative solution

- Default: Y is extremely sensitive to X
  - Results in large changes in regression estimate in response to small changes in input

- Alternate default assumption: Y does not depend on X
  - Prediction is just a horizontal line at $Y = 0$
  - Useless

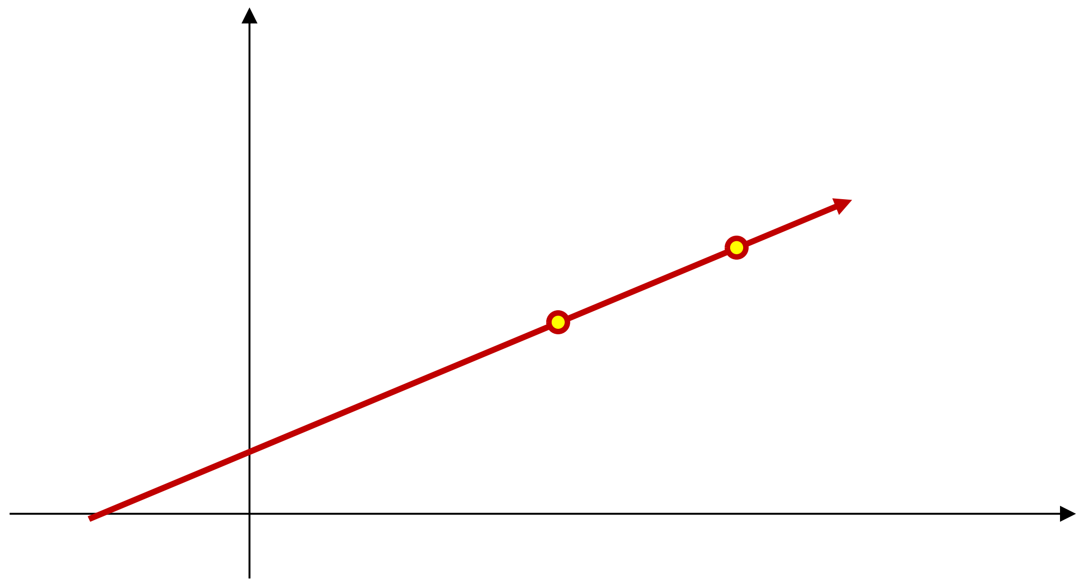
- Conservative Compromise: Y is *weakly* related to X
  - Large increments in X result in small increments in Y
  - Willing to change opinion if we see a large number of instances where a large increment in X resulted in a large change in Y
    - Seeing just a few instances will not satisfy us
      - Reduced sensitivity to outliers

The Believer's Linear Regression

Model:

$$y = \widehat{A}x + \widehat{b} + err$$

$$\widehat{A}, \widehat{b} = \underset{A,b}{\operatorname{argmin}} (Y - (Ax + b))^2$$



- Response of standard regression given only two training instances
  - Belief: Observed data tell the entire truth
    - Model completely fit to trends in data
    - A single point is a trend

The Disbeliever's Linear Regression

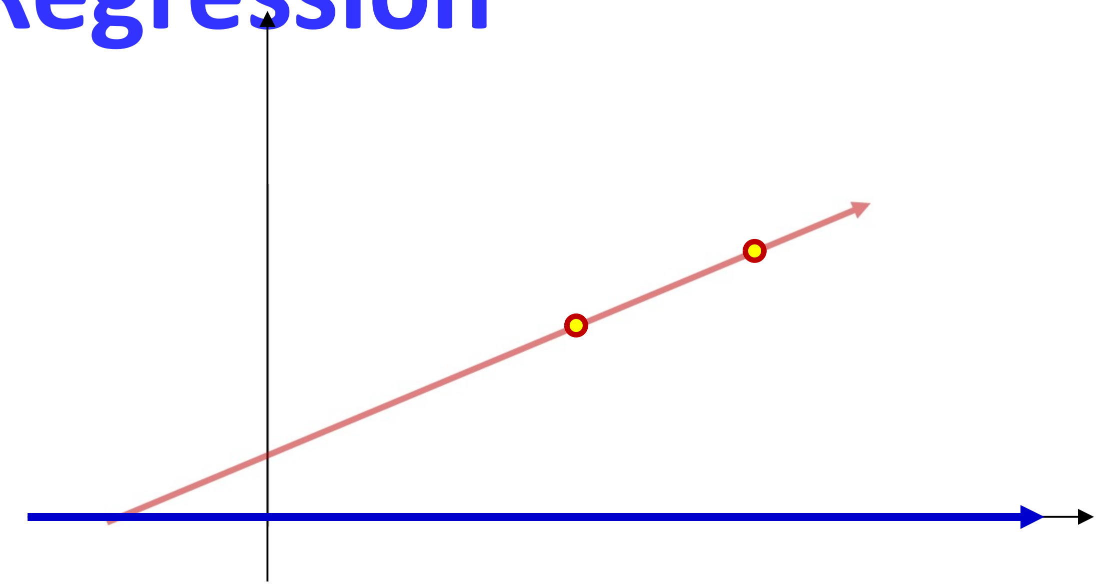
Model:

$y = err$

Alternately stated:

$y = Ax + b + err$

$A = b = 0$



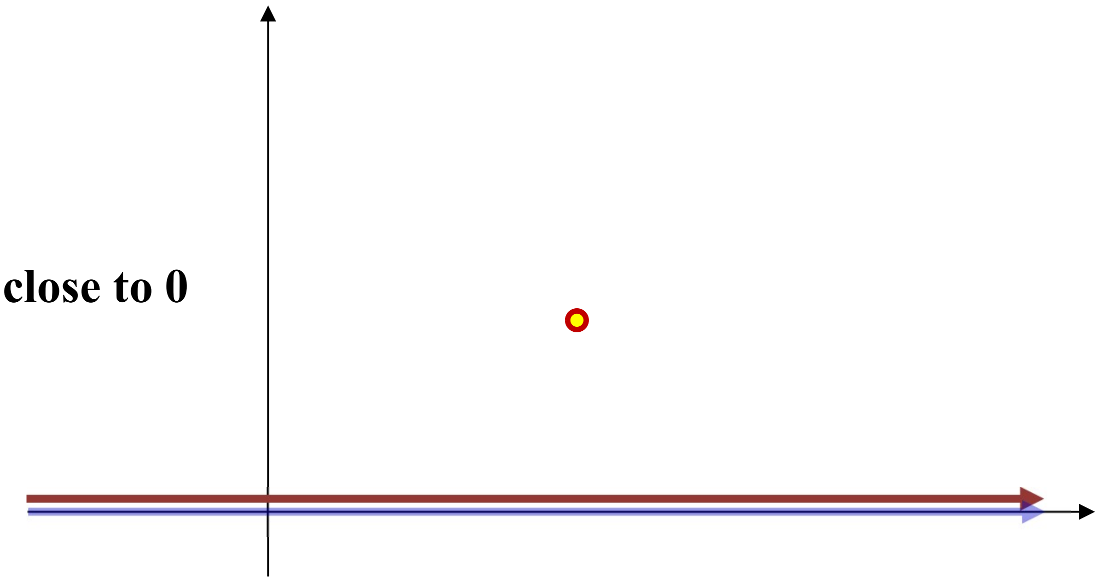
- All data are noise
  - The truth is that Y is a zero-mean random variable
  - The observed data are outcomes of noise variations

The Conservative Regression

Model:

$$y = Ax + b + err$$

Strong belief that A and b are close to 0



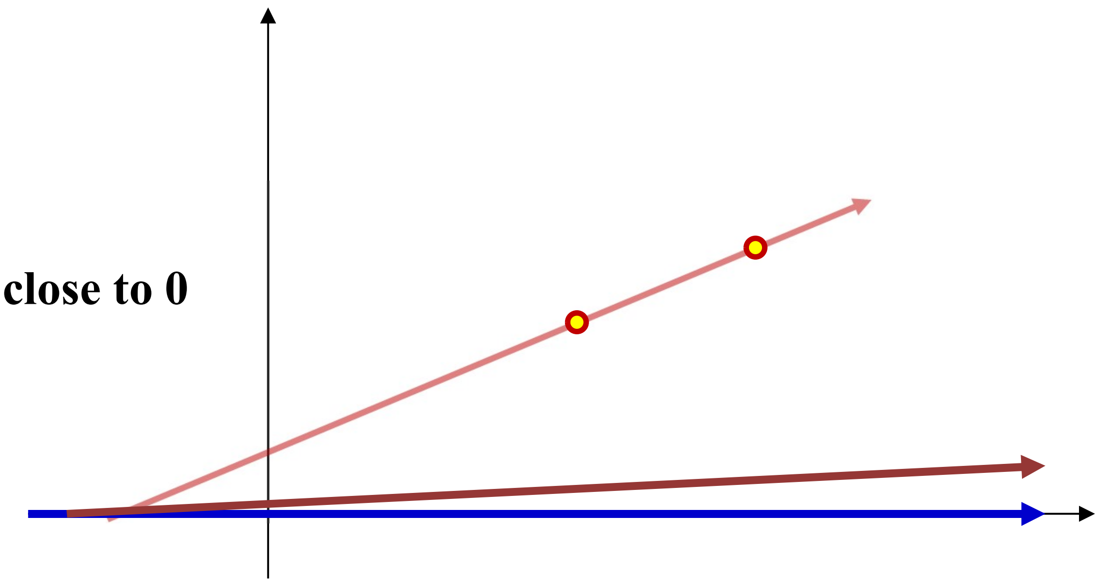
- After seeing only one point..

The Conservative Regression

Model:

$$y = Ax + b + err$$

Strong belief that A and b are close to 0



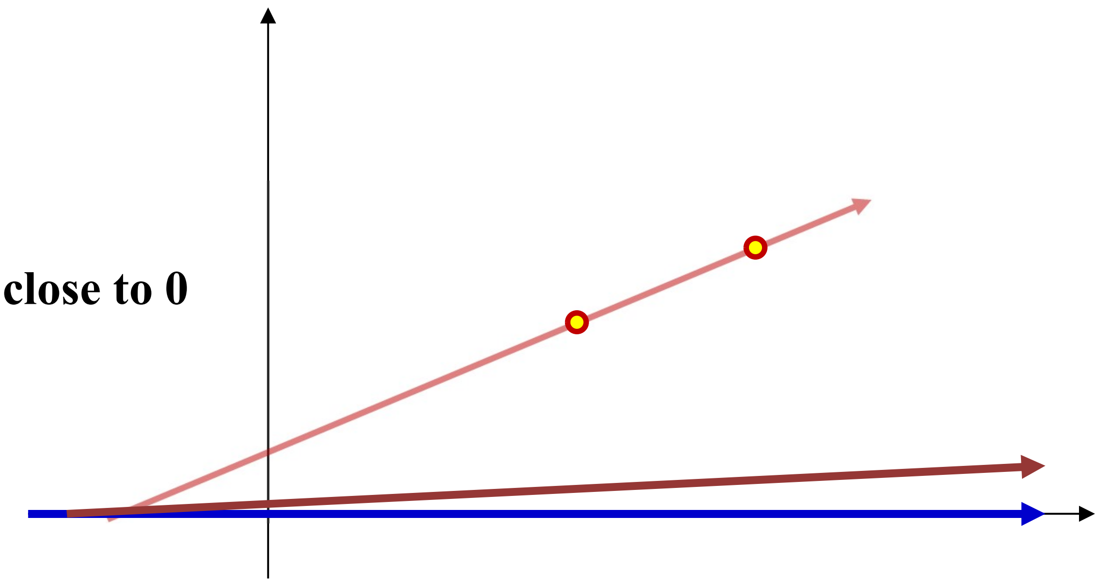
- The data provide evidence, but belief in the default is strong

The Conservative Regression

Model:

$$y = Ax + b + err$$

Strong belief that A and b are close to 0



$$\hat{A}, \hat{b} = \underset{A,b}{\operatorname{argmin}} \sum_i \|y_i - (Ax_i + b)\|^2 + \lambda(A^2 + b^2), \qquad \lambda > 0$$

- Minimize the error of prediction by the model
- But also insist that A and b be as small as possible
  - λ gives measure of "insistence" that A and b be small
  - Externally set

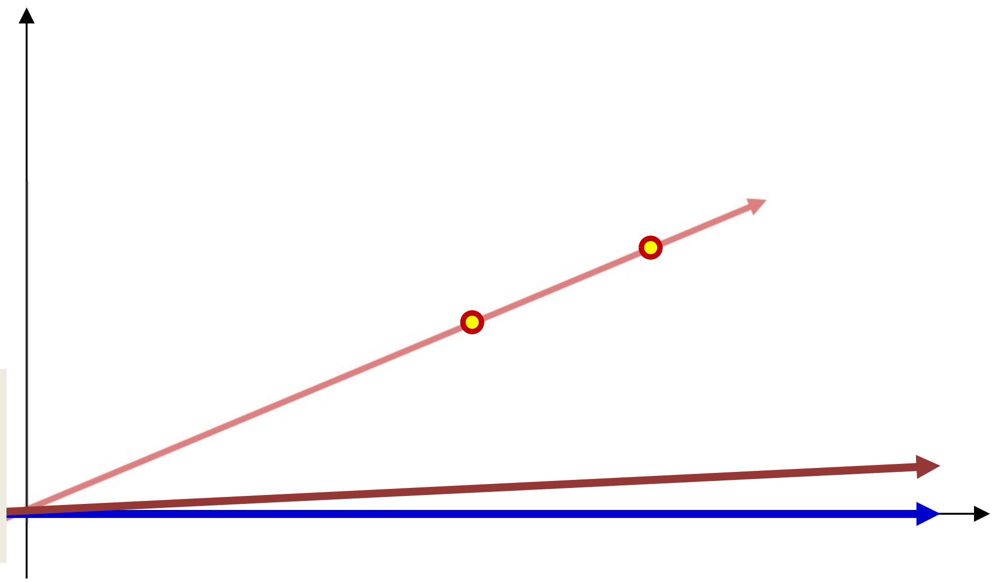
The Conservative Regression

Model:

$$y = A\hat{x} + err$$

Strong belief that A is close to 0

Using the augmented x notation (padding x with a 1) to include bias term



$$\hat{A} = \underset{A}{\operatorname{argmin}} \sum_i \|y_i - A\hat{x}_i\|^2 + \lambda\|A\|_F^2, \qquad \lambda > 0$$

- Minimize the error of prediction by the model
- But also insist that A should be as small as possible
  - λ gives measure of "insistence" that A must be small
  - Externally set

Simple solution

- Conventional solution:

$$\hat{\boldsymbol{A}} = \underset{A}{\operatorname{argmin}} \left\| \boldsymbol{Y} - \boldsymbol{A}\hat{\boldsymbol{X}} \right\|_F^{\mathbf{2}}$$

$$\hat{\boldsymbol{A}} = \boldsymbol{Y}\hat{\boldsymbol{X}}\left(\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^T\right)^{-1}$$

- With regularization

$$\hat{\boldsymbol{A}} = \underset{A}{\operatorname{argmin}} \left\| \boldsymbol{Y} - \boldsymbol{A}\hat{\boldsymbol{X}} \right\|_F^{\mathbf{2}} + \lambda \|\boldsymbol{A}\|_{\boldsymbol{F}}^2$$

- Also called *Tikhonov Regularization* or *Ridge regression*
- Minmization gives us

$$\hat{\boldsymbol{A}} = \boldsymbol{Y}\hat{\boldsymbol{X}}\left(\hat{\boldsymbol{X}}\hat{\boldsymbol{X}}^T + \lambda \boldsymbol{I}\right)^{-1}$$

- This is exactly the same as conventional estimation, with additional diagonal loading of the correlation matrix of $\hat{\boldsymbol{X}}$
  - Can be alternately explained as “stabilizing” the correlation matrix, for inversion

Other forms of regularization: L1 regularization

- An alternate regularization

$$\widehat{\boldsymbol{A}} = \underset{A}{\operatorname{argmin}} \left\| \boldsymbol{Y} - \boldsymbol{A}\widehat{\boldsymbol{X}} \right\|_F^{\mathbf{2}} + \lambda |\mathbf{A}|_1$$

- The one-norm $|\mathbf{A}|_1$ sums the magnitude of components of $\boldsymbol{A}$
  - The minimization causes $\boldsymbol{A}$ to be *sparse*
- No closed form solution
  - Quadratic programming solutions required
- Dual formulation

$$\widehat{\boldsymbol{A}} = \underset{A}{\operatorname{argmin}} \left\| \boldsymbol{Y} - \boldsymbol{A}\widehat{\boldsymbol{X}} \right\|_F^{\mathbf{2}} \quad \text{subject to } |\mathbf{A}|_1 \le t$$

- “LASSO” – Least absolute shrinkage and selection operator

Regularization

$$E = \|\mathbf{y} - \mathbf{a}^T X\|^2 + \Omega(\mathbf{a})$$

Constraints

$$\Omega(\mathbf{a}) = \sigma \|\mathbf{a}\|_2^2$$

Map Estimation

A Maximum Likelihood Estimator maximizes

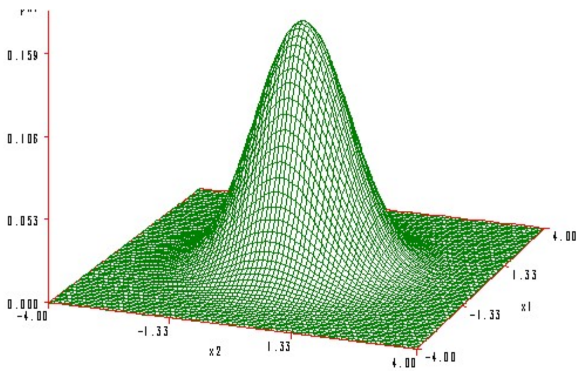
$$\mathbb{P}(\text{data} \mid \text{parameters})$$

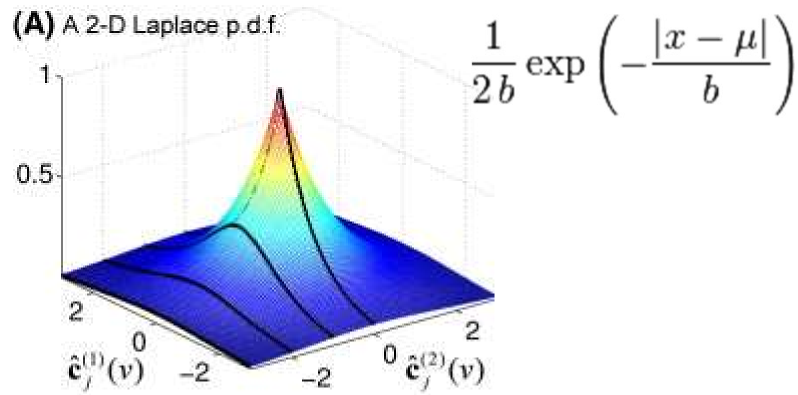
A Maximum A Posteriori Estimator maximizes

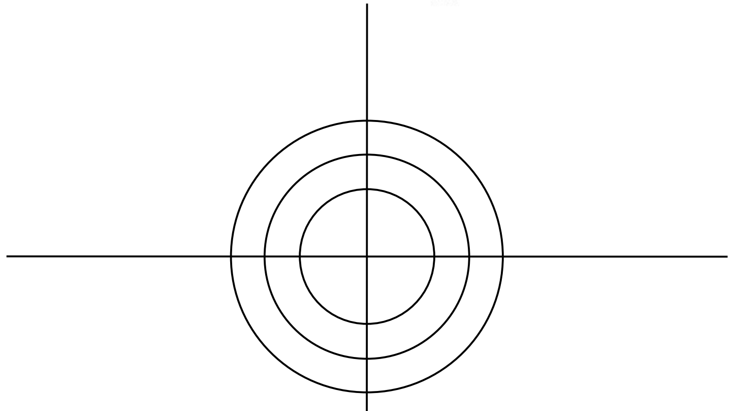
$$\mathbb{P}(\text{parameters} \mid \text{data})$$

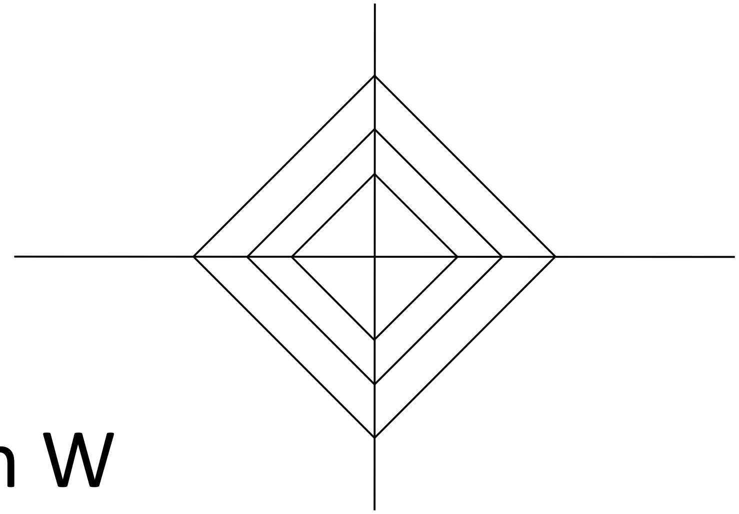
$$\mathbb{P}(\text{parameters} \mid \text{data}) = \frac{\mathbb{P}(\text{data} \mid \text{parameters}) \cdot \mathbb{P}(\text{parameters}}{\mathbb{P}(\text{data})}$$

MAP estimate priors









- Left: Gaussian Prior on W
- Right: Laplacian Prior

MAP estimate of weights

$$dL = \left(2\mathbf{a}^T\mathbf{X}\mathbf{X}^T + 2\mathbf{y}\mathbf{X}^T + 2\sigma\mathbf{I}\right)d\mathbf{a} = 0$$

$$\mathbf{a} = \left(\mathbf{X}\mathbf{X}^T + \sigma\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{Y}^T$$

- Equivalent to *diagonal loading* of correlation matrix
  - Improves condition number of correlation matrix
    - Can be inverted with greater stability
  - Will not affect the estimation from well-conditioned data
  - Also called Tikhonov Regularization
    - Dual form: Ridge regression
- **MAP estimate of *weights***
  - **Not to be confused with MAP estimate of Y**

MAP estimation of weights with Laplacian prior

- Assume weights drawn from a Laplacian
  - $P(\mathbf{a}) = \lambda^{-1}\exp(-\lambda^{-1}|\mathbf{a}|_1)$
- Maximum *a posteriori* estimate

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{A}} C' - (\mathbf{y} - \mathbf{a}^T\mathbf{X})^T(\mathbf{y} - \mathbf{a}^T\mathbf{X})^T - \lambda^{-1}|\mathbf{a}|_1$$

- No closed form solution
  - Quadratic programming solution required
    - Non-trivial

MAP estimation of weights with Laplacian prior

- Assume weights drawn from a Laplacian
  - $P(\mathbf{a}) = \lambda^{-1}\exp(-\lambda^{-1}|\mathbf{a}|_1)$
- Maximum *a posteriori* estimate

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{A}} C' - (\mathbf{y} - \mathbf{a}^T\mathbf{X})^T(\mathbf{y} - \mathbf{a}^T\mathbf{X})^T - \lambda^{-1}|\mathbf{a}|_1$$

- Identical to L_1 regularized least-squares estimation

L_1-regularized LSE

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{A}} C' - (\mathbf{y} - \mathbf{a}^T\mathbf{X})^T(\mathbf{y} - \mathbf{a}^T\mathbf{X})^T - \lambda^{-1}|\mathbf{a}|_1$$

- No closed form solution
  - Quadratic programming solutions required
- Dual formulation

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{A}} C' - (\mathbf{y} - \mathbf{a}^T\mathbf{X})^T(\mathbf{y} - \mathbf{a}^T\mathbf{X})^T \quad \textbf{subject to} \quad |\mathbf{a}|_1 \le t$$

- "LASSO" – Least absolute shrinkage and selection operator

LASSO Algorithms

- Various convex optimization algorithms
- LARS: Least angle regression
- Pathwise coordinate descent..
- Matlab code available from web

Regularized least squares

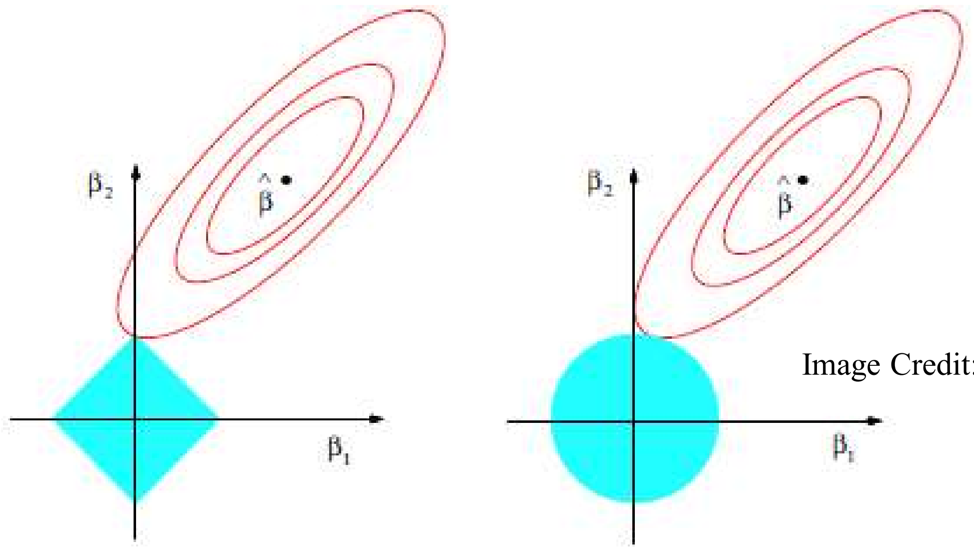


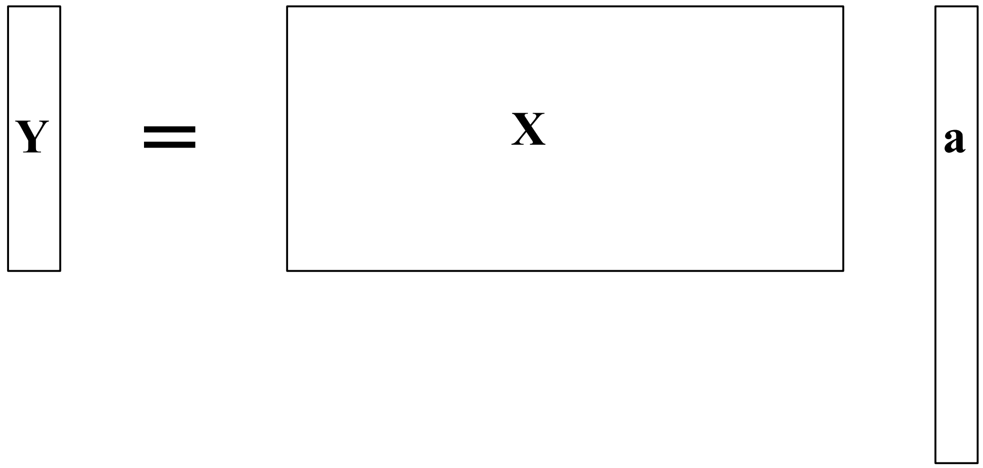
Image Credit: Tibshirani

- Regularization results in selection of suboptimal (in least-squares sense) solution
  - One of the loci outside center
- Tikhonov regularization selects ***shortest*** solution
- L_1 regularization selects ***sparsest*** solution

Next up..

- *Classification* with linear regression models
  - *AKA* linear classifiers

LASSO and Compressive Sensing



- Given $\mathbf{Y}$ and $\mathbf{X}$, estimate sparse $\mathbf{a}$
- LASSO:
  - $\mathbf{X}$ = explanatory variable
  - $\mathbf{Y}$ = dependent variable
  - $\mathbf{a}$ = weights of regression
- CS:
  - $\mathbf{X}$ = measurement matrix
  - $\mathbf{Y}$ = measurement
  - $\mathbf{a}$ = data

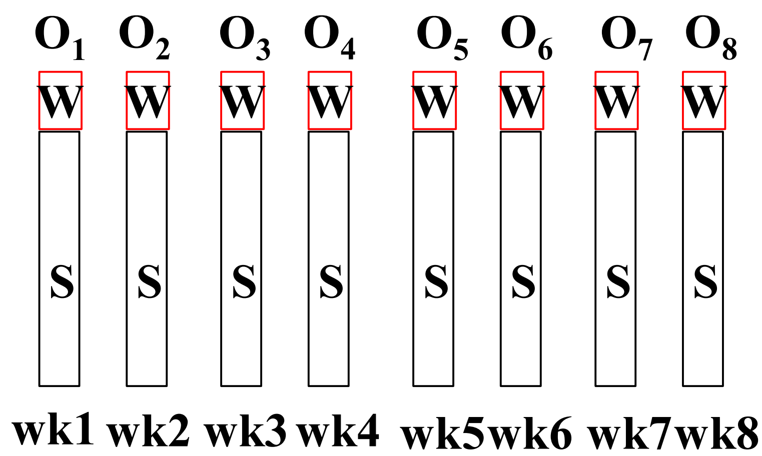
An interesting problem: Predicting War!

- Economists measure a number of social indicators for countries weekly
  - Happiness index
  - Hunger index
  - Freedom index
  - Twitter records
  - …
- Question: Will there be a revolution or war next week?

An interesting problem: Predicting War!

- Issues:
  - Dissatisfaction builds up – not an instantaneous phenomenon
    - Usually
  - War / rebellion build up much faster
    - Often in hours
- Important to predict
  - Preparedness for security
  - Economic impact

Predicting War



Given

- Sequence of economic indicators for each week
- Sequence of unrest markers for each week
  - At the end of each week we know if war happened or not that week

- Predict probability of unrest next week
  - This could be a new unrest or persistence of a current one

Predicting Time Series

- Need *time-series models*
- HMMs – later in the course