

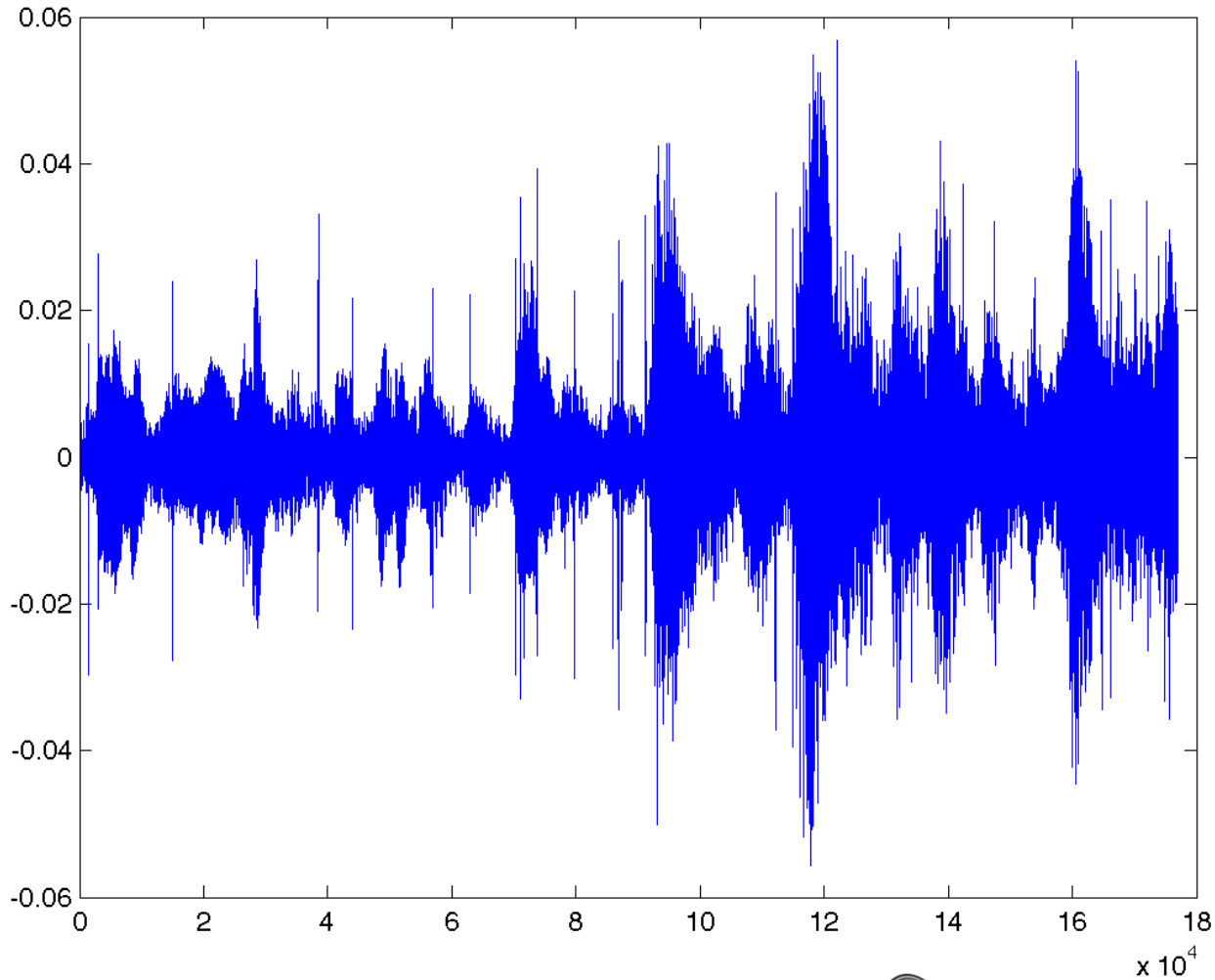
Machine Learning for Signal Processing

Regression and Prediction

Class 16. 29 Oct 2015

Instructor: Bhiksha Raj

A Common Problem



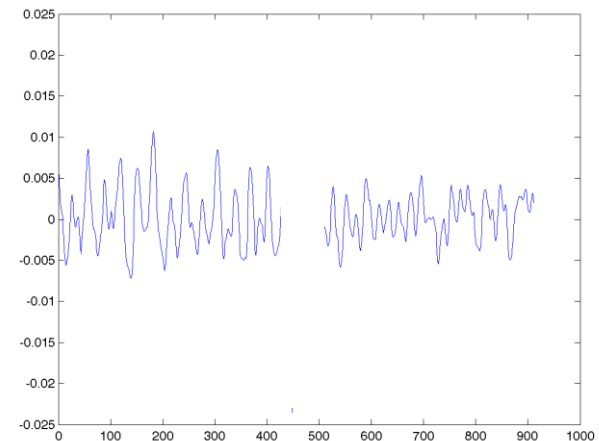
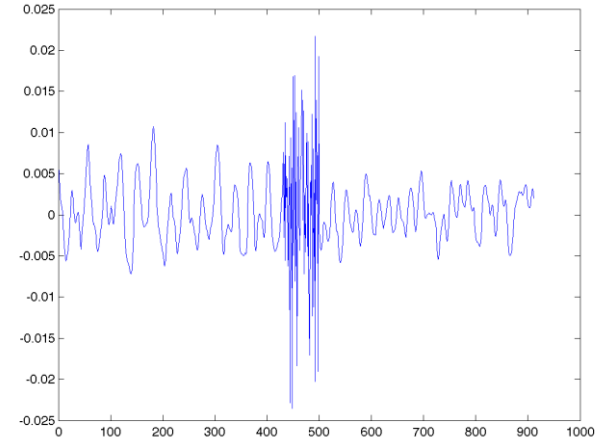
- Can you spot the glitches?



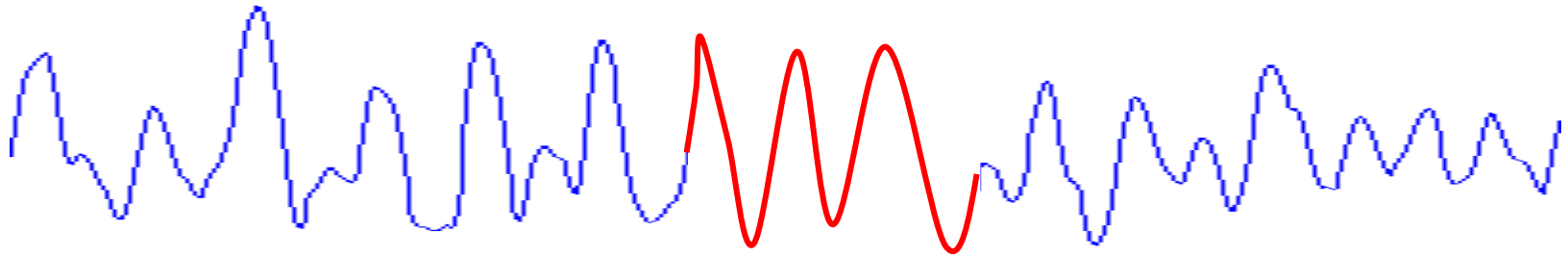
11755/18797

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?

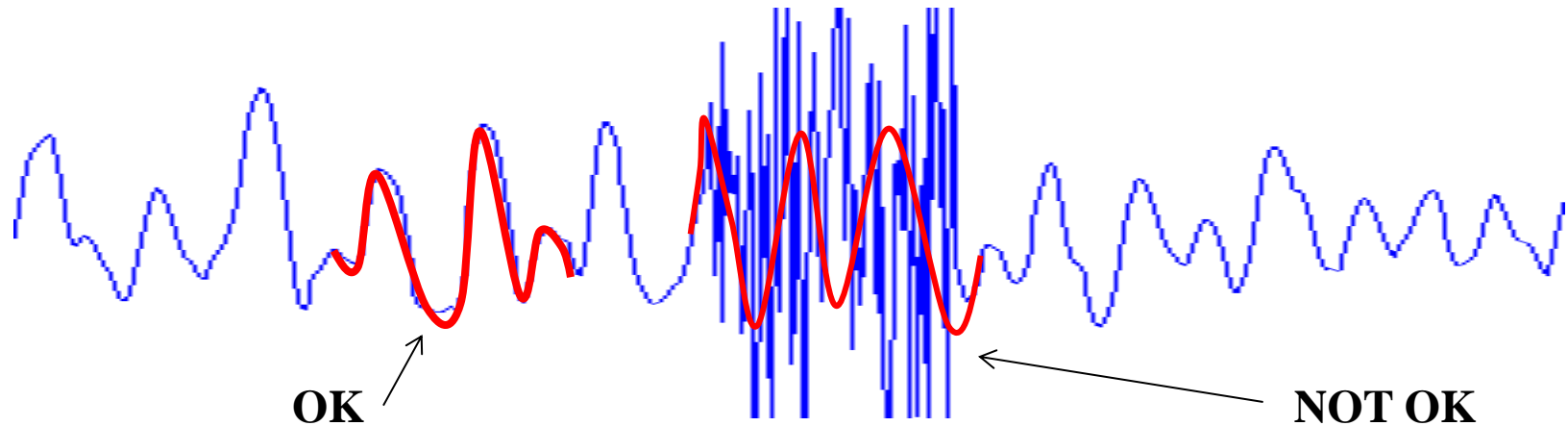


Interpolation..



- “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

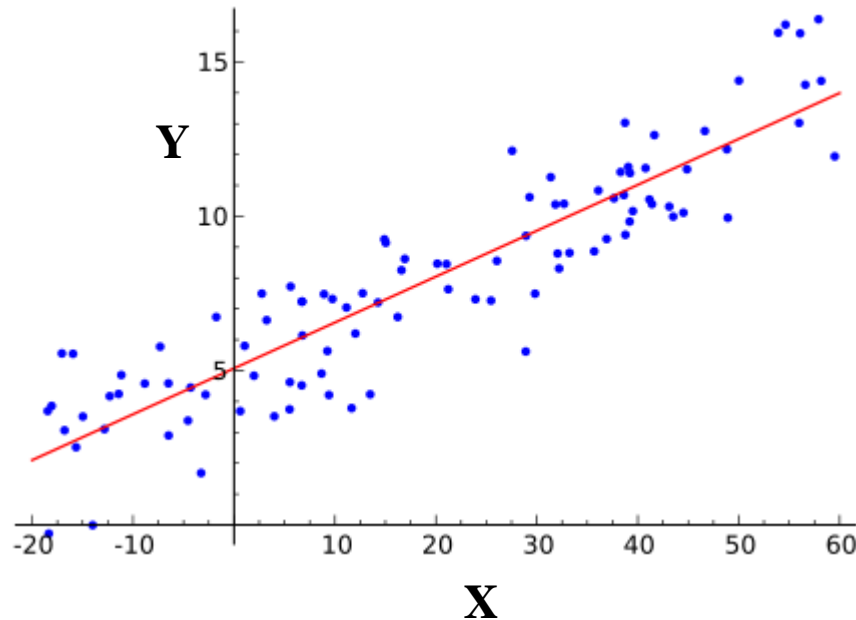
What is a regression

- 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
 - $f(\cdot)$ is a linear or affine function
 - $f(\cdot)$ is a non-linear function
 - $f(\cdot)$ 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..

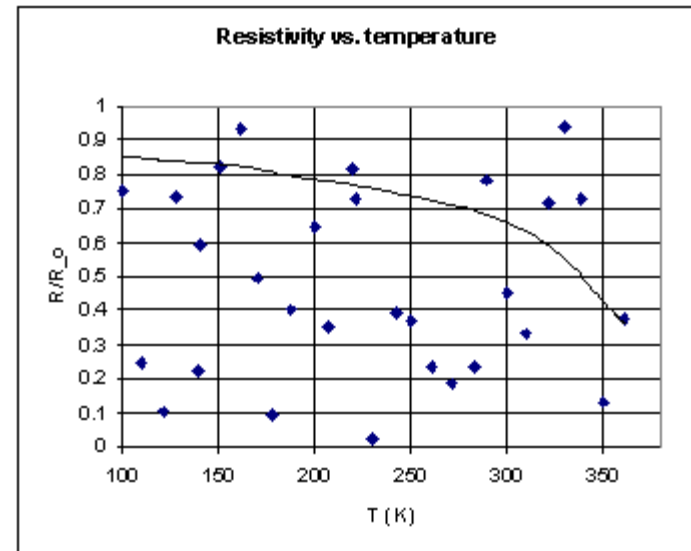
- <http://pages.cs.wisc.edu/~kovar/hall.html>

- 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.

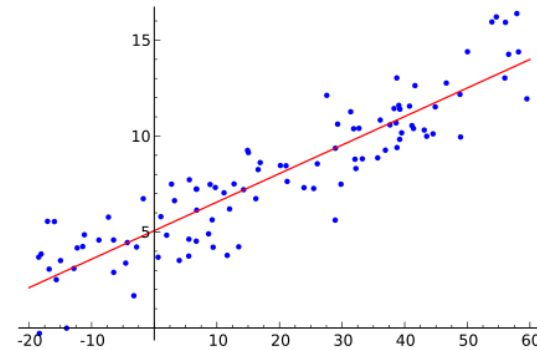
Christ, this was such a waste of my time.

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

$$y_1 = \mathbf{a}^T \mathbf{x}_1 + b + e_1$$

$$y_2 = \mathbf{a}^T \mathbf{x}_2 + b + e_2$$

$$y_3 = \mathbf{a}^T \mathbf{x}_3 + b + e_3$$

- Define:

$$\mathbf{y} = [y_1 \ y_2 \ y_3 \dots]$$

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

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \dots \\ 1 & 1 & 1 & \dots \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix}$$

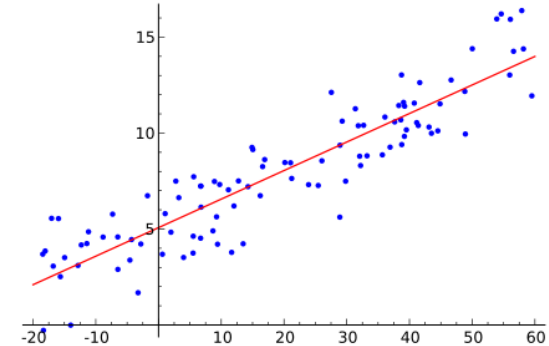
- Rewrite

$$\mathbf{y} = \mathbf{A}^T \mathbf{X} + \mathbf{e}$$

Learning the parameters

$$\mathbf{y} = \mathbf{A}^T \mathbf{X} + \mathbf{e}$$

$$\hat{\mathbf{y}} = \mathbf{A}^T \mathbf{X} \quad \text{Assuming no error}$$



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

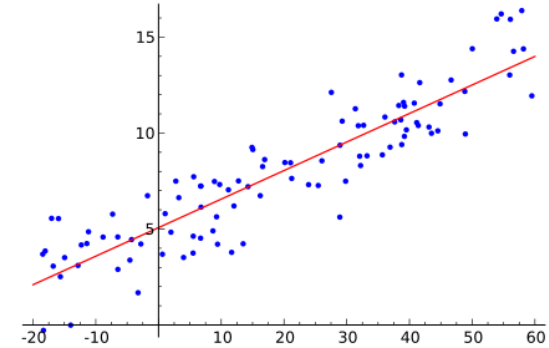
The prediction error as divergence

$$y_1 = \mathbf{a}^T \mathbf{x}_1 + b + e_1$$

$$y_2 = \mathbf{a}^T \mathbf{x}_2 + b + e_2$$

$$y_3 = \mathbf{a}^T \mathbf{x}_3 + b + e_3$$

$$\mathbf{y} = \mathbf{a}^T \mathbf{X} + \mathbf{e} = \hat{\mathbf{y}} + \mathbf{e}$$



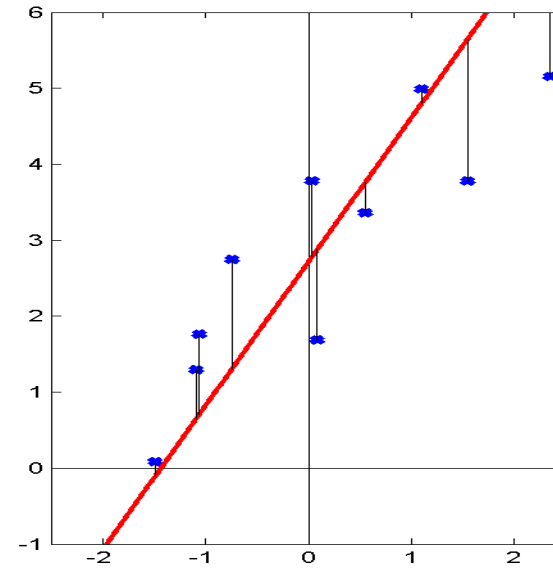
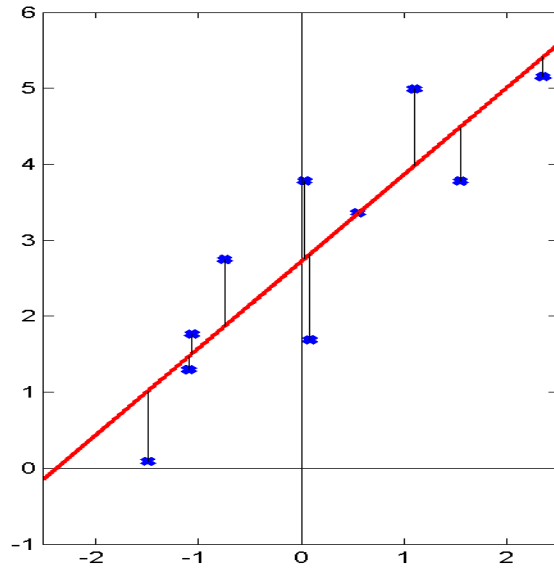
$$\mathbf{D}(\mathbf{y}, \hat{\mathbf{y}}) = \mathbf{E} = e_1^2 + e_2^2 + e_3^2 + \dots$$

$$= (y_1 - \mathbf{a}^T \mathbf{x}_1 - b)^2 + (y_2 - \mathbf{a}^T \mathbf{x}_2 - b)^2 + (y_3 - \mathbf{a}^T \mathbf{x}_3 - b)^2 + \dots$$

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

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

Prediction error as divergence



- $y = \mathbf{A}^T \mathbf{x} + e$
 - e = prediction error
 - Find the “slope” \mathbf{a} such that the total squared length of the error lines is minimized

Solving a linear regression

$$\mathbf{y} = \mathbf{A}^T \mathbf{X} + \mathbf{e}$$

- Minimize squared error

$$\mathbf{E} = \|\mathbf{y} - \mathbf{A}^T \mathbf{X}\|^2$$

$$\mathbf{A}^T = \mathbf{y} \mathit{pinv}(\mathbf{X})$$

$$\mathbf{A} = \mathit{pinv}(\mathbf{X})^T \mathbf{y}^T$$

More Explicitly

- Minimize squared error

$$\begin{aligned}\mathbf{E} &= \|\mathbf{y} - \mathbf{X}^T \mathbf{A}\|^2 = (\mathbf{y} - \mathbf{A}^T \mathbf{X})(\mathbf{y} - \mathbf{A}^T \mathbf{X})^T \\ &= \mathbf{y}\mathbf{y}^T + \mathbf{A}^T \mathbf{X}\mathbf{X}^T \mathbf{A} - 2\mathbf{y}\mathbf{X}^T \mathbf{A}\end{aligned}$$

- Differentiating w.r.t \mathbf{A} and equating to 0

$$d\mathbf{E} = (2\mathbf{A}^T \mathbf{X}\mathbf{X}^T - 2\mathbf{y}\mathbf{X}^T) d\mathbf{A} = 0$$

$$\mathbf{A}^T = \mathbf{y}\mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^{-1} = \mathit{ypinv}(\mathbf{X})$$

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

Regression in multiple dimensions

$$\mathbf{y}_1 = \mathbf{A}^T \mathbf{x}_1 + \mathbf{b} + \mathbf{e}_1$$

$$\mathbf{y}_2 = \mathbf{A}^T \mathbf{x}_2 + \mathbf{b} + \mathbf{e}_2$$

$$\mathbf{y}_3 = \mathbf{A}^T \mathbf{x}_3 + \mathbf{b} + \mathbf{e}_3$$

\mathbf{y}_i is a vector

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

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

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

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

$$\mathbf{y}_i = \mathbf{A}^T \mathbf{x}_i + \mathbf{b} + \mathbf{e}_i \quad \longrightarrow$$

$$y_{i1} = \mathbf{a}_1^T \mathbf{x}_i + b_1 + e_{i1}$$

$$y_{i2} = \mathbf{a}_2^T \mathbf{x}_i + b_2 + e_{i2}$$

$$y_{i3} = \mathbf{a}_3^T \mathbf{x}_i + b_3 + e_{i3}$$

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

Multiple Regression

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

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

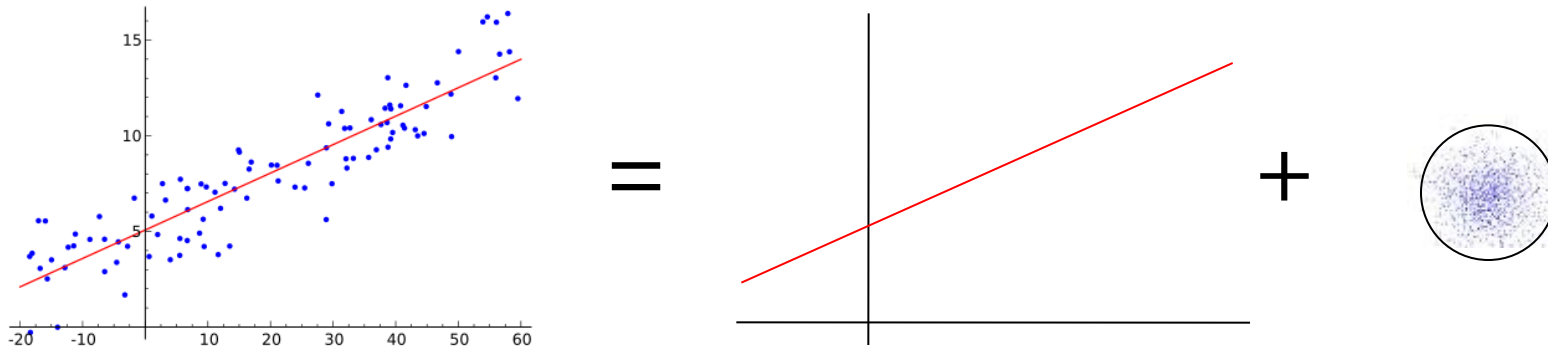
$$DIV = \sum_i \left\| \mathbf{y}_i - \hat{\mathbf{A}}^T \bar{\mathbf{x}}_i \right\|^2$$

- Minimizing

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

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

A Different Perspective



- \mathbf{y} is a noisy reading of $\mathbf{A}^T \mathbf{x}$

$$\mathbf{y} = \mathbf{A}^T \mathbf{x} + \mathbf{e}$$

- Error \mathbf{e} is Gaussian

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

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

The *Likelihood* of the data

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

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

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

- Probability of collection: $\mathbf{\Lambda} = [\boldsymbol{\lambda}^1 \ \boldsymbol{\lambda}^5 \cdots \boldsymbol{\lambda}^N]$ $\mathbf{X} = [\mathbf{x}^1 \ \mathbf{x}^5 \cdots \mathbf{x}^N]$

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

- Assuming IID for convenience (not necessary)

A Maximum Likelihood Estimate

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

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

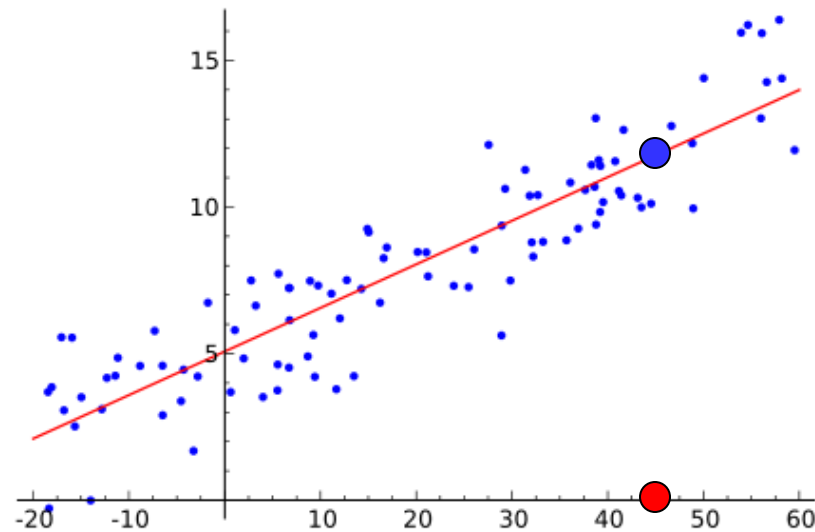
$$\log P(\mathbf{Y} | \mathbf{X}; \mathbf{A}) = C - \sum_i \frac{1}{2\sigma^2} \|\mathbf{y}_i - \mathbf{A}^T \mathbf{x}_i\|^2$$

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

$$\mathbf{A}^T = \mathbf{YX}^T (\mathbf{XX}^T)^{-1} = \mathbf{Y} \mathit{pinv}(\mathbf{X})$$

$$\mathbf{A} = (\mathbf{XX}^T)^{-1} \mathbf{XY}^T$$

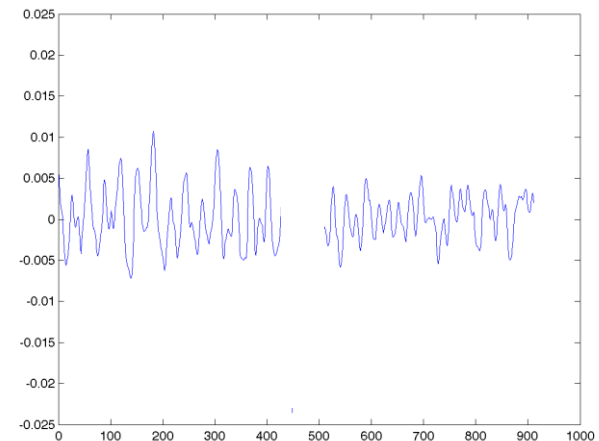
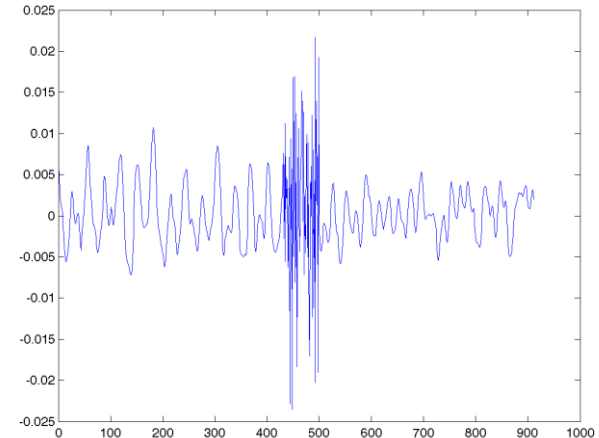
Predicting an output



- 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: $\hat{\mathbf{y}} = \mathbf{A}^T \mathbf{X}$

Applying it to our problem

- Prediction by regression
- Forward regression
- $x_t = a_1x_{t-1} + a_2x_{t-2} \dots a_kx_{t-k} + e.$
- Backward regression
- $x_t = b_1x_{t+1} + b_2x_{t+2} \dots b_kx_{t+k} +$



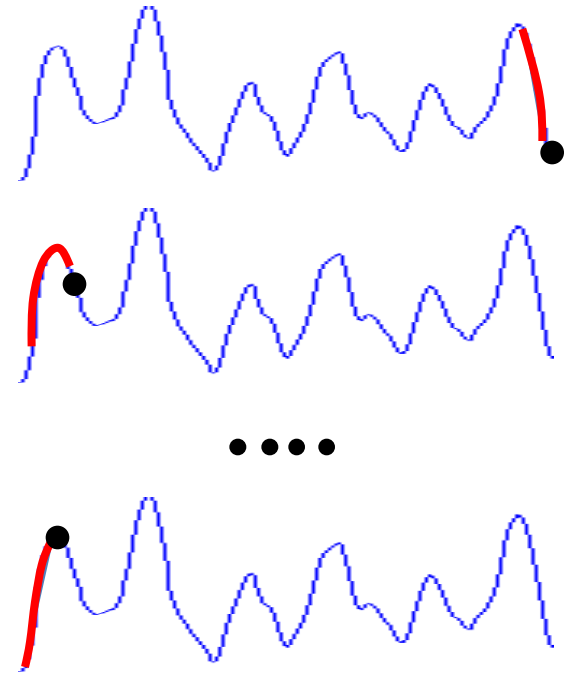
Applying it to our problem

- Forward prediction

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

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

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



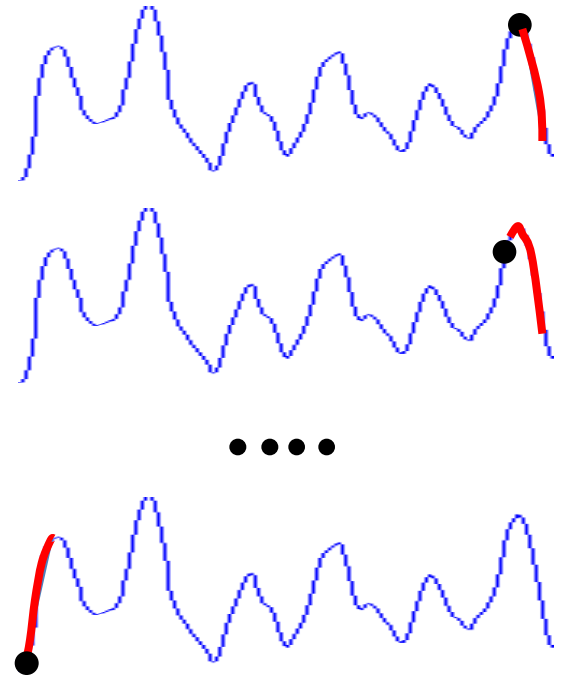
Applying it to our problem

- Backward prediction

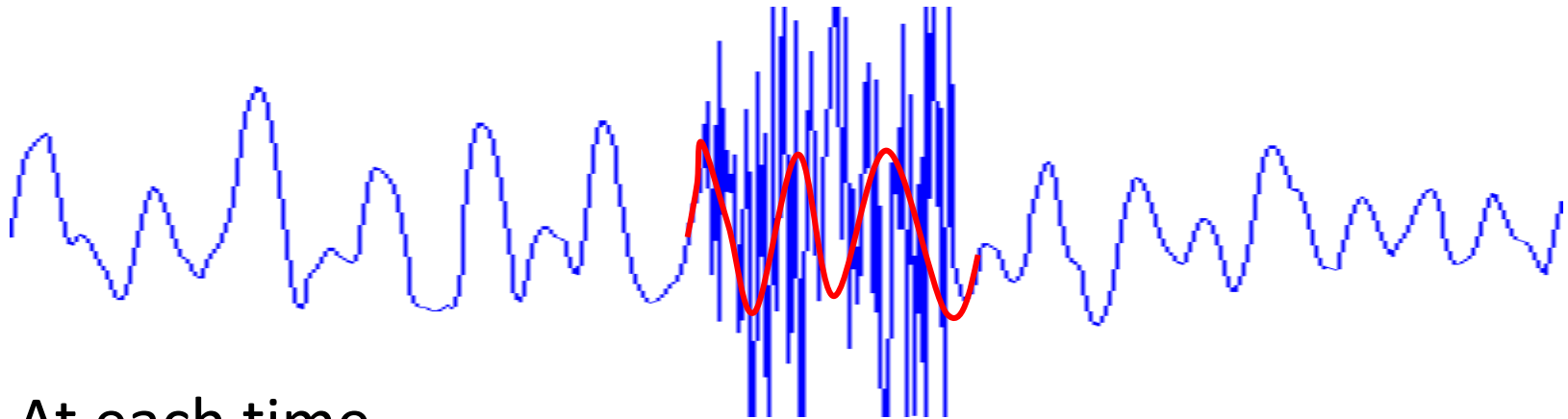
$$\begin{bmatrix} x_{t-K-1} \\ x_{t-K-2} \\ \dots \\ x_1 \end{bmatrix} = \begin{bmatrix} x_t & x_{t-1} & \dots & x_{t-K} \\ x_{t-1} & x_{t-2} & \dots & x_{t-K-1} \\ \dots & \dots & \dots & \dots \\ x_{K+1} & x_K & \dots & x_2 \end{bmatrix} \mathbf{b}_t + \begin{bmatrix} e_{t-K-1} \\ e_{t-K-2} \\ \dots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

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

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

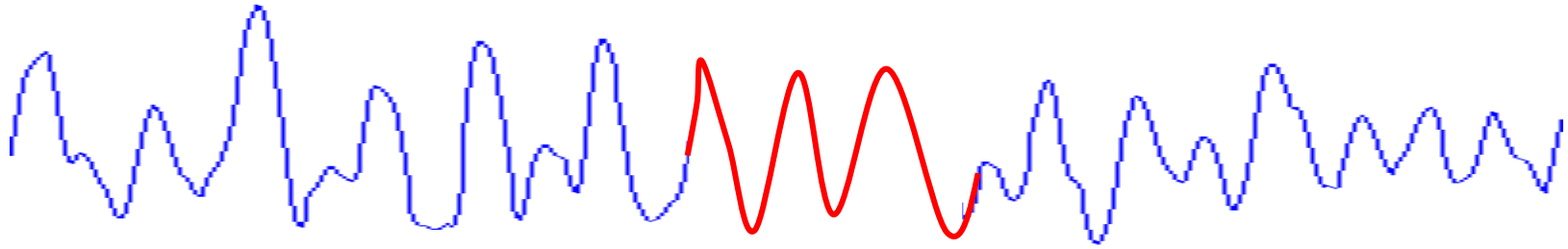


Finding the burst



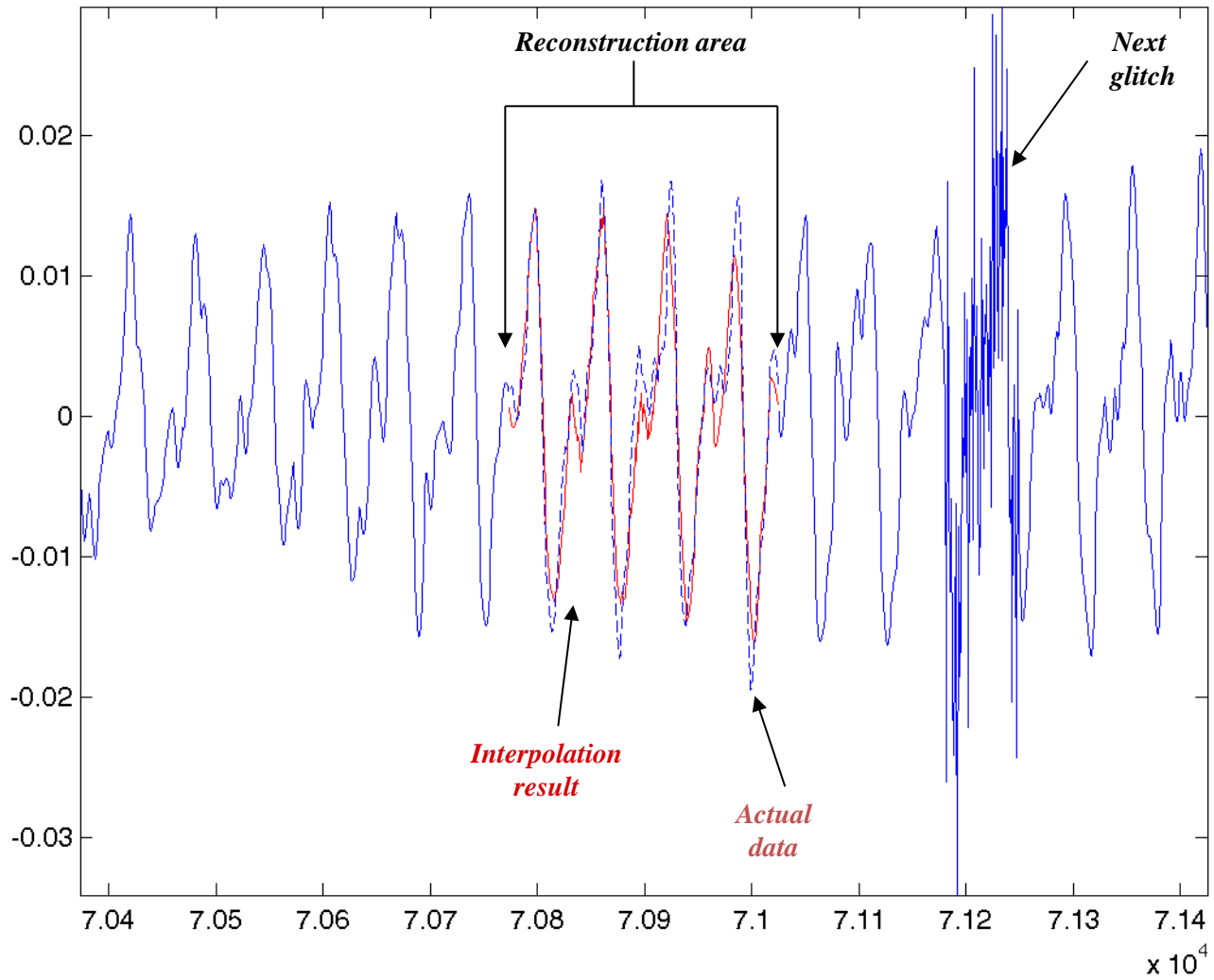
- At each time
 - Learn a “forward” predictor \mathbf{a}_t
 - At each time, predict next sample $x_t^{\text{est}} = \sum_i a_{t,k} x_{t-k}$
 - Compute error: $ferr_t = |x_t - x_t^{\text{est}}|^2$
 - Learn a “backward” predict and compute backward error
 - $berr_t$
 - Compute average prediction error 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 $x_t^{\text{est}} = \sum_i a_{t,k} x_{t-k}$
 - 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 $x_t^{\text{est}} = \sum_i b_{t,k} x_{t+k}$
 - Use estimated samples if real samples are not available
- Average forward and backward predictions

Reconstruction zoom in



Incrementally learning the regression

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

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

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \eta (y_t - \hat{y}_t) \mathbf{x}_t \quad \hat{y}_t = (\mathbf{a}^t)^T \mathbf{x}_t$$

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

Predicting a value

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

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

- 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

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

$$\hat{y} = \mathbf{Y}\hat{\mathbf{X}}^T \hat{\mathbf{x}} = \frac{1}{N} [\mathbf{y}_1 \quad \dots \quad \mathbf{y}_N] \begin{bmatrix} \hat{\mathbf{x}}_1^T \\ \vdots \\ \hat{\mathbf{x}}_N^T \end{bmatrix} \hat{\mathbf{x}} = \sum_i y_i (\hat{\mathbf{x}}_i^T \hat{\mathbf{x}})$$

- What are we doing exactly?

Predicting a value

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

- 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..

What are we doing: A different perspective

$$\hat{\mathbf{y}} = \mathbf{A}^T \mathbf{x} = \mathbf{YX}^T (\mathbf{XX}^T)^{-1} \mathbf{x}$$

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

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

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

High-dimensional regression

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

- $\mathbf{X}^T \mathbf{X}$ 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

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

- Working our way down..

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

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

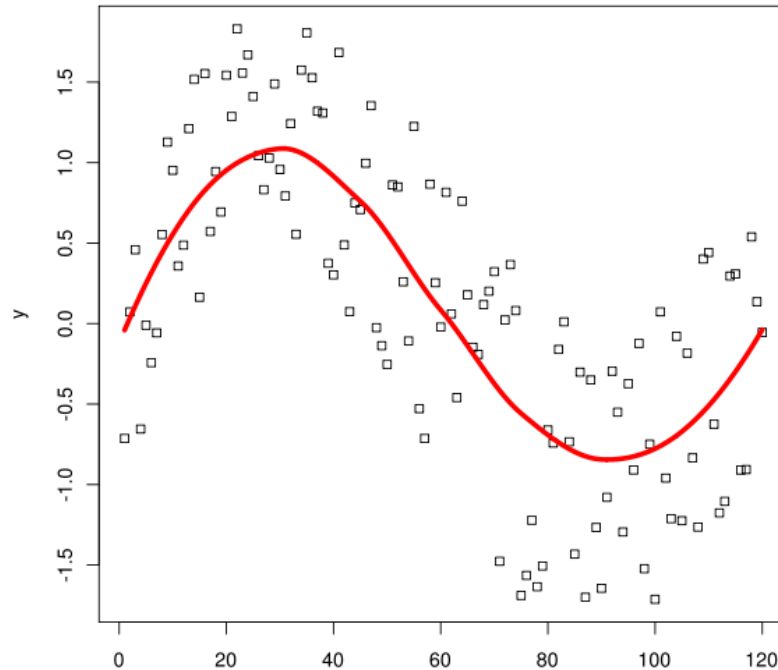
Linear Regression in High-dimensional Spaces

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

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

- 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}

Relationships are not always linear



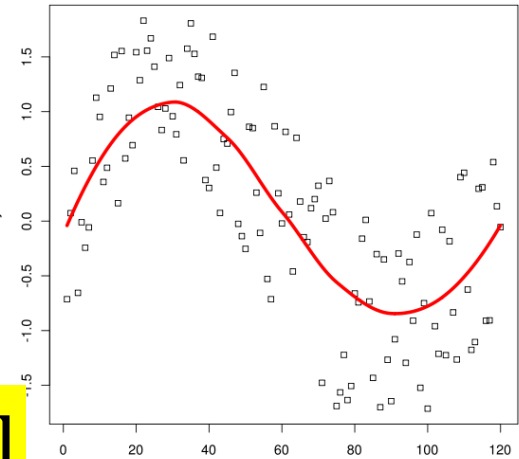
- How do we model these?
- Multiple solutions

Non-linear regression

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

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [\phi_1(\mathbf{x}) \ \phi_2(\mathbf{x}) \ \dots \ \phi_N(\mathbf{x})]$$

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



- $\mathbf{Y} = \mathbf{A}\Phi(\mathbf{X}) + e$

- Replace \mathbf{X} with $\Phi(\mathbf{X})$ in earlier equations for solution

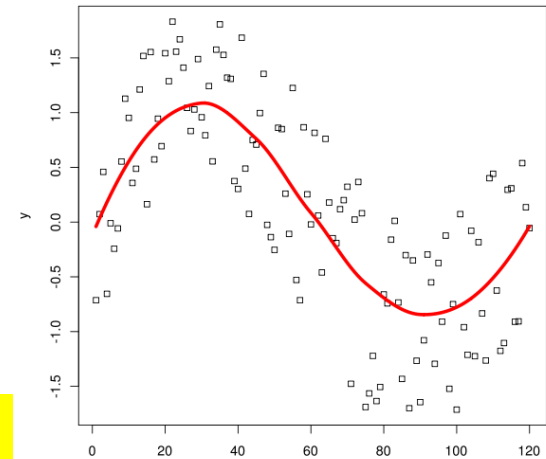
$$\mathbf{A} = \left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T \right)^{-1} \Phi(\mathbf{X})\mathbf{Y}^T$$

Problem

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

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

- $\Phi(\mathbf{X})$ may be in a very high-dimensional space
- The high-dimensional space (or the transform $\Phi(\mathbf{X})$) may be unknown..



The regression is in high dimensions

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

- High-dimensional regression

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

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

$$\hat{\mathbf{y}} = \sum_i \ddot{y}_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x})$$

Doing it with Kernels

- *High-dimensional regression with Kernels:*

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

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

- Regression in Kernel Hilbert Space..

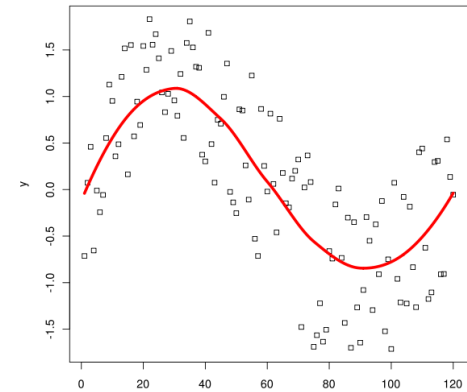
$$\ddot{\mathbf{Y}} = \mathbf{Y}\mathbf{G}^{-1}$$

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

A different way of finding nonlinear relationships: Locally linear regression

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

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

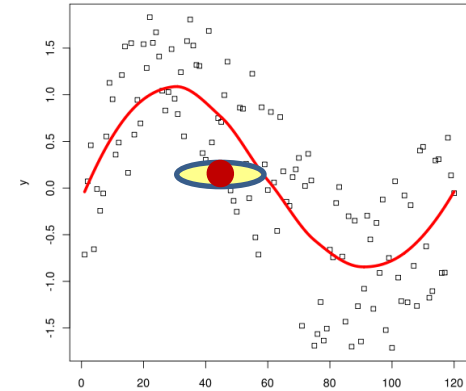


- Single global regression is estimated and applied to all future \mathbf{x}
- Alternative: *Local regression*
- *Learn a regression that is specific to \mathbf{x}*

Being non-committal: Local Regression

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

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



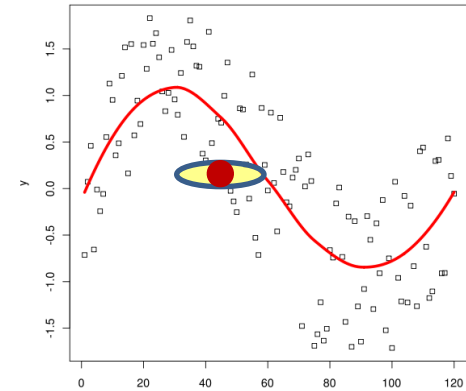
- The resultant regression has the form

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

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

Local Regression

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

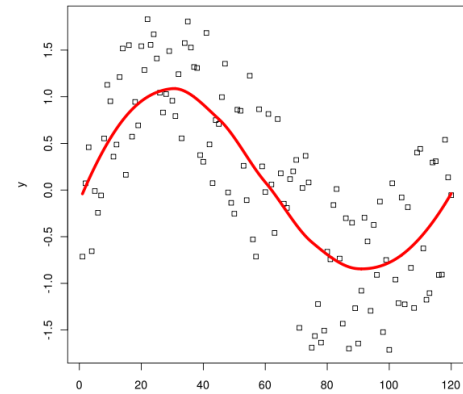


- But what is $d()$?
 - For linear regression $d()$ is an inner product
- More generic form: Choose $d()$ as a function of the *distance between \mathbf{x} and \mathbf{x}_j*
- If $d()$ falls off rapidly with $|\mathbf{x} \text{ and } \mathbf{x}_j|$ the “neighborhood” requirement can be relaxed

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

Kernel Regression: $d() = K()$

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



- 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

Map Estimators

- MAP (*Maximum A Posteriori*): Find a “best guess” for \mathbf{y} (statistically), given known \mathbf{x}

$$\mathbf{y} = \operatorname{argmax}_Y P(\mathbf{Y}|\mathbf{x})$$

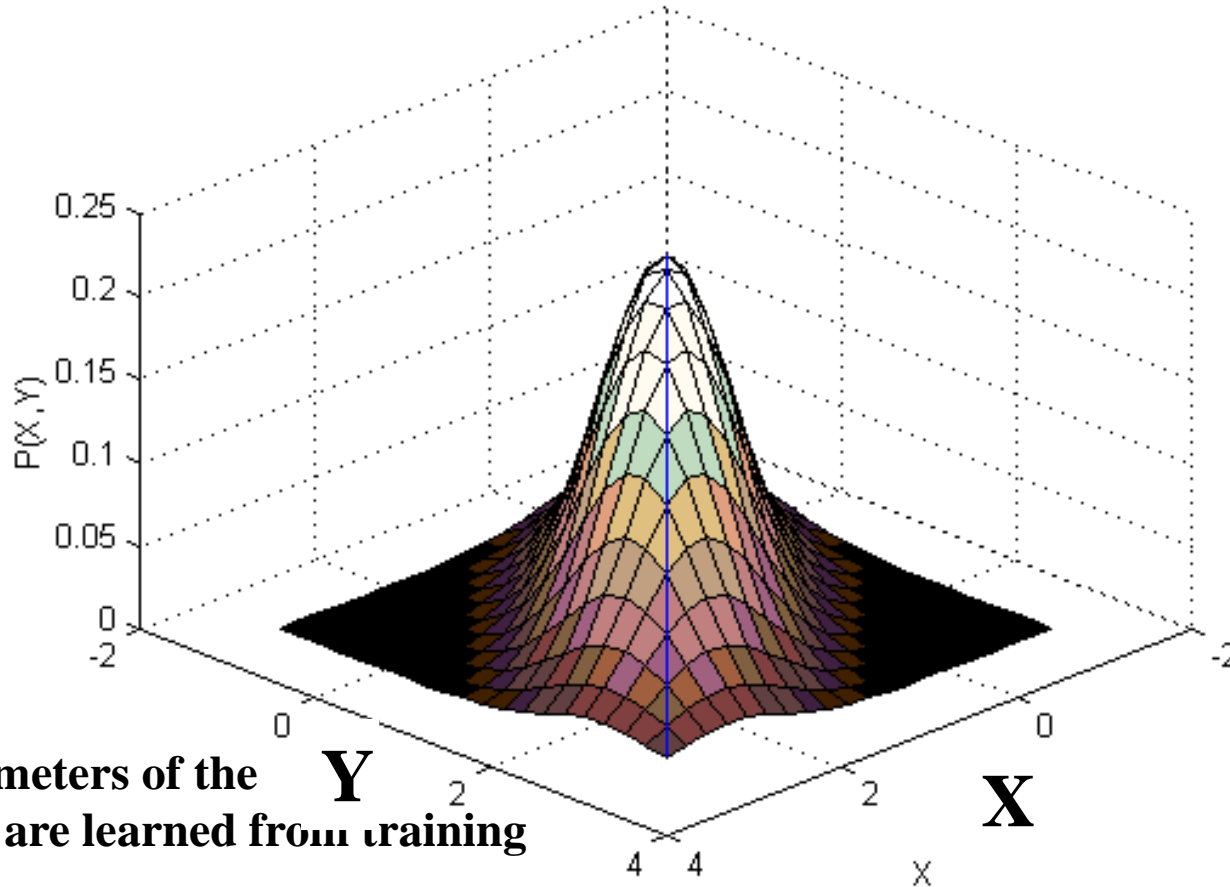
- ML (*Maximum Likelihood*): Find that value of \mathbf{y} for which the statistical best guess of \mathbf{x} would have been the observed \mathbf{x}

$$\mathbf{y} = \operatorname{argmax}_Y P(\mathbf{x}|\mathbf{Y})$$

- MAP is simpler to visualize

MAP estimation: Gaussian PDF

Assume X
and Y are
jointly
Gaussian



The parameters of the Gaussian are learned from training data

Learning the parameters of the Gaussian

$$\mathbf{z} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix}$$

$$\mu_{\mathbf{z}} = \frac{1}{N} \sum_{i=1}^N \mathbf{z}_i$$

$$C_{\mathbf{z}} = \frac{1}{N} \sum_{i=1}^N (\mathbf{z}_i - \mu_{\mathbf{z}})(\mathbf{z}_i - \mu_{\mathbf{z}})^T$$

$$\mu_{\mathbf{z}} = \begin{bmatrix} \mu_{\mathbf{y}} \\ \mu_{\mathbf{x}} \end{bmatrix}$$

$$C_{\mathbf{z}} = \begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix}$$

Learning the parameters of the Gaussian

$$\mu_{\mathbf{z}} = \frac{1}{N} \sum_{i=1}^N \mathbf{z}_i \quad \mathbf{z} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} \quad C_{\mathbf{z}} = \frac{1}{N} \sum_{i=1}^N (\mathbf{z}_i - \mu_{\mathbf{z}})(\mathbf{z}_i - \mu_{\mathbf{z}})^T$$

$$\mu_{\mathbf{z}} = \begin{bmatrix} \mu_{\mathbf{y}} \\ \mu_{\mathbf{x}} \end{bmatrix}$$

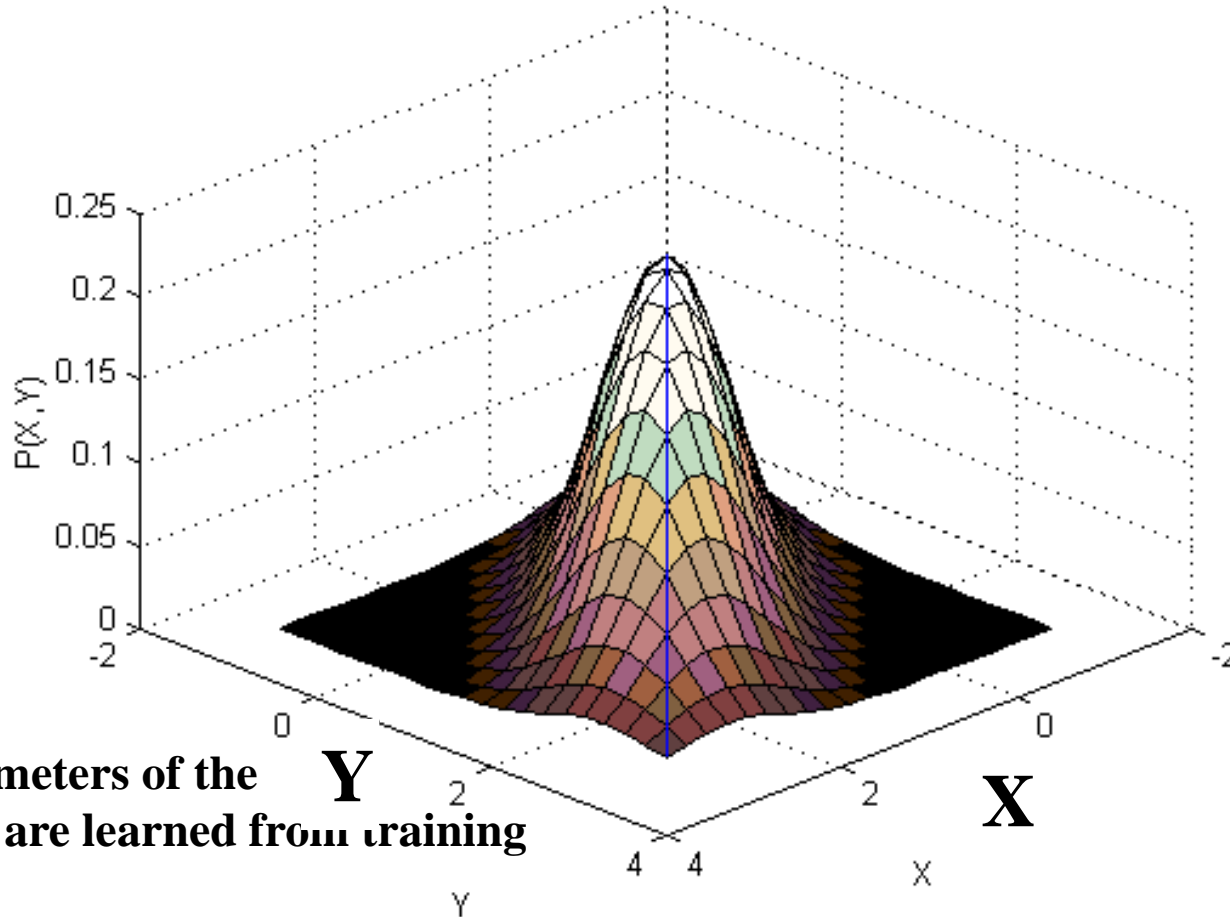
$$C_{\mathbf{z}} = \begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix}$$

$$\mu_{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$$

$$C_{XY} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \mu_{\mathbf{x}})(\mathbf{y}_i - \mu_{\mathbf{y}})^T$$

MAP estimation: Gaussian PDF

Assume X
and Y are
jointly
Gaussian

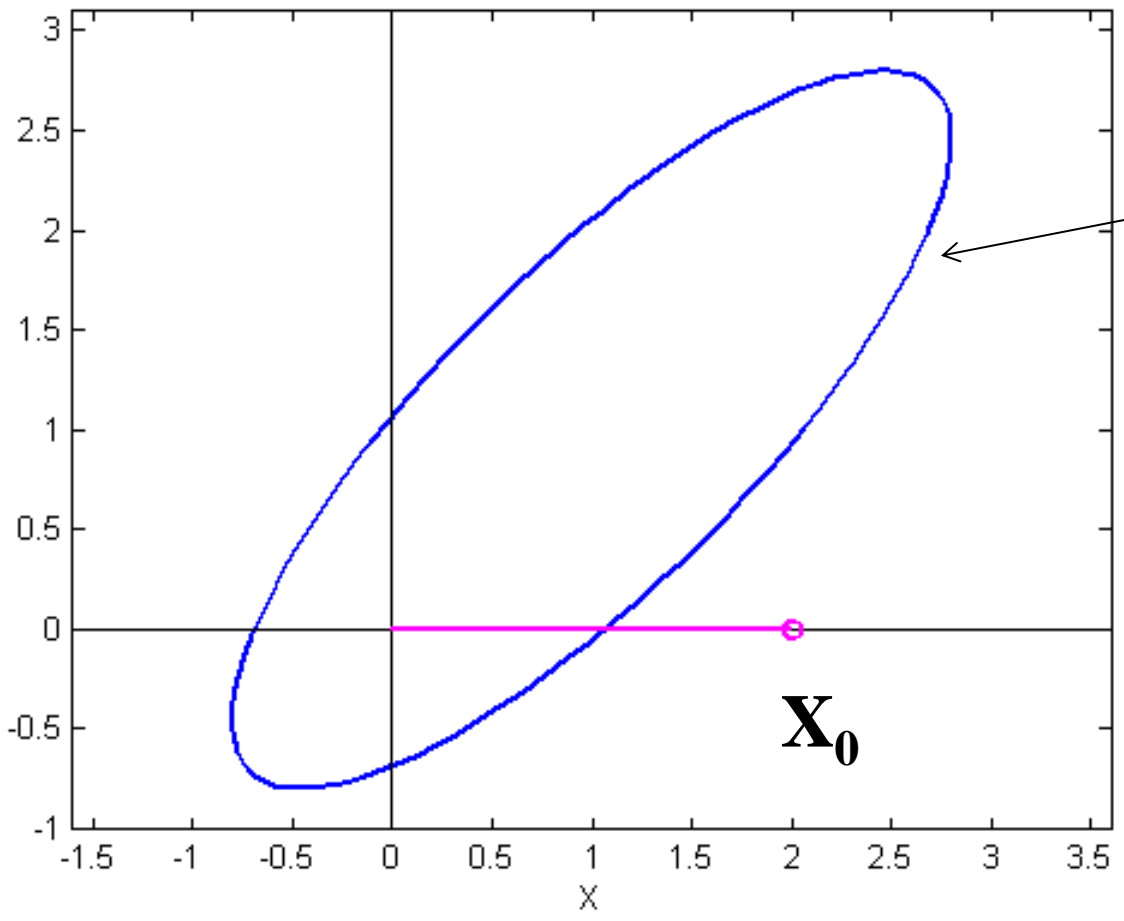


The parameters of the Gaussian are learned from training data

MAP Estimator for Gaussian RV

Assume X and Y are jointly Gaussian

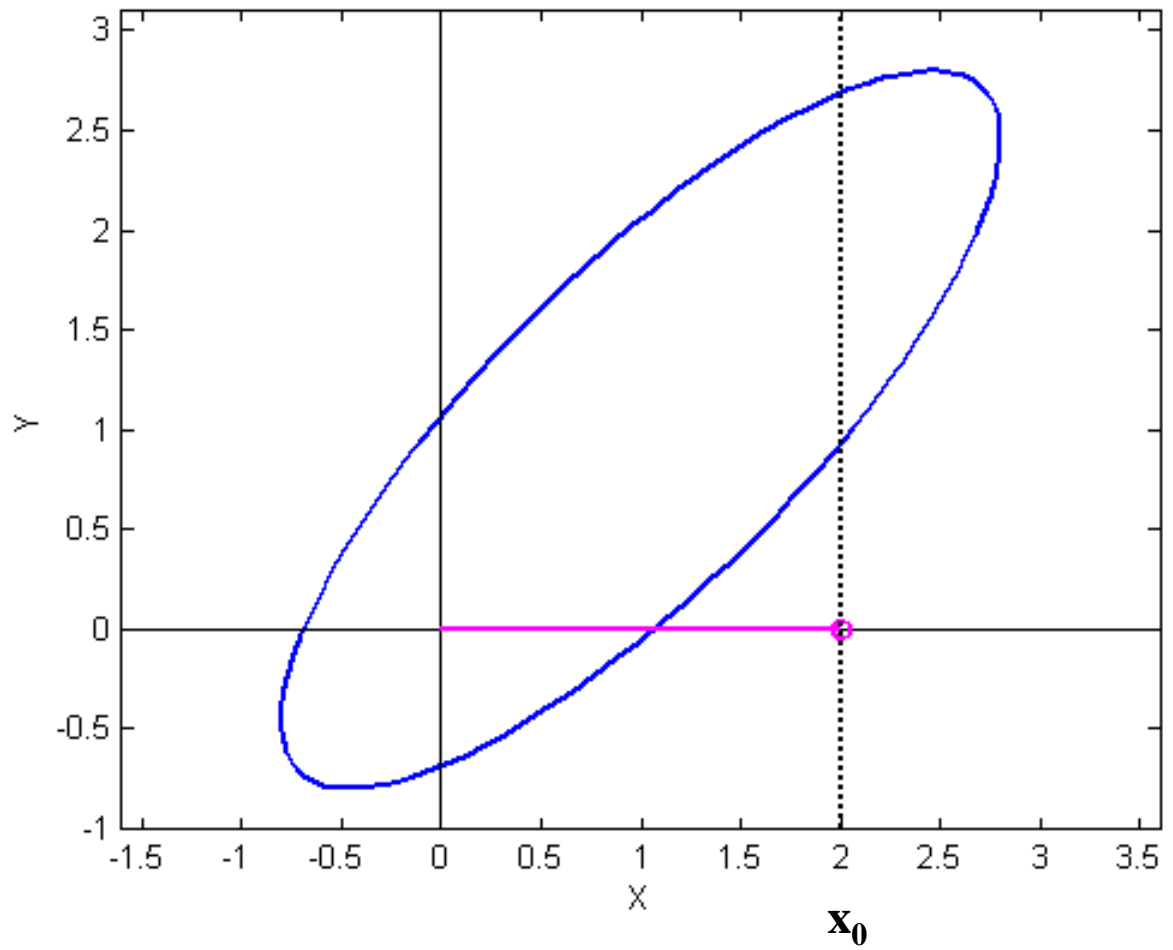
The parameters of the Gaussian \succ are learned from training data



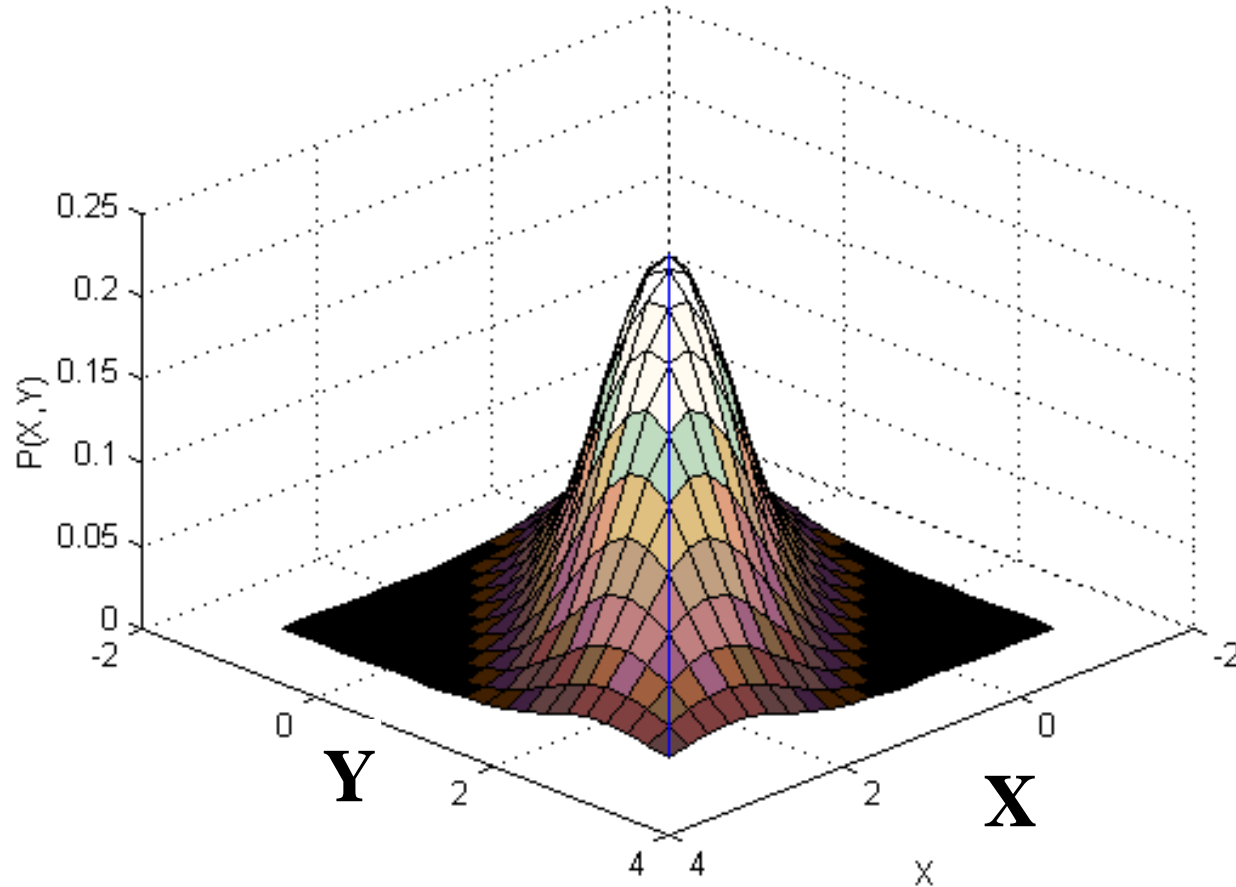
Level set of Gaussian

Now we are given an X , but no Y
What is Y ?

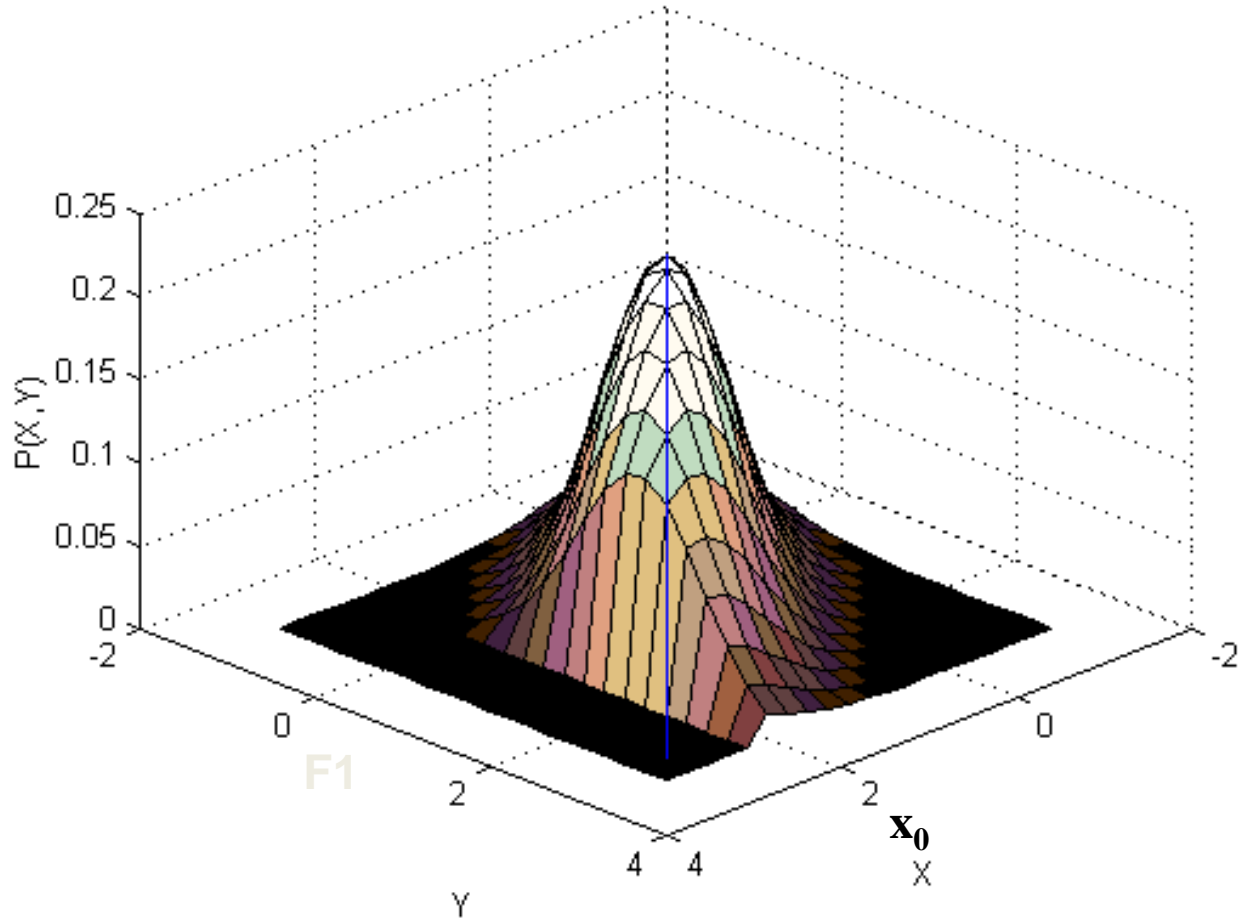
MAP estimator for Gaussian RV



MAP estimation: Gaussian PDF

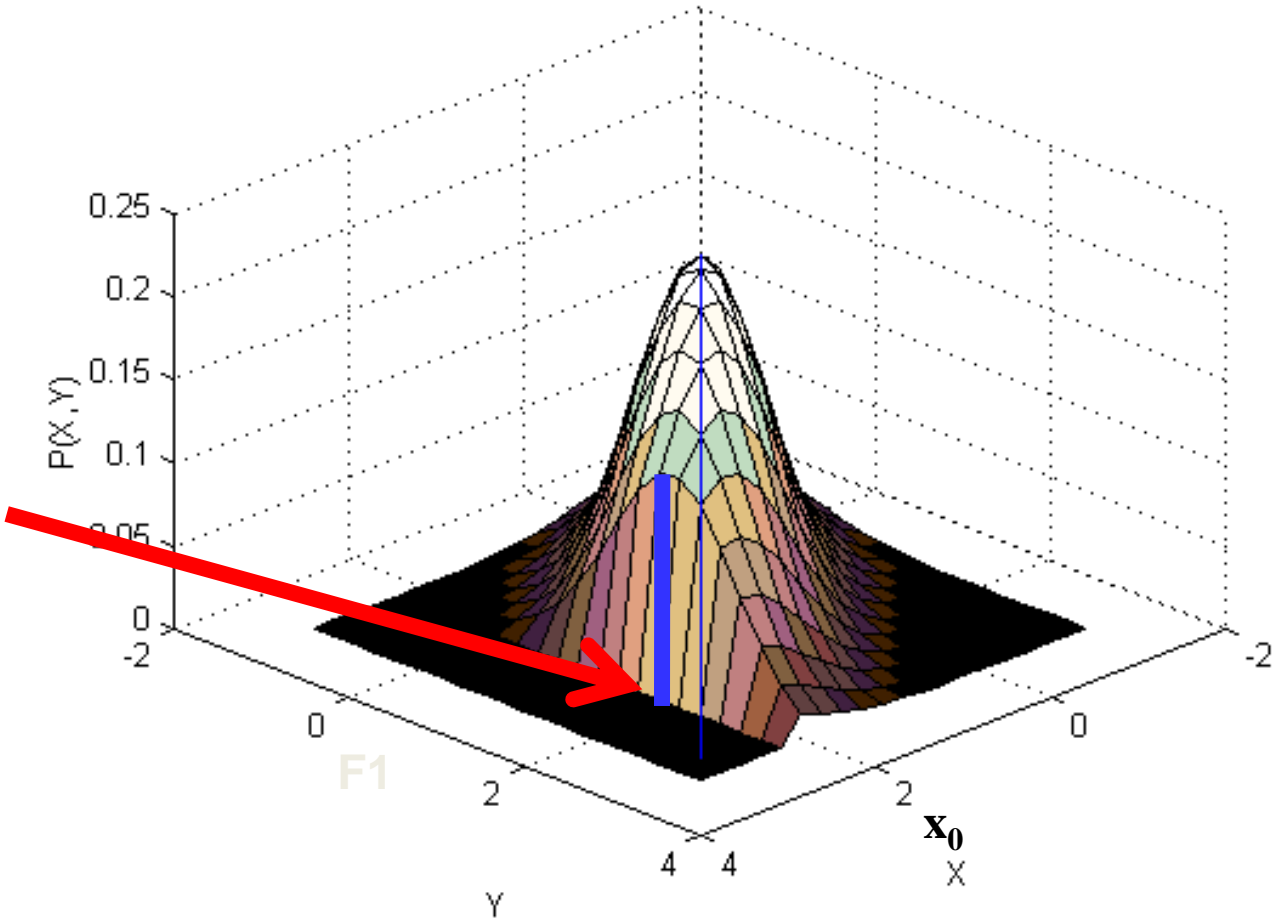


MAP estimation: The Gaussian at a particular value of X



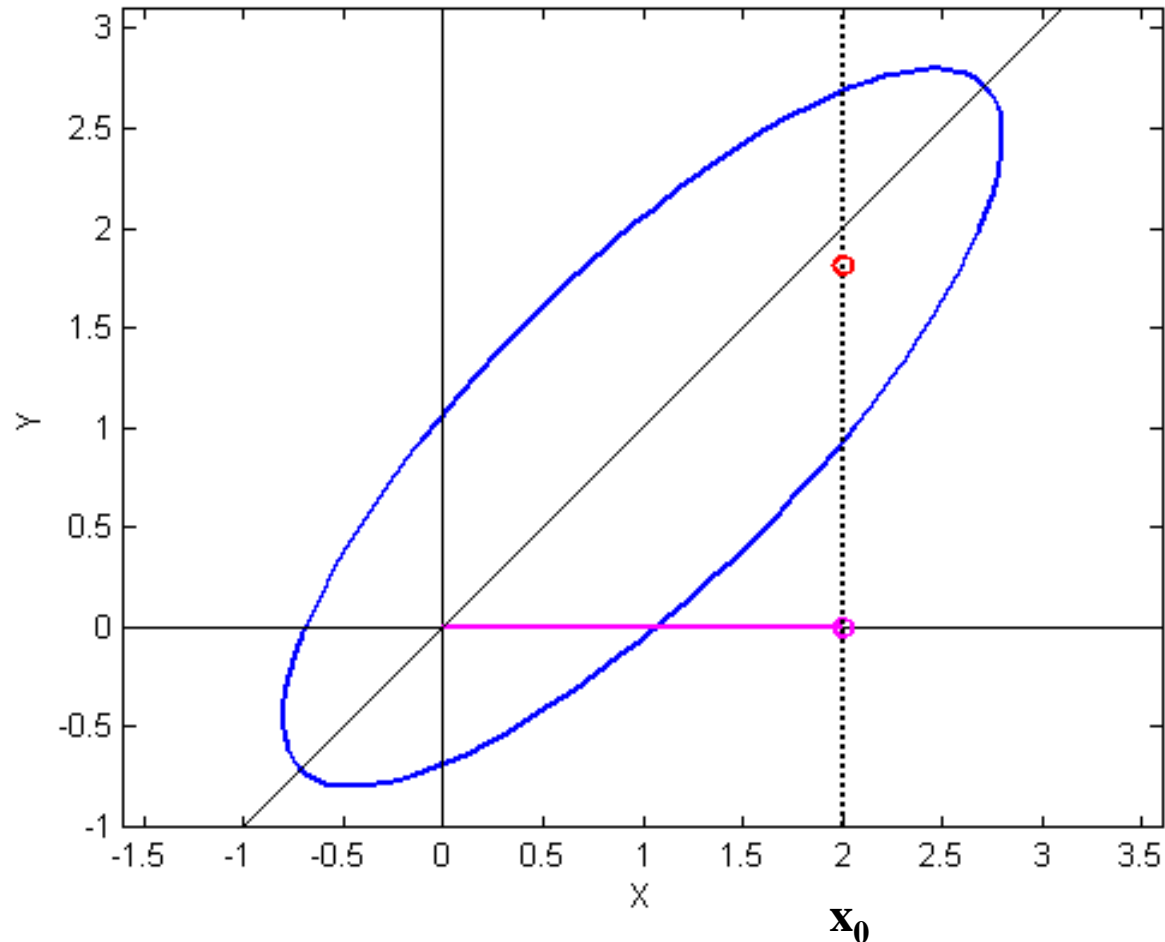
MAP estimation: The Gaussian at a particular value of X

Most likely value

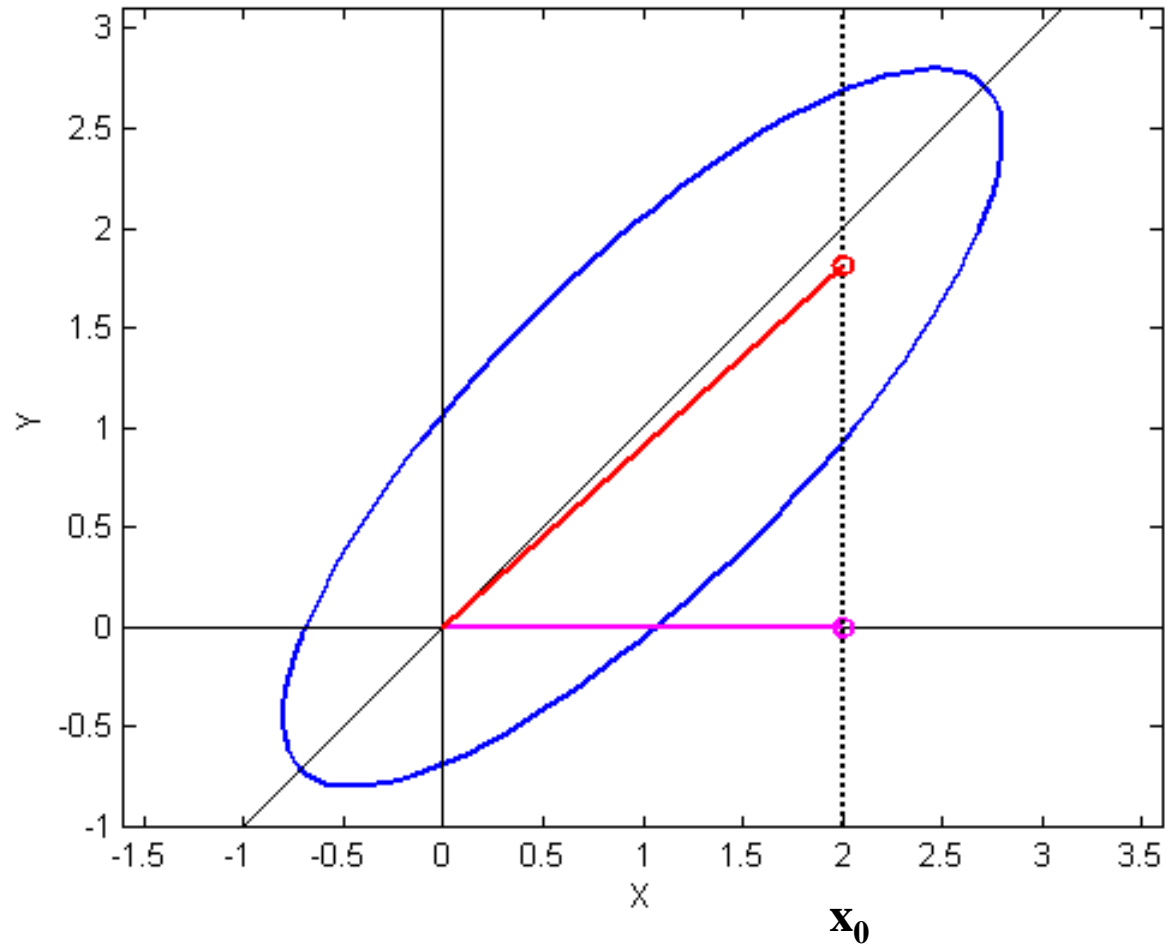


MAP Estimation of a Gaussian RV

$$Y = \operatorname{argmax}_y P(y|X) ???$$

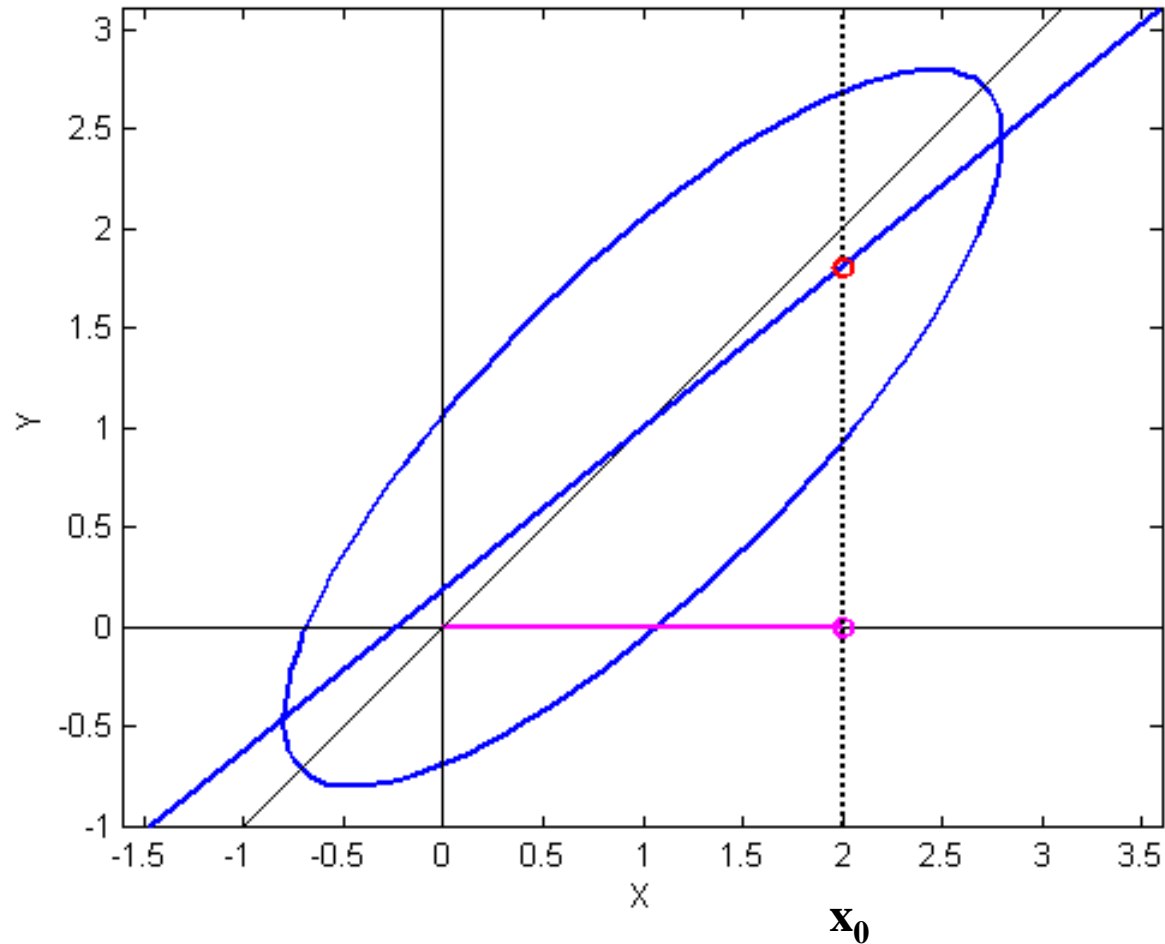


MAP Estimation of a Gaussian RV



MAP Estimation of a Gaussian RV

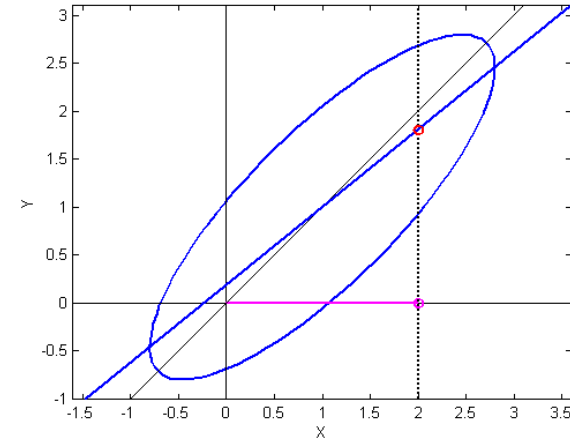
$$Y = \operatorname{argmax}_y P(y|X)$$



So what is this value?

- Clearly a line
- Equation of Line:

$$\hat{y} = \mu_Y + C_{YX} C_{XX}^{-1} (x - \mu_x)$$



- Scalar version given; vector version is identical

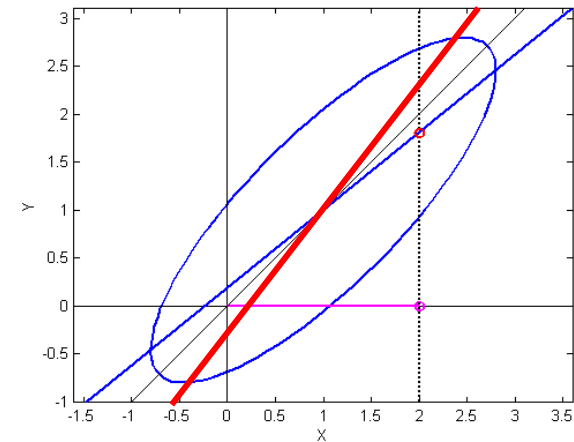
$$\hat{\mathbf{y}} = \mu_Y + C_{YX} C_{XX}^{-1} (\mathbf{x} - \mu_x)$$

- Derivation? Later in the program a bit
 - Note the similarity to regression

This is a *multiple* regression

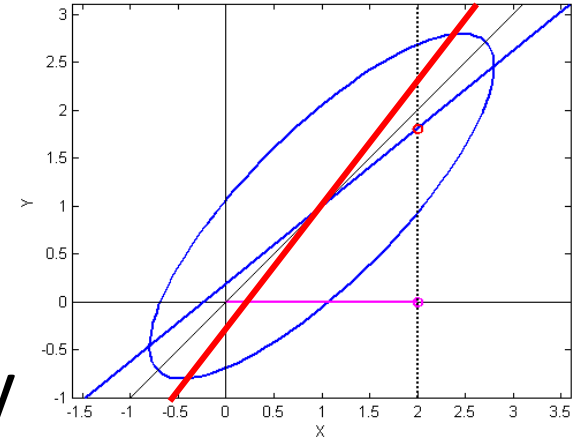
$$\hat{\mathbf{y}} = \mu_Y + C_{YX} C_{XX}^{-1} (\mathbf{x} - \mu_x)$$

- This is the MAP estimate of \mathbf{y}
 - $\mathbf{y} = \operatorname{argmax}_Y P(Y/\mathbf{x})$
- What about the ML estimate of \mathbf{y}
 - $\operatorname{argmax}_Y P(\mathbf{x}|Y)$
- Note: Neither of these may be the *regression* line!
 - MAP estimation of \mathbf{y} is the regression on Y for Gaussian RVs
 - *But this is not the MAP estimation of the regression parameter*



A Closer Look

$$\hat{\mathbf{y}} = \mu_Y + C_{YX} C_{XX}^{-1} (\mathbf{x} - \mu_x)$$



- Assuming 0 mean for simplicity

$$\hat{\mathbf{y}} = C_{YX} C_{XX}^{-1} \mathbf{x}$$

$$\hat{\mathbf{y}} = \mathbf{YX}^T C_{XX}^{-1} \mathbf{x}$$

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

$$\hat{\mathbf{y}} = \sum_{i \in \text{training}} \mathbf{y}_i \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}$$

whitened

- A familiar equation
- Linear regression actually gives you an MAP estimate under Gaussian assumption*

Its also a *minimum-mean-squared error estimate*

- General principle of MMSE estimation:
 - \mathbf{y} is unknown, \mathbf{x} is known
 - Must estimate it such that the *expected* squared error is minimized

$$Err = E[\|\mathbf{y} - \hat{\mathbf{y}}\|^2 | \mathbf{x}]$$

- Minimize above term

Its also a *minimum-mean-squared error estimate*

- Minimize error:

$$Err = E[\|\mathbf{y} - \hat{\mathbf{y}}\|^2 | \mathbf{x}] = E[(\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) | \mathbf{x}]$$

$$Err = E[\mathbf{y}^T \mathbf{y} + \hat{\mathbf{y}}^T \hat{\mathbf{y}} - 2\hat{\mathbf{y}}^T \mathbf{y} | \mathbf{x}] = E[\mathbf{y}^T \mathbf{y} | \mathbf{x}] + \hat{\mathbf{y}}^T \hat{\mathbf{y}} - 2\hat{\mathbf{y}}^T E[\mathbf{y} | \mathbf{x}]$$

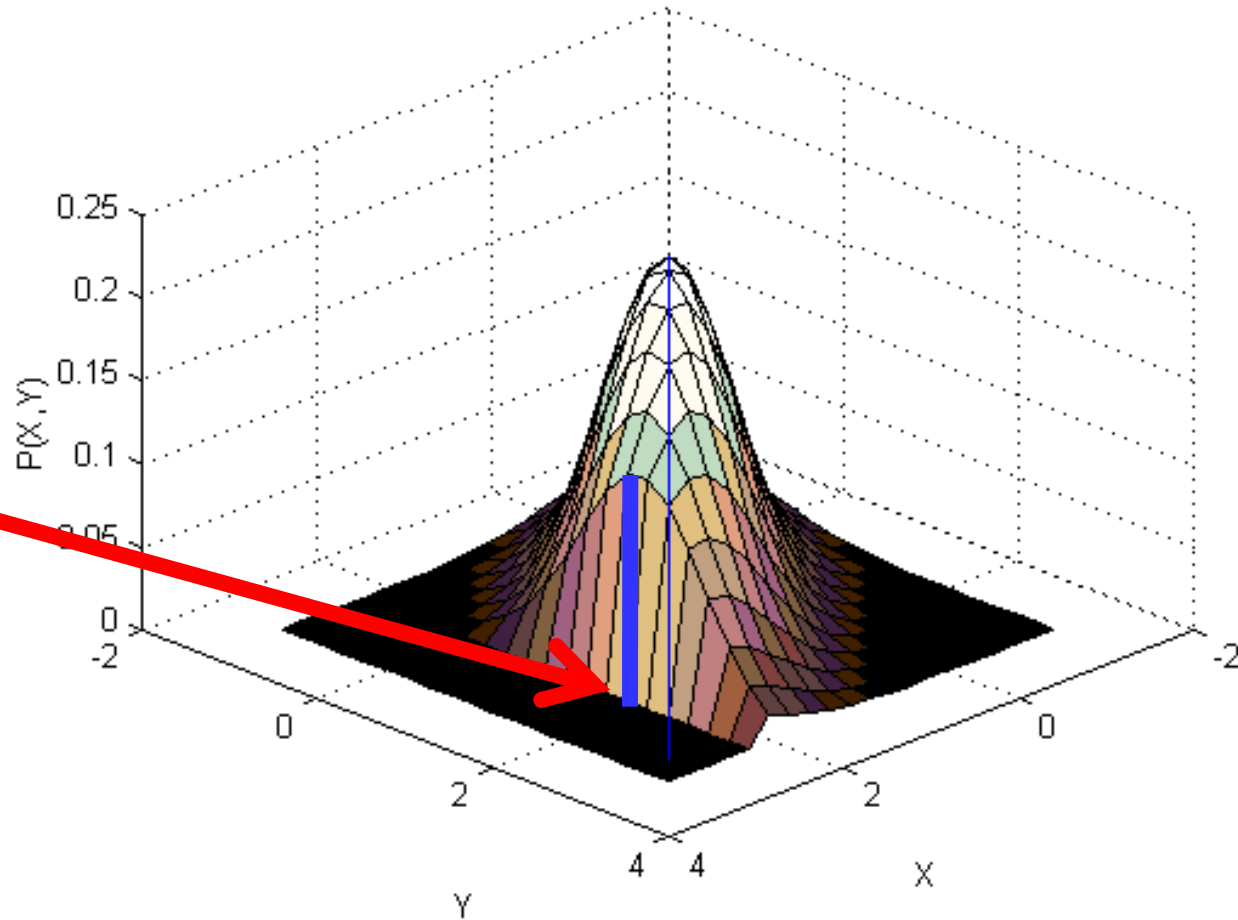
- Differentiating and equating to 0:

$$d.Err = 2\hat{\mathbf{y}}^T d\hat{\mathbf{y}} - 2E[\mathbf{y} | \mathbf{x}]^T d\hat{\mathbf{y}} = 0$$

$$\hat{\mathbf{y}} = E[\mathbf{y} | \mathbf{x}]$$

The MMSE estimate is the mean of the distribution

For the Gaussian: MAP = MMSE



**Most likely
value**

is also

**The MEAN
value**

- Would be true of any symmetric distribution

MMSE estimates for mixture distributions

$$P(\mathbf{y} | \mathbf{x}) = \sum_k P(k)P(\mathbf{y} | k, \mathbf{x})$$

- Let $P(\mathbf{y}|\mathbf{x})$ be a mixture density
- The MMSE estimate of \mathbf{y} is given by

$$E[\mathbf{y} | \mathbf{x}] = \int \mathbf{y} \sum_k P(k)P(\mathbf{y} | k, \mathbf{x}) d\mathbf{y} = \sum_k P(k) \int \mathbf{y} P(\mathbf{y} | k, \mathbf{x}) d\mathbf{y}$$

$$= \sum_k P(k)E[\mathbf{y} | k, \mathbf{x}]$$

- Just a weighted combination of the MMSE estimates from the component distributions

MMSE estimates from a Gaussian mixture

- Let $P(\mathbf{x}, \mathbf{y})$ be a Gaussian Mixture

$$\mathbf{z} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix}$$

$$P(\mathbf{x}, \mathbf{y}) = P(\mathbf{z}) = \sum_k P(k) N(\mathbf{z}; \mu_k, \Sigma_k)$$

- $P(\mathbf{y} | \mathbf{x})$ is also a Gaussian mixture

$$P(\mathbf{y} | \mathbf{x}) = \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{x})} = \frac{\sum_k P(k, \mathbf{x}, \mathbf{y})}{P(\mathbf{x})} = \frac{\sum_k P(\mathbf{x}) P(k | \mathbf{x}) P(\mathbf{y} | \mathbf{x}, k)}{P(\mathbf{x})}$$

$$P(\mathbf{y} | \mathbf{x}) = \sum_k P(k | \mathbf{x}) P(\mathbf{y} | \mathbf{x}, k)$$

MMSE estimates from a Gaussian mixture

- Let $P(\mathbf{y}|\mathbf{x})$ is a Gaussian Mixture

$$P(\mathbf{y} | \mathbf{x}) = \sum_k P(k | \mathbf{x}) P(\mathbf{y} | \mathbf{x}, k)$$

$$P(\mathbf{y}, \mathbf{x}, k) = N([\mathbf{y}; \mathbf{x}]; [\mu_{k,y}; \mu_{k,x}], \begin{bmatrix} C_{k,yy} & C_{k,yx} \\ C_{k,xy} & C_{k,xx} \end{bmatrix})$$

$$P(\mathbf{y} | \mathbf{x}, k) = N(\mathbf{y}; \mu_{k,y} + C_{k,yx} C_{k,xx}^{-1} (\mathbf{x} - \mu_{k,x}), \Theta)$$

$$P(\mathbf{y} | \mathbf{x}) = \sum_k P(k | \mathbf{x}) N(\mathbf{y}; \mu_{k,y} + C_{k,yx} C_{k,xx}^{-1} (\mathbf{x} - \mu_{k,x}), \Theta)$$

MMSE estimates from a Gaussian mixture

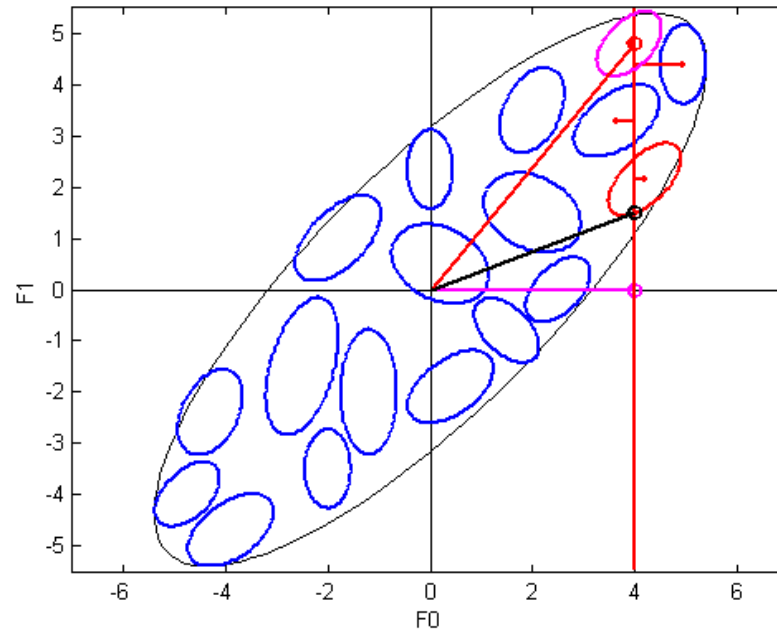
$$P(\mathbf{y} | \mathbf{x}) = \sum_k P(k | \mathbf{x}) N(\mathbf{y}; \mu_{k,y} + C_{k,yx} C_{k,xx}^{-1} (\mathbf{x} - \mu_{k,x}), \Theta)$$

- $P(\mathbf{y} | \mathbf{x})$ is a mixture Gaussian density
- $E[\mathbf{y} | \mathbf{x}]$ is also a mixture

$$E[\mathbf{y} | \mathbf{x}] = \sum_k P(k | \mathbf{x}) E[\mathbf{y} | k, \mathbf{x}]$$

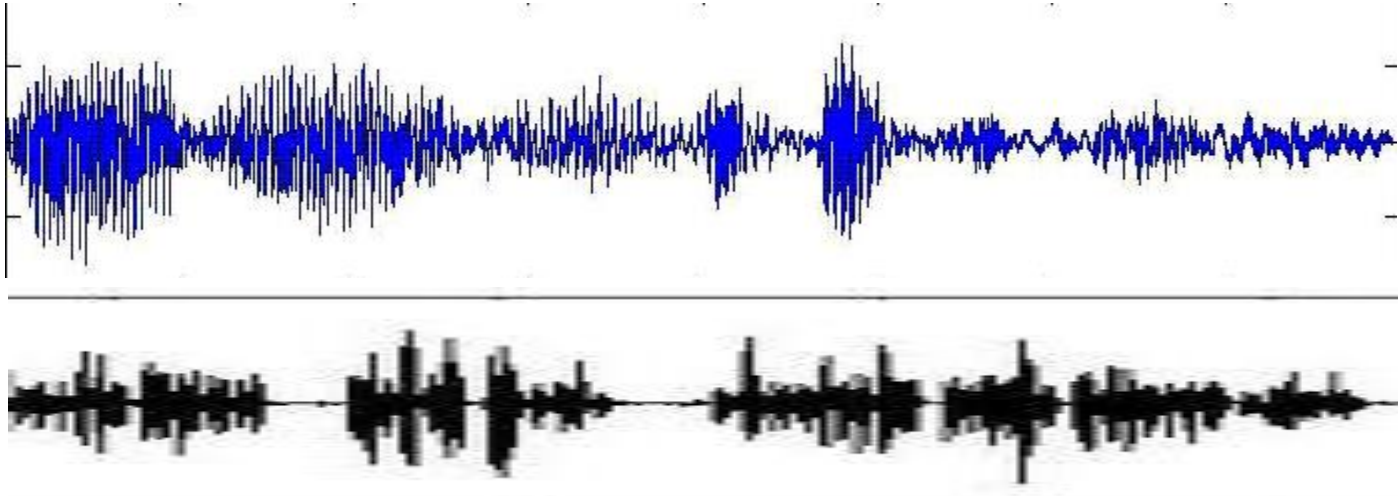
$$E[\mathbf{y} | \mathbf{x}] = \sum_k P(k | \mathbf{x}) \left(\mu_{k,y} + C_{k,yx} C_{k,xx}^{-1} (\mathbf{x} - \mu_{k,x}) \right)$$

MMSE estimates from a Gaussian mixture



- A mixture of estimates from individual Gaussians

















Voice Morphing



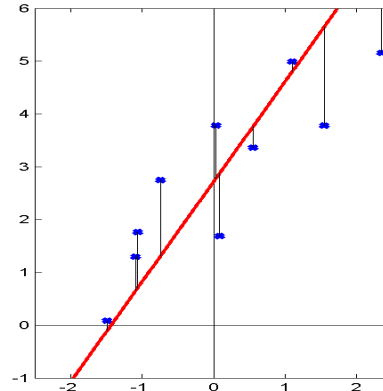
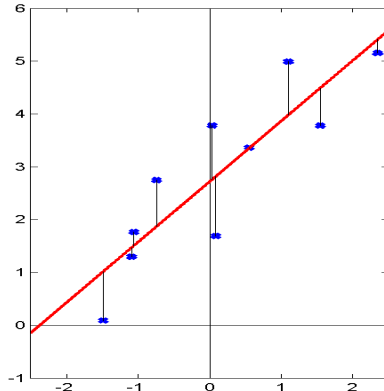
- *Align training recordings from both speakers*
 - Cepstral vector sequence
- Learn a GMM on joint vectors
- Given speech from one speaker, find MMSE estimate of the other
- *Synthesize from cepstra*

MMSE with GMM: Voice Transformation

- Festvox GMM transformation suite (Toda)

	awb	bdl	jmk	slt
awb				
bdl				
jmk				
slt				

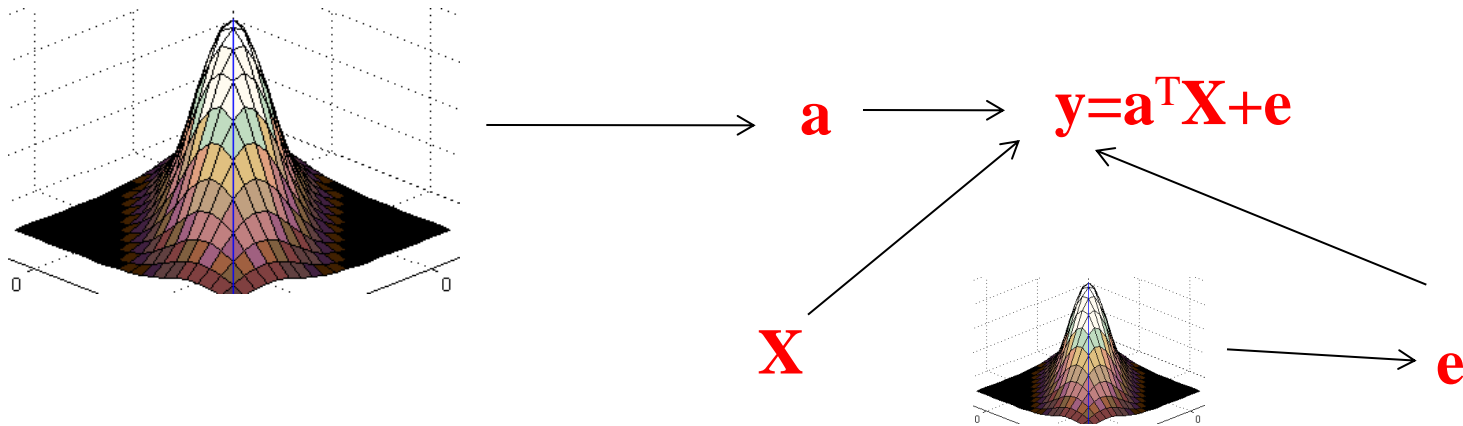
A problem with regressions



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

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

MAP estimation of weights



- Assume weights drawn from a Gaussian
 - $P(\mathbf{a}) = \mathcal{N}(0, \sigma^2 \mathbf{I})$
- Max. Likelihood estimate

$$\hat{\mathbf{a}} = \arg \max_{\mathbf{a}} \log P(\mathbf{y} | \mathbf{X}; \mathbf{a})$$

- Maximum *a posteriori* estimate

$$\hat{\mathbf{a}} = \arg \max_{\mathbf{a}} \log P(\mathbf{a} | \mathbf{y}, \mathbf{X}) = \arg \max_{\mathbf{a}} \log P(\mathbf{y} | \mathbf{X}, \mathbf{a})P(\mathbf{a})$$

MAP estimation of weights

$$\hat{\mathbf{a}} = \arg \max_{\mathbf{A}} \log P(\mathbf{a} | \mathbf{y}, \mathbf{X}) = \arg \max_{\mathbf{A}} \log P(\mathbf{y} | \mathbf{X}, \mathbf{a})P(\mathbf{a})$$

- $P(\mathbf{a}) = N(0, \sigma^2\mathbf{I})$
- $\log P(\mathbf{a}) = C - \log \sigma - 0.5\sigma^{-2} \|\mathbf{a}\|_2^2$

$$\log P(\mathbf{y} | \mathbf{X}, \mathbf{a}) = C - \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{a}^T \mathbf{X})^T (\mathbf{y} - \mathbf{a}^T \mathbf{X})$$

$$\hat{\mathbf{a}} = \arg \max_{\mathbf{A}} C' - \log \sigma - \frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{a}^T \mathbf{X})^T (\mathbf{y} - \mathbf{a}^T \mathbf{X}) - 0.5\sigma^2 \mathbf{a}^T \mathbf{a}$$

- Similar to ML estimate with an additional term

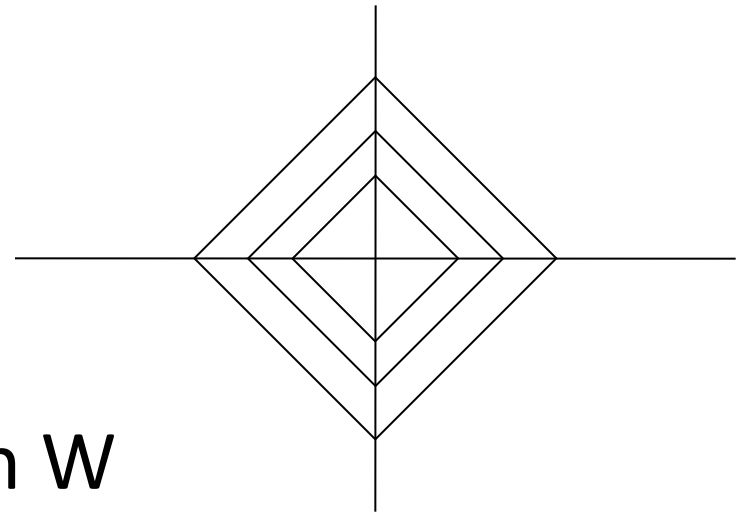
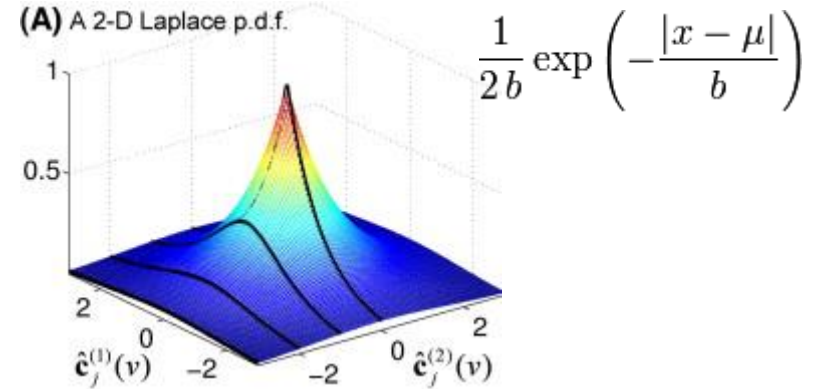
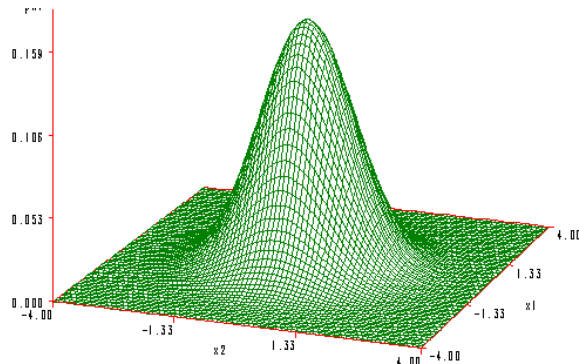
MAP estimate of weights

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

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

- 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 estimate priors



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

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

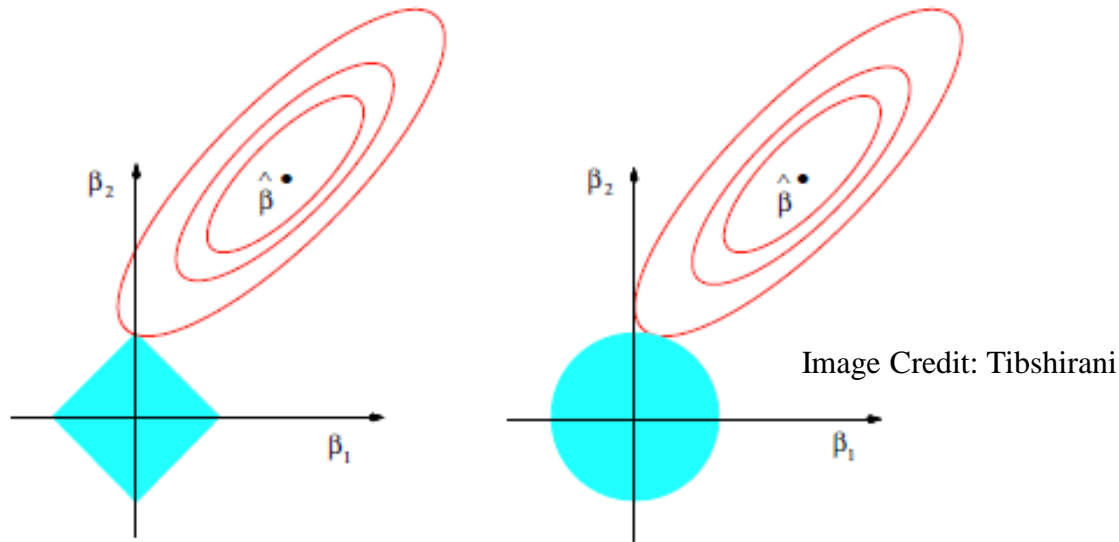
$$\hat{\mathbf{a}} = \arg \max_{\mathbf{A}} C' - (\mathbf{y} - \mathbf{a}^T \mathbf{X})^T (\mathbf{y} - \mathbf{a}^T \mathbf{X})^T \quad \text{subject to} \quad \|\mathbf{a}\|_1 \leq t$$

- “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



- 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

LASSO and Compressive Sensing

$$\mathbf{Y} = \mathbf{X} \mathbf{a}$$

- 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

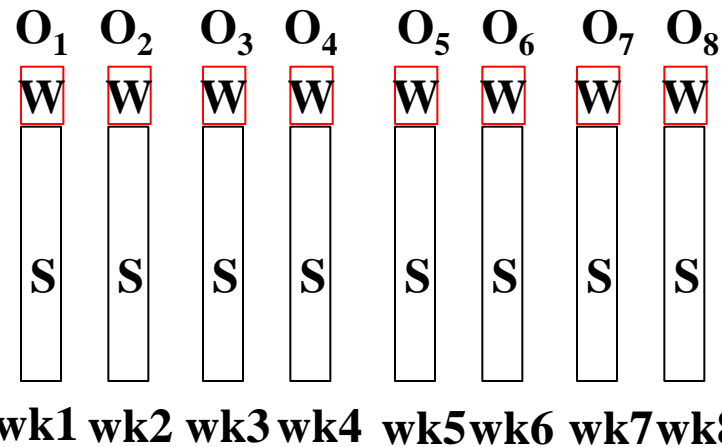
An interesting problem: Predicting War!

- 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