11-755 Machine Learning for Signal Processing

Regression and Prediction

Class 15. 23 Oct 2012

Instructor: Bhiksha Raj

Matrix Identities

$$f(\mathbf{x}) \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_D \end{bmatrix} \qquad df(\mathbf{x}) = \begin{bmatrix} \frac{df}{dx_1} dx_1 \\ \frac{df}{dx_2} dx_2 \\ \ldots \\ \frac{df}{dx_D} dx_D \end{bmatrix}$$

- The derivative of a scalar function w.r.t. a vector is a vector
- The derivative w.r.t. a matrix is a matrix

Matrix Identities

$$f(\mathbf{x}) \qquad \mathbf{x} = \begin{bmatrix} x_{11} & x_{12} & .. & x_{1D} \\ x_{21} & x_{22} & .. & x_{2D} \\ .. & .. & .. & .. \\ x_{D1} & x_{D2} & .. & x_{DD} \end{bmatrix} \qquad df(\mathbf{x}) = \begin{bmatrix} \frac{df}{dx_{11}} dx_{11} & \frac{df}{dx_{12}} dx_{12} & .. & \frac{df}{dx_{1D}} dx_{1D} \\ \frac{df}{dx_{21}} dx_{21} & \frac{df}{dx_{22}} dx_{22} & .. & \frac{df}{dx_{2D}} dx_{2D} \\ .. & .. & .. & .. \\ \frac{df}{dx_{D1}} dx_{D1} & \frac{df}{dx_{D2}} dx_{D2} & .. & \frac{df}{dx_{DD}} dx_{DD} \end{bmatrix}$$

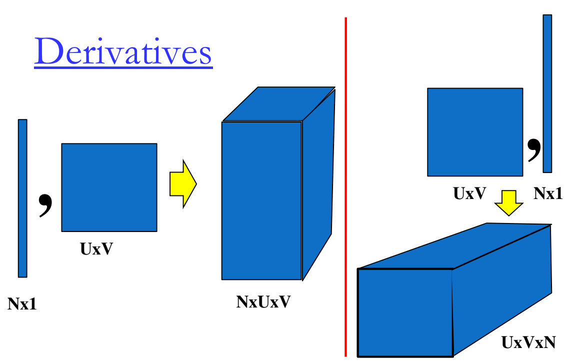
- The derivative of a scalar function w.r.t. a vector is a vector
- The derivative w.r.t. a matrix is a matrix

Matrix Identities

$$\mathbf{F}(\mathbf{x}) \qquad \mathbf{F} = \begin{bmatrix} F_1 \\ F_2 \\ \ldots \\ F_N \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_D \end{bmatrix} \qquad \begin{bmatrix} dF_1 \\ dF_2 \\ \ldots \\ dF_N \end{bmatrix} = \begin{bmatrix} \frac{dF_1}{dx_1} dx_1 & \frac{dF_1}{dx_2} dx_2 & .. & \frac{dF_1}{dx_D} dx_D \\ \frac{dF_2}{dx_1} dx_1 & \frac{dF_2}{dx_2} dx_2 & .. & \frac{dF_2}{dx_D} dx_D \\ .. & .. & .. & .. \\ \frac{dF_N}{dx_D} dx_1 & \frac{dF_N}{dx_2} dx_2 & .. & \frac{dF_N}{dx_D} dx_D \end{bmatrix}$$

- The derivative of a vector function w.r.t. a vector is a matrix
  - Note transposition of order

Derivatives



- In general: Differentiating an MxN function by a UxV argument results in an MxNxUxV tensor derivative

Matrix derivative identities

$d(\mathbf{X}\mathbf{a}) = \mathbf{X}d\mathbf{a} \qquad d(\mathbf{a}^T\mathbf{X}) = \mathbf{X}^T d\mathbf{a}$ — **X** is a matrix, **a** is a vector. Solution may also be $\mathbf{X}^T$

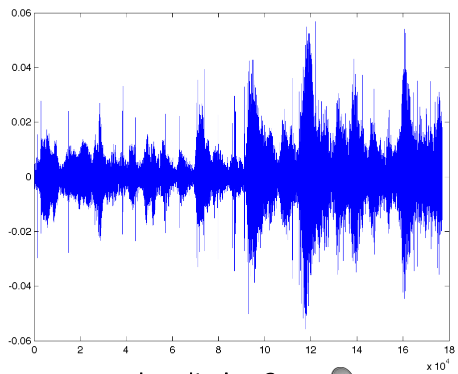
$d(\mathbf{AX}) = (d\mathbf{A})\mathbf{X} \; ; \; d(\mathbf{XA}) = \mathbf{X}(d\mathbf{A})$ — **A** is a matrix

$$d\left(\mathbf{a}^T\mathbf{X}\mathbf{a}\right) = \mathbf{a}^T\left(\mathbf{X} + \mathbf{X}^T\right)d\mathbf{a}$$

$$d\left(trace\left(\mathbf{A}^T\mathbf{X}\mathbf{A}\right)\right) = d\left(trace\left(\mathbf{X}\mathbf{A}\mathbf{A}^T\right)\right) = d\left(trace\left(\mathbf{A}\mathbf{A}^T\mathbf{X}\right)\right) = (\mathbf{X}^T + \mathbf{X})d\mathbf{A}$$

- Some basic linear and quadratic identities

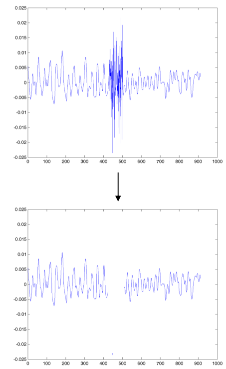
A Common Problem



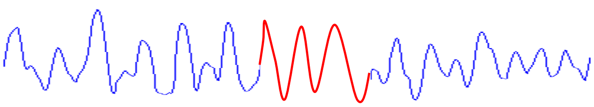
- Can you spot the glitches?

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?

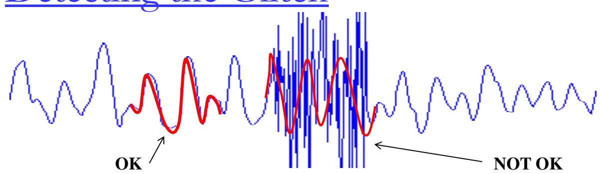


Interpolation..



- “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

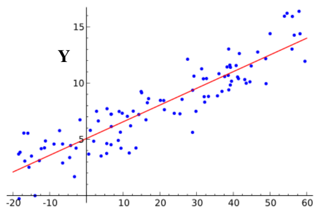
What is a regression

- 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
    - Y is real
    - 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(.) is a linear or affine function
  - f(.) is a non-linear function
  - f(.) is a *time-series* model

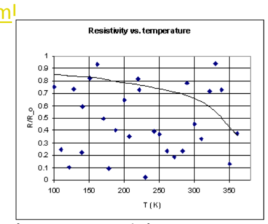
A *linear* regression



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

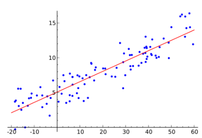
An imaginary regression..

- 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

- $\mathbf{y} = \mathbf{Ax} + \mathbf{b} + \mathbf{e}$
  - e = prediction error



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

Linear Regression to a scalar

$$y_1 = \boldsymbol{a}^T\mathbf{x}_1 + b + e_1$$
$$y_2 = \boldsymbol{a}^T\mathbf{x}_2 + b + e_2$$
$$y_3 = \boldsymbol{a}^T\mathbf{x}_3 + b + e_3$$

- Define:

$$\mathbf{y} = [y_1\ y_2\ y_3 \ldots] \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \ldots \\ 1 & 1 & 1 & \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix}$$
$$\mathbf{e} = [e_1\ e_2\ e_3 \ldots]$$

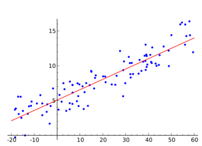
- Rewrite

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

Learning the parameters

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

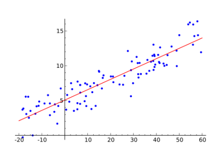
$$\hat{\mathbf{y}} = \mathbf{A}^T\mathbf{X} \quad \text{Assuming no error}$$



- Given training data: several $\mathbf{x},\mathbf{y}$
- Can define a "divergence": $D(\mathbf{y},\hat{\mathbf{y}})$
  - Measures how much yhat differs from 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

$$y_1 = \boldsymbol{a}^T\mathbf{x}_1 + b + e_1$$
$$y_2 = \boldsymbol{a}^T\mathbf{x}_2 + b + e_2$$
$$y_3 = \boldsymbol{a}^T\mathbf{x}_3 + b + e_3$$



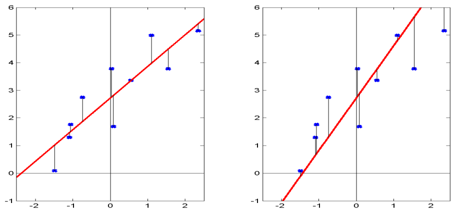
$$\mathbf{y} = \mathbf{A}^T\mathbf{X} + \mathbf{e}$$

$$\mathbf{D}(\mathbf{y},\hat{\mathbf{y}}) = \mathbf{E} = e_1^2 + e_2^2 + e_3^2 + \ldots$$
$$= (y_1 - \mathbf{a}^T b)^2 + (y_1 - \mathbf{a}^T b)^2 + (y_1 - \mathbf{a}^T b)^2 + \ldots$$

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

- Define the divergence as the sum of the squared error in predicting y

Prediction error as divergence



- $y = \mathbf{a}^T\mathbf{x} + e$
  - e = prediction error
  - Find the "slope" **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

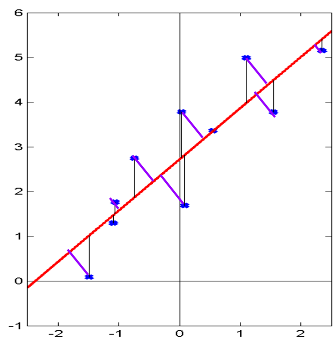
$$\mathbf{E} = \|\mathbf{y} - \mathbf{X}^T\mathbf{A}\|^2 = (\mathbf{y} - \mathbf{A}^T\mathbf{X})(\mathbf{y} - \mathbf{A}^T\mathbf{X})^T$$
$$= \mathbf{y}\mathbf{y}^T + \mathbf{A}^T\mathbf{X}\mathbf{X}^T\mathbf{A} - 2\mathbf{y}\mathbf{X}^T\mathbf{A}$$

- Differentiating w.r.t **A** and equating to 0

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

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

An Aside



- What happens if we minimize the perpendicular instead?

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 **A**

b_j = j^{th} component of **b**

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

$$\mathbf{y}_1 = \mathbf{A}^T\mathbf{x}_1 + \mathbf{b} + \mathbf{e}_1 \Rightarrow \begin{matrix} y_{11} = \mathbf{a}_1^T\mathbf{x}_1 + b_1 + e_{11} \\ y_{12} = \mathbf{a}_2^T\mathbf{x}_2 + b_2 + e_{12} \\ y_{13} = \mathbf{a}_3^T\mathbf{x}_3 + b_3 + e_{13} \end{matrix}$$

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

Multiple Regression

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

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

Dx1 vector of ones

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

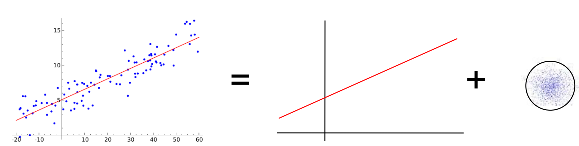
$$DIV = \sum_i \left\|\mathbf{y}_i - \mathbf{A}^T\mathbf{x}_i - \mathbf{b}\right\|^2 = trace\left((\mathbf{Y} - \mathbf{A}^T\mathbf{X})(\mathbf{Y} - \mathbf{A}^T\mathbf{X})^T\right)$$

- Differentiating and equating to 0

$$dDiv = \left(2\mathbf{A}^T\mathbf{X}\mathbf{X}^T - 2\mathbf{Y}\mathbf{X}^T\right)d\mathbf{A} = 0$$

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

A Different Perspective



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

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

- Error **e** is Gaussian

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

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

The *Likelihood* of the data

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

- Probability of observing a specific y, given x, for a particular matrix A

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

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

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

- Assuming IID for convenience (not necessary)

A Maximum Likelihood Estimate

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

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

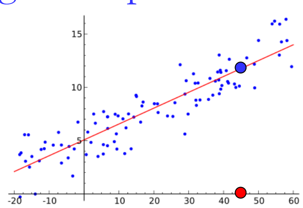
$$\log P(\mathbf{Y} \mid \mathbf{X}; \mathbf{A}) = C - \sum_i \frac{1}{2\sigma^2}\left\|\mathbf{y}_i - \mathbf{A}^T\mathbf{x}_i\right\|^2$$

$$= C - \frac{1}{2\sigma^2} trace\left((\mathbf{Y} - \mathbf{A}^T\mathbf{X})(\mathbf{Y} - \mathbf{A}^T\mathbf{X})^T\right)$$

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

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

Predicting an output

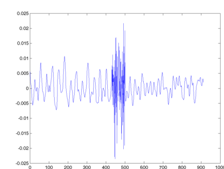


- From a collection of training data, have learned **A**
- Given **x** for a new instance, but not **y**, what is **y**?
- Simple solution:

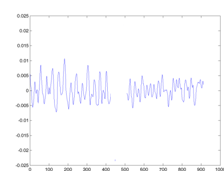
$$\hat{\mathbf{y}} = \mathbf{A}^T\mathbf{X}$$

Applying it to our problem

- Prediction by regression



- Forward regression
- $x_t = a_1x_{t-1} + a_2x_{t-2} \ldots a_kx_{t-k} + e_t$

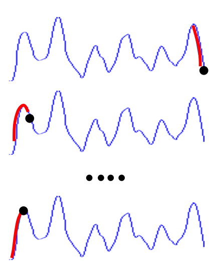


- Backward regression
- $x_t = b_1x_{t+1} + b_2x_{t+2} \ldots b_kx_{t+k} + e_t$

Applying it to our problem

- Forward prediction

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



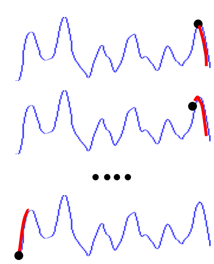
$$\mathbf{x} = \mathbf{a}_t^T\mathbf{X} + \mathbf{e}$$

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

Applying it to our problem

- Backward prediction

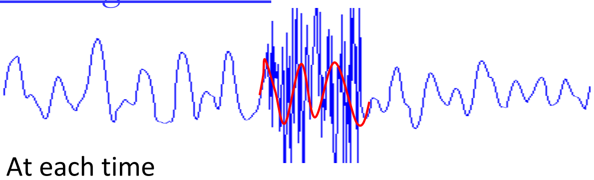
$$\begin{bmatrix} x_{t-K-1} \\ x_{t-K-2} \\ .. \\ x_1 \end{bmatrix} = \mathbf{b}_t^T \begin{bmatrix} x_t & x_{t-1} & .. & x_{K+1} \\ x_{t-1} & x_{t-2} & .. & x_K \\ .. & .. & .. & .. \\ x_{t-K} & x_{t-K-1} & .. & x_2 \end{bmatrix} + \begin{bmatrix} e_{t-K-1} \\ e_{t-K-2} \\ .. \\ e_1 \end{bmatrix}$$



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

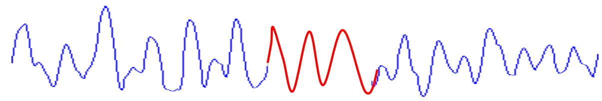
$$\bar{\mathbf{x}}\ pinv(\bar{\mathbf{X}}) = \mathbf{b}_t^T$$

Finding the burst



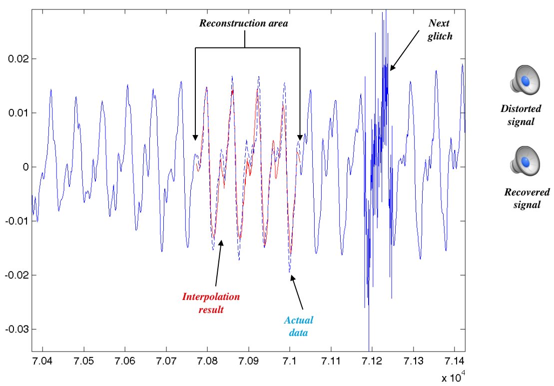
- At each time
  - Learn a "forward" predictor $\mathbf{a}_t$
  - At each time, predict next sample $x_t^{est} = \Sigma_i\, a_{t,k} x_{t-k}$
  - Compute error: $ferr_t = |x_t - x_t^{est}|^2$
  - Learn a "backward" predict and compute backward error
    - $berr_t$
  - Compute average prediction error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

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



Incrementally learning the regression

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

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

- Can we learn A incrementally instead?
  - As data comes in?
- The Widrow Hoff rule

Scalar prediction version

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

error

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

Predicting a value

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

- What are we doing exactly?

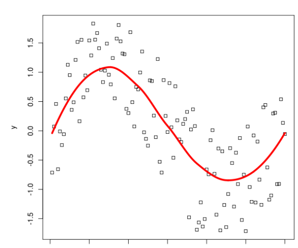
$$\mathbf{C} = \mathbf{X}\mathbf{X}^T$$

- Let $\hat{\mathbf{x}} = \mathbf{C}^{-\frac{1}{2}}\mathbf{x}$
  - Normalizing and rotating space
    - The rotation is irrelevant

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

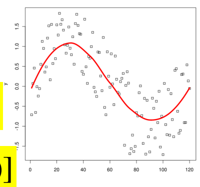
- Weighted combination of inputs

Relationships are not always linear



- How do we model these?
- Multiple solutions

Non-linear regression



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

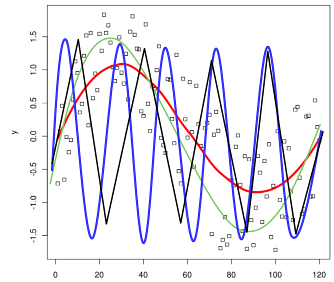
$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [\phi_1(\mathbf{x})\ \ \phi_2(\mathbf{x}) \ldots \phi_N(\mathbf{x})]$$

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

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

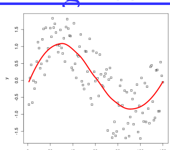
$$\mathbf{A} = \left(\Phi(\mathbf{X})\Phi(\mathbf{X})^T\right)^{-1}\Phi(\mathbf{X})\mathbf{Y}^T$$

What we are doing



- Finding the optimal combination of various function
  - Remind you of something?

Being non-commital: Local Regression



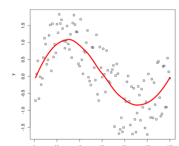
- Regression is usually trained over the *entire* data
  - Must apply everywhere

$$\hat{\mathbf{y}} = \mathbf{Y}\hat{\mathbf{X}}^T\hat{\mathbf{x}} = \sum_i \hat{\mathbf{x}}_i^T\hat{\mathbf{x}}\mathbf{y}_i \qquad \mathbf{y} = \sum_i \mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}_i\mathbf{y}_i + \mathbf{e}$$

- How about doing this locally?
  - For any $\mathbf{x}$

$$\mathbf{y} = \sum_i d(\mathbf{x},\mathbf{x}_i)\mathbf{y}_i + \mathbf{e}$$

Local Regression



- The resulting regression is dependent on x!

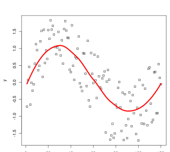
$$\hat{\mathbf{y}} = \sum_i d(\mathbf{x},\mathbf{x}_i)\mathbf{y}_i \qquad \hat{\mathbf{y}} = \sum_i d(\mathbf{x},\mathbf{x}_i)\mathbf{y}_i$$

$$e(\mathbf{x}) = \| \mathbf{y} - \sum_i d(\mathbf{x},\mathbf{x}_i)\mathbf{y}_i \|^2$$

- No closed form solution
  - But can be highly accurate
- But what is $d(\mathbf{x},\mathbf{x}')$??

Kernel Regression

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



- Actually a non-parametric MAP estimator of $\mathbf{y}$
  - Note – an estimator of $\mathbf{y}$, not parameters of regression
  - The "Kernel" is the kernel of a parzen window
- But first.. MAP estimators..

Map Estimators

- MAP (*Maximum A Posteriori*): Find a "best guess" for $\mathbf{y}$ (in a statistical sense), given that we know $\mathbf{x}$

$$\mathbf{y} = argmax_Y\, P(Y|\mathbf{x})$$

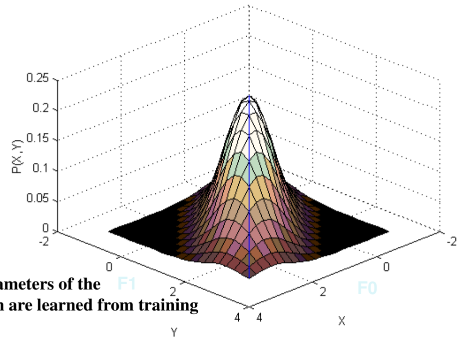
- ML (*Maximum Likelihood*): Find that value of Y for which the statistical best guess of X would have been the observed X

$$\mathbf{y} = argmax_Y\, P(\mathbf{x}|Y)$$

- 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

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

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

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

Learning the parameters of the Gaussian

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

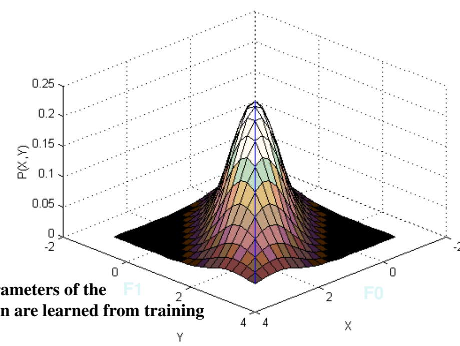
$$\mu_{\mathbf{z}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_i \qquad C_{\mathbf{z}} = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{z}_i - \mu_{\mathbf{z}})(\mathbf{z}_i - \mu_{\mathbf{z}})^T$$

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

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

MAP estimation: Gaussian PDF

Assume X and Y are jointly Gaussian

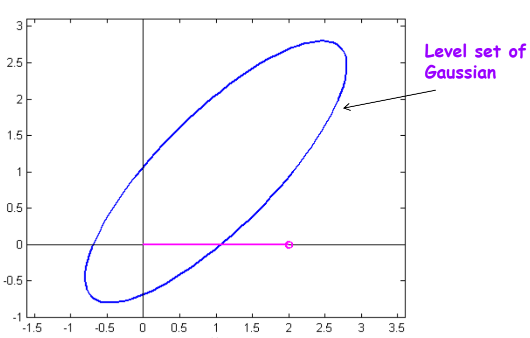


The parameters of the Gaussian are learned from training data

MAP Estimator for Gaussian RV

Assume X and Y are jointly Gaussian

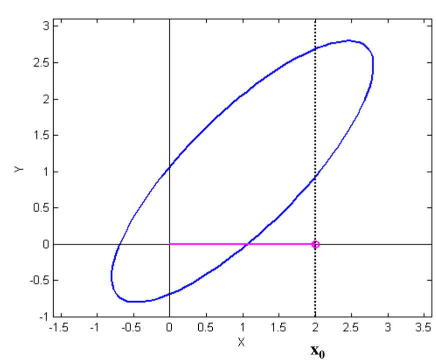
The parameters of the Gaussian are learned from training data



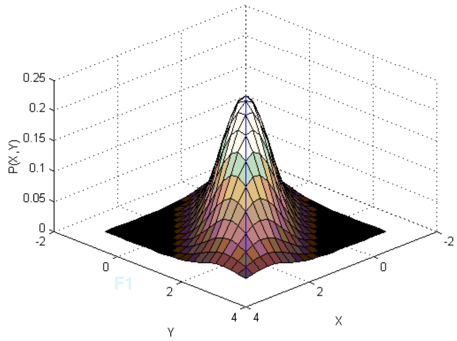
Now we are given an X, but no Y

What is Y?

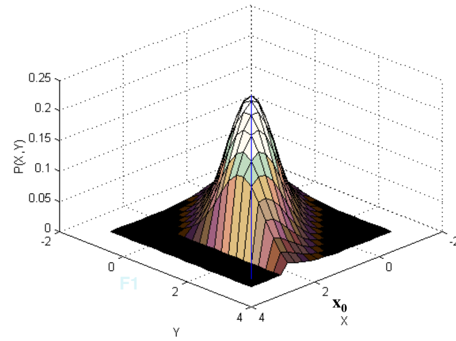
MAP estimator for Gaussian RV



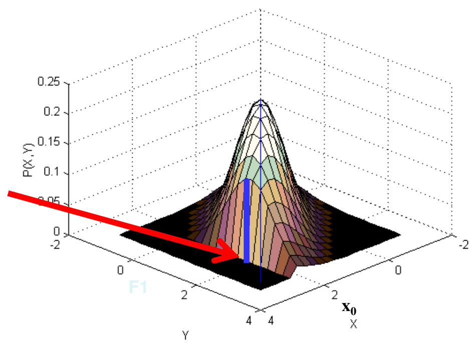
MAP estimation: Gaussian PDF



MAP estimation: The Gaussian at a particular value of X

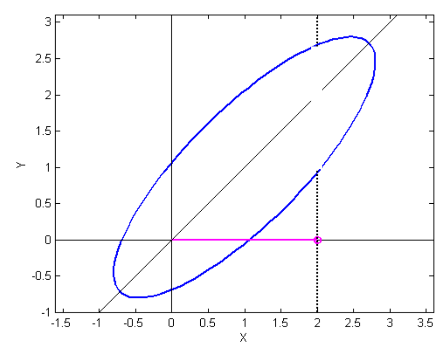


MAP estimation: The Gaussian at a particular value of X

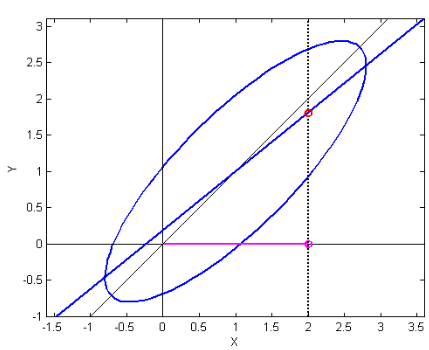


MAP Estimation of a Gaussian RV

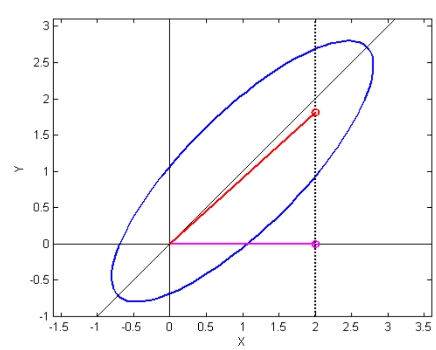
$Y = argmax_y P(y| X)$???



MAP Estimation of a Gaussian RV

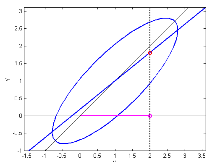


MAP Estimation of a Gaussian RV



So what is this value?

- Clearly a line
- Equation of Line:



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

- Scalar version given; vector version is identical

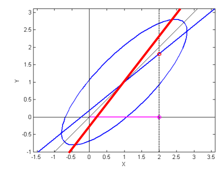
$$\hat{\mathbf{y}} = \mu_X + C_{YX} C_{XX}^{-1} (\mathbf{x} - \mu_{\mathbf{x}})$$

- Derivation? Later in the program a bit

This is a *multiple* regression

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

- This is the MAP estimate of **y**
  - NOT the regression parameter



- What about the ML estimate of **y**
  - Again, ML estimate of **y**, not regression parameter

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

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

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

  - Minimize above term

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

- Minimize error:

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

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

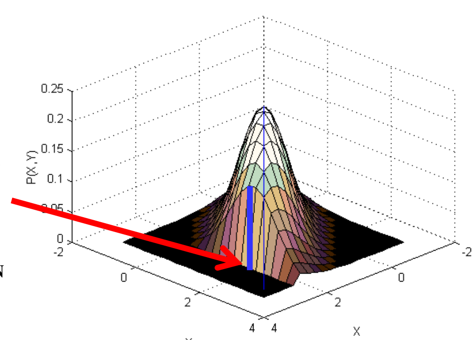
- Differentiating and equating to 0:

$$dErr = 2E[\mathbf{y}^T\mathbf{y} + \hat{\mathbf{y}}^T\hat{\mathbf{y}} - 2\hat{\mathbf{y}}^T\mathbf{y} \mid \mathbf{x}] = 2\hat{\mathbf{y}}^T d\hat{\mathbf{y}} - 2E[\mathbf{y} \mid \mathbf{x}]^T d\hat{\mathbf{y}} = 0$$

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

The MMSE estimate is the mean of the distribution

For the Gaussian: MAP = MMSE



- Would be true of any symmetric distribution

MMSE estimates for mixture distributions

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

- Let P(y|X) be a mixture density
- The MMSE estimate of y is given by

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

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

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

MMSE estimates from a Gaussian mixture

- Let P(**x**,**y**) be a Gaussian Mixture

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

- Let P(**y**|**x**) is also a Gaussian mixture

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

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

MMSE estimates from a Gaussian mixture

- Let P(**y**|**x**) is a Gaussian Mixture

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

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

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

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

MMSE estimates from a Gaussian mixture

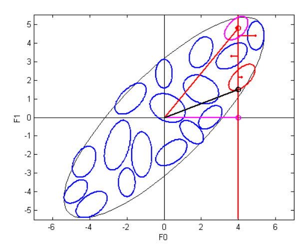
$$P(\mathbf{y} \mid \mathbf{x}) = \sum_k P(k \mid \mathbf{x}) N(\mathbf{y}; \mu_{k,\mathbf{y}} + C_{k,\mathbf{yx}} C_{k,\mathbf{xx}}^{-1} (\mathbf{x} - \mu_{k,\mathbf{x}}), \Theta)$$

- P(y|x) is a mixture density
- E(Y|X) is also a mixture

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

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

MMSE estimates from a Gaussian mixture



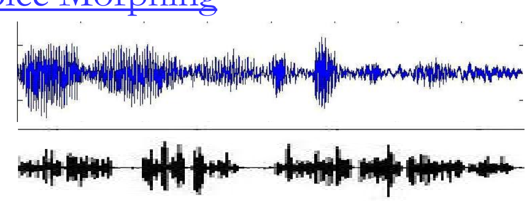
- A mixture of estimates from individual Gaussians

MMSE with GMM: Voice Transformation

- Festvox GMM transformation suite (Toda)

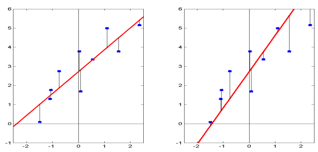
	awb	bdl	jmk	slt
awb				
bdl				
jmk				
slt				

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*

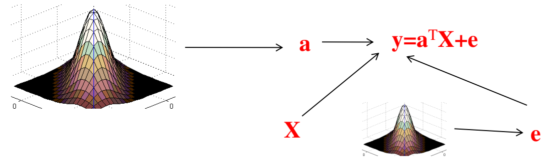
A problem with regressions



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

- ML fit is sensitive
  - Error is squared
  - Small variations in data → large variations in weights
  - Outliers affect it adversely
- Unstable
  - If dimension of X >= no. of instances
    - (XX^T) is not invertible

MAP estimation of weights



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

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

- Maximum *a posteriori* estimate

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

MAP estimation of weights

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

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

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

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

- Similar to ML estimate with an additional term

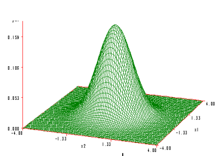
MAP estimate of weights

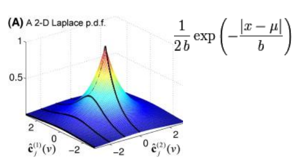
$$dL = \left(2\mathbf{a}^T\mathbf{X}\mathbf{X}^T + 2\mathbf{y}\mathbf{X}^T + 2\sigma\mathbf{I}\right)d\mathbf{a} = 0$$

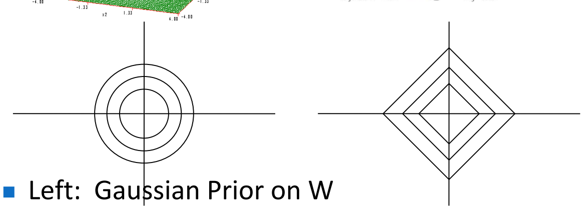
$$\mathbf{a} = \left(\mathbf{X}\mathbf{X}^T + \sigma\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{Y}^T$$

- 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 L1 regularized least-squares estimation

L1-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

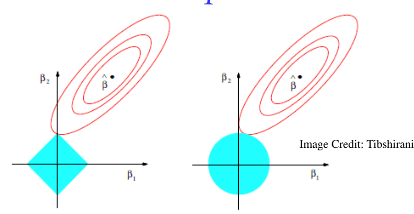
$$\hat{\mathbf{a}} = \arg\max_{\mathbf{A}} C' - (\mathbf{y} - \mathbf{a}^T\mathbf{X})^T(\mathbf{y} - \mathbf{a}^T\mathbf{X})^T \quad \textbf{subject to} \quad |\mathbf{a}|_1 \leq t$$

- "LASSO" – Least absolute shrinkage and selection operator

LASSO Algorithms

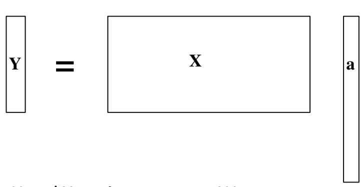
- Various convex optimization algorithms
- LARS: Least angle regression
- Pathwise coordinate descent..
- Matlab code available from web

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
- L1 regularization selects ***sparsest*** solution

LASSO and Compressive Sensing



- Given Y and X, estimate sparse W
- LASSO:
  - X = explanatory variable
  - Y = dependent variable
  - a = weights of regression
- CS:
  - X = measurement matrix
  - Y = measurement
  - a = data

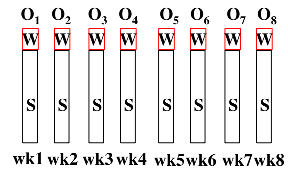
An interesting problem: Predicting War!

- 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!

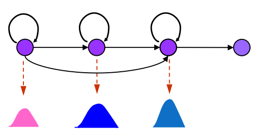
- Issues:
  - Dissatisfaction builds up – not an instantaneous phenomenon
    - Usually
  - War / rebellion build up much faster
    - Often in hours
- Important to predict
  - Preparedness for security
  - Economic impact

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

A Step Aside: Predicting Time Series

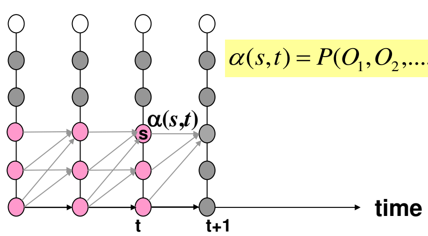


- An HMM is a model for time-series data
- How can we use it predict the future?

Predicting with an HMM

- Given
  - Observations $O_1..O_t$
  - All HMM parameters
    - Learned from some training data
- Must estimate future observation O_{t+1}
  - Estimate must consider *entire* history ($O_1..O_t$)
  - No knowledge of actual state of the process at any time

Predicting with an HMM

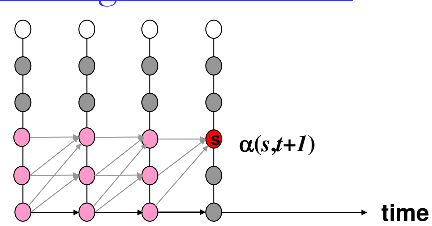


$$\alpha(s,t) = P(O_1, O_2, ..., O_t, state(t) = s)$$

- Given $O_1..O_t$
  - Compute $P(O_1..O_t, s)$
  - Using the forward algorithm – computes $\alpha(s,t)$

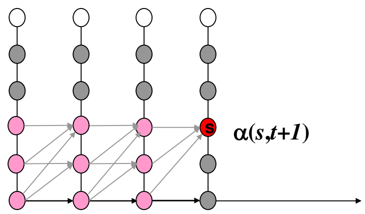
$$P(s_t = s \mid O_{1..t}) = \frac{P(s_t = s, O_{1..t})}{\sum_{s'} P(s_t = s', O_{1..t})} = \frac{\alpha(s,t)}{\sum_{s'} \alpha(s',t)}$$

Predicting with an HMM



- Given $P(s_t=s \mid O_{1..t})$ for all s
- $P(s_{t+1} = s \mid O_{1..t}) = \Sigma_{s'} P(s_t=s' \mid O_{1..t})P(s \mid s')$
- $P(O_{t+1}, s \mid O_{1..t}) = P(O \mid s) P(s_{t+1}=s \mid O_{1..t})$
- $P(O_{t+1} \mid O_{1..t}) = \Sigma_s P(O_{t+1}, s \mid O_{1..t})$
  $= \Sigma_s P(O \mid s) P(s_{t+1}=s \mid O_{1..t})$
- This is a mixture distribution

Predicting with an HMM



- $P(O_{t+1}|O_{1..t}) = \Sigma_s P(O_{t+1},s|O_{1..t})$
  $= \Sigma_s P(O|s) P(s_{t+1}=s|O_{1..T})$
- MMSE estimate of O_{t+1} given $O_{1..t}$
  - $E[O_{t+1} | O_{1..t}] = \Sigma_s P(s_{t+1}=s|O_{1..T}) E[O|s]$
- A weighted sum of the state means

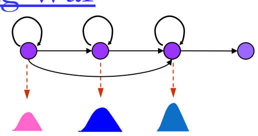
Predicting with an HMM

- MMSE Estimate of $O_{t+1} = E[O_{t+1}|O_{1..T}]$
  - $E[O_{t+1} | O_{1..t}] = \Sigma_s P(s_{t+1}=s|O_{1..T}) E[O|s]$
- If $P(O|s)$ is a GMM
  - $E(O|s) = \Sigma_k P(k|s) \mu_{k,s}$

$$\hat{O}_{t+1} = \sum_s P(s|O_{1..t}) \sum_k w_{k,s}\mu_{k,s}$$

$$\hat{O}_{t+1} = \sum_s \frac{\alpha(t,s)}{\sum_{s'} \alpha(t,s')} \sum_k w_{k,s}\mu_{k,s}$$

Predicting War



- Train an HMM on z = [w, s]
- After the t[th] week, predict probability distribution:
  - $P(z_t | z_1 \dots z_t) = P(w, z | z_1..z_t)$
- Marginalize out x (not known for next week)

$$P(w|z_{1..t}) = \int P(w,s|z_{1..t})ds$$

- War? → $E[w | z_1..z_t]$