Regression and Prediction

Class 15. 23 Oct 2012

Instructor: Bhiksha Raj
Matrix Identities

\[ f(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \]

\[ df(\mathbf{x}) = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \\ \vdots \\ \frac{df}{dx_D} \end{bmatrix} \]

- 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

\[
f(x) = \begin{bmatrix}
    x_{11} & x_{12} & \cdots & x_{1D} \\
    x_{21} & x_{22} & \cdots & x_{2D} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{D1} & x_{D2} & \cdots & x_{DD}
\end{bmatrix}
\]

\[
\frac{df}{dx_1} = \begin{bmatrix}
    \frac{df}{dx_{11}} & \frac{df}{dx_{12}} & \cdots & \frac{df}{dx_{1D}} \\
    \frac{df}{dx_{21}} & \frac{df}{dx_{22}} & \cdots & \frac{df}{dx_{2D}} \\
    \vdots & \vdots & \ddots & \vdots \\
    \frac{df}{dx_{D1}} & \frac{df}{dx_{D2}} & \cdots & \frac{df}{dx_{DD}}
\end{bmatrix}
\]

- 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

\[ \mathbf{F}(\mathbf{x}) = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \]

\[ \begin{bmatrix} \frac{dF_1}{dx_1} & \frac{dF_1}{dx_2} & \frac{dF_1}{dx_D} \\ \frac{dF_2}{dx_1} & \frac{dF_2}{dx_2} & \frac{dF_2}{dx_D} \\ \vdots & \vdots & \vdots \\ \frac{dF_N}{dx_1} & \frac{dF_N}{dx_2} & \frac{dF_N}{dx_D} \end{bmatrix} \]

- 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

23 Oct 2012
Matrix derivative identities

\[ d(Xa) = Xda \quad d(a^T X) = X^T da \]

\[ d(AX) = (dA)X \quad d(XA) = X(dA) \]

\[ d(a^T Xa) = a^T (X + X^T) da \]

\[ d(\text{trace}(A^T XA)) = d(\text{trace}(XAA^T)) = d(\text{trace}(AA^TX)) = (X^T + X)dA \]

- Some basic linear and quadratic identities

\[ X \] is a matrix, \( a \) is a vector.
Solution may also be \( X^T \)

\( A \) is a matrix
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

- \( y = f(x; \Theta) + e \)

Different possibilities

- \( y \) is a scalar
  - \( Y \) is real
  - \( Y \) is categorical (classification)

- \( y \) is a vector

- \( x \) is a vector
  - \( x \) is a set of real valued variables
  - \( x \) is a set of categorical variables
  - \( x \) 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 $x$ and $y$
  - $y = \textit{dependent}$ variable
  - $x = \textit{explanatory}$ variable
  - Given $x$, $y$ can be predicted as an affine function of $x$
An imaginary regression..

- 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

- \( y = Ax + b + e \)
  - \( e \) = prediction error

- Given a “training” set of \( \{x, y\} \) values: estimate \( A \) and \( b \)
  - \( y_1 = Ax_1 + b + e_1 \)
  - \( y_2 = Ax_2 + b + e_2 \)
  - \( y_3 = Ax_3 + b + e_3 \)
  - ... 

- If \( A \) and \( b \) are well estimated, prediction error will be small
Linear Regression to a scalar

\[ y_1 = a^T x_1 + b + e_1 \]
\[ y_2 = a^T x_2 + b + e_2 \]
\[ y_3 = a^T x_3 + b + e_3 \]

- Define:
  \[ y = [y_1 \ y_2 \ y_3 \ldots] \]
  \[ X = \begin{bmatrix} x_1 & x_2 & x_3 & \ldots \end{bmatrix} \]
  \[ A = \begin{bmatrix} a \\ b \end{bmatrix} \]

- Rewrite

\[ y = A^T X + e \]
Learning the parameters

\[ y = A^T X + e \]

\[ \hat{y} = A^T X \]  \quad \text{Assuming no error}

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

\[ y_1 = a^T x_1 + b + e_1 \]
\[ y_2 = a^T x_2 + b + e_2 \]
\[ y_3 = a^T x_3 + b + e_3 \]

\[ y = A^T X + e \]

\[ D(y, \hat{y}) = E = e_1^2 + e_2^2 + e_3^2 + ... \]

\[ = (y_1 - a^T x_1 - b)^2 + (y_2 - a^T x_2 - b)^2 + (y_3 - a^T x_3 - b)^2 + ... \]

\[ E = (y - A^T X)(y - A^T X)^T = \|y - A^T X\|^2 \]

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

\[ y = a^T x + e \]

- \( e \) = prediction error
- Find the “slope” \( a \) such that the total squared length of the error lines is minimized
Solving a linear regression

\[ y = A^T X + e \]

- Minimize squared error

\[ E = \| y - X^T A \|^2 = (y - A^T X)(y - A^T X)^T \]

\[ = yy^T + A^T XX^T A - 2yX^T A \]

- Differentiating w.r.t. \( A \) and equating to 0

\[ dE = \left( 2A^T XX^T - 2yX^T \right) dA = 0 \]

\[ A^T = yX^T (XX^T)^{-1} = ypinv(X) \]

\[ A = (XX^T)^{-1} Xy^T \]
What happens if we minimize the perpendicular instead?
Regression in multiple dimensions

\[ y_1 = A^T x_1 + b + e_1 \]
\[ y_2 = A^T x_2 + b + e_2 \]
\[ y_3 = A^T x_3 + b + e_3 \]

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

\[ y_1 = A^T x_1 + b + e_1 \]
\[ y_{11} = a_1^T x_1 + b_1 + e_{11} \]
\[ y_{12} = a_2^T x_2 + b_2 + e_{12} \]
\[ y_{13} = a_3^T x_3 + b_3 + e_{13} \]

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

\[ Y = [y_1 \ y_2 \ y_3 ...] \quad X = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots \\ 1 & 1 & 1 & \cdots \end{bmatrix} \quad A = \begin{bmatrix} A \\ b \end{bmatrix} \]

\[ E = [e_1 \ e_2 \ e_3 ...] \]

\[ Y = A^T X + E \]

\[ DIV = \sum_i \| y_i - A^T x_i - b \|^2 = trace( (Y - A^T X)(Y - A^T X)^T ) \]

- Differentiating and equating to 0

\[ dDIV = \left( 2A^T XX^T - 2YX^T \right) dA = 0 \]

\[ A^T = YX^T (XX^T)^{-1} = Y \text{pinv}(X) \]

\[ A = (XX^T)^{-1} XY^T \]
A Different Perspective

- $y$ is a noisy reading of $A^T x$
  
  $y = A^T x + e$

- Error $e$ is Gaussian
  
  $e \sim N(0, \sigma^2 I)$

- Estimate $A$ from
  
  $Y = [y_1 \ y_2 \ldots y_N] \quad X = [x_1 \ x_2 \ldots x_N]$
The **Likelihood** of the data

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

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

\[
P(y \mid x; A) = N(A^T x, \sigma^2 I) \]

- Probability of the collection: \( Y = [y_1, y_2\ldots y_N] \quad X = [x_1, x_2\ldots x_N] \)

\[
P(Y \mid X; A) = \prod_{i} N(A^T x_i, \sigma^2 I) \]

- Assuming IID for convenience (not necessary)
A Maximum Likelihood Estimate

\[ y = A^T x + e \quad e \sim N(0, \sigma^2 I) \quad Y = [y_1 \ y_2 \ldots y_N] \quad X = [x_1 \ x_2 \ldots x_N] \]

\[
P(Y \mid X) = \prod_i \frac{1}{\sqrt{(2\pi\sigma^2)^D}} \exp \left( -\frac{1}{2\sigma^2} \| A^T x_i \|^2 \right)
\]

\[
\log P(Y \mid X; A) = C - \sum_i \frac{1}{2\sigma^2} \| y_i - A^T x_i \|^2
\]

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

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

\[
A^T = YY^T \left( XX^T \right)^{-1} = Y \text{pinv}(X)
\]

\[
A = \left( XX^T \right)^{-1} XY^T
\]
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:

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

\[
\begin{bmatrix}
  x_t \\
  x_{t-1} \\
  \vdots \\
  x_{K+1}
\end{bmatrix} = a_t^T \begin{bmatrix}
  x_{t-1} & x_{t-2} & \ldots & x_K \\
  x_{t-2} & x_{t-3} & \ldots & x_{K-1} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{t-K} & x_{t-K-1} & \ldots & x_1
\end{bmatrix} + \begin{bmatrix}
  e_t \\
  e_{t-1} \\
  \vdots \\
  e_{K+1}
\end{bmatrix}
\]

\[\mathbf{x} = a_t^T \mathbf{X} + \mathbf{e}\]

\[\mathbf{x} \text{ pinv}(\mathbf{X}) = a_t^T\]
Applying it to our problem

- Backward prediction

\[
\begin{bmatrix}
    x_{t-K-1} \\
    x_{t-K-2} \\
    \vdots \\
    x_1
\end{bmatrix}
= \mathbf{b}_t^T
\begin{bmatrix}
    x_t & x_{t-1} & \ldots & x_{K+1} \\
    x_{t-1} & x_{t-2} & \ldots & x_K \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{t-K} & x_{t-K-1} & \ldots & x_2
\end{bmatrix}
+ \begin{bmatrix}
    e_{t-K-1} \\
    e_{t-K-2} \\
    \vdots \\
    e_1
\end{bmatrix}
\]

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

\[
\bar{x} \; \text{pinv}(\bar{X}) = \mathbf{b}_t^T
\]
Finding the burst

- At each time
  - Learn a “forward” predictor $a_t$
  - At each time, predict next sample $x_t^{est} = \sum_i a_{t,k}x_{t-k}$
  - 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 \( x_t^{\text{est}} = \sum_i a_{t,k} x_{t-k} \)
    - 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

![Graph showing reconstruction of a signal with distortion and recovery.](chart)

- **Reconstruction area**
- **Distorted signal**
- **Recovered signal**
- **Interpolation result**
- **Actual data**
- **Next glitch**

23 Oct 2012
Incrementally learning the regression

\[ A = (XX^T)^{-1}XY^T \]

Can we learn A incrementally instead?

- As data comes in?

The Widrow Hoff rule

\[ a^{t+1} = a^t + \eta (y_t - \hat{y}_t)x_t \]
\[ \hat{y}_t = (a^t)^T x_t \]

Note the structure

- Can also be done in batch mode!
Predicting a value

\[ A = (XX^T)^{-1} XY^T \]

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

- What are we doing exactly?

\[ C = XX^T \]

- Let

\[ \hat{x} = C^{-2} \]

  - Normalizing and rotating space
  - The rotation is irrelevant

\[ \hat{y} = Y\hat{X}^T \hat{x} = \sum_{i} \hat{x}_i^T \hat{y}_i \]

- Weighted combination of inputs
Relationships are not always linear

- How do we model these?
- Multiple solutions
Non-linear regression

- $y = \varphi(x) + e$

$x \rightarrow \varphi(x) = [\phi_1(x) \ \phi_2(x) \ldots \phi_N(x)]$

$X \rightarrow \Phi(X) = [\varphi(x_1) \ \varphi(x_2) \ldots \varphi(x_K)]$

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

Replace $X$ with $\Phi(X)$ in earlier equations for solution

$$A = \left(\Phi(X)\Phi(X)^T\right)^{-1} \Phi(X)Y^T$$
What we are doing

- Finding the optimal combination of various function
  - Remind you of something?
Being non-committal: Local Regression

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

\[ \hat{y} = Y\hat{X}^T\hat{x} = \sum_i \hat{x}_i \hat{y}_i \]

- How about doing this locally?
  - For any \( x \)

\[ y = \sum_i x^T C^{-1} x_i y_i + e \]

\[ y = \sum_i d(x, x_i) y_i + e \]
Local Regression

- The resulting regression is dependent on $x$!

$$\hat{y}(x) = \sum_i d(x, x_i)y_i$$

- No closed form solution
  - But can be highly accurate

- But what is $d(x, x')$??
Kernel Regression

\[ \hat{y} = \frac{\sum_{i} K_h(x - x_i) y_i}{\sum_{i} K_h(x - x_i)} \]

- Actually a non-parametric MAP estimator of \( y \)
  - Note – an estimator of \( y \), 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 $y$ (in a statistical sense), given that we know $x$
  \[ y = \arg\max_y P(Y|x) \]

- ML (Maximum Likelihood): Find that value of $Y$ for which the statistical best guess of $X$ would have been the observed $X$
  \[ y = \arg\max_y P(x|Y) \]

- 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

\[ z = \begin{bmatrix} y \\ x \end{bmatrix} \]

\[ \mu_z = \frac{1}{N} \sum_{i=1}^{N} z_i \]

\[ C_z = \frac{1}{N} \sum_{i=1}^{N} (z_i - \mu_z)(z_i - \mu_z)^T \]

\[ \mu_z = \begin{bmatrix} \mu_y \\ \mu_x \end{bmatrix} \]

\[ C_z = \begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix} \]
Learning the parameters of the Gaussian

\[ \mu_z = \frac{1}{N} \sum_{i=1}^{N} z_i \]

\[ \mu_z = \begin{bmatrix} \mu_y \\ \mu_x \end{bmatrix} \]

\[ z = \begin{bmatrix} y \\ x \end{bmatrix} \]

\[ C_z = \frac{1}{N} \sum_{i=1}^{N} (z_i - \mu_z)(z_i - \mu_z)^T \]

\[ C_z = \begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix} \]

\[ \mu_x = \frac{1}{N} \sum_{i=1}^{N} x_i \]

\[ C_{XY} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y)^T \]
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 $X$

Most likely value
MAP Estimation of a Gaussian RV

\[ Y = \arg \max_y P(y|X) \]
MAP Estimation of a Gaussian RV
MAP Estimation of a Gaussian RV

![Graph showing MAP estimation of a Gaussian RV](image)
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

- Derivation? Later in the program a bit
This is a *multiple* regression

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

- 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
It's 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

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

- Minimize above term
It's also a minimum-mean-squared error estimate

- Minimize error:

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

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

- Differentiating and equating to 0:

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

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

The MMSE estimate is the mean of the distribution
For the Gaussian: $\text{MAP} = \text{MMSE}$

Most likely value is also the mean value.

- Would be true of any symmetric distribution.
MMSE estimates for mixture distributions

Let $P(y|X)$ be a mixture density.

The MMSE estimate of $y$ is given by

\[ P(y|x) = \sum_k P(k)P(y|k,x) \]

\[ E[y|x] = \int y \sum_k P(k)P(y|k,x)dy = \sum_k P(k)\int yP(y|k,x)dy = \sum_k P(k)E[y|k,x] \]

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

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

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

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

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

$$P(y|x) = \sum_k P(k|x)P(y|x,k)$$
MMSE estimates from a Gaussian mixture

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

\[
P(y|x) = \sum_k P(k|x)P(y|x,k)
\]

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

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

\[
P(y|x) = \sum_k P(k|x)N(y; \mu_{k,y} + C_{k,yx}C_{k,xx}^{-1}(x - \mu_{k,x}), \Theta)
\]
MMSE estimates from a Gaussian mixture

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

- \( P[y \mid x] \) is a mixture density
- \( E[y \mid x] \) is also a mixture

\[ E[y \mid x] = \sum_k P(k \mid x)E[y \mid k, x] \]

\[ E[y \mid x] = \sum_k P(k \mid x)\left(\mu_{k,y} + C_{k,yx}C_{k,xx}^{-1}(x - \mu_{k,x})\right) \]
MMSE estimates from a Gaussian mixture

- A mixture of estimates from individual Gaussians
MMSE with GMM: Voice Transformation

- Festvox GMM transformation suite (Toda)

<table>
<thead>
<tr>
<th></th>
<th>awb</th>
<th>bdl</th>
<th>jmk</th>
<th>slt</th>
</tr>
</thead>
<tbody>
<tr>
<td>awb</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
</tr>
<tr>
<td>bdl</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
</tr>
<tr>
<td>jmk</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
</tr>
<tr>
<td>slt</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
<td>♫</td>
</tr>
</tbody>
</table>
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

- ML fit is sensitive
  - Error is squared
  - Small variations in data $\rightarrow$ large variations in weights
  - Outliers affect it adversely

- Unstable
  - If dimension of $X \geq$ no. of instances
    - $(XX^T)$ is not invertible

$$A = (XX^T)^{-1} XY^T$$
**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
  \[
  \hat{\mathbf{a}} = \arg \max_\mathbf{a} \log P(\mathbf{a} | \mathbf{y}, \mathbf{X}) = \arg \max_\mathbf{A} \log P(\mathbf{y} | \mathbf{X}, \mathbf{a})P(\mathbf{a})
  \]

\[ \mathbf{y} = \mathbf{a}^T \mathbf{X} + \mathbf{e} \]
MAP estimation of weights

\[ \hat{a} = \arg \max_a \log P(a \mid y, X) = \arg \max_a \log P(y \mid X, a)P(a) \]

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

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

\[ \hat{a} = \arg \max_a C' - \log \sigma - \frac{1}{2\sigma^2} (y - a^T X)^T (y - a^T X) - 0.5\sigma^2 a^T a \]

- Similar to ML estimate with an additional term
**MAP estimate of weights**

\[ dL = \left( 2a^T XX^T + 2yX^T + 2\sigma I \right) da = 0 \]

\[ a = \left( XX^T + \sigma I \right)^{-1} XY^T \]

- 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
  
  \[ P(a) = \lambda^{-1} \exp(-\lambda^{-1}|a|_1) \]

- Maximum \textit{a posteriori} estimate

\[
\hat{a} = \arg \max_A C' - (y - a^T X)^T (y - a^T X)^T - \lambda^{-1}|a|_1
\]

- No closed form solution
  
  - Quadratic programming solution required
    
    - Non-trivial
MAP estimation of weights with laplacian prior

- Assume weights drawn from a Laplacian
  \[ P(a) = \lambda^{-1} \exp(-\lambda^{-1} |a|_1) \]

- Maximum a posteriori estimate

\[
\hat{a} = \arg \max_A C' - (y - a^T X)^T (y - a^T X)^T - \lambda^{-1} |a|_1
\]

- Identical to L1 regularized least-squares estimation
L1-regularized LSE

\[ \hat{a} = \arg \max_A C'(y - a^T X)^T (y - a^T X)^T - \lambda^{-1}|a|_1 \]

- No closed form solution
  - Quadratic programming solutions required

- Dual formulation

\[ \hat{a} = \arg \max_A C'(y - a^T X)^T (y - a^T X)^T \quad \text{subject to} \quad |a|_1 \leq t \]

- “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:
  - $X = \text{explanatory variable}$
  - $Y = \text{dependent variable}$
  - $a = \text{weights of regression}$
- CS:
  - $X = \text{measurement matrix}$
  - $Y = \text{measurement}$
  - $a = \text{data}$

$$Y = X \cdot a$$
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
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

- **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 $s$
- $P(s_{t+1} = s \mid O_{1..t}) = \sum_{s'} P(s_{t}=s' \mid O_{1..t})P(s\mid s')$
- $P(O_{t+1},s\mid O_{1..t}) = P(O\mid s) \ P(s_{t+1}=s\mid O_{1..t})$
- $P(O_{t+1}\mid O_{1..t}) = \sum_s P(O_{t+1}\mid s\mid O_{1..t})$
  \[ = \sum_s P(O\mid s) \ P(s_{t+1}=s\mid O_{1..t})\]
- This is a mixture distribution

23 Oct 2012

11755/18797
Predicting with an HMM

- $P(O_{t+1}|O_{1..t}) = \sum_s P(O_{t+1},s|O_{1..t})$
  $= \sum_s P(O|s) P(s_{t+1}=s|O_{1..T})$

- MMSE estimate of $O_{t+1}$ given $O_{1..t}$
  - $E[O_{t+1} | O_{1..t}] = \sum_s P(s_{t+1}=s|O_{1..T}) E[O|s]$

- A weighted sum of the state means

23 Oct 2012

11755/18797

85
Predicting with an HMM

- **MMSE Estimate of** $O_{t+1} = E[O_{t+1}|O_{1..T}]$
  - $E[O_{t+1} | O_{1..t}] = \sum_s P(s_{t+1}=s|O_{1..T}) E[O|s]$

- **If** $P(O|s)$ is a GMM
  - $E(O|s) = \sum_k P(k|s) \mu_{k,s}$

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

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

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

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

- War? $\Rightarrow E[w | z_1..z_t]$