Machine Learning for Signal Processing
Regression and Prediction

Class 16. 28 Oct 2014

Instructor: Bhiksha Raj
Matrix Identities

\[ f(\mathbf{x}) \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \quad df(\mathbf{x}) = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \\ \vdots \\ \frac{df}{dx_D} \end{bmatrix} \]

- The derivative of a scalar function w.r.t. a vector is a vector
**Matrix Identities**

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

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

- 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

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

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

- The derivative of a vector function w.r.t. a vector is a matrix
  - Note transposition of order
• In general: Differentiating an $M \times N$ function by a $U \times V$ argument results in an $M \times N \times U$ tensor derivative.
Matrix derivative identities

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

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

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

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

• Some basic linear and quadratic identities
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
  – \( f(.) \) 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..

- 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:
\[ y = [y_1 \ y_2 \ y_3 \ldots] \]
\[ x = \begin{bmatrix} x_1 & x_2 & x_3 & \ldots \end{bmatrix} \]
\[ e = [e_1 \ e_2 \ e_3 \ldots] \]

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

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

\[ y = A^T X + e = \hat{y} + e \]

\[ D(y, \hat{y}) = E = e_1^2 + e_2^2 + e_3^2 + ... \]
\[ = (y_1 - a^T x_1 - b)^2 + (y_2 - a^T x_2 - b)^2 + (y_3 - a^T x_3 - b)^2 + ... \]

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

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

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

\[ A = (XX^T)^{-1} Xy^T \]
Regression in multiple dimensions

- Also called *multiple regression*
- Equivalent of saying:
  \[ y_i = A^T x_i + b + e_i \]
  
  \[ y_1 = A^T x_1 + b + e_1 \]
  \[ y_2 = A^T x_2 + b + e_2 \]
  \[ y_3 = A^T x_3 + b + e_3 \]

  \[ y_i \text{ is a vector} \]

  \[ y_{ij} = \text{\(j\)th component of vector } y_i \]

  \[ a_i = \text{\(i\)th column of } A \]

  \[ b_j = \text{\(j\)th component of } b \]

  \[ y_{i1} = a_1^T x_i + b_1 + e_{i1} \]
  \[ y_{i2} = a_2^T x_i + b_2 + e_{i2} \]
  \[ y_{i3} = a_3^T x_i + b_3 + e_{i3} \]

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

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

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

\[ Y = \hat{A}^T X + E \]

\[ DIV = \sum_i \left\| y_i - \hat{A}^T \bar{x}_i \right\|^2 = trace( (Y - \hat{A}^T X)(Y - \hat{A}^T X)^T ) \]

• Differentiating and equating to 0

\[ d.Div = -2(Y - \hat{A}^T X) X^T d\hat{A} = 0 \quad \text{YX}^T = \hat{A}^T XX^T \]

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

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

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

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

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

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

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

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

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

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

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

\[ A^T = YX^T \left(XX^T\right)^{-1} = Ypinv(X) \]

\[ A = \left(XX^T\right)^{-1} XY^T \]
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: $\hat{y} = A^TX$
Applying it to our problem

• Prediction by regression

• Forward regression

\[ x_t = a_1 x_{t-1} + a_2 x_{t-2} \ldots a_k x_{t-k} + e. \]

• Backward regression

\[ x_t = b_1 x_{t+1} + b_2 x_{t+2} \ldots b_k x_{t+k} + \]
Applying it to our problem

• Forward prediction

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

\[x = Xa_t + e\]

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

- Backward prediction

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

\[
\bar{x} = \overline{X}b_t + e
\]

\[
pinv(\overline{X})\bar{x} = b_t
\]
Finding the burst

• At each time
  – Learn a “forward” predictor $a_t$
  – 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
Filling the hole

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

- Average forward and backward predictions
Reconstruction zoom in

Reconstruction area

Interpolation result

Distorted signal

Recovered signal

Actual data

Next glitch
Incrementally learning the regression

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

Requires knowledge of all (x,y) pairs

• Can we learn A incrementally instead?
  – As data comes in?

• The Widrow Hoff rule

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

Scalar prediction version

• 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?
  - For the explanation we are assuming no “b” (X is 0 mean)
  - Explanation generalizes easily even otherwise

\[ C = XX^T \]

- Let \( \hat{x} = C^{-\frac{1}{2}} x \) and \( \hat{X} = C^{-\frac{1}{2}} X \)
- Whitening x
- \( N^{-0.5} C^{-0.5} \) is the whitening matrix for x

\[ \hat{y} = YX^T C^{-\frac{1}{2}} \hat{X}^{-\frac{1}{2}} X = Y\hat{X}^T \hat{x}_i \]
Predicting a value

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

\[ \hat{y} = Y\hat{X}^T \hat{x} = \frac{1}{N} \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix} \begin{bmatrix} \hat{x}_1^T \\ \vdots \\ \hat{x}_N^T \end{bmatrix} = \sum_{i} y_i \left( \hat{x}_i^T \hat{x}_i \right) \]

- What are we doing exactly?
Predicting a value

\[
\hat{y} = \sum_i y_i (\hat{x}^T \hat{x})
\]

- Given training instances \((x_i, y_i)\) for \(i = 1..N\), estimate \(y\) for a new test instance of \(x\) with unknown \(y\):
- \(y\) is simply a *weighted sum of the \(y_i\) instances from the training data*
- The weight of any \(y_i\) is simply the inner product between its corresponding \(x_i\) and the new \(x\)
  - With due whitening and scaling..
What are we doing: A different perspective

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

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

\[ A = X(X^T X)^{-1} Y^T \quad \hat{y} = Y(X^T X)^{-1} X^T x \]
High-dimensional regression

\[ \hat{y} = Y(X^T X)^{-1} X^T x \]

- \( X^T X \) is the “Gram Matrix”

\[ G = \begin{bmatrix}
    x_1^T x_1 & x_1^T x_2 & \ldots & x_1^T x_N \\
    x_2^T x_1 & x_2^T x_2 & \ldots & x_2^T x_N \\
    \vdots & \vdots & \ddots & \vdots \\
    x_N^T x_1 & x_N^T x_2 & \ldots & x_N^T x_N
\end{bmatrix} \]

\[ \hat{y} = YG^{-1}X^T x \]
High-dimensional regression

\[ \hat{y} = YG^{-1}X^T x \]

• Normalize \( Y \) by the inverse of the gram matrix

\[ \bar{Y} = YG^{-1} \]

• Working our way down..

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

\[ \hat{y} = \sum_i \bar{y}_i x_i^T x \]
Linear Regression in High-dimensional Spaces

\[ \hat{y} = \sum_{i} \bar{y}_i x_i^T x \]

\[ \bar{Y} = YG^{-1} \]

- Given training instances \((x_i, y_i)\) for \(i = 1..N\), estimate \(y\) for a new test instance of \(x\) with unknown \(y\):
- \(y\) is simply a weighted sum of the normalized \(y_i\) instances from the training data
  - The normalization is done via the Gram Matrix
- The weight of any \(y_i\) is simply the inner product between its corresponding \(x_i\) and the new \(x\)
Relationships are not always linear

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

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

\[ \mathbf{x} \rightarrow \phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_N(\mathbf{x})] \]

\[ \mathbf{X} \rightarrow \Phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_K)] \]

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

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

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

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

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

- \( \Phi(X) \) may be in a very high-dimensional space
- The high-dimensional space (or the transform \( \Phi(X) \)) may be unknown..
The regression is in high dimensions

• **Linear regression:**
  \[ \hat{y} = \sum_{i} \bar{y}_i x_i^T x \]
  \[ \bar{Y} = YG^{-1} \]

• **High-dimensional regression**

\[
G = \begin{bmatrix}
\Phi(x_1)^T \Phi(x_1) & \Phi(x_2)^T \Phi(x_2) & \ldots & \Phi(x_1)^T \Phi(x_N) \\
\Phi(x_2)^T \Phi(x_1) & \Phi(x_2)^T \Phi(x_2) & \ldots & \Phi(x_2)^T \Phi(x_N) \\
\vdots & \vdots & \ddots & \vdots \\
\Phi(x_1)^T \Phi(x_1) & \Phi(x_N)^T \Phi(x_2) & \ldots & \Phi(x_N)^T \Phi(x_N)
\end{bmatrix}
\]

\[ \bar{Y} = YG^{-1} \]

\[ \hat{y} = \sum_{i} \bar{y}_i \Phi(x_i)^T \Phi(x) \]
Doing it with Kernels

• *High-dimensional regression with Kernels*:

\[ K(x, y) = \Phi(x)^T \Phi(y) \]

\[
G = \begin{bmatrix}
K(x_1, x_1) & K(x_1, x_1) & \ldots & K(x_1, x_N) \\
K(x_2, x_1) & K(x_2, x_2) & \ldots & K(x_2, x_N) \\
\vdots & \vdots & \ddots & \vdots \\
K(x_N, x_1) & K(x_N, x_2) & \ldots & K(x_N, x_N)
\end{bmatrix}
\]

• Regression in Kernel Hilbert Space..

\[
\bar{Y} = YG^{-1}
\]

\[
\hat{y} = \sum_i \bar{y}_i K(x_i, x)
\]
A different way of finding nonlinear relationships: Locally linear regression

• Previous discussion: Regression parameters are optimized over the entire training set
• Minimize
  \[
  E = \sum_{\text{all } i} \left| y_i - A^T x_i - b \right|^2
  \]
• Single global regression is estimated and applied to all future \( x \)
• Alternative: *Local regression*
• *Learn a regression that is specific to* \( x \)
Being non-committal: Local Regression

• Estimate the regression to be applied to any \( x \) using training instances near \( x \)

\[
E = \sum_{x_j \in \text{neighborhood}(x)} \left\| y_i - A^T x_i - b \right\|^2
\]

• The resultant regression has the form

\[
y = \sum_{x_j \in \text{neighborhood}(x)} d(x, x_j)y_j + e
\]

– Note: this regression is specific to \( x \)

• A separate regression must be learned for every \( x \)
Local Regression

\[ y = \sum_{x_j \in \text{neighborhood}(x)} d(x, x_j) y_j + e \]

• But what is \( d() \)?
  – For linear regression \( d() \) is an inner product

• More generic form: Choose \( d() \) as a function of the distance between \( x \) and \( x_j \)

• If \( d() \) falls off rapidly with \( |x \text{ and } x_j| \) the “neighborhood” requirement can be relaxed

\[ y = \sum_{\text{all}} d(x, x_j) y_j + e \]
Kernel Regression: \( d() = K() \)

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

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

• MAP (Maximum A Posteriori): Find a “best guess” for $y$ (statistically), given known $x$
  $$y = \arg\max_y P(Y|x)$$

• 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.
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

\[ Y = \arg\max_y P(y \mid X) \]

![Graph showing MAP estimation of a Gaussian RV]
MAP Estimation of a Gaussian RV

![Graph showing MAP estimation of a Gaussian RV]
MAP Estimation of a Gaussian RV

\[ Y = \text{argmax}_y P(y \mid X) \]
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
  \[
  \hat{y} = \mu_Y + C_{YX} C_{XX}^{-1} (\mathbf{x} - \mu_x)
  \]
- Derivation? Later in the program a bit
  – Note the similarity to regression
This is a *multiple* regression

\[ \hat{y} = \mu_Y + C_{yx} C^{-1}_{xx} (x - \mu_x) \]

- This is the MAP estimate of \( y \)
  - \( y = \arg\max_Y P(Y|x) \)

- What about the ML estimate of \( y \)
  - \( \arg\max_Y P(x|Y) \)

- Note: Neither of these may be the *regression* line!
  - MAP estimation of \( y \) is the regression on \( Y \) for Gaussian RVs
  - *But this is not the MAP estimation of the 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

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

- Minimize above term
Its also a \textit{minimum-mean-squared error} estimate

- Minimize error:

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

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

- Differentiating and equating to 0:

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

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

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

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

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

- 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

<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>

11755/18797
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 $\mathbf{X}$ $\geq$ no. of instances
    - $(\mathbf{X}\mathbf{X}^T)$ is not invertible

$$A = (\mathbf{X}\mathbf{X}^T)^{-1} \mathbf{X} \mathbf{Y}^T$$
MAP estimation of weights

- Assume weights drawn from a Gaussian
  \[ P(a) = \mathcal{N}(0, \sigma^2 I) \]
- Max. Likelihood estimate
  \[ \hat{a} = \arg \max_a \log P(y \mid X; a) \]
- Maximum \textit{a posteriori} estimate
  \[ \hat{a} = \arg \max_a \log P(a \mid y, X) = \arg \max_a \log P(y \mid X, a) P(a) \]
MAP estimation of weights

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

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

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

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

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

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

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

• 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

\[
\frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right)
\]
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

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

• 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 \textit{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 L\(_1\) regularized least-squares estimation
**L$_1$-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

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

- “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

Image Credit: Tibshirani
LASSO and Compressive Sensing

- Given $Y$ and $X$, estimate sparse $a$
- 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
Predicting Time Series

• Need *time-series models*

• HMMs – later in the course