Linear Units

A. Preliminaries

\[ y = s = w^T x \]
Model Assumptions and Parameters

- Training examples \((x(k), d(k))\) drawn randomly, second order zero mean sequences.

- Parameters
  - Inputs: \(x(k) \in \mathbb{R}^n\)
  - Weights: \(w(k) \in \mathbb{R}^n\)
  - Outputs: \(y(k) = w(k)^T x(k)\)
  - Desired outputs: \(d(k)\)
  - Error: \(e(k) = d(k) - y(k)\)

- Error criterion (MSE)
  \[
  \min J(w) = E \left[ .5(e(k))^2 \right]
  \]
Define $P = E(x(k)d(k))$ and $R = E(x(k)x(k)^T)$.

$J(w) = .5 \ E[(d(k)-y(k))^2]$

$= .5E(d(k)^2) - E(x(k)d(k))^T w + w^T E(x(k)x(k)^T) w$

$= .5E[d(k)^2] - P^T w + .5w^TRw$

Note $J(w)$ is a quadratic function of $w$. To minimize $J(w)$ find gradient, $\nabla J(w)$ and set to 0.

$\nabla J(w) = -P + Rw = 0$

$Rw = P$ (Wiener solution)

If $R$ is nonsingular, then $w = R^{-1}P$.

Resulting MSE $= .5E[d(k)^2] - .5P^TR^{-1}P$
Gradient based iterative algorithms

- Steepest descent algorithm (move in direction of negative gradient)
  \[ w(k+1) = w(k) - \mu \nabla J(w(k)) = w(k) + \mu (P - Rw(k)) \]

- Least mean square algorithm (approximate gradient from training example)
  \[ \hat{\nabla} J(w(k)) = -e(k)x(k) \]
  \[ w(k+1) = w(k) + \mu e(k)x(k) \]
Steepest Descent Convergence

- \( w(k+1) = w(k) + \mu (P-Rw(k)) \); Let \( w^* \) be solution.

  Center weight vector \( v = w - w^* \)

- \( v(k+1) = v(k) - \mu (Rv(k)) \); Assume \( R \) is nonsingular.

  Decorrelate weight vector \( u = Q^{-1}v \) where \( R = Q\Lambda Q^{-1} \) is the transformation that diagonalizes \( R \).

- \( u(k+1) = (I - \mu \Lambda)u(k) \), \( u(k) = (I - \mu \Lambda)^k u(0) \).

  Conditions for convergence \( 0 < \mu < 2/\lambda_{\text{max}} \).
Step Size $\mu$

$\mu$ too large

$\mu$ too small
Rate of Convergence

- Rate of convergence depends on eigenvalues, $\lambda_i$ as convergence rate for this eigenvalue is $(1 - \mu \lambda_i)$. Key eigenvalues are $\lambda_{\text{min}}$ and $\lambda_{\text{max}}$.

- Fastest rate of convergence achieved when setting $\mu = 2 / (\lambda_{\text{min}} + \lambda_{\text{max}})$. This results in smallest and largest eigenvalue having same convergence rate.

- Convergence of SD depends on condition number of matrix $\lambda_{\text{max}} / \lambda_{\text{min}}$.
Energy Function

- Energy Function:
  \[ J(w) = \frac{1}{2} \sigma_d^2 P^T w + \frac{1}{2} w^T R w \]
  For optimal weight \( R w^* = P \) and
  \[ J_{\text{min}} = J(w^*) = \frac{1}{2} \sigma_d^2 - \frac{1}{2} P^T w^* \]
- SD energy function behavior
  \[ J(w(k)) = J_{\text{min}} + \frac{1}{2} (w(k) - w^*)^T R (w(k) - w^*) \]
  \[ = J_{\text{min}} + \frac{1}{2} u(k)^T \Lambda u(k) \]
  \[ = J_{\text{min}} + \frac{1}{2} \sum_i (I - \mu \lambda_i)^{2k} u_i(0)^2 \]
LMS Algorithm

- SD requires knowledge of R and P. In many applications these second order statistics are unknown.
- Least mean square algorithm
  \[ \nabla J(w(k)) = -e(k)x(k) \]
  \[ w(k+1) = w(k) + \mu e(k)x(k) \]
- LMS algorithm is an iterative noisy gradient descent algorithm that approximates SD from one training example.
- LMS algorithm attempts to find weight that minimizes mean squared error cost function, J(w).
LMS Algorithm Properties

- Steepest Descent and LMS algorithm convergence depends on step size $\mu$ and eigenvalues of $R$.
- LMS algorithm is simple to implement.
- LMS algorithm convergence is relatively slow.
- Tradeoff between convergence speed and excess MSE.
- LMS algorithm can track training data that is time varying.
LMS Convergence Behavior

- Assumptions: \( x(n) \) iid sequence, \( x(n) \) independent of \( d(n-k), \ k > 0 \), \( d(n) \) independent of \( y(n-k), \ k > 0 \), \( x(n) \) and \( d(n) \) are jointly Gaussian.

- Mean convergence analysis: Let \( e^*(k) = d(k) - w^T x(k) \), denote error from optimal weight at time \( k \).
  - \( E(v(k+1)) = (I - \mu R) E(v(k)) + \mu E(x(k)e^*(k)) \)
  - Asymptotically assuming step size is chosen correctly, then \( \lim_k E(v(k)) = 0 \) and \( E(w(k)) \) converges to \( w^* \)

- Mean squared analysis studies cost function \( J(w(k)) \). Note \( \text{tr}(R) > \lambda_{\text{max}} \) and more conservative bound given by \( 0 < \mu < 2 / \text{tr}(R) \).
Iterative Algorithm Comments

- Algorithms based on descending energy surface by examining first and second derivatives.
- LMS (stochastic gradient descent), tradeoffs between algorithm complexity and convergence speed.
- Can use other cost functions besides quadratic cost functions: Absolute error, Minkowski error, entropy function.
- Can apply to nonlinear activation units or multi-layer networks.
- Levenberg-Marquardt algorithm: another approximation of energy function using Taylor series. Uses pseudo inverse and can approximate Newton’s method or gradient descent.
Least Squares Algorithm

- Let \((x(k),d(k)), 1 \leq k \leq m\) then LS algorithm finds weight \(w\) such that squared error is minimized. Let \(e(k) = d(k) - w^Tx(k)\), then cost function for LS algorithm given by \(J(w) = .5\sum_k e(k)^2\).

- In matrix form can represent
  \[ J(w) = .5 \|d-Xw\|^2 = .5\|d\|^2 - d^TXw + .5w^TX^TXw \]
  where \(d\) is vector of desired outputs and \(X\) contains inputs arranged in rows.
Least Squares Solution

- Let $X$ be the data matrix, $d$ the desired output, and $w$ the weight vector.
- Previously we showed that
  
  $$J(w) = 0.5 \|d - Xw\|^2 = 0.5\|d\|^2 - d^T Xw + 0.5w^T X^T X w$$

  where $d$ is vector of desired outputs and $X$ contains inputs arranged in rows.
- LS solution given by $X^T X w^* = X^T d$ (normal equation) with $w^* = X^d$. If $X^T X$ is of full rank then $X^d = (X^T X)^{-1} X^T$.
- Output $y = X w^*$ and error $e = d - y$
- Desired output often of form $d = X w^* + v$
Adaptive Filter

\[ y(n) = \sum w_i u(n-i) \]

\[ e(n) = d(n) - \sum y(n) \]
Data Presentation

Windowed data

\[ X = \begin{bmatrix}
  u(k) & u(k-1) & \cdots & u(k-n+1) \\
  u(k-1) & u(k-2) & \cdots & u(k-n) \\
  \vdots & \vdots & \ddots & \vdots \\
  u(k-m+1) & u(k-1) & \cdots & u(k-n-m)
\end{bmatrix} \]

\[ d = \begin{bmatrix}
  d(k) \\
  d(k-1) \\
  \vdots \\
  d(k-m+1)
\end{bmatrix} \]

Fixed window, growing window, exponential weighted window
Least Squares Solution Comments

- Note LS solution approximates Wiener solution as window size gets large: \( R \approx \frac{1}{m} X^T X, \ P \approx \frac{1}{m} X^T d \)
- Principle of orthogonality (Projection theorem): error orthogonal to data \( e^T X = 0 \) which results in
  \[
  J(w^*) = .5||d||^2 - d^T X w^* = .5||d||^2 - .5d^T X X^\dagger d
  \]
- Normal equations are derived from principle of orthogonality (scalar representation):
  \[
  \Sigma w_j^* \Sigma u(i-k)u(i-j) = \Sigma u(i-k)d(i) \quad k=0\ldots m-1
  \]
- Ridge regression: add regularization term to get
  \[
  w^* = (\lambda I + X^T X)^{-1} X^T d
  \]
Time Correlations

- Let $\Phi = X^T X$, or $\Phi(k,j) = \Sigma u(i-k)u(i-j)$ represent time correlation data matrix
  - Symmetric, positive semi-definite, eigenvalues are nonnegative real numbers
- Let $z = X^T d$, or $z(k) = \Sigma u(i-k)d(i)$ represent time cross-correlation
- LS solution given by $\Phi w^* = z$
Least Squares statistical properties

- Estimate of weight $w^*$ is unbiased
  \[ d = X w^* + v \]

- When measurement error process is white with zero mean and variance $\sigma^2$ the covariance matrix of the LS estimate $w^*$ equals $\sigma^2 \Phi^{-1}$

- When measurement error process is white with zero mean, the LS estimate $w^*$ is the best linear unbiased estimate

- In addition when the measurement process is Gaussian the LS estimate is the same as the maximum likelihood estimate and achieves the Cramer-Rao lower bound
Singular Value Decomposition

- LS solution given by $w^* = X^+d$ involves computing the pseudoinverse of the Moore-Penrose generalized inverse of the matrix $X$. (matlab $w = X\backslash d$ or $w = \text{pinv}(X)*d$)
- Any matrix $X$ can be decomposed into a general eigenvector / eigenvalue decomposition.
  - If $X$ is symmetric we have that $X = Q\Lambda Q^T$
  - For arbitrary $X$ we have that $X = U S V^T$ were $S$ contains singular values and $U$ and $V$ are unitary matrices
SVD Continued

- $X$ (m by n) can be decomposed using SVD to get that $X = USV^T$ where
- $S$ (m by n) with first $\min(m,n)$ rows being a diagonal matrix containing square root of eigenvalues of $XX^T$ and $X^TX$. Rest of rows are zeros
- $U$ (m by m) is unitary and contains eigenvectors of $XX^T$
- $V$ (n by n) is unitary and contains eigenvectors of $X^TX$
- $S = U^TXV$ (matlab $[U,S,V] = \text{svd}(X)$)
SVD Continued

- Note $X = USV^T$ and correlation data matrix $\Phi = X^TX = VS^TSV^T$.
- As number of observations in window grows large we have $R \approx (1/m) \Phi$ and $R = QAQ^T$. Therefore $Q \approx V$ and $\lambda_i \approx (1/m) s_i^2$.
- Methods of obtaining SVD. Matrix operations such as Givens rotations and Householder transformations.
Singular Values and Pseudo-Inverse

- One useful benefit of SVD is that it is easy to express pseudoinverse in terms of SVD terms,
  \[ X^\dagger = V(S_i)^T U^T \]

where \( S_i = \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} \)

- For LS problem assuming \( m > n \) (over determined system and assume \( X^TX \) is of full rank \( n \), then can easily show by substituting \( X=USV^T \) that pseudoinverse is given by above.
Recursive Least Square Algorithm

- Can develop an on-line version of LS algorithm called Recursive LS (RLS) algorithm
- Algorithm based on using Sherman-Morrison-Woodbury formula:

\[
(A + vv^T)^{-1} = A^{-1} - A^{-1} v(1 + v^T A^{-1} v) v^T A^{-1}
\]

where \(A = X^T X\) contains old data and \(v = x(m+1)\) contains new data at time \(m+1\)
- Similar to Kalman filter equations where we update estimate recursively adding new information or innovations.
- Update is \(O(n^2)\) operations
RLS Algorithm Comments

- Often exponentially weighted algorithm implemented. Update correlation matrix, gain factor, and weights.
- Parameters of RLS algorithm: Initial correlation matrix and weight decay factor.
- Convergence is typically an order of magnitude faster than LMS algorithm. Algorithm theoretically converges to zero excess mean squared error and convergence does not depend on eigenvalues.
- Many variations to account for more stable matrix computations: QR and Cholesky factorizations.
Linear Filter Applications

- **Inverse Modeling: Channel Equalization**
- **Adaptive Beamforming**
  - Radar
  - Sonar
  - Speech enhancement
- **System Identification: Plant modeling**
- **Prediction**
- **Adaptive Interference Cancellation: Echo Cancellation**
Nonlinear Methods

- Multilayer feedforward networks: error back propagation algorithm
- Kernel methods:
  - Support Vector Machines (SVM)
  - Least squares methods
  - Radial Basis Functions (RBF)