Lecture 8: Regression Based Learning and Support Vector Machines

Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu (860) 486-2890

Fall 2018
November 5, 2018

Reading List

- Duda, Hart and Stork, Chapter 5
- Murphy, Chapters 14 and 16
- Bishop, Chapters 4, 6, 7 and 11
- Theodiridis, Chapters 3-6

Review So Far

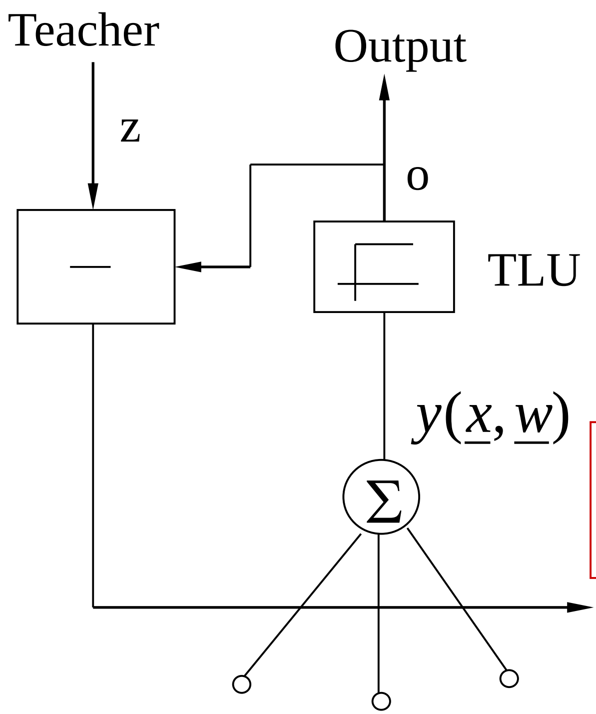
- LDA & QDA
- Generative versus discriminative
- Generative
  - ML or Bayesian
- Discriminative
  - Logistic (binary) or softmax (multi-class)
  - IRLS
  - Variational
  - Log-sum-exponent bounds
  - Perceptrons
  - Decision Trees
- Density-based
  - PNN
  - kNN

Lecture Outline

- Regression Based Learning
- Optimization Techniques
- LMS and Convergence Analysis
- Better Methods (RLS, Gauss-Newton,..)
- Ho-Kashyap Procedures
- Support Vector Machines $\Rightarrow$ QP
- Subgradient-based SVM
- SVM and Elastic Net
- SVM Regression $\Rightarrow$ QP
- Summary

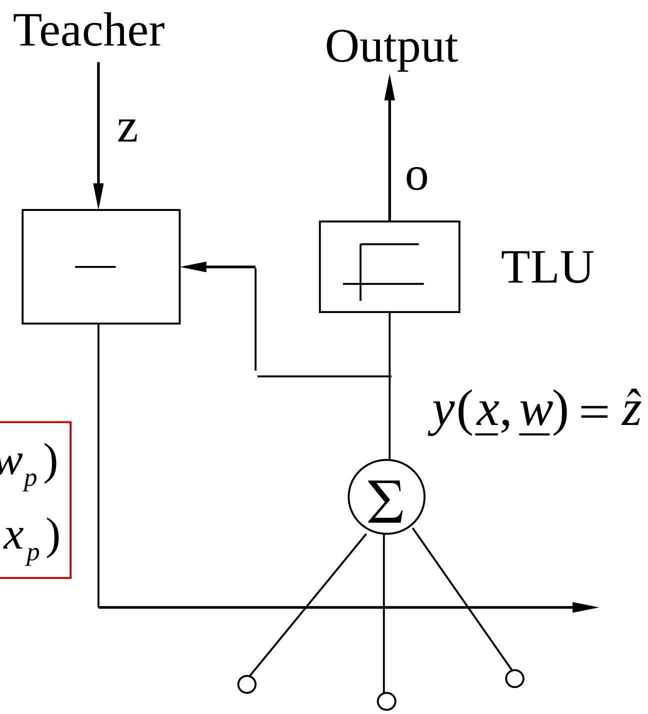
Regression Based Learning - 1

❑ Approximation (Regression based) learning



$$\underline{w}^T = (w_0 \; w_1 \cdots\cdots w_p)$$

$$\underline{x} = (-1 \; x_1 \; x_2 \cdots\cdots x_p)$$



In Perceptron learning, the teacher is available **after** the threshold logic unit (TLU)

In Approximation-based learning (also called Delta learning), the teacher is available **before** the threshold logic unit (TLU)

Regression Based Learning - 2

$$\min J(\underline{w}) = \frac{1}{2}\sum_{n=1}^{N}(z^n - y(\underline{x}^n, \underline{w}))^2 = \frac{1}{2}\sum_{n=1}^{N} e_n^2$$

- For multiple outputs, say C

$$J(\underline{w}) = \frac{1}{2}\sum_{k=1}^{C}\underbrace{\left[\sum_{n=1}^{N}(z_k{}^n - \hat{z}_k{}^n)^2\right]}_{\text{for ouput } k} = \frac{1}{2}\sum_{k=1}^{C}\sum_{n=1}^{N} e_{nk}^2$$

can minimize one output at a time

$$tr[(\underline{z}^n - \hat{\underline{z}}^n)(\underline{z}^n - \hat{\underline{z}}^n)^T] = (\underline{z}^n - \hat{\underline{z}}^n)^T(\underline{z}^n - \hat{\underline{z}}^n) = e_n^2$$

In general

$$J(\underline{w}) = \frac{1}{2}\sum_{n=1}^{N} e_n^2(\underline{x}^n, \underline{w})$$

~ sometimes we scale it by 1/N. then this corresponds to MSE

This is a nonlinear least squares problem (if y is nonlinear).
All the techniques use first and second order derivative information.

Regression Based Learning - 3

$$\nabla J(\underline{w}) = \underline{g} = \begin{bmatrix} \partial J / \partial w_o \\ \partial J / \partial w_1 \\ \vdots \\ \partial J / \partial w_p \end{bmatrix} = \sum_{i=1}^{N} e_n(\underline{x}^n, \underline{w}) \nabla e_n(\underline{x}^n, \underline{w}) = \sum_{i=1}^{N} e_n(\underline{x}^n, \underline{w}) \underline{g}^n$$

$$= \begin{bmatrix} \underline{g}^1 & \underline{g}^2 & \cdots\cdots & \underline{g}^N \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} = J^T \underline{e}$$

J = Jacobian matrix

$$\nabla^2 J(\underline{w}) = \begin{bmatrix} \dfrac{\partial^2 J}{\partial w_0{}^2} & \cdots & \cdots & \dfrac{\partial^2 J}{\partial w_0 w_p} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \dfrac{\partial^2 J}{\partial w_0 w_p} & \cdots & \cdots & \dfrac{\partial^2 J}{\partial w_p{}^2} \end{bmatrix} = \sum_{n=1}^{N} \underline{g}^n \underline{g}^{nT} + \underbrace{\sum_{n=1}^{N} e_n(\underline{x}^n, \underline{w}) \nabla^2 e_n(\underline{x}^n, \underline{w})}_{negelect \Rightarrow Gauss-Newton\,(EKF)}$$

$(p+1)$ by $(p+1)$ symmetric matrix

❑ Optimization Techniques

$$\underline{w}^{(0)} \rightarrow \underline{w}^{(1)} \rightarrow \cdots\cdots\cdots \rightarrow \underline{w}^{(k)} \rightarrow \cdots\cdots \rightarrow \underline{w}^{*}$$

$$J(\underline{w}^{(0)}) \geq J(\underline{w}^{(1)}) \geq \cdots\cdots\cdots \geq J(\underline{w}^{*})$$

Note that $\underline{g}^{(k)} = \nabla J(\underline{w}^{(k)})$ is the direction of increase at $\underline{w} = \underline{w}^{(k)}$, then $-\underline{g}^{(k)}$ is the direction of local decrease in J at $\underline{w}^{(k)}$.

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \alpha^{k}\underline{g}^{(k)}$$ steepest descent or Gradient or Cauchy method

$$\alpha^{(k)} = \text{step size} > 0$$

For α_k small $J(\underline{w}^{(k+1)}) = J(\underline{w}^{(k)}) - \alpha^{(k)}\underline{g}^{(k)\mathrm{T}}\underline{\mathrm{g}}^{(\mathrm{k})} < J(\underline{w}^{(k)})$

$\exists$ *Other directions that decrease* J

$$J(\underline{w}^{(k)} + \alpha^{(k)}\underline{d}^{(k)}) \approx J(\underline{w}^{(k)}) + \alpha^{(k)}\underline{g}^{(k)T}\underline{d}^{(k)}$$

$$\Rightarrow \text{ Need directions } \ni \underline{g}^{(k)T}\underline{d}^{(k)} < 0$$

such directions are called descent directions.

Optimization Techniques - 2

suppose take $\underline{d}^{(k)} = -H^{(k)}\underline{g}^{(k)}$; $\quad H^{(k)} > 0 \Rightarrow -\underline{g}^{(k)T} H_k \underline{g}^{(k)} < 0$

α^k is obtained via line search. NN typically select α^k heuristically, <u>*they do not perform line search*</u>.

Direction $\underline{d}^k$ determines the type of algorithm.

1. ***SD*** $\qquad H^{(k)} = I$
2. ***Diagonally scaled SD***

$$H^{(k)} = \left\{ \left[\frac{\partial^2 J}{\partial w_i^{(k)^2}} \right]^{-1} \right\} = \left[1 / \frac{\partial^2 J}{\partial w_i^{(k)^2}} \right]$$

3. ***Newton*** $\quad H^{(k)} = \left[\nabla^2 J(\underline{w}^{(k)}) \right]^{-1}$

If J is a quadratic function in $\underline{w}$, Newton's method will converge in one step (iteration).

Convergence of SD versus Newton

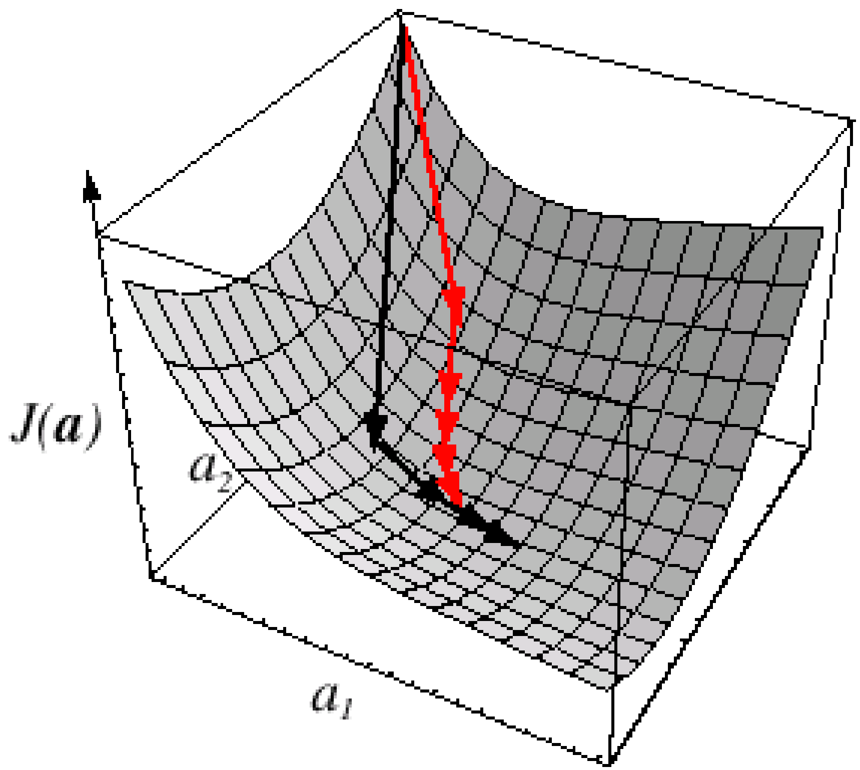


Figure. The sequence of weight vectors given by a simple gradient descent method (red) and by Newton's (second order) algorithm (black). Newton's method typically leads to greater improvement.

Optimization Techniques - 3

Two problems
- Hessian not PD
- Hessian is expensive to compute

4. Diagonally modified Newton

4.1

$$H^{(k)} = \begin{cases} \left[\nabla^2 J(\underline{w}^{(k)})\right]^{-1} & \text{if PD} \\ \left[\nabla^2 J(\underline{w}^{(k)}) + \mu_k I\right]^{-1} & \text{if not PD} \end{cases}$$

can be done via LDL^T factorization or Cholesky decompostion

4.2 Periodic Hessian or Periodic Newton

$$H^{(kT+i)} = \left[\nabla^2 J(\underline{w}^{(kT)})\right]^{-1} \qquad \begin{aligned} i &= 0,1,2,\cdots\cdots,T-1 \\ k &= 0,1,2,\cdots\cdots \end{aligned}$$

5. Discretized Newton

Backward difference:

$$\frac{\partial^2 J(\underline{w}^{(k)})}{\partial w_i^2} = \frac{\dfrac{\partial J(\underline{w}^{(k)})}{\partial w_i} - \dfrac{\partial J(\underline{w}^{(k)} - \underline{e}_i h)}{\partial w_i}}{h}$$

***h* = discretization step**

Optimization Techniques - 4

$$\frac{\partial^2 J(\underline{w}^{(k)})}{\partial w_i \partial w_j} = \frac{\dfrac{\partial J(\underline{w}^{(k)})}{\partial w_i} - \dfrac{\partial J(\underline{w}^{(k)} - \underline{e}_j h)}{\partial w_i}}{h}$$

where $\underline{e}_i$ and $\underline{e}_j$ **are unit vectors**

Central difference:

$$\frac{\partial^2 J(\underline{w}^{(k)})}{\partial w_i^2} = \frac{\dfrac{\partial J(\underline{w}^{(k)} + \underline{e}_i h)}{\partial w_i} - \dfrac{\partial J(\underline{w}^{(k)} - \underline{e}_i h)}{\partial w_i}}{2h}$$

$$\frac{\partial^2 J(\underline{w}^{(k)})}{\partial w_i \partial w_j} = \frac{\dfrac{\partial J(\underline{w}^{(k)} + \underline{e}_j h)}{\partial w_i} - \dfrac{\partial J(\underline{w}^{(k)} - \underline{e}_j h)}{\partial w_i}}{2h}$$

In NN, $\dfrac{\partial J}{\partial w_i}$ **can be obtained via back propagation.**

6. *Gauss-Newton or Outer approximation to Hessian*

Recall $\nabla^2 J(\underline{w}) = \sum_{n=1}^{N} g^n g^{nT} + \sum_{n=1}^{N} e_n(\underline{x}^n,\underline{w})\nabla^2 e_n(\underline{x}^n,\underline{w})$

Near optimum, $e_n \to 0$ so,

$$\nabla^2 J(\underline{w}) \approx \sum_{n=1}^{N} g^n g^{nT}$$ Exact for linear $y(\underline{x},\underline{w}) = \hat{z} = \underline{w}^T \underline{x}$

so, Gauss-Newton method uses

$$H^{(k)} = \begin{cases} \left[\sum_{n=1}^{N} g^{n(k)} g^{n(k)^T}\right]^{-1} & \text{if invertible} \\ \left[\sum_{n=1}^{N} g^{n(k)} g^{n(k)^T} + \mu_k I\right]^{-1} & \text{if not} \end{cases}$$

The latter modification is called Levenberg-Marquardt update. This approach is quite useful in ML identification of linear dynamic systems (recall C-R bound) and Extended Kalman filters.

7. Conjugate Gradient method

It has the property of **quadratic termination**, that is, finds the optimal solution in *p+1* iterations if *J* is a quadratic function in $\underline{w}$.

CG method generates directions without having to form H_k

$$\underline{d}^{(k)} = -\underline{g}^{(k)} + \beta^{(k)} \underline{d}^{(k-1)}$$

$$\beta_k = \begin{cases} \dfrac{\underline{g}^{(k)T} \underline{g}^{(k)}}{\underline{g}^{(k-1)T} \underline{g}^{(k-1)}} & \text{Fletcher - Reeves update} \\ \dfrac{\underline{g}^{(k)T} \left[\underline{g}^{(k)} - \underline{g}^{(k-1)}\right]}{\underline{g}^{(k-1)T} \underline{g}^{(k-1)}} & \text{Polak - Ribiere - Polyak update} \\ \dfrac{\underline{g}^{(k)T} \left[\underline{g}^{(k)} - \underline{g}^{(k-1)}\right]}{\left[\underline{g}^{(k)} - \underline{g}^{(k-1)}\right]^T \underline{d}^{(k-1)}} & \text{Sorensen - Wolfe update} \end{cases}$$

- Sensitive to step size suboptimality

8. *Quasi-Newton (secant) methods*

These methods strive to approximate **inverse** Hessian $H^{(k)}$, or Hessian, $B^{(k)}$.

- Inverse Hessian:

$$H^{(k+1)} = H^{(k)} + \frac{\underline{p}^{(k)}\underline{p}^{(k)^T}}{\underline{p}^{(k)^T}\underline{q}^{(k)}} - \frac{H^{(k)}\underline{q}^{(k)}\underline{q}^{(k)^T}H^{(k)}}{\underline{q}^{(k)^T}H^{(k)}\underline{q}^{(k)}}$$

Davidon-Fletcher-Powell (DFP) update

- Hessian:

$$B^{(k+1)} = B^{(k)} + \frac{\underline{q}^{(k)}\underline{q}^{(k)^T}}{\underline{q}^{(k)^T}B^{(k)}\underline{q}^{(k)}} - \frac{B^{(k)}\underline{p}^{(k)}\underline{p}^{(k)^T}B^{(k)}}{\underline{p}^{(k)^T}B^{(k)}\underline{p}^{(k)}}$$

Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

In both of these updates

$$\underline{p}^{k} = \underline{w}^{(k+1)} - \underline{w}^{(k)}$$

$$\underline{q}^{k} = \underline{g}^{(k+1)} - \underline{g}^{(k)}$$

Optimization Techniques - 8

- *Zero memory versions: Inner products only!*
  - *DFP version*

$$H^{(k+1)} = I + \frac{\underline{p}^{(k)}\underline{p}^{(k)^T}}{\underline{p}^{(k)^T}\underline{q}^{(k)}} - \frac{\underline{q}^{(k)}\underline{q}^{(k)^T}}{\underline{q}^{(k)^T}\underline{q}^{(k)}}$$

$$\underline{d}^{(k+1)} = -\underline{g}^{(k+1)} - \frac{\underline{p}^{(k)^T}\underline{g}^{(k+1)}}{\underline{p}^{(k)^T}\underline{q}^{(k)}}\underline{p}^{(k)} + \frac{\underline{q}^{(k)^T}\underline{g}^{(k+1)}}{\underline{q}^{(k)^T}\underline{q}^{(k)}}\underline{q}^{(k)}$$

  - *BFGS version*

$$\underline{d}^{(k+1)} = -\underline{g}^{(k+1)} + [\frac{\underline{q}^{(k)^T}\underline{g}^{(k+1)}}{\underline{p}^{(k)^T}\underline{q}^{(k)}} - (1 + \frac{\underline{q}^{(k)^T}\underline{q}^{(k)}}{\underline{p}^{(k)^T}\underline{q}^{(k)}})\frac{\underline{p}^{(k)^T}\underline{g}^{(k+1)}}{\underline{p}^{(k)^T}\underline{q}^{(k)}}]\underline{p}^{(k)}$$

$$+ \frac{\underline{p}^{(k)^T}\underline{g}^{(k+1)}}{\underline{p}^{(k)^T}\underline{q}^{(k)}}\underline{q}^{(k)}$$

Linear Regression Models

- Let us consider the linear case first

  → ADALINE (Adaptive Linear Element)

$$y(\underline{x},\underline{w}) = \underline{w}^T\,\underline{x}$$

Also, linear if know the form of the polynomial

$w_0 + w_1x_1 + \cdots\cdots + w_px_p + w_{12}x_1x_2 + \cdots + w_{pp}x_1x_p + \cdots + w_{12}x_p{}^2$ etc.

or transformed $\underline{x}$, $\phi(\underline{x})$, e.g., Chebyshev, Legendre polynomials

$$\min J(\underline{w}) = \frac{1}{2}\sum_{n=1}^{N}(z^n - \underline{w}^T\,\underline{x}^n)^2$$

$$\nabla J(\underline{w}) = \underline{g} = -\sum_{n=1}^{N}(z^n - \underline{w}^T\,\underline{x}^n)\underline{x}^n = -\sum_{n=1}^{N}e_n\,\underline{x}^n$$

Least Squares Solution

- Key: "Gradient is sum over training patterns"

$$\nabla^2 J(\underline{w}) = \sum_{n=1}^{N} (\underline{x}^n \left(\underline{x}^n\right)^T) = \text{"information matrix"}$$

"Hessian is sum over training patterns as well"

Let us look at several possibilities:

Optimum $\Rightarrow \nabla J(\underline{w}) = 0$

$$\Rightarrow \sum_{n=1}^{N} \underline{x}^n z^n = \left[\sum_{n=1}^{N} \underline{x}_n \underline{x}_n^T \right] \underline{w}$$

Let $X = \begin{bmatrix} \left(\underline{x}^1\right)^T \\ \left(\underline{x}^2\right)^T \\ \vdots \\ \left(\underline{x}^N\right)^T \end{bmatrix} \sim N$ by $(p+1)$ matrix; $\quad \underline{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}$

Orthogonalization Methods

$$X^T X \underline{w} = X^T \underline{z}$$

$$\underline{w} = (X^T X)^{-1} X^T \underline{z} = X^{\dagger} \underline{z}$$

$X^{\dagger} =$ pseudo inverse of X, a $(p+1)$ by N matrix ($generalized\ \ inverse$)

corresponds to solving

$X\underline{w} = \underline{z}$ in a **least squares** sense

How to find generalized inverse?

1. **Orthogonalization methods (Gram-Schmidt, Givens, Householder):**

- Gram-Schmidt: $X = QR$ $\qquad Q$ is N by $(p+1)$, R is $(p+1)$ by $(p+1)$

Singular Value Decomposition

- Givens and Householder:

$$X = \tilde{Q}\tilde{R} \qquad \tilde{Q} \text{ is } N \text{ by } N,\ \tilde{R} \text{ is } N \text{ by } (p+1);\ \tilde{Q} = N[\overset{p+1}{Q} \vdots \overset{N-p-1}{Q_1}];\ \tilde{R} = \begin{bmatrix} R \\ \cdots \\ 0 \end{bmatrix}$$

$$= QR$$

so, $\quad \underline{w} = (X^T X)^{-1} X^T \underline{z} = (R^T R)^{-1} R^T Q^T \underline{z}$

or, $\quad \underline{w} = R^{-1} Q^T \underline{z} \qquad$ if R is full rank

2. **Singular Value Decomposition (SVD)**

$${}_N\overset{p+1}{X} = {}_N\overset{N}{U}\ {}_N\overset{p+1}{\Sigma}\ {}_{(p+1)}\overset{(p+1)}{V^T} \qquad U, V \text{ orthogonal} \Rightarrow U^{-1} = U^T,\ V^{-1} = V^T$$

$$= \sum_{i=1}^{p+1} \sigma_i \underline{u}_i \underline{v}_i^T$$

$$X^+ = \sum_{i=1}^{p+1} \frac{\underline{v}_i \underline{u}_i^T}{\sigma_i} \qquad \underline{w} = \sum_{i=1}^{p+1} \left(\frac{\underline{u}_i^T \underline{z}^N}{\sigma_i} \right) \underline{v}_i \in R(X^T)$$

Can we process data sequentially?

Tests of Significance & Feature Selection

- Prediction at training inputs

$$\underline{\hat{z}} = X\underline{\hat{w}} = XX^{\dagger}\underline{z} = X(X^TX)^{-1}X^T\underline{z}$$

- Estimate of measurement noise variance

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{n=1}^{N}(z^n - \hat{z}^n)^2$$

- To test if $w_j = 0$, form Z-score

$$\beta_j = \frac{\hat{w}_j}{\hat{\sigma}\sqrt{diag\left[(X^TX)^{-1}\right]_{jj}}} \sim t(\beta_j; N-p-1)$$

$$If\ |\beta_j| < 2, consider\ dropping\ \ feature\ x_j.$$

- Two feature sets: $\{S\}$ and $\{S'\}$

$$F = \frac{\left[J(\underline{w}_S) - J(\underline{w}_{S'})\right]/\left(|S'| - |S|\right)}{J(\underline{w}_{S'})/(N - |S'| - 1)}$$

Add feature with the largest F and stop when F<95 percentile of $F_{1,(N-|S'|-1)}$

This is the error function (NOT expectation)

Variations : Shrinkage Methods

- Ridge (L_2) Regression

  $J(\underline{w}) = \| \underline{z} - X\underline{w} \|_2^2 + \mu \| \underline{w} \|_2^2$ .....*assume input and output data is centered*

  $$\Rightarrow \underline{\hat{w}} = \left(X^T X + \mu I_p \right)^{-1} X^T \underline{z} = X^T \left(X X^T + \mu I_N \right)^{-1} \underline{z} \text{ ....via Matrix Inversion Lemma}$$

  $$(or)\ \underline{\hat{w}} = X^T \alpha; \underline{\alpha} = \left(X X^T + \mu I_N \right)^{-1} \underline{z} \text{....} \textit{Kernel form} \Rightarrow \hat{z} = \underline{\hat{w}}^T \underline{x} = \sum_{i=1}^{N} \alpha_i \underline{x}_i^T \underline{x}$$

  $$so,\ \underline{\hat{z}} = \underbrace{X \left(X^T X + \mu I_p \right)^{-1} X^T}_{P_\mu} \underline{z} = \sum_{j=1}^{p} \left(\frac{\lambda_j^2}{\lambda_j^2 + \mu} \right) \underline{u}_j \underline{u}_j^T \underline{z}; \lambda_j^2 = \textit{eigen value of } X^T X$$

- L_1 Regression (Compressed Sensing, Lasso)

  $$J(\underline{w}) = \| \underline{z} - X\underline{w} \|_2^2 + \mu \| \underline{w} \|_1$$

  LASSO: least absolute shrinkage and selection operator

- Combined L_1 and L_2 Regression ("Elastic Nets"… related to SVM)

  https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9856/10002

  $$J(\underline{w}) = \| \underline{z} - X\underline{w} \|_2^2 + \mu_1 \| \underline{w} \|_1 + \mu_2 \| \underline{w} \|_2^2$$

- You can use reduced dimensioned features

  $$X_r = X V_r$$

  $\boxed{\textbf{\textit{PCA}}}$

  V_r = first r singular vectors corresponding to largest r singular values

LOOCV for Ridge Regression

- Key: Can compute $\hat{z}_{i,-i}$, prediction of z_i when trained on all data except i, from $\hat{z}_i$ and $H = (X^T X + \mu I)^{-1}$ or $P_\mu = X(X^T X + \mu I)^{-1} X^T$

$$\hat{z}_{-i} = \underline{x}_i^T \underline{\hat{w}}_{-i} = \underline{x}_i^T \left(X^T X + \mu I - \underline{x}_i \underline{x}_i^T \right)^{-1} (X^T \underline{z} - \underline{x}_i z_i)$$

$$= \underline{x}_i^T \left[H + \frac{H \underline{x}_i \underline{x}_i^T H}{1 - \underline{x}_i^T H \underline{x}_i} \right] (X^T \underline{z} - \underline{x}_i z_i); H = \left(X^T X + \mu I \right)^{-1}$$

$$= \underline{x}_i^T H X^T \underline{z} - \underline{x}_i^T H \underline{x}_i z_i + \frac{\underline{x}_i^T H \underline{x}_i \underline{x}_i^T H X^T \underline{z}}{1 - \underline{x}_i^T H \underline{x}_i} - \frac{\underline{x}_i^T H \underline{x}_i \underline{x}_i^T H \underline{x}_i z_i}{1 - \underline{x}_i^T H \underline{x}_i}$$

$$= (1 + \frac{\underline{x}_i^T H \underline{x}_i}{1 - \underline{x}_i^T H \underline{x}_i}) \left[\underline{x}_i^T \underbrace{H X^T \underline{z}}_{\underline{\hat{w}}} - \underline{x}_i^T H \underline{x}_i z_i \right]$$

$$= \frac{\hat{z}_i - \underline{x}_i^T H \underline{x}_i z_i}{1 - \underline{x}_i^T H \underline{x}_i} = \frac{\hat{z}_i - P_{\mu,ii} z_i}{1 - P_{\mu,ii}}$$

- Normalized Validation Error with LOOCV

$$J_{LOOCV} = \frac{1}{N} \sum_{i=1}^{N} (z_i - \hat{z}_{-i})^2 = \frac{1}{N} \sum_{i=1}^{N} \left(z_i - \frac{\hat{z}_i - \underline{x}_i^T H \underline{x}_i z_i}{1 - \underline{x}_i^T H \underline{x}_i} \right)^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left(\frac{z_i - \hat{z}_i}{1 - \underline{x}_i^T H \underline{x}_i} \right)^2 = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{z_i - \hat{z}_i}{1 - P_{\mu,ii}} \right)^2$$

It is rare to have such closed-form expressions for validation error

Incremental Gradient (ADALINE) method - 1

- **Incremental Gradient Method:**

Consider the gradient method

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \eta^k \underline{g}^{(k)}$$

Know $$\underline{g}^{(k)} = \sum_{n=1}^{N} \underline{g}^{n(k)} = -\sum_{n=1}^{N} \underbrace{(z^n - \underline{w}^{(k)T} \underline{x}^n)}_{\text{Error } E^n} \underline{x}^n = -\sum_{n=1}^{N} E_n \underline{x}^n$$

To go from k to $(k+1)$, what if I process the data sequentially as follows:

We use $\eta^{(k)}$ in place of $\alpha^{(k)}$ to be consistent with the literature.

Do until $\underline{w}^{(k)}$ converges

$$\underline{\psi}^{(1)} = \underline{w}^{(k)}$$

Do $n = 1, 2, \cdots\cdots, N$

$$\underline{\psi}^{(n+1)} = \underline{\psi}^{(n)} + \eta^{(k)} \underbrace{[z^n - \underline{\psi}^{(n)T} \underline{x}^n]}_{\text{Error}} \underline{x}^n$$

End Do

$$\underline{w}^{(k+1)} = \underline{\psi}^{(N+1)}$$

End Do

Incremental Gradient (ADALINE) method - 2

- This is precisely what happens in LMS, Recursive least squares (incremental Gauss-Newton) and back propagation algorithms. We can drop $\underline{\psi}$ and write weight update as

$$\underline{w}^{(n+1)} = \underline{w}^{(n)} + \eta^n E_n \underline{x}^n$$

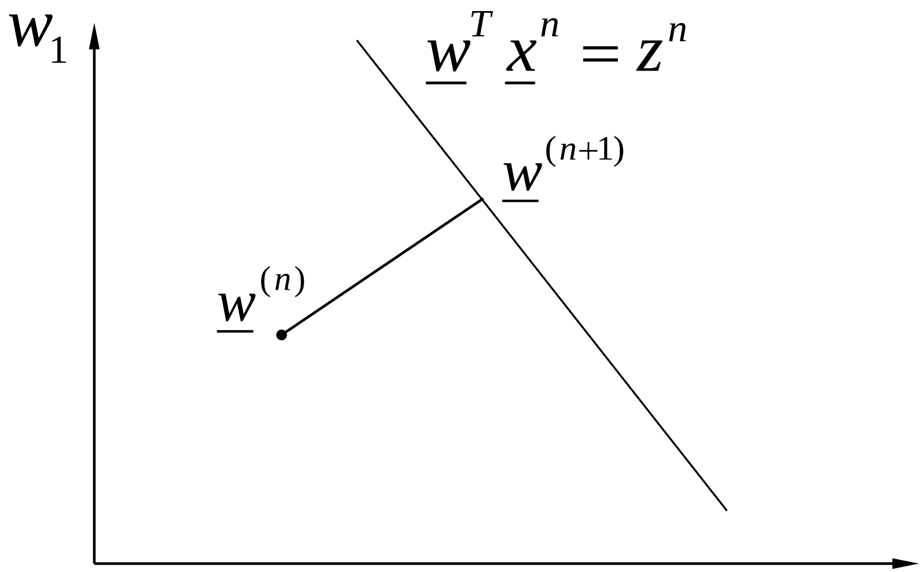
If $\eta^n = \dfrac{\lambda}{\left\|\underline{x}^n\right\|^2}$ where $0 < \lambda < 2$, the algorithm converges.

Typically $0 < \lambda \le 1$ is preferred. Indeed, when $\lambda = 1$, the update can be interpreted as a projection.

$$\min_{\underline{w}} \frac{1}{2}\left\|\underline{w} - \underline{w}^{(n)}\right\|^2 \qquad \text{s.t. } \underline{w}^T \underline{x}^n = z^n$$

$$L(\underline{w}, \mu) = \frac{1}{2}(\underline{w} - \underline{w}^{(n)})^T(\underline{w} - \underline{w}^{(n)}) + \mu\left[z^n - \underline{w}^T \underline{x}^n\right]$$

Incremental Gradient (ADALINE) method - 3



$$\nabla_{\underline{w}} L = \underline{w} - \underline{w}^{(n)} - \mu \underline{x}^n = 0 \qquad \text{using } \underline{w}^T \underline{x}^n = z^n$$

$$\Rightarrow \mu = \frac{z^n - \underline{w}^{(n)T} \underline{x}^n}{\left(\underline{x}^n\right)^T \underline{x}^n}$$

$$\Rightarrow w^{(n+1)} = w^{(n)} + \frac{(z^n - \underline{w}^{(n)T} \underline{x}^n)}{\left\|\underline{x}^n\right\|^2} \underline{x}^n$$

Stochastic Convergence Analysis - 1

$\underline{w}^n$ are random because data is random. So, we can talk about two types of convergence,

- *Convergence in the mean*

$$E[\underline{w}^{(n)}] \to \underline{w}^* \text{ as } n \to \infty \quad \text{provided } \eta \text{ is sufficiently small.}$$

- *Convergence in mean square*

$$E\left\{\left[z^n - \underline{w}^{(n)T}\underline{x}^n\right]^2\right\} \to \text{constant as } n \to \infty$$

- *Convergence in the mean*

$$E\left[\underline{w}^{(n+1)}\right] = E\left[\underline{w}^{(n)}\right] + \eta E[(z^n - \underline{w}^{(n)T}\underline{x}^n)\underline{x}^n]$$

$$\underline{\hat{w}}^{(n+1)} = \underline{\hat{w}}^{(n)} + \eta E\left[z^n \underline{x}^{(n)}\right] - \eta E[\underline{x}^{(n)}\underline{x}^{(n)T}]\underline{\hat{w}}^{(n)} \quad \text{(LMS assumption)}$$

$$\underline{\hat{w}}^{(n+1)} = [I - \eta \mathbf{R}_x]\underline{\hat{w}}^{(n)} + \eta \underline{r}_{z\underline{x}}$$

Stochastic Convergence Analysis - 4

- If it converges, $\mathbf{R}_x \underline{w}^* = \underline{r}_{zx}$

$R_x = E[\underline{x}\,\underline{x}^T]$ ~ Correlation matrix of data

$\underline{r}_{zx}$ ~ Cross-correlation vector between $\underline{x}^n$ and z^n

Since $\mathbf{R}_x$ is symmetric, $\exists$ an orthogonal matrix Q such that,

$$\mathbf{R}_x = Q \Lambda Q^T$$

Let $\underline{v}^{(n+1)} = \underline{w}^{(n+1)} - \underline{w}^*$

$$= [I - \eta R_x]\underline{v}^{(n)} = Q[I - \eta\Lambda]Q^T \underline{v}^{(n)}$$

$$\boxed{\begin{array}{l} -1 < 1 - \eta\lambda_k < 1 \forall k \\ UB: 0 < 2 - \eta\lambda_k \forall k \\ \Rightarrow \eta\lambda_k - 2 < 0 \forall k \\ \Rightarrow \eta < \frac{2}{\lambda_k} \forall k \Rightarrow \eta < \frac{2}{\lambda_{\max}} \\ LB: -\eta\lambda_k < 0 \Rightarrow \eta > 0 \because \lambda_k > 0 \forall k \end{array}}$$

Let $\underline{\tilde{v}}^{(n+1)} = Q^T \underline{v}^{(n+1)} \Rightarrow \underline{\tilde{v}}^{(n+1)} = [I - \eta\Lambda]\underline{\tilde{v}}^{(n)}$

Convergence in mean if $|1 - \eta\lambda_k| < 1 \ \forall\, k \Rightarrow 0 < \eta < \dfrac{2}{\lambda_{\max}(\mathbf{R}_x)}$

Heavy Ball Method

- Convergence in mean square if $0 < \eta < \dfrac{2}{Tr(\mathbf{R}_x)}$

$$Tr(\mathbf{R}_x) = \frac{1}{N}\sum_{n=1}^{N}\left(\underline{x}^n\right)^T \underline{x}^n = \text{ input power}$$

Ref: Haykin

Convergence in mean square $\Rightarrow$ Convergence in mean

- ***Alternate selection of*** η

❖ $$\eta^n = \frac{\eta_o}{1+\dfrac{n}{\tau}}$$

The incremental gradient algorithm converges provided $\sum_{n=1}^{N}\underline{x}^n\left(\underline{x}^n\right)^T$ is PD

❖ A popular modification to LMS is the so called heavy ball (or momentum) method

$$\underline{w}^{(n+1)} = \underline{w}^{(n)} + \eta(z^n - \underline{w}^{(n)T}\underline{x}^n)\underline{x}^n + \mu(\underline{w}^{(n)} - \underline{w}^{(n-1)})$$

Bold Driver Technique

$$\left.\begin{array}{l}\Delta \underline{w}^{(n)} = \mu \Delta \underline{w}^{(n-1)} + \eta e_n \underline{x}^n \\ \text{clearly need } \mu < 1\end{array}\right\}$$

$$\eta = \frac{1.5}{N}, \quad \mu = 1 - \frac{\eta}{3.5}$$

An alternative is to take μ to correspond to β of conjugate gradient

$$\mu = \frac{\underline{g}^{(n)T}\left[\underline{g}^{(n)} - \underline{g}^{(n-1)}\right]}{\underline{g}^{(n-1)T}\,\underline{g}^{(n-1)}} = \frac{e_n\left(\underline{x}^n\right)^T\left[e_n \underline{x}^n - e_{n-1}\underline{x}^{n-1}\right]}{e_{n-1}^2\left(\left(\underline{x}^{(n-1)}\right)^T \underline{x}^{(n-1)}\right)}$$

- Bold driver technique (adaptive η)

$$\eta^{new} = \begin{cases}\rho\ \eta^{old} & \text{if } \Delta J < 0 \Rightarrow \text{improved} \\ \sigma\ \eta^{old} & \text{if } \Delta J > 0\end{cases}$$

$$\rho = 1.1 \quad \sigma \approx 0.5$$

Quickprop Method

❖ Quickprop (Fahlman)

$$w_i^{(n+1)} - w_i^{(n)} = \frac{g_i^{(n)}}{g_i^{(n-1)} - g_i^{(n)}}\left[w_i^{(n)} - w_i^{(n-1)}\right]$$

$$\boxed{\Delta w_i^{(n+1)} = \frac{g_i^{(n)}}{g_i^{(n-1)} - g_i^{(n)}}\Delta w_i^{(n)}}$$

Problem when gradients are nearly equal!

$$\mathit{Parabola}: a w_i^2 + b w_i + c$$

$$\min\ at\ w_i = -\frac{b}{2a}$$

$$2a w_i^{(n-1)} + b = g_i^{(n-1)}$$

$$2a w_i^{(n)} + b = g_i^{(n)} \quad \ldots\ldots (1)$$

$$\Rightarrow 2a = \frac{g_i^{(n)} - g_i^{(n-1)}}{w_i^{(n)} - w_i^{(n-1)}}$$

$$\Rightarrow w_i^{(n+1)} = -\frac{b}{2a} = w_i^{(n)} - \frac{g_i^{(n)}}{2a} \ldots \text{ from (1)}$$

$$\Rightarrow w_i^{(n+1)} = w_i^{(n)} + \frac{g_i^{(n)}}{g_i^{(n-1)} - g_i^{(n)}}\left(w_i^{(n)} - w_i^{(n-1)}\right)$$

$$\Rightarrow w_i^{(n+1)} = \frac{g_i^{(n-1)} w_i^{(n)} - g_i^{(n)} w_i^{(n-1)}}{g_i^{(n-1)} - g_i^{(n)}}$$

- **Extension to a single layer Neural Network**

$$J(\underline{w}) = \frac{1}{2}\sum_{n=1}^{N}(z^n - \hat{z}^n)^2 = \frac{1}{2}\sum_{n=1}^{N} e_n^2$$

$$\hat{z}^n = g[y(\underline{w}, \underline{x}^n)] = g(\underline{w}^T \underline{x}^n) = \frac{1}{1 + e^{-\underline{w}^T \underline{x}^n}}$$

Single Layer Network

$$\nabla J(\underline{w}) = -\sum_{n=1}^{N} (z^n - \hat{z}^n)\nabla \hat{z}^n$$

$$= -\sum_{n=1}^{N} (z^n - \hat{z}^n)\hat{z}^n(1-\hat{z}^n)\underline{x}^n;\ \hat{z}^n = g^n$$

$$= -\sum_{n=1}^{N} e_n \hat{z}^n (1-\hat{z}^n)\underline{x}^n \qquad e_n = z^n - g(\underline{w}^T \underline{x}^n)$$

$$= -\sum_{n=1}^{N} e_n \cdot \underbrace{g'}_{\substack{\text{local gradient} \\ \text{of neuron}}} \underline{x}^n$$

To avoid saturation: replace g' by $\max(g',0.1)$ or $(g'+0.1)$ ~ Fahlman

Saturation resistant activation:

$$g(a) = \frac{\alpha a}{1+|a|} \qquad \text{with } g'(a) = \frac{\alpha}{(1+|a|)^2} = \frac{1}{\alpha a^2}[g(a)]^2$$

Incremental Newton Method

- Incremental Newton ≡ Incremental Gauss Newton ≡ RLS ≡ EKF

$$\nabla^2 J(\underline{w}) = \sum_{n=1}^{N} \underline{x}^n \left(\underline{x}^n\right)^T \qquad \nabla J(\underline{w}) = -\sum_{n=1}^{N} e_n \underline{x}^n$$

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} + \left[\sum_{n=1}^{N} \underline{x}^n \left(\underline{x}^n\right)^T\right]^{-1} \sum_{n=1}^{N} [z^n - \underline{w}^{(k)^T} \underline{x}^n]\underline{x}^n$$

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} + \left[\sum_{n=1}^{N} \underline{x}^n \left(\underline{x}^n\right)^T\right]^{-1} [\sum_{n=1}^{N} z^n \underline{x}^n] - \underline{w}^{(k)} = \underline{w}^*$$

Needs only one iteration to converge.

Can also update weights recursively, let $\left[\sum_{i=1}^{n} \underline{x}^i \left(\underline{x}^i\right)^T\right]^{-1} = \Sigma^{(n)}$

$$\Sigma^{(n)-1} = \Sigma^{(n-1)-1} + \underline{x}^n \left(\underline{x}^n\right)^T \qquad \Sigma^{(n)} = \Sigma^{(n-1)} - \frac{\Sigma^{(n-1)} \underline{x}^n \left(\underline{x}^n\right)^T \Sigma^{(n-1)}}{1 + \left(\underline{x}^n\right)^T \Sigma^{(n-1)} \underline{x}^n}$$

Recursive Least Squares

- RLS

Initialize $\underline{w}^0$, $\Sigma^0 = 10^6 I$...... large value

Do $n = 1,2,\cdots\cdots,N$

$$\underline{k}^n = \frac{\Sigma^{(n-1)}\underline{x}^n}{1+\left(\underline{x}^n\right)^T \Sigma^{(n-1)}\underline{x}^n}$$

$$\underline{w}^{(n)} = \underline{w}^{(n-1)} + \underline{k}^n[z^n - \left(\underline{x}^n\right)^T \underline{w}^{(n-1)}]$$

$$\Sigma^{(n)} = \Sigma^{(n-1)} - \underline{k}^n\left(\underline{x}^n\right)^T \Sigma^{(n-1)}$$

*

End Do

$$\underline{w}^* = \underline{w}^{(N)}$$

Fading memory filter

Note: If want to put more emphasis on recent data, set

$$\Sigma^n \leftarrow \frac{\Sigma^n}{\lambda} \quad 0 < \lambda \le 1 \quad \text{at } *$$

Linearization of Neuron

➢ *Extension to Neurons*

$$e_n = z^n - g(\underline{w}^T \underline{x}^n)$$

Have $\underline{w}^{(n-1)}$ when we process $\underline{x}^n$

$$\text{So, } e_n \approx z^n - g\left(\left(\underline{w}^{(n-1)}\right)^T \underline{x}^n\right) - g'.\left(\underline{x}^n\right)^T (\underline{w} - \underline{w}^{(n-1)})$$

$$= \underbrace{[z^n - \underbrace{g\left(\left(\underline{w}^{(n-1)}\right)^T \underline{x}^n\right)}_{\hat{z}^n} + g'.\left(\underline{x}^n\right)^T \underline{w}^{(n-1)}]}_{\tilde{z}^n} - \underbrace{g'.\left(\underline{x}^n\right)^T}_{\left(\tilde{\underline{x}}^n\right)^T} \underline{w}$$

$$= \tilde{z}^n - \left(\tilde{\underline{x}}^n\right)^T \underline{w}$$

For logistic: $\tilde{\underline{x}}^n = g(\underline{w}^{(n-1)T} \underline{x}^n)[1 - g(\underline{w}^{(n-1)T} \underline{x}^n)]\underline{x}^n = \hat{z}^n(1 - \hat{z}^n)\underline{x}^n$

➢ *Modified RLS = EKF*

Initialize $\underline{w}$, Σ

Extended Kalman Filter (EKF)

Do until $\underline{w}$ converges

Do $n = 1, 2, \cdots\cdots, N$

Compute $\tilde{z}^n$ and $\underline{\tilde{x}}^n$ (or) $\hat{z}^n$ and and $\underline{\tilde{x}}^n$ at current $\underline{w}$

$$\underline{k} \leftarrow \frac{\Sigma \underline{\tilde{x}}^n}{1 + \left(\underline{\tilde{x}}^n\right)^T \Sigma \, \underline{\tilde{x}}^n}$$

$$\underline{w} \leftarrow \underline{w} + \underline{k}[\tilde{z}^n - \left(\underline{\tilde{x}}^n\right)^T \underline{w}] = \underline{w} + \underline{k}[z^n - \hat{z}^n]$$

$$\Sigma \leftarrow \Sigma - \underline{k}\left(\underline{\tilde{x}}^n\right)^T \Sigma$$

$$\Sigma \leftarrow \Sigma / \lambda$$

End Do

End Do

Fisher's Linear Discriminant

- Recall Total Covariance, S_T (HW problem)

$$S_T = S_W + S_B$$

$$S_T = \sum_{n=1}^{N} (\underline{x}^n - \underline{\mu})(\underline{x}^n - \underline{\mu})^T; \underline{\mu} = \frac{1}{N}\sum_{n=1}^{N} \underline{x}^n$$

$$S_W = \sum_{k=1}^{C} S_k; S_k = \sum_{\substack{n=1 \\ n:z^n=k}}^{N} (\underline{x}^n - \underline{\mu}_k)(\underline{x}^n - \underline{\mu}_k)^T; \underline{\mu}_k = \frac{\sum_{\substack{n=1 \\ n:z^n=k}}^{N} \underline{x}^n}{n_k}$$

$$n_k = \sum_{n=1}^{N} \delta_{z^n k}; \delta_{z^n k} = Kronecker\ delta\ function$$

$$S_B = \sum_{k=1}^{C} n_k (\underline{\mu}_k - \underline{\mu})(\underline{\mu}_k - \underline{\mu})^T$$

PCA versus LDA

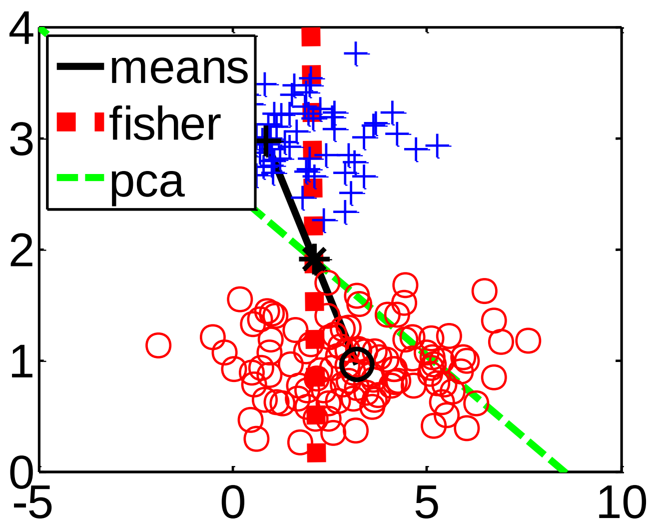
PCA: Dimensionality reduction while preserving as much of the variance in the high dimensional space as possible.

LDA: Dimensionality reduction while preserving as much of the class discriminatory information as possible.

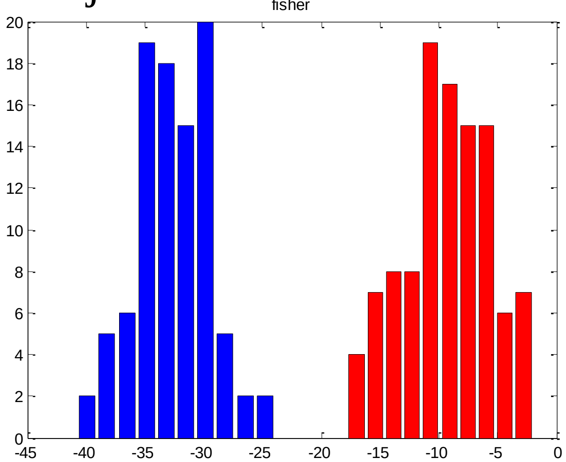
$Find\ discriminant\ functions\ \underline{g} = W^T \underline{x} \ni tr\{(WS_W W^T)^{-1}(WS_B W^T)\}$ is maximum

$\Rightarrow$ Normalized Eigen vectors of $(S_W^{-1} S_B)$ corresponding to $(C$-1) largest eigen values

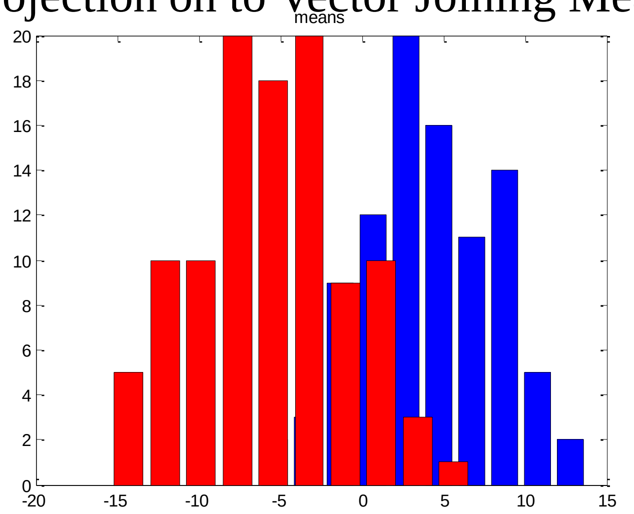
Fisher's Linear Discriminant



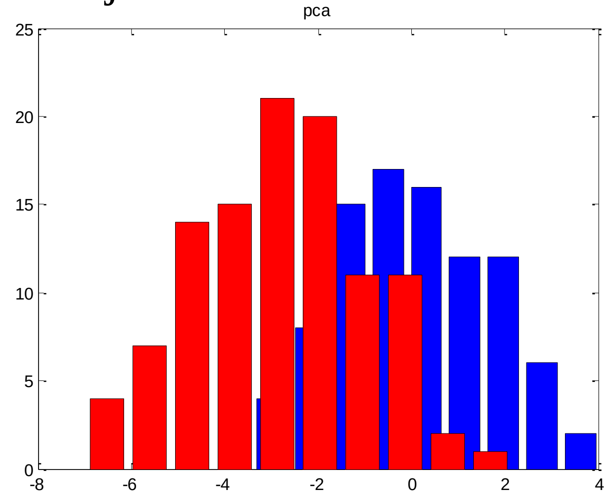
Projection on to Fisher Vector



Projection on to Vector Joining Means

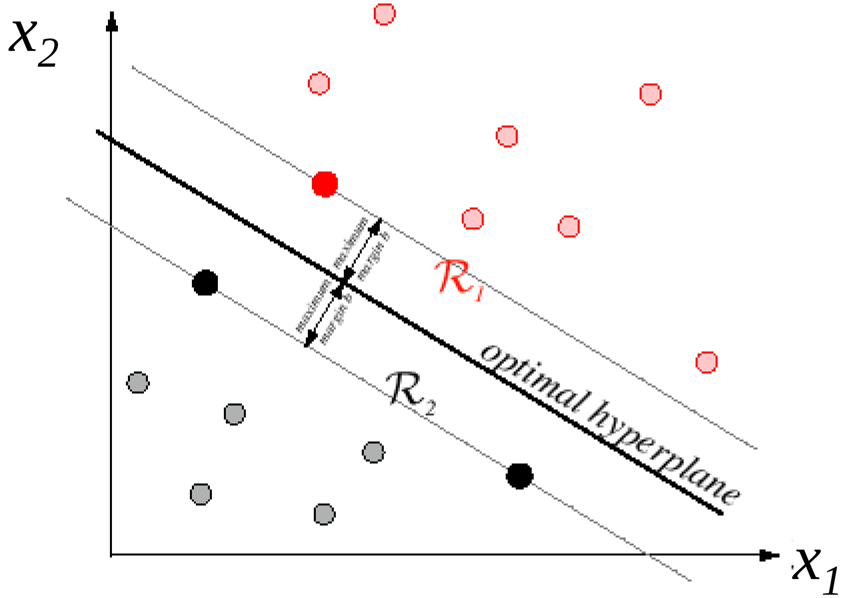


Projection on to PCA Vector



Support vector Machines

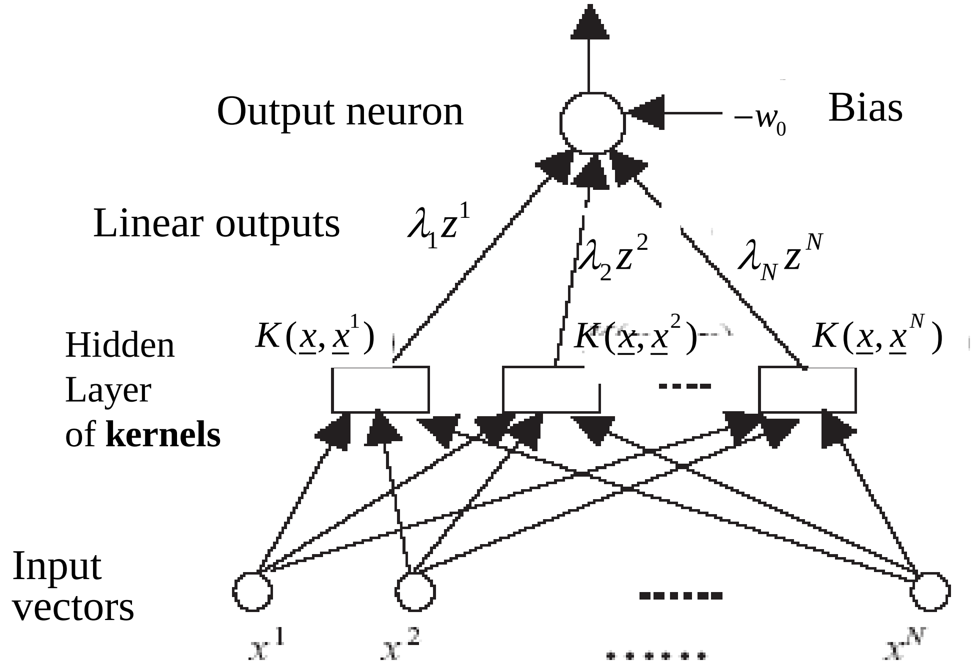
- ***Can we find a hyperplane with the largest separation (margin) between two classes?***



SVM formulates the problem of finding the largest margin as a quadratic programming problem. It maximizes the distance from the nearest training patterns. *Excellent Method*.

SVM : Key Idea

- Idea: **Nonlinearly transforms data** into a higher dimensional feature space such that the classes are linearly separable and **finds an optimal hyperplane** separating each pair of classes in the new space



Kernels allow you to transform data for linear separability. Kernels exploit inner product between data points.

$K = [K(\underline{x}^i, \underline{x}^j)]$ *is a Kernel if* $K \geq 0$

Mercer's theorem

SVM: Maximum Margin Classification

- ***Recall the problem in the $\underline{x}$ space***

  separating hyperplane: $\underline{w}^T \underline{x} - w_0 = 0$

Projection Problem:

$$\min_{\underline{x}_p} \| \underline{x} - \underline{x}_p \|_2^2$$

$$s.t. \; \underline{w}^T \underline{x}_p - w_0 = 0$$

$\dfrac{|w_0|}{\|\underline{w}\|}$ is the perpendicular distance from the hyperplane to the origin

$$Need: \quad \underline{w}^T \underline{x}^i - w_0 \geq 1 \; for \; class \; 1 \Rightarrow z^i = 1$$

$$\underline{w}^T \underline{x}^i - w_0 \leq -1 \; for \; class \; 2 \Rightarrow z^i = -1$$

Can always scale $\underline{w}$ to satisfy these.

$$\Rightarrow \quad z^i(\underline{w}^T \underline{x}^i - w_0) - 1 \geq 0 \quad \forall i$$

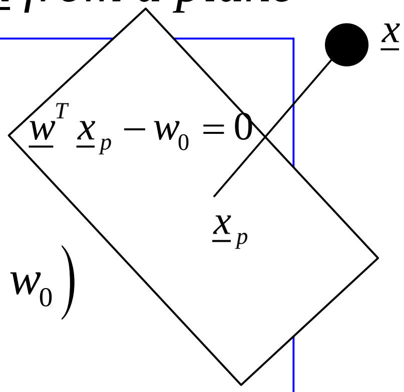
Note: z^i is a class label here

$$\text{Key: distance between } \underline{w}^T \underline{x} - w_0 = 1 \text{ and } \underline{w}^T \underline{x} - w_0 = -1 \text{ is } \frac{2}{\|\underline{w}\|}$$

$$\text{so, why not maximize } \frac{2}{\|\underline{w}\|} \quad \textit{or} \text{ minimize } \|\underline{w}\|?$$

Projection Problem

- *Projection Problem: Minimum Distance of a Point $\underline{x}$ from a plane*



$$\min_{\underline{x}_p} \| \underline{x} - \underline{x}_p \|_2^2$$

$$s.t.\ \underline{w}^T \underline{x}_p - w_0 = 0$$

$$\textbf{Lagrangian: } L(\underline{x}_p, \lambda) = \left(\underline{x} - \underline{x}_p\right)^T \left(\underline{x} - \underline{x}_p\right) + \lambda\left(\underline{w}^T \underline{x}_p - w_0\right)$$

$$\nabla_{\underline{x}_p} L = \underline{0} \Rightarrow -2\left(\underline{x} - \underline{x}_p\right) + \lambda \underline{w} = \underline{0} \Rightarrow \underline{x}_p = \underline{x} - \frac{\lambda}{2}\underline{w}$$

$$\nabla_{\lambda} L = 0 \Rightarrow \underline{w}^T \underline{x}_p - w_0 = 0 \ \Rightarrow \underline{w}^T \underline{x} - \frac{\lambda}{2}\underline{w}^T \underline{w} = w_0 \Rightarrow \lambda = 2\left(\frac{\underline{w}^T \underline{x} - w_0}{\underline{w}^T \underline{w}}\right)$$

$$so,\ \underline{x}_p = \underline{x} - \left(\frac{\underline{w}^T \underline{x} - w_0}{\underline{w}^T \underline{w}}\right)\underline{w}$$

$$\| \underline{x} - \underline{x}_p \|_2 = \left(\frac{| \underline{w}^T \underline{x} - w_0 |}{\underline{w}^T \underline{w}}\right) \| \underline{w} \|_2$$

$$\underline{x} = \underline{0} \Rightarrow \| \underline{x}_p \|_2 = \frac{| w_0 |}{\| \underline{w} \|_2}$$

Distance of the plane from the origin

Quadratic Programming (QP) Problem

- ***QP Problem***

$$\min_{\underline{w}} \quad \frac{1}{2}\underline{w}^T\underline{w}$$

$$z^i(\underline{w}^T\underline{x}^i - w_0) - 1 \geq 0 \quad \forall i$$

$$\max_{\underline{w}, w_0} \min\{\|\underline{x} - \underline{x}_i\| : \underline{x} \in R^p,$$
$$z^i\left(\underline{w}^T\underline{x} - w_0\right) \geq 1, i = 1, 2, .., N\}$$

- Lagrangian

$$L(\underline{w}, \underline{\lambda}) = \frac{1}{2}\underline{w}^T\underline{w} - \sum_{i=1}^{N}\lambda_i[z^i(\underline{w}^T\underline{x}^i - w_0) - 1]$$

$$= \frac{1}{2}\underline{w}^T\underline{w} - \sum_{i=1}^{N}\lambda_i z^i(\underline{w}^T\underline{x}^i - w_0) + \sum_{i=1}^{N}\lambda_i$$

Optimal solution for given $\{\lambda_i\}$ satisfies:

$$\underline{w}^*(\underline{\lambda}) = \sum_{i=1}^{N}\lambda_i z^i\underline{x}^i \quad \textbf{and} \quad \sum_{i=1}^{N}\lambda_i z^i = 0$$

Dual QP problem

- ***Dual QP Problem is one of maximizing:***

$$q(\underline{\lambda}) = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z^i z^j \left(\underline{x}^i\right)^T \underline{x}^j$$

$$\textit{subject to}: \sum_{i=1}^{N}\lambda_i z^i = 0 \textbf{ and } \lambda_i \geq 0$$

$$\max_{\underline{\lambda}\geq \underline{0}} \quad \underline{\lambda}^T \underline{e} - \frac{1}{2}\underline{\lambda}^T V^T V \underline{\lambda}; V = [z^1\underline{x}^1, z^2\underline{x}^2, \ldots, z^N\underline{x}^N]$$

$$\textit{subject to}: \underline{\lambda}^T \underline{z} = 0$$

Concave QP Problem

- In the solution, those points for which $\lambda_i > 0$ are called *support vectors* (primal constraints are active). Support vectors are critical elements of the training set. They lie closest to the decision boundary!!
- Can transform $\underline{x}$ into $K(\underline{x})$: Gaussian RBF, Polynomial, MLP, etc. are used as Kernels. **Kernels exploit inner products between data points**

$$\Rightarrow \textit{replace} \left(\underline{x}^i\right)^T \underline{x}^j \textit{ by } K(\underline{x}^i, \underline{x}^j) \textit{ in the dual.}$$

$$Ex: K(\underline{x}^i, \underline{x}^j) = e^{-\left\|\underline{x}^i - \underline{x}^j\right\|_2^2 / 2\sigma^2}; (\left(\underline{x}^i\right)^T \underline{x}^j + 1)^d; \tanh(\gamma \left(\underline{x}^i\right)^T \underline{x}^j + r)$$

$$\textit{They all satisfy } K = [K(\underline{x}^i, \underline{x}^j)] \geq 0 \ (\textit{Mercer's Theorem})$$

Examples of Kernels -1

$$1. K(\underline{x},\underline{u}) = (\underline{x}^T\underline{u}+c)^2 = \Phi^T(\underline{x})\Phi(\underline{u})$$

$$Example: p=2: (x_1u_1+x_2u_2+c)^2$$

$$=[x_1^2\ \sqrt{2}x_1x_2\ \sqrt{2c}x_1\ \sqrt{2c}x_2\ x_2^2\ c][u_1^2\ \sqrt{2}u_1u_2\ \sqrt{2c}u_1\ \sqrt{2c}u_2\ u_2^2\ c]^T$$

$$\Rightarrow \Phi(\underline{x}) = [x_1^2\ \sqrt{2}x_1x_2\ \sqrt{2c}x_1\ \sqrt{2c}x_2\ x_2^2\ c]\ for\ \ p=2$$

$$2.\ K(\underline{x},\underline{u}) = \exp\left(-\frac{\|\underline{x}-\underline{u}\|_2^2}{2\sigma^2}\right) = \exp\left(-\frac{\underline{x}^T\underline{x}}{2\sigma^2}\right)\exp\left(\frac{\underline{x}^T\underline{u}}{\sigma^2}\right)\exp\left(-\frac{\underline{u}^T\underline{u}}{2\sigma^2}\right)$$

$\underline{x}^T\underline{x}, \underline{x}^T\underline{u}, \underline{u}^T\underline{u}$ *are kernels*

Exponential transformations of Kernels are Kernels.

$$3.\ K(\underline{x},\underline{u}) = \exp\left(-\frac{1}{2}(\underline{x}-\underline{u})^T\Sigma^{-1}(\underline{x}-\underline{u})\right)$$

If Σ *is diagonal,*

$$K(\underline{x},\underline{u}) = \exp\left(-\frac{1}{2}\sum_{i=1}^{p}\frac{(x_i-u_i)^2}{\sigma_i^2}\right);\ \sigma_i =$$ **characteristic length scale of dimension** i

Examples of Kernels -2

4. $K(\underline{x},\underline{u}) = \sum_{i=1}^{m} P(i)\, p(\underline{x}\,|\,i)\, p(\underline{u}\,|\,i)$ .... like SVD and outerproduct representation

5. $K(\underline{x},\underline{u}) = \int_{\underline{\theta}} p(\underline{x}\,|\,\underline{\theta})\, p(\underline{u}\,|\,\underline{\theta}) p(\underline{\theta}) d\underline{\theta}$

6. $K(\underline{x},\underline{u}) = [\nabla_{\underline{\theta}} \ln p(\underline{x}\,|\,\underline{\theta})][\nabla_{\underline{\theta}} \ln p(\underline{u}\,|\,\underline{\theta})]^T$ ...Fisher kernel

7. *Kernels for comparing documents* (cos*ine similarity*)

$$K(\underline{x},\underline{u}) = \frac{\underline{x}^T \underline{u}}{\|\underline{x}\|_2 \|\underline{u}\|_2}$$

8. *Term frequency inverse document frequency* (*tf* $-$ *idf*)

x_{ij} = *number of times word j occurs in document i*

$$tf\,(x_{ij}) \triangleq \ln(1+x_{ij});\, idf\,(j) = \frac{N}{1+\sum_{i=1}^{N} I(x_{ij}>0)};\, N = \text{number of documents}$$

$$\Phi(\underline{x}_i) \triangleq [tf\,(x_{ij}) \text{x}\, idf\,(j)]_{j=1}^{w};\, w = \text{number of words}$$

$$K(\underline{x}_i,\underline{x}_k) = \frac{\Phi^T(\underline{x}_i)\Phi(\underline{x}_k)}{\|\Phi(\underline{x}_i)\|_2 \|\Phi(\underline{x}_k)\|_2}$$

9. *String* ker*nels*: $K(\underline{x},\underline{u}) = \sum_{s} w_s \phi_s(\underline{x}) \phi_s(\underline{u});\, s = \text{substring},\, w_s = \text{weight of } s$

$\phi_s(\underline{x}) =$ number of times substring s occurs in $\underline{x}$

SVM for Nonseparable Case

- ***Change conditions to***

Need: $\underline{w}^T \underline{x}^i - w_0 \geq 1 - \alpha_i$ **for class 1** $\Rightarrow z^i = 1$

$\underline{w}^T \underline{x}^i - w_0 \leq -1 + \alpha_i$ **for class 2** $\Rightarrow z^i = -1$

$$\Rightarrow \quad z^i(\underline{w}^T \underline{x}^i - w_0) - 1 + \alpha_i \geq 0 \quad \forall i \Rightarrow z^i(\underline{w}^T \underline{x}^i - w_0) \geq 1 - \alpha_i$$

- QP problem (C-SVM or *soft margin classifier*)

$$\min_{\underline{w}} \quad \frac{1}{2}\underline{w}^T \underline{w} + C\sum_{i=1}^{N}\alpha_i$$

$$z^i(\underline{w}^T \underline{x}^i - w_0) - 1 + \alpha_i \geq 0 \quad \textit{and } \alpha_i \geq 0 \quad \forall i$$

- Dual QP problem

$$\max_{\underline{\lambda}} q(\underline{\lambda}) = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z^i z^j \underline{x}^{iT} \underline{x}^j$$

$$= \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\|V\underline{\lambda}\|_2^2; V = [z^1\underline{x}^1, z^2\underline{x}^2, \ldots, z^N\underline{x}^N]$$

$$\textit{subject to}: \sum_{i=1}^{N}\lambda_i z^i = 0 \textbf{ and } 0 \leq \lambda_i \leq C$$

SVM and Hinge Loss Function

$$Let\ y_i = \underline{w}^T \Phi(\underline{x}^i) - w_0$$

$$z^i y_i \geq 1 \Rightarrow \alpha_i = 0$$

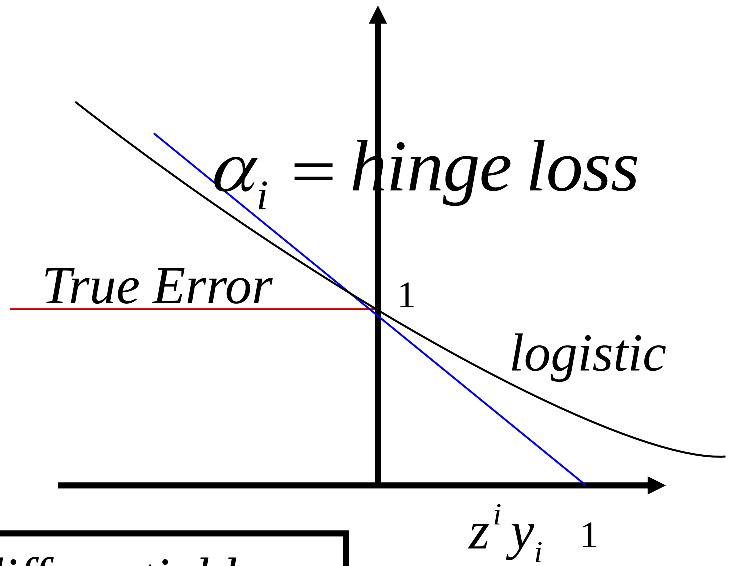
$$z^i y_i < 1 \Rightarrow \alpha_i = 1 - z^i y_i$$

$$\Rightarrow \alpha_i = [1 - z^i y_i]^+ = \max(0., 1 - z^i y_i)$$

$$So, C - SVM\ Cost:$$

$$\frac{\beta}{2} \| \underline{w} \|_2^2 + C \sum_{i=1}^{N} [1 - z^i (\underline{w}^T \Phi(\underline{x}^i) - w_0)]^+$$

β **regularization weight**



Non-differentiable

$$Logistic\ (differentiable)\ approximation:$$

$$[1 - z^i y_i]^+ \approx \ln(1 + \exp(-z^i y_i))] / \ln 2$$

$$So, Logistic - \mathrm{Re}gression\ \ Cost:$$

$$\frac{\beta}{2} \| \underline{w} \|_2^2 + C \sum_{i=1}^{N} \ln(1 + \exp(-z^i y_i))] / \ln 2$$

The piecewise linear and flat nature of SVM cost function leads to sparse solutions

SVM and Subgradient Method

QP-based SVM is very slow for large data. Stochastic gradient methods are used for "big data" SVM.

$$C-SVM\ Cost: J(\underline{\tilde{w}}) = (\frac{\beta}{2}\|\underline{\tilde{w}}\|_2^2 + \frac{1}{N}\sum_{i=1}^{N}[1-z^i\underline{\tilde{w}}^T\underline{\tilde{\Phi}}(\underline{x}^i)]^+;\ \beta \text{ regularization weight}; \underline{\tilde{w}} = \begin{bmatrix}\underline{w}\\ w_0\end{bmatrix}; \underline{\tilde{\Phi}}(\underline{x}^i) = \begin{bmatrix}\underline{\Phi}(\underline{x}^i)\\ -1\end{bmatrix}$$

Piece-wise linear + Quadratic $\Rightarrow$ convex (not strongly, though) $\Rightarrow$ unique minimum

(see http://machinelearning.wustl.edu/mlpapers/paper_files/icml2007_Shalev-ShwartzSS07.pdf)

$$\text{Subgradient: } \partial J(\underline{\tilde{w}}) = \begin{cases}\beta\underline{\tilde{w}} - z^i\underline{\tilde{\Phi}}(\underline{x}^i) \text{ if } z^i\underline{\tilde{w}}^T\underline{\tilde{\Phi}}(\underline{x}^i) < 1\\ \beta\underline{w};\text{otherwise}\end{cases};$$

Pegasos Algorithm:

Inputs: Training data, D; Regularization weight, β; Maximum number of Iteration, $T_{\max}$; Batch size, b (≈ 128)

Initialize: Choose $\underline{\tilde{w}}_1 \ni \|\underline{\tilde{w}}_1\|_2 \le 1/\sqrt{\beta}$ .... why? optimal solution has this property

For t=1,2,..,T

Choose a subset $A_t \subseteq D$ such that $|A_t| = b$

Misclassified samples in A_t : $A_t^+ = \{(\underline{x}^i, z^i) \in A_t : z^i\underline{\tilde{w}}^T\underline{\tilde{\Phi}}(\underline{x}^i) < 1\}$

Set $\eta_t = \dfrac{1}{\beta t}$

Set $\underline{u} = (1-\eta_t\beta)\underline{\tilde{w}}_t + \dfrac{\eta_t}{b}\sum_{(\underline{x}^i, z^i)\in A_t^+} z^i\underline{\tilde{\Phi}}(\underline{x}^i)$

$\underline{\tilde{w}}_{t+1} = \min\{1, \dfrac{1/\sqrt{\beta}}{\|\underline{u}\|_2}\}\underline{u}$

end

Output: $\underline{\tilde{w}}_{T+1}$

There are other ways of handling the bias w_0

SVM for Nonseparable Case

- **ν-SVM**

  QP Problem: Replace parameter C by $\nu \in [0,\ 1]$

  ν=lower bound on the number of support vectors

$$\min_{\underline{w}, w_0, \underline{\alpha} \geq \underline{0}, \rho \geq 0} \frac{1}{2} \|\underline{w}\|_2^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\alpha_i$$

$$s.t.\ \ z^i(\underline{w}^T\Phi(\underline{x}^i) - w_0) \geq \rho - \alpha_i$$

❖ QP problem and its dual

$$L(\underline{w}, w_0, \underline{\alpha}, \rho, \underline{\lambda}) = \frac{1}{2}\|\underline{w}\|_2^2 - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}[\lambda_i(z^i(\underline{w}^T\Phi(\underline{x}^i) - w_0) - \rho + \alpha_i) + \beta_i\alpha_i] - \delta\rho$$

$$Dual: q(\underline{\lambda}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j z^i z^j K(\underline{x}^i, \underline{x}^j)$$

$$subject\ to: \sum_{i=1}^{N}\lambda_i z^i = 0 \textbf{ and } 0 \leq \lambda_i \leq \frac{1}{N}; \sum_{i=1}^{N}\lambda_i \geq \nu$$

$$K(\underline{x}^i, \underline{x}^j) = \Phi(\underline{x}^i)^T\Phi(\underline{x}^j)$$

$$\Rightarrow \underline{w} = \sum_{i=1}^{N}\lambda_i z^i\Phi(\underline{x}^i); \lambda_i + \beta_i = \frac{1}{N}; \sum_{i=1}^{N}\lambda_i z^i = 0; \sum_{i=1}^{N}\lambda_i - \delta = \nu$$

$$\nu \in [0,1] \Rightarrow easy\ to\ experiment$$

$$C = \frac{1}{N\rho}$$

SVM and Elastic Net

- Elastic Net is related to dual QP problem of SVM with squared hinge loss

$J(\underline{w}) = \frac{1}{2}\| \underline{z} - X\underline{w}\|_2^2 + \mu_2 \|\underline{w}\|_2^2; \|\underline{w}\|_1 \le 1$ (can always assume 1 by scaling $\underline{w}$ and $\underline{z}$); X is nxp (1)

$\underline{w} = \underline{w}^+ - \underline{w}^-; \underline{w}^+ \ge 0; \underline{w}^- \ge 0;\ w_i^+ w_i^- = 0 \Rightarrow$ only one of the components is non-zero.

No loss in generality in assuming $\|\underline{w}\|_1 = \|\underline{w}^+\|_1 + \|\underline{w}^-\|_1 = 1 \Rightarrow \underline{e}^T \hat{\underline{w}} = 1; \hat{\underline{w}} = \begin{bmatrix} \underline{w}^+ \\ \underline{w}^- \end{bmatrix} \ge \underline{0}; 2p$ vector

Noting that $\underline{z} = \underline{z}\underline{e}^T \hat{\underline{w}}$, then (1) is: $J(\hat{\underline{w}}) = \frac{1}{2}\|V\hat{\underline{w}}\|_2^2 + \mu_2 \|\hat{\underline{w}}\|_2^2; \underline{e}^T \hat{\underline{w}} = 1; \hat{\underline{w}} \ge \underline{0}; V = [\underline{z}\underline{e}^T - X, (\underline{z}\underline{e}^T + X)]$

- SVM with *squared* hinge loss

minimum versus maximum

$\hat{w}_{i,elastic\,net} = \frac{\lambda_{i,SVM}}{\|\lambda\|_1}$

Primal (no bias): $\underline{\beta} \in R^n$; Samples $N=2p$ Dual:

$\min_{\underline{\beta}} J(\underline{\beta}) = \frac{1}{2}\underline{\beta}^T\underline{\beta} + C\sum_{i=1}^{2p}\alpha_i^2$

$\max_{\underline{\lambda} \ge \underline{0}} \sum_{i=1}^{2p}\lambda_i - \frac{1}{2}\|V\underline{\lambda}\|_2^2 - \frac{1}{4C}\underline{\lambda}^T\underline{\lambda}; V = [z^1\underline{x}^1, z^2\underline{x}^2, \ldots, z^{2p}\underline{x}^{2p}]$

$z^i \underline{\beta}^T \underline{x}^i - 1 + \alpha_i \ge 0$ *and* $\alpha_i \ge 0\ \forall i = 1,2,..,2p$

- SVM algorithms (e.g., CG + Newton) can be used to solve Elastic Net Problem. Newton steps can be parallelized. *Liblinear* is an algorithm specifically suited for this. Two orders of magnitude faster.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008

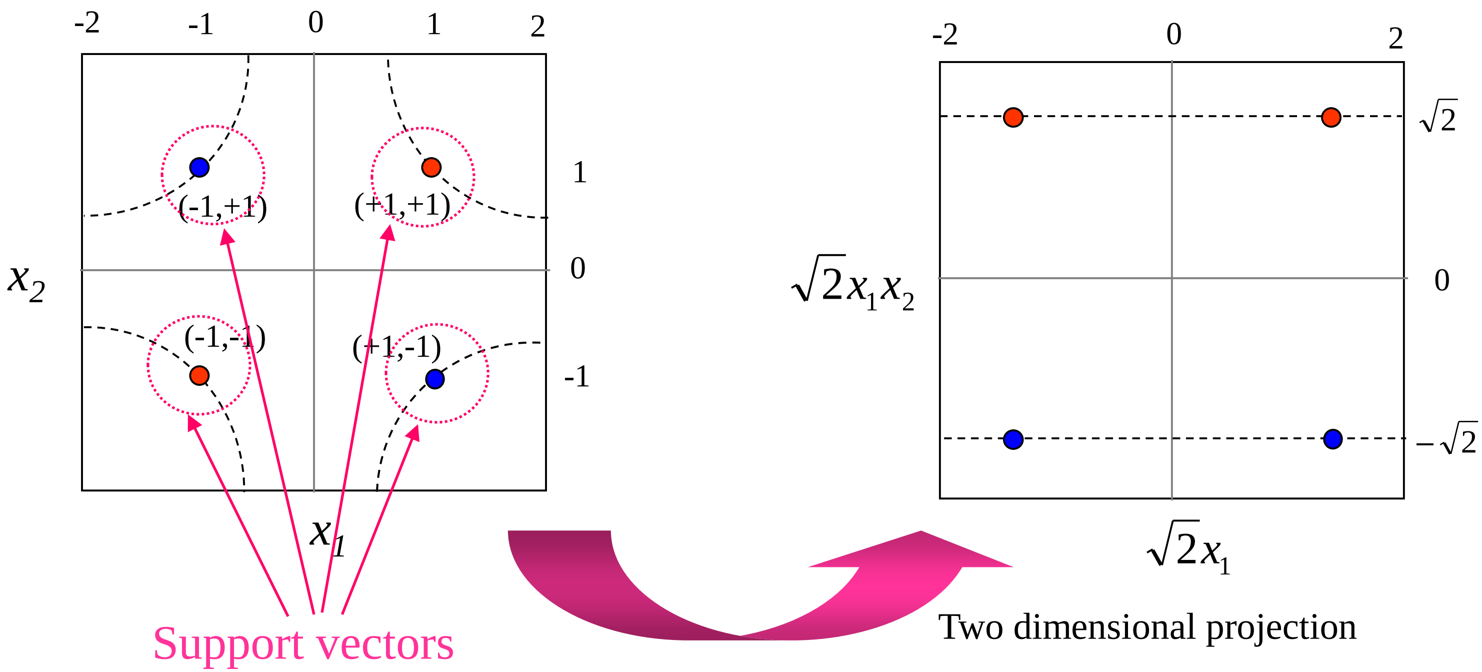
Open Source SVM Packages

- **RHUL-SVM** (C++) :
  Royal Holloway University of London
  http://www.ece.umn.edu/groups/ece8591/softwaresvm.html
- **SVM-Light** (C):
  Thorsten Joachims, University Dortmond/ Dept of CS/ AI group http://svmlight.joachims.org/
- **ISIS-SVM** (MATLAB):
  Steve Gunn, Image Speech & Intelligent System group in University of Southampton
  http://www.ece.umn.edu/groups/ece8591/software/svm.html
- **S-SVM** (C&MATLAB):
  Xuhui Shao, ECE Dept in University of Minnesota
  http://www.ece.umn.edu/groups/ece8591/software/svm.html
- **SSVM** (MATLAB):
  Yuh-Jye Lee and O. L. Mangasarian, University of Wisconsin Madison
  http://www.cs.wisc.edu/dmi/svm/ssvm/
- **LIBSVM** (C++ & JAVA):
  Chih-Chung Chang and Chih-Jen Lin, National Taiwan University
  http://www.csie.ntu.edu.tw/~cjlin/libsvm/#nuandone

SVM for the XOR Problem

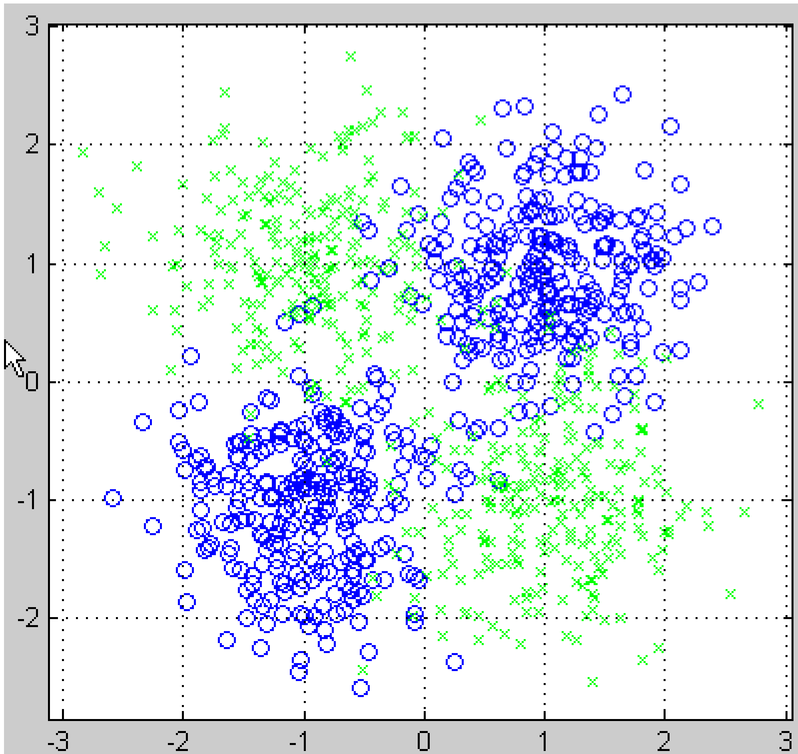
- The *exclusive-OR (XOR) problem* is the simplest problem that cannot be solved using a linear discriminant operating directly on the features.
- *SVM approach*: preprocess the features to map them to a higher dimension space where they can be linearly separated.
- An Example:

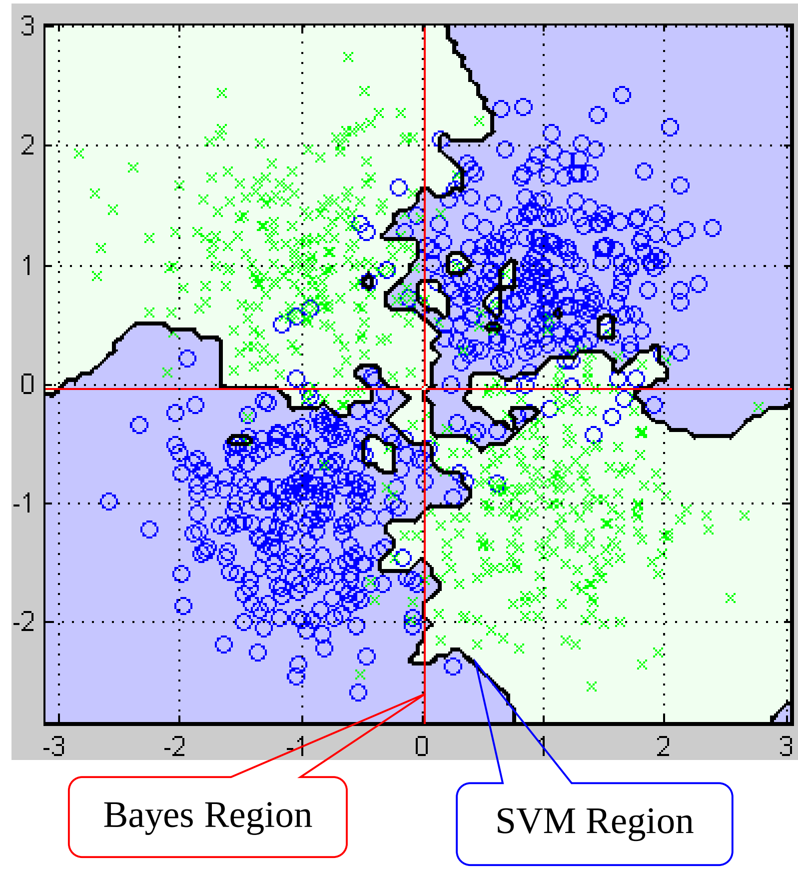
$$x_1, x_2 \Rightarrow 1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2$$



SVM for the XOR Problem (Cont.)

- Gaussian Case with spherical covariance:





Gaussion distributed data sets with 1000 samples :

((1,1); 0.6 I_2), ((-1,-1); 0.6 I_2)

((1,-1); 0.6 I_2), ((-1,1); 0.6 I_2)

leave-one-out cross-validation

SVM for Text Categorization

- Goal: classification of documents into a fixed number of predefined categories
- Significantly better than other methods, such as Naive Bayes, linear regression, k-nearest neighbor, and decision tree. New approaches: sequence prediction algorithms (LSTM, Transformer, Attention models)

Text → Features → Categories

Feature formation
Feature selection
Feature score calculations

Features:
Single tokens ☆
Phrases
Stemmed tokens

High dimensional input space (many features)
Few irrelevant features
Document Vectors are sparse
Many text categorization problems are linearly separable

Notation:
tf_{fd} : the occurrence frequency of feature f in document d
D : the total number of documents in the training set
df_f : the number of documents containing the feature f

Feature weighting schemes:

IDF(inv. doc. freq.): $\log(D/df_f)$

TF(term freq.): tf_{fd}

TFIDF: $tf_{fd}\log(D/df_f)$

☆ LOGTFIDF: $\log(tf_{fd}+0.5)\,\log(D/df_f)$

LOGTF: $\log(tf_{fd})$

BINARY: 1 or 0

Playing with Data

</REUTERS>

<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="5562" NEWID="19">

```
# Reuters category"corporate aquisitions" (training examples: 1000 positive/ 1000 negative)
1 6:0.0198403253586671 15:0.0339873732306071 29:0.0360280968798065 31:0.0378103484117687 41:0.0456787263779904
1 6:0.0292418053787394 11:0.0438009834096617 15:0.0500925330294462 26:0.0210944325344804 27:0.0141908540227881
1 6:0.028662086648757 26:0.124057412723749 29:0.0520475554655016 31:0.0546222636376468 63:0.0309765241093428 6
...
1 6:0.0201930401036759 15:0.0691831790313298 31:0.0384825282857204 60:0.0766336384026414 63:0.0218236097489283
```

<u>Data set</u>: Reuters-21578, Distribution 1.0
text categorization test collection
<u>Available</u>: http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
<u>Categorization Question</u>:
Which Reuters articles are about "Corporate mergers/acquisitions (ACQ)"?
<u>Train data</u>: 9947 feature vectors, 2000 articles, half of them are about ACQ
<u>Test data</u>: 600 articles
<u>Test Results</u>: Using LIBSVM, RBF kernel with parameter gamma=1.2, precision=97.3% (584 out of 600),
vs. SVM-light with the same setting, precision=95.4%. [Joachims 98]

</BODY></TEXT>

Documents represented as feature vectors

Initial article in SGML format

Kernel Regression

$$D = \{(z_1, \underline{x}_1), (z_2, \underline{x}_2), ..., (z_N, \underline{x}_N)\}$$

$$error = z - y(\underline{x});\ y(\underline{x}) = \underline{w}^T \underline{\phi}(\underline{x}) + w_0$$

$$Linear\ estimate: \underline{w} = \sum_{n=1}^{N} \alpha_n \underline{\phi}(\underline{x}_n) = \Phi \underline{\alpha}$$

$$\Phi = \begin{bmatrix} \underline{\phi}(\underline{x}_1) & \underline{\phi}(\underline{x}_2) & . & \underline{\phi}(\underline{x}_N) \end{bmatrix};\ p\ by\ N\ matrix$$

$$\underline{\alpha} = [\alpha_1 \quad \alpha_2 \quad . \quad \alpha_N]^T$$

$\Phi^T \Phi = [K(\underline{x}_i, \underline{x}_j)] = [\underline{\phi}^T(\underline{x}_i)\underline{\phi}(\underline{x}_j)]$ **symmetric Kernel (Grammian)**

$$J_N = \frac{1}{2}\sum_{n=1}^{N}\left(z_n - \underline{w}^T \underline{\phi}(\underline{x}_n) - w_0\right)^2 + \frac{\lambda}{2}\underline{w}^T \underline{w} = \frac{1}{2}\sum_{n=1}^{N}\left(z_n - \underline{\alpha}^T \Phi^T \underline{\phi}(\underline{x}_n) - w_0\right)^2 + \frac{\lambda}{2}\underline{\alpha}^T \underbrace{\Phi^T \Phi}_{[K(\underline{x}_i, \underline{x}_j)]}\underline{\alpha}$$

$$= \frac{1}{2}(\underline{z} - w_0 \underline{e})^T(\underline{z} - w_0 \underline{e}) + \frac{1}{2}\underline{\alpha}^T K^2 \underline{\alpha} - \underline{\alpha}^T K(\underline{z} - w_0 \underline{e}) + \frac{\lambda}{2}\underline{\alpha}^T K \underline{\alpha}$$

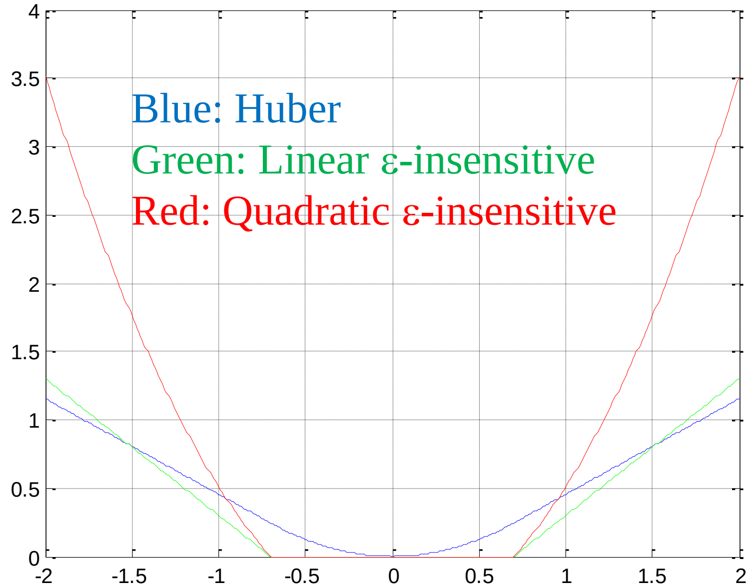
$$\boxed{\begin{aligned} \frac{\partial J_N}{\partial w_0} &= -\underline{e}^T(\underline{z} - w_0 \underline{e}) + \underline{\alpha}^T K \underline{e} = 0 \\ \Rightarrow w_0 &= \frac{\underline{e}^T(\underline{z} - K\underline{\alpha})}{\underline{e}^T \underline{e}} = \frac{\underline{e}^T(\underline{z} - K\underline{\alpha})}{N} \end{aligned}}$$

$$\underline{e} = [1 \quad 1 \quad . \quad 1]^T$$

$$\left.\begin{aligned} \hat{\underline{\alpha}} &= (K^2 + \lambda K)^{-1} K(\underline{z} - w_0 \underline{e}) = (K + \lambda I)^{-1}(\underline{z} - \hat{w}_0 \underline{e}) \\ \hat{w}_0 &= \frac{\underline{e}^T}{N}(\underline{z} - K\hat{\underline{\alpha}}) = \bar{z} - \bar{\hat{z}} \\ \Rightarrow \hat{\underline{\alpha}} &= ([I_N - \frac{\underline{e}\underline{e}^T}{N}]K + \lambda I)^{-1}(\underline{z} - \bar{z}\underline{e});\ \bar{z} = \frac{\underline{e}^T \underline{z}}{N} \end{aligned}\right\} \begin{bmatrix} K + \lambda I & \underline{e} \\ \underline{e}^T K & N \end{bmatrix}\begin{bmatrix} \hat{\underline{\alpha}} \\ \hat{w}_0 \end{bmatrix} = \begin{bmatrix} \underline{z} \\ \underline{e}^T \underline{z} \end{bmatrix}$$

At a test point $\underline{x}$: $y(\underline{x}) = \hat{\underline{w}}^T \underline{\phi}(\underline{x}) + \hat{w}_0 = \hat{\underline{\alpha}}^T \Phi^T \underline{\phi}(\underline{x}) + \hat{w}_0 = \underbrace{\left(\underline{z} - \bar{z}\underline{e}\right)^T ([I_N - \frac{\underline{e}\underline{e}^T}{N}]K + \lambda I)^{-1}}_{\text{Pre-computed}} \underbrace{\begin{bmatrix} \underline{\phi}^T(\underline{x}_1)\underline{\phi}(\underline{x}) \\ \underline{\phi}^T(\underline{x}_2)\underline{\phi}(\underline{x}) \\ . \\ \underline{\phi}^T(\underline{x}_N)\underline{\phi}(\underline{x}) \end{bmatrix}}_{\text{inner products}} + \hat{w}_0$

SVM Regression - 1



$$e = z - y(\underline{x})$$
$$y(\underline{x}) = \underline{w}^T \underline{\Phi}(\underline{x}) + w_0$$

Huber:

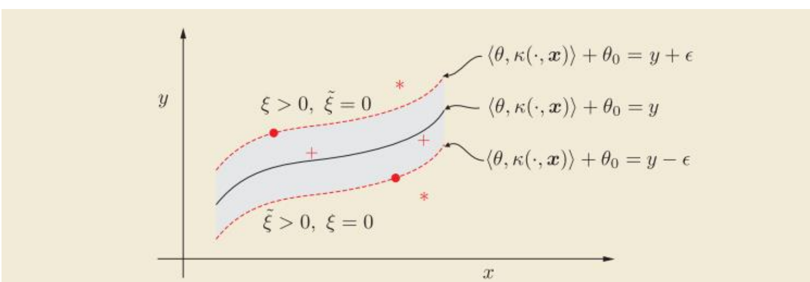
$$L_\varepsilon(e) = \begin{cases} \varepsilon |e| - \dfrac{\varepsilon^2}{2}; |e| > \varepsilon \\ \dfrac{e^2}{2}; |e| \le \varepsilon \end{cases}$$

Linear ε – insensitive:

$$L_\varepsilon(e) = \begin{cases} |e| - \varepsilon; |e| > \varepsilon \\ 0; |e| \le \varepsilon \end{cases}$$

Quadratic ε – insensitive:

$$L_\varepsilon(e) = \begin{cases} e^2 - \varepsilon^2; |e| > \varepsilon \\ 0; |e| \le \varepsilon \end{cases}$$



The tube around the nonlinear regression curve. Points outside the tube have either $\tilde{\xi} > 0$ and $\xi = 0$ or $\xi > 0$ and $\tilde{\xi} = 0$. The rest of the points have $\tilde{\xi} = \xi = 0$. Points which are inside the tube correspond to zero Lagrange multipliers.

Primal: Linear ε-insensitive

$$J = C\sum_{n=1}^{N} L_\varepsilon(e_n) + \frac{1}{2}\|\underline{w}\|^2 = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\underline{w}\|^2$$

$$y(\underline{x}^n) - \varepsilon - \hat{\xi}_n \le z^n \le y(\underline{x}^n) + \varepsilon + \xi_n; n = 1, 2, .., N$$

$$\Rightarrow y(\underline{x}^n) + \varepsilon + \xi_n - z^n \ge 0 \text{ .....multiplier } \lambda_n$$

$$z^n - y(\underline{x}^n) + \varepsilon + \hat{\xi}_n \ge 0 \text{......multiplier } \hat{\lambda}_n$$

$$\xi_n \ge 0; \hat{\xi}_n \ge 0$$

SVM Regression - 2

Primal: Linear ε-insensitive

$$J = C\sum_{n=1}^{N} L_\varepsilon(e_n) + \frac{1}{2}\|\underline{w}\|^2 = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\underline{w}\|^2$$

$$y(\underline{x}^n) - \varepsilon - \hat{\xi}_n \le z^n \le y(\underline{x}^n) + \varepsilon + \xi_n;\ n = 1,2,..,N$$

$$\Rightarrow y(\underline{x}^n) + \varepsilon + \xi_n - z^n \ge 0 \text{ .....multiplier } \lambda_n$$

$$z^n - y(\underline{x}^n) + \varepsilon + \hat{\xi}_n \ge 0 \text{......multiplier } \hat{\lambda}_n$$

$$\xi_n \ge 0;\ \hat{\xi}_n \ge 0$$

Dual: Linear ε-insensitive

$$q(\underline{\lambda}, \underline{\hat{\lambda}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(\lambda_n - \hat{\lambda}_n)(\lambda_m - \hat{\lambda}_m)K(\underline{x}^n, \underline{x}^m)$$

$$s.t. \quad 0 \le \lambda_n \le C$$

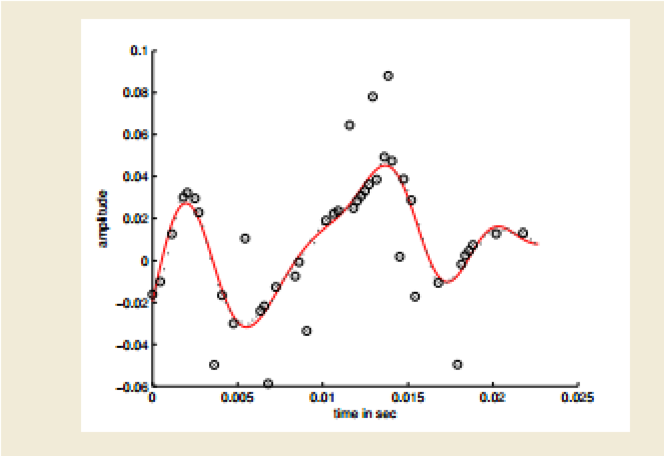
$$0 \le \hat{\lambda}_n \le C$$

$$\underline{w}^* = \sum_{n=1}^{N}(\lambda_n - \hat{\lambda}_n)\underline{\Phi}(\underline{x}^n)$$

$$w_0 = \frac{1}{(\#\xi_n = 0) + (\#\hat{\xi}_n = 0)}\left(\sum_{n:\xi_n=0}[z^n - y(\underline{x}^n) - \varepsilon] + \sum_{n:\hat{\xi}_n=0}[z^n + \varepsilon - y(\underline{x}^n)]\right)$$

$$y(\underline{x}) = \sum_{n=1}^{N}(\lambda_n - \hat{\lambda}_n)\underline{\phi}^T(\underline{x}^n)\underline{\phi}(\underline{x}) + w_0 = \sum_{n=1}^{N}(\lambda_n - \hat{\lambda}_n)K(\underline{x}^n, \underline{x}) + w_0$$

The original data were samples from a music recording from Blade Runner. A white Gaussian noise was then added at a 15dB level and a number of outliers were intentionally randomly introduced and "hit" some of the values (10%). The SVM regression was used, employing the Gaussian kernel with $\sigma = 0.004$ and $\varepsilon = 0.003$.



Summary

- Regression Based Learning
- Optimization Techniques
- LMS and Convergence Analysis
- Better Methods (RLS, Gauss-Newton,..)
- Ho-Kashyap Procedures
- Support Vector Machines $\Rightarrow$ QP
- Subgradient-based SVM
- SVM and Elastic Net
- SVM Regression $\Rightarrow$ QP