Recap

- We have been considering linear classifiers:

$$\begin{aligned} h(X) &= 1 \text{ if } \sum_{i=1}^{d'} w_i \phi_i(X) + w_0 > 0 \\ &= 0 \text{ Otherwise} \end{aligned}$$

  where ϕ_i are fixed functions.
- We take $h(X) = \text{sign}(W^T \Phi(X))$ for simplicity of notation. (Augumented feature vector).
- Perceptron is a classical algorithm to learn such a classifier.

Recap

- We also discussed linear regression. The objective is to learn a model:
$$\hat{y}(X) = W^T X$$
(We could use $\phi_i(X)$ in place of x_i).
- The least squares criterion is to minimize
$$J(W) = \frac{1}{2} \sum_{i=1}^{n} (W^T X_i - y_i)^2$$
- The minimizer is the linear least squares solution
$$W^* = (A^T A)^{-1} A^T Y$$
Where A is a matrix whose rows are X_i^T and Y is a vector whose components are y_i.

Recap

- We can also minimize J by iterative gradient descent.
- A stochastic gradient descent on J is the LMS algorithm.
- The LMS algorithm is:

$$W(k+1) = W(k) - \eta \left(X(k)^T W(k) - y(k)\right) X(k)$$

  where $(X(k), y(k))$ is the (random) sample picked and $W(k)$ is the weight vector at iteration k.
- This is used in many adaptive signal processing problems.

Recap

- We can model the conditional distribution of y given X as Gaussian with mean $W^T X$.
- Then the ML estimate for W is same as the least squares solution.
- Thus, we can think of least squares criterion as appropriate when the measurements are corrupted by additive gaussian noise.

- The Gaussian noise assumption is alright for a regression problem.
- In a 2-class classification problem, where $y \in \{0,\ 1\}$, Gaussian noise does not make sense.
- So, we will investigate a slightly different model for the classification problem.

- We showed that if we want to predict y as a function of X to minimize $E[\ (f(X) - y)^2\ ]$, then the optimal function is
$$f^*(X)\ =\ E\left[y \mid X\right]$$
- Suppose $y \in \{0,\ 1\}$. Then
$$f^*(X)\ =\ E\left[y \mid X\right]\ =\ P[y = 1 \mid X]\ = q_1(X)$$
- Hence in the least squares method we are learning an approximation to posterior probability.
- We can ask what would be a reasonable model for posterior probability.

By Bayes rule

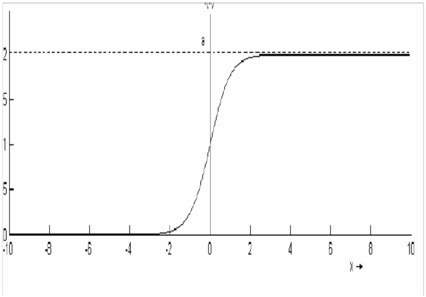
$$\begin{aligned} q_0(X) &= \frac{f_0(X)\,p_0}{f_0(X)\,p_0 \,+\, f_1(X)\,p_1} \\ &= \frac{1}{1+\frac{f_1(X)\,p_1}{f_0(X)\,p_0}} \\ &= \frac{1}{1+\exp(-\xi)} \quad \text{where} \end{aligned}$$

$$\xi = -\ \ln\left(\frac{f_1(X)\,p_1}{f_0(X)\,p_0}\right) \,=\, \ln\left(\frac{f_0(X)\,p_0}{f_1(X)\,p_1}\right)$$

- Suppose f_0, f_1 are Gaussian with equal covariance matrices.
- Then
$$\ln\left(\frac{f_0(X)\,p_0}{f_1(X)\,p_1}\right) = W^T X + w_0$$
where
$$W = \Sigma^{-1}(\mu_0 - \mu_1); \quad w_0 = \frac{1}{2}(\mu_1^T \Sigma^{-1}\mu_1 - \mu_0^T \Sigma^{-1}\mu_0) + \ln\left(\frac{p_0}{p_1}\right)$$
- Thus, in this case we have
$$q_0(X) = f_{y|X}(0|X) = \frac{1}{1+\exp(-W^T X - w_0)} = \sigma(W^T X + w_0)$$
This is called the logistic function or the sigmoid function.

Logistic Function

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



Logistic Regression

- The logistic function is a good model for posterior probability.
- We want to learn a probability model:

$$f_{y|X}(1|X) = P[y = 1|X] = \frac{1}{1 + \exp(-W^T X)}$$

  (where we are assuming augumented feature vector)
- This is a discriminative model.
- We can learn the model using maximum likelihood approach
- Such a method is called logistic regression.

Logistic Regression

- Let the data be $\mathcal{D} = \{(X_1, y_1), \cdots, (X_n, y_n)\}$.
- We are modelling y conditioned on X as Bernoulli with parameter $\sigma(W^T X)$.
- Hence the likelihood function is given by

$$L(W \mid \mathcal{D}) = \prod_{i=1}^{n} \theta_i^{y_i}(1-\theta_i)^{1-y_i}, \quad \theta_i = \sigma(W^T X_i)$$

- Hence the log likelihood is given by

$$l(W|\mathcal{D}) = \sum_{i=1}^{n} \left[y_i \ln(\sigma(W^T X_i)) + (1-y_i)\ln(1-\sigma(W^T X_i)) \right]$$

- To maximize this, we need its gradient.
- Note that the derivative of logistic function is

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$$

- The log likelihood is given by

$$l(W|\mathcal{D}) = \sum_{i=1}^{n} \left[y_i \ln(\sigma(W^T X_i)) + (1 - y_i) \ln(1 - \sigma(W^T X_i)) \right]$$

- Its gradient is

$$\begin{aligned}
\frac{\partial l}{\partial W} &= \sum_{i=1}^{n} \Bigg[y_i \frac{\sigma(W^T X_i)(1 - \sigma(W^T X_i))X_i}{\sigma(W^T X_i)} + \\
&\qquad (1 - y_i) \frac{(-1)\sigma(W^T X_i)(1 - \sigma(W^T X_i))X_i}{1 - \sigma(W^T X_i)} \Bigg] \\
&= \sum_{i=1}^{n} \left[y_i (1 - \sigma(W^T X_i))X_i + (1 - y_i)(-1)\sigma(W^T X_i)X_i \right] \\
&= \sum_{i=1}^{n} \left[\left(y_i - y_i\sigma(W^T X_i) - \sigma(W^T X_i) + y_i\sigma(W^T X_i) \right) X_i \right] \\
&= \sum_{i=1}^{n} \left[\left(y_i - \sigma(W^T X_i) \right) X_i \right]
\end{aligned}$$

- In logistic regression we need to maximize this log likelihood.
- Hence the algorithm is

$$W(k+1) = W(k) + \eta \sum_{i=1}^{n} \left[\left(y_i - \sigma(W^T X_i) \right) X_i \right]$$

- The least squares algorithm to minimize $\sum_{i=1}^{n} (y_i - W^T X_i)^2$ through gradient descent is

$$W(k+1) = W(k) + \eta \sum_{i=1}^{n} \left[\left(y_i - W^T X_i \right) X_i \right]$$

- So, in logistic regression we are simply taking the model output as $\sigma(W^T X_i)$.
- In Least squares we would threshold $W^T X$ at 0.5 to classify a new X.
- In logistic regression, we threshold $\sigma(W^T X)$ at 0.5 to classify a new X.

- Since we know that logistic function is a good model for posterior probability we could (alternately) try and minimize

$$J(W) = \sum_{i=1}^{n} \left(y_i - \sigma(W^T X_i)\right)^2$$

- Gradient descent on this would give us

$$W(k+1) = W(k) + \eta \sum_{i=1}^{n} \left[\left(y_i - \sigma(W^T X_i)\right) g(W, X_i) X_i\right]$$

where $g(W, X) = \sigma(W^T X_i)(1 - \sigma(W^T X_i))$.

- Very similar to the logistic regression algorithm.

- We are fitting a model $f(X) = W^T \, \Phi(X)$ to the data.
- We want to rate different W for their ‘goodness of fit’.
- $\sum_i \left(W^T \, \Phi(X_i) - y_i\right)^2$ is the ‘data error’.
- But it does not tell whole story of how good is W.

- We do not want the algorithm to 'overfit'.
- That is, low training error but high test error or generalization error
- Overfitting often happens if the model we learnt is 'too complex'.
- Hence we should have an algorithm that prefers 'simple' models.
- One way of doing this is through the so called regularization.

Regularization

- We can change our criterion to

$$\begin{aligned} J(W) &= \text{Data error} \; + \; \lambda \; \text{model complexity} \\ &= \frac{1}{2} \sum_{i=1}^{n} (W^T \Phi(X_i) - y_i)^2 \; + \; \lambda \, \Omega(W) \end{aligned}$$

- Here $\Omega(W)$ is some measure of how ‘complex’ the model is.
- This is called regularized least squares and λ is called the regularization constant.

Ridge Regression

- In linear least squares regression, we often choose $\Omega(W) = \frac{1}{2}||W||^2$. Called Tikhanov regularization.
- Now the criterion is

$$\begin{aligned} J(W) &= \frac{1}{2} \sum_{i=1}^{n} (W^T \Phi(X_i) - y_i)^2 + \frac{\lambda}{2} W^T W \\ &= \frac{1}{2} (AW - Y)^T (AW - Y) + \frac{\lambda}{2} W^T W \end{aligned}$$

where, as earlier, A is the matrix whose rows are $\Phi(X_i)$.

- Equating the gradient of J to zero, we get

$$A^T\,(AW - Y)\, +\, \lambda\, W = 0$$

- This gives us

$$(A^T A + \lambda\, I)W = A^T Y \quad \Rightarrow \quad W^* = (A^T A + \lambda\, I)^{-1} A^T Y$$

- $A^T A$ is positive semidefinite ($z^T A^T A z = (Az)^T (Az) \geq 0$).
- Hence, $(A^T A + \lambda I)$ would have all eigen values above λ and hence is always invertible.
- So, regularization helps the condition number of the linear equations and thus the learning is more ‘stable’.

- In ridge regression we are minimizing

$$J(W) = \frac{1}{2} \sum_{i=1}^{n} (W^T X_i - y_i)^2 \; + \; \frac{\lambda}{2} W^T W$$

- This is essentially same as solving the problem

$$\min_{W} \sum_{i=1}^{n} (W^T X_i - y_i)^2, \;\; s.t. \;\; ||W|| \leq B$$

  (for an appropriate choice of λ).
- Here the norm used is the L_2 norm.
- So, ridge regression is said to use L_2 penalty.

- We could use other norms, e.g., L_1 norm.
- Thus we could reformulate this as minimizing

$$J(W) = \frac{1}{2} \sum_{i=1}^{n} (W^T X_i - y_i)^2 \; + \; \frac{\lambda}{2} \mid\mid W \mid\mid_1$$

  where $\mid\mid W \mid\mid_1 = \sum_{i=1}^{d} |w_i|$.
- This formulation is called Lasso:

$$\min_W \sum_{i=1}^{n} (W^T X_i - y_i)^2, \;\; s.t. \;\; ||W||_1 \leq B$$

- This is also a convex optimization problem.
- It promotes sparsity in the solution and hence is also suitable for ‘feature selection’.

- Another way to look at regularized least squares is from a Bayesian framework.
- We saw that the least squares solution can also be derived as a ML estimate of parameters of a (reasonable) probabilty model for conditional distribution of y given X.
- The regularized least squares can be derived as a Bayesian (MAP) estimate of the parameters of the same model.

- As earlier, take the probabilty model as

$$f(y \mid X, W, \sigma) \; = \; \frac{1}{\sigma \sqrt{2\pi}} \; \exp\left(- \frac{1}{2} \, \frac{(y - W^T X)^2}{\sigma^2} \right)$$

- We want to estimate W from n *iid* observations $\{y_i(X_i), \; i = 1, \cdots, n\}$.

- We take the prior density of W as

$$f(W) = \left(\frac{1}{\alpha\sqrt{2\pi}}\right)^d \exp\left(-\frac{W^T W}{2\alpha^2}\right)$$

  which is zero-mean normal with diagonal covariance matrix; α is a hyper-parameter.

Now the posterior density is given by

$$\begin{aligned} f(W \mid Y) &\propto \prod_{i=1}^{n} f(y_i \mid X_i, W, \sigma)\, f(W) \\ &\propto \exp\left(- \sum_{i=1}^{n} \frac{(y_i - W^T X_i)^2}{2\sigma^2} - \frac{1}{2\alpha^2} W^T W \right) \end{aligned}$$

- To find the MAP estimate we need to maximize the posterior density
- We can maximize log of the posterior.

The log posterior is of the form

$$\ln(f(W \mid Y)) = -\frac{1}{2} \sum_{i=1}^{n} (y_i - W^T X_i)^2 - \lambda W^T W + K$$

- Maximizing this is same as minimizing the regularized least squares criterion.
- Hence the MAP estimate is the regularized least squares solution.

Bayesian Linear Regression

- We saw that we can look at linear least squares estimation in a Bayesian framework also.
- The MAP estimate corresponds to regularized least squares.
- MAP is a point estimate obtained from the posterior.
- In Bayesian framework, the estimate is whole of the posterior.
- We can also calculate the distribution of prediction variable based on data.

The posterior density

- In general, the prior can be

$$f(W) = \mathcal{N}(W|\boldsymbol{\mu_0}, \Sigma_0)$$

- The probability model for y is

$$f(Y|A, W, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(y_i|X_i^T W, \sigma^2) = \mathcal{N}(Y|AW, \sigma^2 I)$$

- The posterior would also be a Gaussian which can be calculated using the technique of ‘completing the squares’

The posterior density

- The posterior would be

$$f(W|Y, A, \sigma^2) = \mathcal{N}(W|\boldsymbol{\mu_n}, \Sigma_n)$$

where

$$\boldsymbol{\mu_n} = \Sigma_n \left(\sigma^2 A^T Y + \Sigma_0^{-1}\boldsymbol{\mu_0}\right)$$
$$\Sigma_n^{-1} = (\Sigma_0^{-1} + \sigma^2 A^T A)$$

The Predictive Distribution

- We can also Calculate (for any given new X)

$$f(y|X, A, Y, \sigma^2) = \int f(y|X, W, \sigma^2)\ f(W|A, Y, \sigma^2)\ dW$$

  which is $\mathcal{N}(y|\boldsymbol{\mu_n}^T X, A^T \Sigma_n A + \sigma^2)$.
- We can use this to predict the y for any X.

- For linear regression we want to minimize

$$J(W) = (1/2) \sum_i (W^T X_i - y_i)^2$$

- We can think of this as

$$J(W) = \sum_i L(W^T X_i,\ y_i)$$

  where L is a loss function.
- The squared error is only one choice for the loss.
- We can have others, e.g., absolute value of error.

Robust Regression

- The squared error criterion is sensitive to outliers. The absolute value of error is more robust to outliers.
- There are multiple viewpoints possible on this.
- One way of understanding it is that mean is more sensitive to outliers compared to median.
- Another way of looking at it is that the Gaussian noise is a 'light-tailed' distribution. (Recall that we get the criterion of squared error when we assume Gaussian noise and estimate a discriminative model).

Robust Regression

- The squared error criterion is sensitive to outliers. The absolute value of error is more robust to outliers.
- There are multiple viewpoints possible on this.
- One way of understanding it is that mean is more sensitive to outliers compared to median.
- Another way of looking at it is that the Gaussian noise is a 'light-tailed' distribution.
- Instead of Gaussian noise we can assume Laplace distribution for noise:

$$f_{\mathsf{Lap}}(x|\mu, b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right), \; -\infty < x < \infty$$

- Now maximizing likelihood would result in minimizing absolute value of error.
- This is a 'heavy-tailed' distribution.

- Suppose we want to learn W to minimize

$$J(W) = \sum_{i=1}^{n} |\,(y_i - W^T X_i)\,| = ||\,(AW - Y)\,||_1$$

- Optimizing absolute value of error is difficult.
- The objective function is convex but non-smooth.
- We can use some convex optimiztion techniques.
- Another interesting method for this is Iterated Reweighted Least Squares (IRLS).

Weighted Least Squares

- Suppose we want to minimize

$$J(W) = \sum_{i=1}^{n} b_i (W^T X_i - y_i)^2, \;\; b_i > 0$$

- Define a diagonal matrix $B = \text{diag}(\sqrt{b_i})$.
- Now minimizing J is same as least squares where we multiply X_i and y_i by B_{ii}.
- With A and Y as earlier, let $\tilde{A} = BA$ and $\tilde{Y} = BY$.
- So, now we get the solution as $W^* = (\tilde{A}^T \tilde{A})^{-1} \tilde{A} \tilde{Y}$.
- Thus, we can easily solve the least squares problem with any positive weights.

- Suppose we know how to minimize over W (for any b)

$$C(W,b) = \sum_{i=1}^{n} b_i \, g_i(W) \quad \left[g_i(W) = (W^T X_i - y_i)^2\right]$$

- We actually want to minimize

$$C_h(W) = \sum_{i=1}^{n} h(g_i(W)) \quad \left[h(x) = \sqrt{x}\right]$$

- Heuristically, we want to ensure descent on C should be descent on C_h.

$$\nabla_W C(W,b) = \nabla_W C_h(W) \ \Rightarrow \ b_i \nabla_W g_i(W) = h'(g_i(W)) \nabla_W g_i(W)$$

- Hence, we can take $b_i = h'(g_i(W))$.
  (In the iterative method, we use previous iteration value of W).

IRLS algorithm for minimizing absolute error

- We want to minimize

$$J(W) = \sum_{i=1}^{n} h(g_i(W)) = \sum_{i=1}^{n} \sqrt{(W^T X_i - y_i)^2}$$

- Note that for $h(x) = \sqrt{x}, \quad h'(x) = 0.5(x)^{-0.5}$
- Hence, we get the iterative method as

$$W(k+1) = \arg\min_{W} \sum_{i=1}^{n} b_i^k (W^T X_i - y_i)^2$$

where

$$b_i^k = h'(g_i(W(k))) = \frac{1}{2} |W(k)^T X_i - y_i|^{-1}$$

- We solve the minimization using least squares.
- One can show this is a descent method for J.

- Our notation: $g_i(W) = (W^T X_i - y_i)^2$
- We want to minimize $J(W) = \sum_{i=1}^n h(g_i(W))$ where h is concave. (For us $h(x) = \sqrt{x}$).
- Let us write: $r_i^k = g_i(W(k)), \;\; r_i^{k+1} = g_i(W(k+1))$
- Our iterations are

$$W(k+1) = \arg\min_W \sum_{i=1}^n h'\left(r_i^k\right) \; g_i(W)$$

- Hence we have

$$\sum_{i=1}^n h'\left(r_i^k\right) \; r_i^{k+1} < \sum_{i=1}^n h'\left(r_i^k\right) \; r_i^k$$

- Using this we need to show

$$J(W(k+1)) = \sum_{i=1}^n h\left(r_i^{k+1}\right) < \sum_{i=1}^n h\left(r_i^k\right) = J(W(k))$$

- Since h is concave, we have

$$h\left(r_i^{k+1}\right) \leq h\left(r_i^k\right) \;+\; h'\left(r_i^k\right)\;\left(r_i^{k+1} - r_i^k\right)$$

- By summing the above for $i = 1$ to n

$$\sum_{i=1}^{n} h\left(r_i^{k+1}\right) \leq \sum_{i=1}^{n} h\left(r_i^k\right) + K$$

where

$$K = \sum_{i=1}^{n} h'\left(r_i^k\right)\; r_i^{k+1} - \sum_{i=1}^{n} h'\left(r_i^k\right)\; r_i^k < 0$$

- Hence we get

$$J(W(k+1)) = \sum_{i=1}^{n} h\left(r_i^{k+1}\right) < \sum_{i=1}^{n} h\left(r_i^k\right) = J(W(k))$$

Huber Loss

- The squared error criterion is easy to optimize but is sensitive to outliers.
- The absolute value of error is more robust to outliers. But the optimiztion problem is more difficult because the absolute value function is non-differentiable at zero.
- An interesting approach is to use a hybrid of the two – use Huber loss defined by

$$\begin{aligned} L_{H_\delta}(a,b) &= 0.5(a-b)^2 && \text{if } |a-b| \leq \delta \\ &= \delta|a-b| - 0.5\delta^2 && \text{if } |a-b| > \delta \end{aligned}$$

- We learn W to minimize $J(W) = \sum_i L_{H_\delta}(y_i, W^T X_i)$.
- Using $\frac{d}{dr}|r| = \text{sign}(r)$, this loss function is differentiable and hence the resulting optimization problem has a smooth objective function.

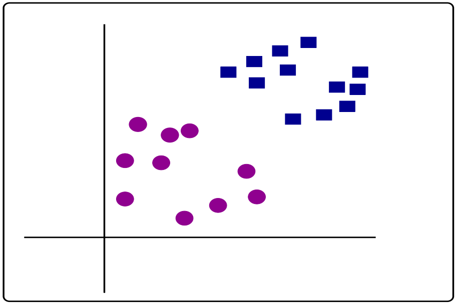
Fisher Linear Discriminant

- Minimizing mean squared error is one of the possible methods for learning linear classifiers and regressors.
- Fisher Linear Discriminant is another way of constructing linear classifiers.

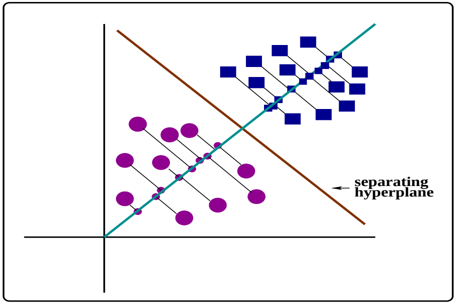
Fisher Linear Discriminant

- A linear discriminant function based classifier is:
  Decide $X \in C$*-1* *if* $W^T X + w_0 > 0$
- We project the data along the direction W.
- Hence One can think of the best W as the direction along which the two classes are well separated.
- Such a method is called Fisher Linear Discriminant.

- Consider the following 2-class example



- A good direction to project the data here is as shown.



- Fisher Linear Discriminant is based on formalizing this notion

Fisher Linear Discriminant

- The idea is to find a direction W such that the training data of the two classes are well-separated if projected onto this direction.
- We consider the 2-class case.

- Let $\{(X_i,\ y_i),\ i = 1, \cdots, n\}$ be the data.
- Let $y_i \in \{0,\ 1\}$.
- Let C_0 and C_1 denote the two classes. Thus, if $y_i = 0$ then $X_i \in C_0$ and if $y_i = 1$ then $X_i \in C_1$.
- Let n_0 and n_1 denote the number of examples of each class. $(n = n_0 + n_1)$
- For any W, let $z_i = W^T X_i$.
- z_i are the one dimensional data that we get after projection.

- Let M_0 and M_1 be the means of data from the two classes:

$$M_0 = \frac{1}{n_0} \sum_{X_i \in C_0} X_i; \quad M_1 = \frac{1}{n_1} \sum_{X_i \in C_1} X_i$$

The corresponding means of the projected data would be

$$m_0 = W^T M_0 \quad \text{and} \quad m_1 = W^T M_1$$

- The difference $(m_0 - m_1)$ gives us an idea of the separation between samples of the two classes after projecting the data onto the direction W.
- Hence, we may want a W that maximizes $(m_0 - m_1)^2$.
- However, we have to make this scale independent.
- Also, the distance between means should be viewed relative to the variances.

- Define

$$s_0^2 = \sum_{X_i \in C_0} (W^T X_i - m_0)^2; \qquad s_1^2 = \sum_{X_i \in C_1} (W^T X_i - m_1)^2$$

  These give us the variances (upto a factor) of the two classes in the projected data.
- We want large separation between m_0 and m_1 relative to the variances.

- Hence we can take our objective to be to maximize

$$J(W) = \frac{(m_1 - m_0)^2}{s_0^2 + s_1^2}$$

- We now rewrite J into a more convenient form.
- We have

$$\begin{aligned}(m_1 - m_0)^2 &= (W^T M_1 - W^T M_0)^2 \\ &= [W^T(M_1 - M_0)][W^T(M_1 - M_0)]^T \\ &= W^T(M_1 - M_0)(M_1 - M_0)^T W\end{aligned}$$

- This is a popular 'trick'. In general $(X^T Y)^2 = (X^T Y)(Y^T X) = X^T YY^T X$

- Thus we have $(m_1 - m_0)^2 = W^T S_B W$ where

$$S_B = (M_1 - M_0)(M_1 - M_0)^T$$

  is a $d \times d$ matrix (note that $X_i \in \Re^d$).
- It is called *between class* scatter matrix.
- We can similarly write s_0^2 and s_1^2 also as quadratic forms.

We have

$$\begin{aligned}
s_0^2 &= \sum_{X_i \in C_0} (W^T X_i - W^T M_0)^2 \\
&= \sum_{X_i \in C_0} [W^T (X_i - M_0)]^2 \\
&= \sum_{X_i \in C_0} W^T (X_i - M_0)(X_i - M_0)^T W \\
&= W^T \left[\sum_{X_i \in C_0} (X_i - M_0)(X_i - M_0)^T \right] W
\end{aligned}$$

- Similarly, we get

$$s_1^2 \;=\; W^T \left[\sum_{X_i \in C_1} (X_i - M_1)(X_i - M_1)^T \right] W$$

- Thus we can write $s_0^2 + s_1^2 = W^T S_w W$, where

$$S_w = \sum_{X_i \in C_0} (X_i - M_0)(X_i - M_0)^T + \sum_{X_i \in C_1} (X_i - M_1)(X_i - M_1)^T$$

- S_w is also $d \times d$ matrix and is called *within class* scatter matrix.

- Hence we can now write J as

$$J(W) = \frac{W^T S_B W}{W^T S_w W}$$

- We want to find a W that maximizes $J(W)$.
- Note that $J(W)$ is not affected by scaling of W.
- Given the data we can calculate the S_B and S_w.
- Maximizing ratio of quadratic forms is a standard optimization problem.

- We need to maximize

$$J(W) = \frac{W^T S_B W}{W^T S_w W}$$

- Differentiating w.r.t. W and equating to zero, we get

$$\frac{2S_B W}{W^T S_w W} - \frac{W^T S_B W}{(W^T S_w W)^2} 2S_w W = 0$$

- Implies, $S_B W$ is in the same direction as $S_w W$.

- Thus, any maximizer of $J(W)$ has to satisfy

$$S_w W \;=\; \lambda\, S_B W$$

  for some constant λ.
- This is known as the generalized eigen value problem.
- There are standard methods to solve this problem using, e.g., LU decomposition.
- By solving the generalized eigen value problem we can find the best direction W.

- Often, the real symmetric matrix S_w would be invertible.
- Recall that

$$S_w = \sum_{X_i \in C_0} (X_i - M_0)(X_i - M_0)^T + \sum_{X_i \in C_1} (X_i - M_1)(X_i - M_1)^T$$

- This is a sum of large number of rank 1 matrices.
- Also each term in S_w is proportional to the sample-mean-estimate of the covariance matrix of one of the class conditional densities.

- If S_w is invertible, then we can write

$$W \; = \; S_w^{-1} S_B W$$

- We have

$$S_B W \; = \; (M_1 - M_0)(M_1 - M_0)^T W = k(M_1 - M_0)$$

  where k is some constant. (note $k = (m_1 - m_0)$)
- Now we get (since scale factor in W is not relevant)

$$W \; = \; S_w^{-1}(M_1 - M_0)$$

Obtaining Fisher Linear Discriminant

- We can sum-up the process as follows.
- Given the training data, we first form the scatter matrix S_w and also calculate the means M_0 and M_1.
- If S_w is invertible, we calculate W by $W = S_w^{-1}(M_1 - M_0)$.
- Even if S_w is not invertible, there are techniques to find the maximizer of $J(W)$ by solving the generalized eigen value problem.
- Thus we can find the best direction W.

- The final (linear) classifier is sign$(W^T X + b)$.
- So far, we have seen how to obtain best W.
- We still have to learn the best b also. But this is a simple threshold learning problem.
- We can do a simple line search to find the threshold b to maximize probability of correct classification.
- Or, we can take the one dimensional projected data and learn the best classifier by, e.g., modelling the class conditional densities as normal.

- Fisher Linear Discriminant is also a popular classifier.
- Though the method looks quite different from that of linear least squares there are close connections between the two.

- Given the original training data $\{(X_i, y_i)\}$ we form new training data $\{(X_i, y_i')\}$ as follows.
- We take $y_i' = n/n_0$ if $y_i = 0$ and $y_i' = -n/n_1$ if $y_i = 1$.
- We now treat this as a data for a regression problem and learn a model $\hat{y} = W^T X + b$ using linear least squares.
- It can be shown that the least squares solution for W would be same as that of FLD.
- Thus FLD can be viewed as a special case of linear least squares.

- We have considered various methods of learning linear classifiers and regression functions.
- In the case of regression, we have considered only estimating real-valued functions. We can generalize this to vector-valued functions.
- In classification we considered only 2-class problems. This can also be generalized to multi-class case.

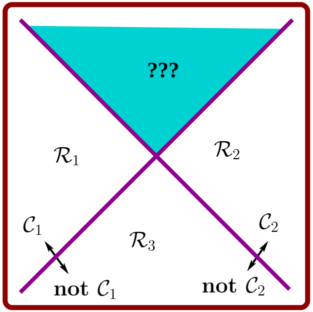
- First consider estimating vector-valued functions.
- Now the training data is $\{(X_i, y_i),\ i = 1, \cdots, n\}$ where $X_i \in \Re^d$ and $y_i = (y_i^1, \cdots, y_i^m) \in \Re^m$.
- For any given X we want to predict the target $y = (y^1, \cdots, y^m)$.
- Thus we want to learn W_j, b_j, $j = 1, \cdots, m$ so that

$$\hat{y}^j \ = \ W_j^T X + b_j,\ j = 1, \cdots, m$$

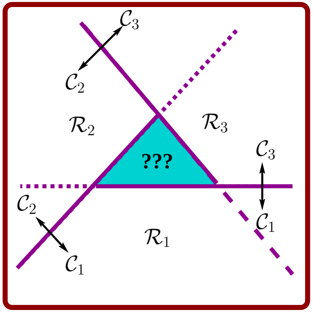
- We can obtain these by simply solving m number of linear least squares regression problems.

- Now let us consider the multi-class problem.
- Suppose we have K classes: $C_1, \cdots, C_K$.
- We can solve the multi-class problem by learning a number of 2-class classifiers.
- For example, we can learn K two class classifiers: 'C_i Vs not-C_i'. (Called one versus rest).
- Or we can learn $K(K-1)/2$ number of 2-class classifiers: 'C_i Vs C_j'
- But neither of these approaches are really satisfactory for generalizing linear discriminant functions.

- Suppose we try ‘one versus rest’ approach. Then there may be regions of feature space where classification is ambiguous.



- Similar problem exists when we try 'C_i Vs C_j' approach



- A better way to formulate a linear discriminant function based classifier for the multi-class case is as follows.
- We will have K functions, g_s, $s = 1, \cdots, K$, given by

$$g_s(X) \ = \ W_s^T X + b_s$$

- Now the classifier would assign class C_j to X if

$$g_j(X) \geq g_s(X), \ \forall s$$

- Essentially, we are approximating the posterior probability of j^{th} class by g_j.
- Recall that this is the structure of Bayes classifier in multi-class case.
- In the above we would have a fixed (may be arbitrary) rule for breaking ties.

- Now, to learn a linear classifier for the K-class case, we need to learn the K functions g_s.
- The simplest way to do this is to make the class label to be a vector of K components.
- If $X_i \in C_j$ then y_i would be a K-vector with j^{th} component one and all others zero.
- Now learning the K functions is same as linear regression with vector valued targets.

Logistic Regression – multi-class case

- We can similarly generalize logistic regression also for multi-class case.
- Let us recall the main idea in logistic regression in the two class case.
- We approximate posterior probability as

$$q_1(X) = h(W^T X + w_0)$$

where h is the logistic function

$$h(a) = \frac{1}{1 + \exp(-a)}$$

- The motivation for using the logistic function is

$$\begin{aligned} q_1(X) &= \frac{f_1(X)\,p_1}{f_0(X)\,p_0 \;+\; f_1(X)\,p_1} \\ &= \frac{1}{1+\exp(-\xi)} \quad \text{where} \end{aligned}$$

$$\xi = -\;\ln\left(\frac{f_0(X)\,p_0}{f_1(X)\,p_1}\right) \;=\; \ln\left(\frac{f_1(X)\,p_1}{f_0(X)\,p_0}\right)$$

- We now use the same Bayes rule to find a convenient model for posterior probabilities in the multiclass case.

- In the multi-class case, Bayes rule gives

$$q_j(X) = \frac{f_j(X)p_j}{\sum_s f_s(X)p_s} = \frac{\exp(a_j)}{\sum_s \exp(a_s)}$$

  where $a_s = \ln(f_s(X)p_s)$.
- The idea is that we approximate $a_s = W_s^T X + w_{s0}$.
- The above function is a good candidate for modeling posterior probabilities in the multi-class case.

- In the two class case we want to know which of $f_1(X)p_1$ and $f_0(X)p_0$ is greater.
- This can be done by looking at sign of $\ln\left(\frac{f_1(X)\,p_1}{f_0(X)\,p_0}\right)$.
- 'sign' is a discontinuous function and the logistic function is a kind of continuous analog for this.
- In the multiclass case, we need to find the maximum of $f_i(X)p_i$, $i = 1, \cdots, K$.
- So, we need a smooth function to approximate the maximum computation.

- Define a function $g : \Re^K \rightarrow \Re^K$, with $g(a) = [g_1(a) \ \cdots \ g_K(a)]^T$ and for $j = 1, \cdots, K$,

$$g_j(a) = \frac{\exp(a_j)}{\sum_s \exp(a_s)}, \ a = (a_1, \cdots, a_K)^T \in \Re^K.$$

- This is known as the softmax function.
- Essentially if a_j is the maximum of the components of a then $g_j(a)$ would be closer to one and all other components of g would be closer to zero.
- We note, for later use, that

$$\frac{\partial g_k}{\partial a_j} = g_k(a)(\delta_{kj} - g_j(a))$$

where $\delta_{kj} = 0$ if $k \neq j$ and $\delta_{kj} = 1$ if $k = j$.

- We now take, for each s, $a_s = W_s^T X + w_{s0}$ and learn all W_s and w_{s0}.
- Using augumented feature vector, we can write $a_s = W_s^T X$. Let W be a matrix with columns W_s.
- After learning, W, given a new X, we calculate $g(W^T X)$ and then put X in class C_j if the j^{th} component of $g(W^T X)$ is the highest.

- In the 2-class logistic regression, we used the logistic function to model posterior probability.
- In the multi-class case we want to use the softmax function for modeling the conditional distribution of y given X.
- Let us now write

$$g_j(W, X) = \frac{\exp(W_j^T X)}{\sum_s \exp(W_s^T X)}$$

- Let us represent the class label as a one-hot vector:

$$y = (y^1, \cdots, y^K)^T; \;\; y^j \in \{0,\ 1\}; \;\; \sum_j y^j = 1$$

- y takes only K different values and let us represent them as $e_1, \cdots, e_K$.
- We take (as our probability model) $P[y = e_j \mid X] = g_j(W, X)$.

- The probability model for the conditional distribution now is

$$f(y \mid X, W) = \prod_{i=1}^{K} (g_i(W, X))^{y^i}$$

- Let the data be $\mathcal{D} = \{(X_1, y_1), \cdots, (X_n, y_n)\}$.
- The likelihood now is

$$L(W_1, \cdots, W_K | \mathcal{D}) = \prod_{i=1}^{n} f(y_i | X_i, W) = \prod_{i=1}^{n} \prod_{j=1}^{K} (g_j(W, X_i))^{y_i^j}$$

- The log likelihood is

$$l(W_1, \cdots, W_K \mid \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{K} y_i^j \ln (g_j(W, X_i))$$

- We need to maximize this to learn the W.

- The log likelihood is

$$l(W_1, \cdots, W_K \mid \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{K} y_i^j \ln \left(g_j(W, X_i) \right)$$

- By differentiating this and using earlier formula for derivative of g_j, we can show that

$$\nabla_{W_j} l(W_1, \cdots, W_K \mid \mathcal{D}) = \sum_{i=1}^{n} (y_i^j - g_j(W, X_i)) X_i$$

- Hence an iterative algorithm for ML estimate of W_j, $j = 1, \cdots, K$, is

$$W_j(k+1) = W_j(k) + \eta \sum_{i=1}^{n} (y_i^j - g_j(W, X_i)) X_i$$

- This is the multi-class logistic regression.

- We can generalize Fisher linear discriminant also to multi-class case.
- In the 2-class case we are interested in finding a direction or a one-dimensional subspace onto which we project the data.
- In the K-class case, we want to find a $(K-1)$-dimensional subspace onto which we project the data.
- The idea is to find a subspace where in the projected data, the means of the two classes have maximum separation relative to the variances.

Learning and generalization

- The problem of designing a classifier is essentially one of learning from examples.
- Given training data, we want to find an appropriate classifier.
- It amounts to searching over a family of classifiers to find one that minimizes ‘error’ over training set.
- For example, in least squares approach we are searching over the family of linear classifiers for minimizing square of error.

- As we discussed earlier, performance on training set is not the real issue.
- We would like the learnt classifier to perform well on new data.
- This is the issue of generalization. Does the learnt classifier generalize well?

- In practice one assesses the generalization of the learnt classifier by looking at the error on a separate set of labelled data called test set.
- Since the test set would not be used in training, error on that data could be a good measure of the performance of the learnt classifier.
- But here we are more interested in formalizing the notion of generalization error.
- we look at the specific issues of practice later on. Currently our focus would be on theoretical analysis of how to say whether a learning algorithm would generalize well.

- We can see the main issue through a simple example of regression
- Suppose we have data $\{(X_i, y_i)\}$, $X_i, y_i \in \Re$.
- We want to learn a function f so that we can predict y as $f(X)$.
- This is a simple regression problem and we can use least squares for it based on the form of f.

- Suppose we choose polynomial function

$$f(X) = w_0 + w_1 X + w_2 X^2 + \cdots + w_m X^m$$

- As we discussed earlier we can easily learn this using linear least squares algorithm.
- One question is what m to choose.
- We have looked at regularized least squares for this. (It does not tell best m but helps learn a model with good generalization for a given m).
- There are other methods (e.g., BIC)

- But specifically, let us ask can our data error tell what m is proper.
- Firstly the fact that we get less error for m' compared to m does not necessarily mean m'-degree is a better fit.
- For a particular m if we get very low data error, can we say it is good?
- We know that if we search over all polynomials, we can never really learn anything. Can we formalize such notions precisely?
- There are different ways of addressing this issue (MDL, VC-theory etc).

- Any learning algorithm takes training data as the input and outputs a specific classifier/function.
- For this, it searches over some chosen family of functions to find one that optimizes a chosen criterion function.

$$\{(X_i, y_i)\} \rightarrow \boxed{\begin{array}{l}\text{Learning Algorithm} \\ \text{(searching over } \mathcal{F}\text{)}\end{array}} \rightarrow f \in \mathcal{F}$$

- The question is: how can we formalize correctness of learning?
- There are many ways of addressing this issue (MDL, VC-theory etc).

- For example, a generic approach is what is called **Minimum Description Length** principle.
- Suppose we want to send the data over a communication channel.
- we can send the $2n$ numbers, X_i, y_i using some number of bits.
- Or we can send X_i, the function f and the errors $y_i - f(X_i)$.

- If the fit is good, the errors $y_i - f(X_i)$ would be small and we may be able to send them using smaller number of bits compared sending y_i.
- However, we also need to send f.
- If f is very complex, then what we save in bits by sending errors instead of y_i may be more than offset by the bits needed to send description of f.
- Hence we can rate different f by the total number of bits we need.
- This can balance the data error and model complexity in a natural way.

- We will follow a different statistical approach to address the issue of correctness of learning.
- We begin with a simple formalism where there is no ‘noise’ and the goal of learning is well-defined.