Revisiting linear regression.

We earlier saw that under a LMS objective the optimal classifier under risk min is the conditional expectation.

Also if $Y \in \{0,1\}$, we know that

$$E[Y|X] = P[Y=1|X] = \eta_1(X).$$

Thus, Linear regression can be viewed as a parametric (linear) approximation for the posterior.

The idea of separability.

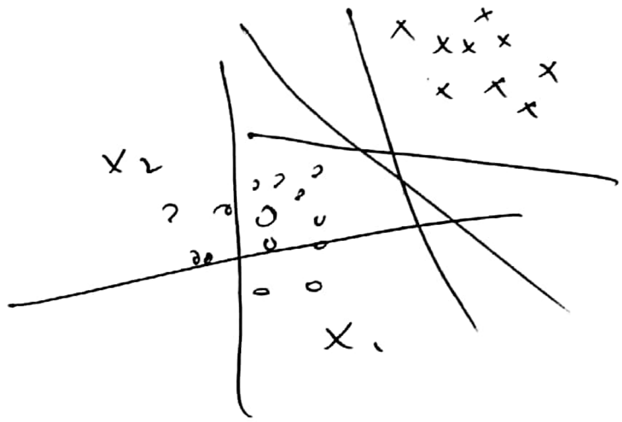
Training data $D : \{(x_i, y_i),\ i=1,\dots,n\}$ is called linearly separable, if $\exists\ W^*$ such that

$$x_i^T W^* > 0 \text{ if } y_i = 1$$
$$< 0 \text{ if } y_i = 0$$

(or $\phi(x_i)^T W^*$ for the Kernalized case).

W^* is called the separating hyperplane.

If data is linearly separable, $\exists$ infinitely many W_s^*



The family of linear classifiers aims to find out one such hyperplane under certain optimality criteria.

Note that $q_1(x) \approx h(x) \approx \operatorname{sign}(W^T x)$.

We saw examples of Baye's classifiers reducing to linear & quadratic discrimi-nant functions under certain conditions.

We were looking at the OLS algo to find the discriminant function.

[notice that the formulation is very similar to both classification & regression].

We also saw that $W^*_{OLS} = (\Phi^T\Phi)^{-1}\Phi^T y$.

We also saw that in general,

Error with a classifier = $Bias^2$ + variance + noise.
β

For a fixed value of error, unit decrease in bias results in more than quadratic increase in variance.

Thus, in practice, one always trades decreased variance with increase bias in order to reduce the total error - [first overfit & then reduce the model capacity]
This process is generally referred to as the Regularization.

Regularized Least Square

Suppose the true modell is a cubic polynomial. But h(x) is assumed to be a 10^{th} degree polynomial.

Now if one fits h(x) using LS, the co-efficients of higher degrees [after 3] will be very high with alternating signs.

In other words, if h(x) should be closer to the 'true target' most of the Ws should be zero. or W vector should be sparse.

Thus, we want to minimize LMS with a constraint that W should be sparse.

Thus, our new objective,

$$W^*_{reg} = \arg\min_W \| Y^T - W^T \Phi^T \|_2^2$$

$$\text{s.t } W^T W < \alpha$$

$[\|W\|_p < \alpha, \text{ In general}]$

Now, our new objective f^n becomes,

$$L = \| Y^T - W^T \Phi^T \| + \lambda W^T W.$$

Can be shown that

$$W^*_{reg} = \underline{(\Phi^T \Phi + \lambda I)^{-1} \Phi^T Y}.$$

One can now expect $\hat{W}_{reg}$ to be a having lesser MSE compared to $\hat{W}_{OLS}$.

The Bayesian view for regularization..

Earlier we saw that OLS is nothing but an MLE for the posterior with the following parametric form

$$f(y|x; w, \sigma) \sim N(\phi^T w, \sigma)$$

observe here that w is the parameter for this density.

Suppose we have some prior 'information' about w.

i.e, lets model

$$f(w) \sim N(w; 0, \alpha)$$

$$f(w) \propto \exp\left(\frac{-w^T w}{2\alpha^2}\right).$$

Now posterior for the parameter becomes,

$$f(w|y) \propto \prod_{i=1}^{n} f(y_i | x_i; w, \sigma) f(w)$$

Assume $\phi = I$

$$\propto \exp\left(\sum_{i=1}^{n} (y_i - w^T x_i)^2 - \frac{1}{2\alpha^2} w^T w\right)$$

for MAP estimate, we need to find the mode of the posterior.

$$\therefore \ln\left[f(w|y)\right] = \frac{-1}{2\sigma^2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 - \frac{1}{2\alpha^2} w^T w + c$$

$\Rightarrow$ maximizing the above is equivalent to minimizing the cost of the reg. least squares.

Thus, regularized least square is the MAP estimate with some prior assumption on the parameter.

Iterative algorithms for LR:

we have for OLS,

$$L(w) = \frac{1}{2}\sum_{i=1}^{n}(x_i^T w - y_i)^2$$

We saw a closed-form expression for this cost. However, this can be solved through an iterative scheme.

We know that the gradient of a fn at any point gives the direction of steepest descent.

Thus, moving small steps along the gradient at every point could potentially lead us to the optima. - This is the idea behind the gradient descent algo.

$$\nabla_w L = \sum_{i=1}^{n} x_i (x_i^T w - y_i)$$

Thus, one can obtain an iterative algo as follows.

$$w(k+1) = w(k) - \underset{\text{learning rate}}{\eta} \sum_{i=1}^{n} x_i (x_i^T w(k) - y_i)$$

The above version is called the 'batch version' of the gradient descent algorithm - It uses all the 'training samples' at once for each iteration.

Instead one can have an ~~bat~~ incremental version when only one training sample is used per iteration. in which case the update steps become : $w(k+1) = w(k) - \eta x_k \cancel{(k)} [x_k^T w_k - y_k]$.

The perceptron learning algorithm.

One of the earliest algos for learning W^*

Iterative algorithm iterating over all training samples, picking one at a time.

Let W_K, X_K & Y_K denote the estimate for the ~~se~~ W^* at the K^{th} iteration, K^{th} feature vector & the labels respectively.

The perceptron algo:

Let $\Delta w_K = w_{K+1} - w_K$, then

$$\Delta w_K = 0 \quad \text{if} \quad \begin{array}{l} w_K^T x_K > 0 \ \& \ y_K = 1 \quad \text{or} \\ w_K^T x_K < 0 \ \& \ y_K = 0 \end{array}$$

$$= x_K \quad \text{if} \quad w_K^T x_K \leq 0 \ \& \ y_K = 1$$

$$= -x_K \quad \text{if} \quad w_K^T x_K \geq 0 \ \& \ y_K = 0.$$

$$w_{K+1} = \underline{w_K + \Delta w_K}.$$

This is an error correction algo that corrects the error locally.

Consider what is it doing.

when $w_k^T x_k \le 0$ & $y_k = 1$,

$$w_{k+1}^T x_k = w_k^T x_k + x_k^T x_k$$
$$\ge w_k^T x_k$$

Illy, when $w_k^T x_k \ge 0$ & $y_k = 0$

$$w_{k+1}^T x_k \le w_k^T x_k.$$

Thus, it is puting the hyperplane towards 'Correctness' at every sample. (Intuitive)

~~At~~ However, there is no guarentee that $w_{k+1}^T x_k$ has the correct sign

Also, correcting w_k to take care of x_k may misclassify some other feature vector,

However, it works always!.

Claim: The perceptron algorithm converges to ~~the~~ (finds a seperating hyperplane) in finite no of steps if it exists.

Proof: Let us multiply all x_i in training set with $y_i = 0$ by -1. (for the sake of ease of algebra).

Now, if w is a separating hyperplane if

$$w^T x_i > 0 \quad \forall i.$$

under this notation, perceptron algo is

$$w_{k+1} = w_k + x_k \quad \forall k \text{ s.t } w_k^T x_k \leq 0.$$

An iteration is counted only when there is an update.

with this perceptron algo is

$$w_{k+1} = w_k + x_k, \quad w_k^T x_k \leq 0, \quad k = 0, 1, \ldots$$

stop when w^* is found.

Goal: To show k is bounded.

Proof by Contradiction: Assume that the algo fails to find a separating hyperplane.

If algo fails to find a separating hyperplane then we must have

$$w_K x_K^T \leq 0, \quad \forall K$$ [note that K is only when corrections are made]

we have,

$$w_{K+1} = w_K + x_K \quad \forall K.$$

$$\|w_{K+1}\|^2 = \|w_K + x_K\|^2$$

$$= \|w_K\|^2 + \|x_K\|^2 + 2 w_K^T x_K$$

$$\leq \|w_K\|^2 + \|x_K\|^2 \quad -(1)$$

$$\because w_K^T x_K \leq 0 \quad \forall K.$$

recurring on (1),

$$\|w_K\|^2 \leq \|w_0\|^2 + \sum_{i=0}^{K-1} \|x_i\|^2$$

without loss of gen, assume $w_0 = 0$.

Let $M = \max_i \|x_i\|^2$

$$\|w_k\|^2 \leq \|w_0\|^2 + \sum_{i=0}^{k-1} \|x_i\|^2$$

$$\leq kM.$$

Also, $w_{k+1} = w_k + x_k \;\; \forall k,$

$$w_k = w_{k-1} + x_{k-1}$$

recursing,

$$w_k = w_0 + \sum_{i=0}^{k-1} x_i$$

[w_k is a linear combo of x_i]

Since data is linearly separable,

$\exists\, w^*$, s.t $x_i^T w^* > 0 \;\; \forall i$.

Let $\gamma = \min_i x_i^T w^*$

$$w_k^T w^* = \left(\sum_{i=0}^{k-1} x_i\right)^T w^* \geq k\gamma > 0.$$

$$K^2 \nu^2 \leq |W_K^T W^*|^2$$

$$\leq \|W_K\|^2 \|W^*\|^2$$

$$\leq \|W^*\|^2 KM$$

The above should be true for all K if the algo keeps updating W.

or $K^2 \nu^2 \leq \|W^*\|^2 KM$

However this can be true only till,

$$K \leq \underbrace{\frac{\|W^*\|^2 M}{\nu^2}}_{\text{finite.}}$$

Thus, the algo always finds a separating hyperplane in finitely many iterations.

This is the first provably 'correct' ML algorithm.

Perceptron can also be learned in the batch vs incremental versions.

Let $S_K = \{ j : w_K^T x_j \leq 0 \}$

then perceptron algo:

$$w_{K+1} = w_K + \sum_{j \in S_K} x_j$$

Perceptron can also be looked at from a risk minimization perspective.

Let $L(y, W^T x) = 0$ if $\cancel{y W^T x}$ $\text{sign}(W^T x) > 0$

$= W^T x$ if $\text{sign}(W^T x) < 0$

Then empirical risk

$$R(h) = \sum_{i=1}^{n} L(y_i, W^T x_i)$$

$$= \sum_{i : W^T x_i \leq 0} W^T x_i$$

Now, Lets do a gradient descent on this cost

$$w_{k+1} = w_k - \eta \nabla L(w_k)$$

$$= w_k + \eta \sum_{j : w^T x_j \leq 0} x_j$$

which is exactly the batch version of perceptron.

Thus, changing objectives functions (loss) results in different algorithms.

The idea of gradient descent.

We saw: ALL ML problems involve minimization of an objective function [often Empirical risk].

While closed-form solution can be obtained in a few cases its often not the case:
Risk is often non-converse & high-dimensional.

Thus, its a good idea to minimize the cost (risk) iteratively.

In other words, change the parameters (w) such that every step reduces the cost.

This is the basic idea behind gradient descent.

Question: which direction to move towards in each iteration?

Let $R(w)$ be the risk function.

Objective: Find a direction of w in which R decreases the fastest (most).

We know that directional derivative of a function is the slope of the fn along that dire

Let w denote a unit vector. we want

$$w^* = \min_{w,\, w^Tw=1} w^T \nabla_w R(w)$$

$$= \min \|w\| \, \|\nabla_w R(w)\|_2 \cos\theta.$$

min will occur when $\theta = \pi$. $\theta \to$ angle b/w w & $\nabla_w R(w)$

Thus, w^* has to point to a direction opposite to the gradient.

Thus, $w_{k+1} = w_k - \eta \nabla_w R(w)$ is an iterative procedure one can use to minimize the risk: One can use this in linear least sq as well.

Fisher linear discriminant.

A linear disc fn based classifier:

Decide $x \in C\text{-}1$ if $W^T x + w_0 > 0$

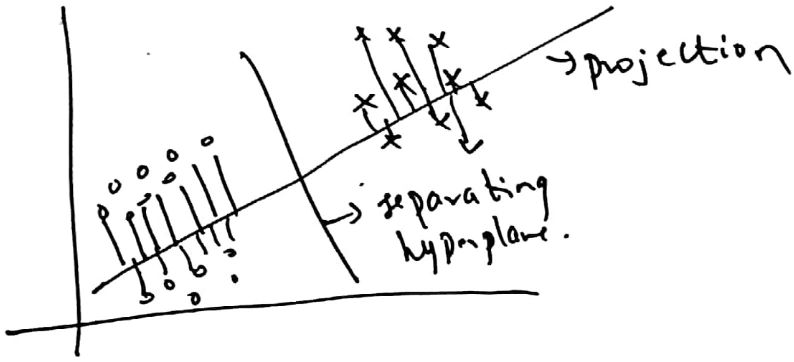
we saw OLS as a way to compute W.

~~But~~ $g(x) = W^T x + w_0$ is basically a projec-
-tion of data (x) on to a 1-D hyperplane.

In the linearly separable case, we saw that there can exist infinite seperating hyperplanes.

Question: Is there a way to quantify the goodness of projected w?

One way: separation b/w projected points of different classes should be 'more'.
FLD formalizes this notion.



FLDA idea: find a W s.t training data is well-separated along this W.

Consider 2- class case.

let $D = \{(x_i, y_i), i = 1, \dots, n\}$ $y_i \in \{0, 1\}$.

Let C_0 & C_1 denote two classes.

Let n_0 & n_1 be the no of examples of each class.

Let M_0 & M_1 be the means of data from C_0 & C_1 resp.

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

let m_0 & m_1 be the means of the projected data.

i.e, $m_0 = W^T M_0$ & $m_1 = W^T M_1$ [all these are vectors]

Intuitively, $(m_0 - m_1)$ gives a measure of the separation b/w the samples of 2 classes after projection onto W.

Thus, we want a W that maximizes

$(m_0 - m_1)^2$.

only having this in the cost has 2 problems

1) scaling the data would change W.
2) The intra-class variability could be lost [trivially by projecting all points in a class to a single point].

(1.6.7) .

Thus, let us define

$$S_0^2 = \sum_{x_i \in C_0} (W^T x_i - m_0)^2 ;$$

$$S_1^2 = \sum_{x_i \in C_1} (W^T x_i - m_1)^2$$

S_0^2 & S_1^2 quantifies the projected variances

$\therefore$ $S_0^2 + S_1^2$ is a measure of total within cluster ~~va~~ scatter.

Thus, $\dfrac{(m_0 - m_1)^2}{S_0^2 + S_1^2}$ quantifies the separability w.r.t to the total variance of the projected data.

~~As~~ Thus $L(\omega)_{FLDA} = \dfrac{(m_1-m_0)^2}{S_0^2+S_1^2}$

$$(m_1-m_0)^2 = (W^T M_1 - W^T M_0)^2$$

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

$$= W^T S_B W$$

$$\underset{d\times d}{S_B} = (M_1-M_0)(M_1-M_0)^T$$

Between class scatter matrix.

Now, $S_0^2 = \sum_{x_i \in C_0} (W^T x_i - W^T M_0)^2$

$$= \sum_{x_i \in C_0} [W^T(x_i - M_0)]^2$$

$$= \sum_{x_i \in C_0} W^T (x_i - M_0)(x_i - M_0)^T W$$

$$= W^T \left[\sum_{x_i \in C_0} (x_i - M_0)(x_i - M_0)^T\right] W$$

(149)

$$\text{IIIly}, \quad S_1^2 = W^T \left[\sum_{x_i \in C_1} (x_i - M_1)(x_i - M_1)^T \right] W$$

Thus, $S_0^2 + S_1^2 = W^T S_W W$ where

$$\underset{d \times d}{S_W} = \sum_{x_i \in C_0} (x_i - M_0)(x_i - M_0)^T + \sum_{x_i \in C_1} (x_i - M_1)(x_i - M_1)^T$$

within class scatter matrix.

$$\therefore \quad L_{FLDA}(W) = \frac{W^T S_B W}{W^T S_W W} \rightarrow$$ max b/w class scatter & min within class scatter.

Note that L is not affected by scale of W.

$$\text{Now,} \quad W^*_{FLDA} = \arg\min L(W)$$

$$\Rightarrow \frac{\partial L}{\partial W} = 0 \Rightarrow \frac{2 S_B W}{W^T S_W W} - \frac{W^T S_B W}{(W^T S_W W)^2} 2 S_W W = 0$$

$$\Rightarrow S_W W = \lambda S_B W \quad \text{for some cnt } \lambda.$$

(150).

$S_W W = \lambda S_B W$ $\qquad$ $Ax = \lambda Bx$

Generalized EV problem - solved using LU decomposition.

However, often S_W is invertible [symmetric] in which case,

$$\underline{W = \lambda S_W^{-1} S_B W}$$

Also, $S_B W = (M_1 - M_0)(M_1 - M_0)^T W = K(M_1 - M_0)$

$K = (m_1 - m_0)$

$$\therefore \underline{W_{FLDA} = S_W^{-1}(M_1 - M_0)}.$$

The final classifier would be $\underline{\text{sign}(W^T x + b)}$.

FLDA as a special case of OLS.

Given original training data, ~~oth~~ Lets construct a new training data as follows,

$y_i' = \frac{n}{n_0}$ if $y_i = 0$ & $y_i' = \frac{-n}{n_1}$ if $y_i = 1$

Now, lets get $\hat{y} = W^T x + b$ using LLS.

It can be shown that LLS gives FLDA.