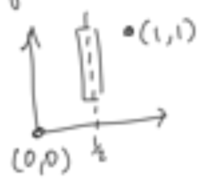
First, quick review of last class's concept check

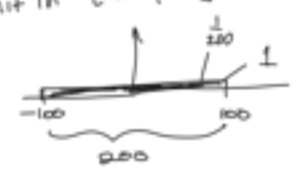


⇒ noted small noise → parabola will go through the pts.

let's suppose: $\hat{y} = a_0 + a_1 x + a_2 x^2$

plug in (0,0): $a_0 = 0$
plug in (1,1): $1 + a_1 + a_2$; write : $a_2 = 1 - a_1$

put an improper prior on a_2 → easier a_2: unif in $[-100, 100]$
$p(a_2) = \frac{1}{200}$



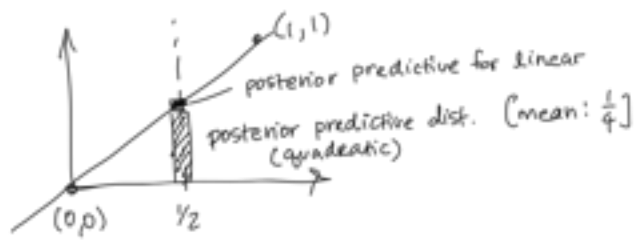
a_1: unif in $[-1, 1]$ $p(a_1) = \frac{1}{2}$

what is the posterior predictive for $x = \frac{1}{2}$

$$y(x=\tfrac{1}{2}) = a_1\left(\tfrac{1}{2}\right) + a_2\left(\tfrac{1}{4}\right)$$
$$= a_1\left(\tfrac{1}{2}\right) + (1-a_1)\left(\tfrac{1}{4}\right)$$
$$= \tfrac{1}{4} + \tfrac{1}{4}a_1$$

} linear ; uniform prior over a_1

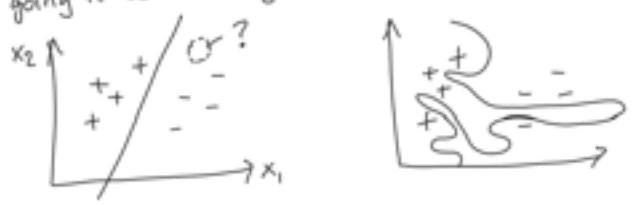
$a_1 = -1$: $\hat{y} = 0$
$a_1 = 1$: $\hat{y} = \frac{1}{2}$



if we had a linear model?

Onto classification.

$\hat{y}$ is going to be a categorical variable.



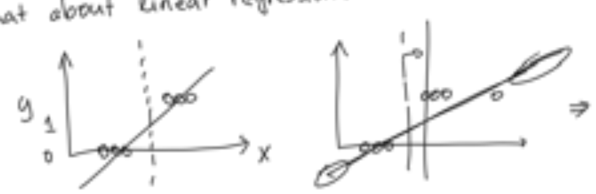
Conceptually same framework as regression...
→ we need decide on a model class? (what kind of separators?)
→ we need decide on a loss function?

Notation: $\boxed{\hat{y} \in \underbrace{C_1 \ldots C_K}_{\text{discrete/countable.}}}$ ⇒ if y is binary, $y = -1/1$ or $y = 0/1$

if y is K-ary ; $\underbrace{[0\ 0\ 1\ 0\ 0]}_{\text{"one hot" encoding}}$

Can we just use our regression tools?
→ KNN still works (nonparametric regression → classification)
(avg) (majority vote)

→ what about linear regression?



⇒ can't just apply linear reg to the 0/1 encoding of classes.

Linear Classification

$\hat{y} = \text{sign}(w^Tx + w_0) = \text{sign}(w^Tx)$ here $\hat{y} \in \{-1, 1\}$

if $w^Tx < 0$, $-1 = \hat{y}$

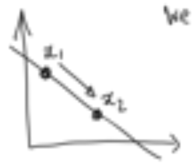
$w^Tx > 0$, $1 = \hat{y}$

$w^Tx + w_0 = 0$ is the decision boundary

in 2D case: $0 = w_0 + w_1 x_1 + w_2 x_2$

$$x_2 = \underbrace{\frac{-w_1}{w_2}}_{\text{slope}} x_1 + \underbrace{\frac{-w_0}{w_2}}_{\text{offset}}$$ clearly a line

generalize to multiple dims $(D>2)$: consider a vector connecting 2 points on the boundary:



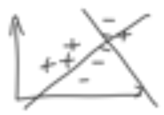
we know: $w^Tx_1 + w_0 = 0$, $w^Tx_2 + w_0 = 0$ by definition

consider $s = x_2 - x_1$ lies on the boundary

$$w^Ts = w^Tx_2 - w^Tx_1 = \underbrace{w^Tx_2 + w_0}_{0} \underbrace{- w_0 - w^Tx_1}_{0} = 0$$

implication is that w is orthogonal to the boundary

Now: let's introduce a loss function. [esp. important if we cannot perfectly separate the data]:



- $l_{0/1}(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{else} \end{cases}$

$$\mathcal{L}(w) = \sum_n l_{0/1}\underbrace{\Big(-\underbrace{y_n}_{\substack{\text{true value}\\(\pm 1)}} \underbrace{(w^Tx_n + w_0)}_{\substack{\text{predicted value, before}\\\text{sign function}}}\Big)}_{\substack{\text{this is positive IF sign of}\\ y_n \neq \text{sign of } w^Tx_n + w_0}}$$

$\Rightarrow$ uninformative gradient!

$=$ # of mis-classified examples.

- alternative: hinge loss / rectified linear function

$l_{hinge} = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{else} \end{cases}$ $\max(0, z)$

$$\mathcal{L}(w) = \sum_n l_{hinge}(-y_n (w^Tx + w_0))$$ $\rightarrow$ ASIDE: use any $f(x, w)$ here

$$= \sum_{\substack{n \text{ that}\\ \text{are incorrect}}} -(w^Tx_n + w_0)y_n$$

we can take gradients! $\frac{\partial}{\partial w}\mathcal{L}(w) = -\sum_{\substack{\text{bad}\\ y\text{'s}}} y_n x_n$

Interesting connection: old algorithm, 1958, Perceptron Algorithm (for linear classification)

- loop through the data
  - for each mis-classified pt. if $\hat{y}_n \neq y_n$,

    $w \leftarrow w + y_n x_n$

until no errors; will converge if a perfect separator exists.

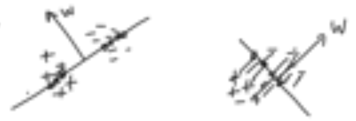
New idea: stochastic gradient descent, use a minibatch of the data when computing the derivative.

relevant: big data.

properties re convergence

↳ look at percepreon: that's just SGD w/ minibatch size = 1

Just briefly: one more loss function: if w projects x onto 1D, $(w^T x)$ suppose we want some kind of "separation" or "clustering" of the different classes...



(Fisher's Discriminant)

define empirical means & variances: $m_1 = \frac{1}{N_1}\sum x_n$ s.t. $y_n = C_1$

$m_2 = \frac{1}{N_2}\sum x_n$ s.t. $y_n = C_2$

$S_1 = \frac{1}{N_1}\sum (x_n - m_1)^T(x_n - m_1)$

$S_2 = \frac{1}{N_2}\sum (x_n - m_2)^T(x_n - m_2)$

then: $w^T x$: $m_1' = w^T m_1$ $\quad v_1 = w^T S_1 w$

$m_2' = w^T m_2$ $\quad v_2 = w^T S_2 w$

$$\min \; \mathcal{L}(w) = \frac{-(m_1' - m_2')^2}{v_1 + v_2} \quad \begin{matrix} \} \text{ put means far apart} \\ \} \text{ make variances small} \end{matrix}$$

we can optimize this loss in closed form: $w \propto (S_1 + S_2)^{-1}(m_1 - m_2)$

Take a step back & discuss additional metrics

Errors:

y	$\hat{y}$	
1	1	True positive
0	1	False positive
0	0	True Neg.
1	0	False neg.

⇒ option: directly put costs on each error.

now, we can combine these diff. errors to get various quantities:

accuracy: $\frac{TP + TN}{TP + TN + FP + FN}$

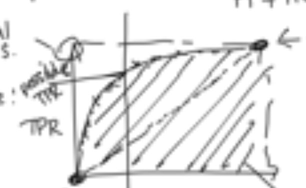
precision: $\frac{TP}{TP + FP}$

recall: $\frac{TP}{TP + FN}$ } all positives

true pos rate: $\frac{TP}{TP + FN}$

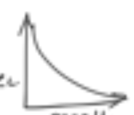
false pos rate: $\frac{FP}{FP + TN}$

ROC curve:



$w^T x$ is a number; we can sweep over w_0 to get diff classifiers...

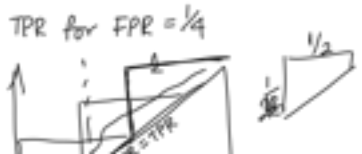
may also see



Concept Check w/ AUCs:

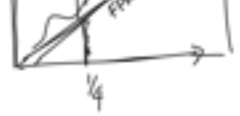
suppose you have a model w/ AUC = $\frac{1}{2} + \frac{1}{8} = \frac{5}{8} = .625$ (low AUC)

1. What are possible TPR for FPR = 1/4



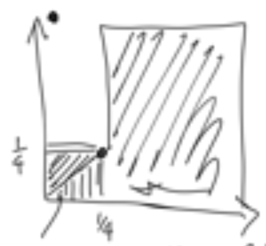
$TPR = \frac{1}{4}$ $\quad FPR = \frac{1}{4}$

TPR = 1 $\quad$ FPR = 1



TPR $= \frac{1}{2}$, FPR $= \frac{[illegible]}{4}$

2. How does this change if AUC was bigger? What's the highest AUC you can have and still TPR = 1/4 , FPR = 1/4 ?



$\frac{1}{16} + \frac{.3}{4} = \frac{10}{16} = .8125$

Classification