ROC Analysis

10/19

Binary Classification

Classification w/ $k=2$

True Positive (TP): # of positive pts predicted as pos
False Positive (FP): # of positive pts predicted as neg
False Neg (FN): # of neg pts predicted as pos
True Neg (TN): # of neg pts predicted as neg

Sensitivity (True positive rate) $TPR = recall_p = \frac{TP}{TP+FN}$

Specificity (True neg rate) $TNR = recall_N = \frac{TN}{FP+TN}$

False Neg Rate: $FNR = 1 - \text{sensitivity}$

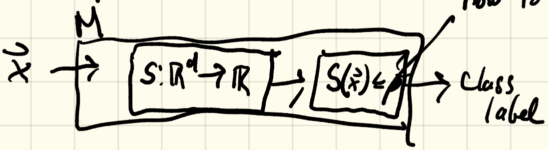
False Pos Rate: $FPR = 1 - \text{specificity}$

ROC Analysis
↳ Receiver Operating Characteristic

- binary classifier w/ cutoff ρ
- gives preformane over all values of ρ
- plot of FPR (1-specifcty) vs true positive rate (sensitivity)

Setup for classifier M

how to pick ρ



Let S be a score function $S: \mathbb{R}^d \to \mathbb{R}$
$\vec{x}_i \in D$

$$\rho^{min} = \min_i \{ S(\vec{x}_i) \}$$

$$\rho^{max} = \max_i \{ S(\vec{x}_i) \}$$

$\forall$ distinct values of ρ (e.g. each $S(\vec{x}_i)$)
compute $R_1(\rho) = \{ \vec{x}_i \in D \mid S(\vec{x}_i) > \rho \}$

→ compute TP & FP
→ FPR vs TPR
→ a point

Note:

$\rho^{max} \Rightarrow$ all pts are neg
$\Rightarrow$ FPR = 0
TPR = 0
→ gives pt $(0,0)$

	True	
Predicted	Pos	Neg
Pos	0	0
Neg	*FN*	*TN*

(a) Initial: all negative

$\rho^{min} \Rightarrow$ all pts are pos
$\Rightarrow$ FPR = 1
TPR = 1
→ $(1,1)$ for the plot

	True	
Predicted	Pos	Neg
Pos	*TP*	*FP*
Neg	0	0

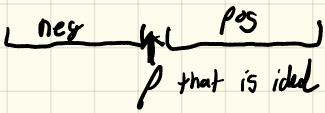
(b) Final: all positive

"best" value of ρ
FPR = 0
TPR = 1
→ $(0,1)$ in the plot

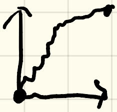
	True	
Predicted	Pos	Neg
Pos	*TP*	0
Neg	0	*TN*

(c) Ideal classifier

Good score function



Typical ROC plot

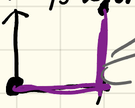


Ideal Plot

area is 1

about as good as random guessing

always wrong



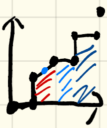
area is 0

area is $\frac{1}{2}$

Area under the ROC curve (AUC)

- measurement of classifiers performance
- volue between [0, 1]
  - ↳ 1 is good
  - 0 is bad

1. sort $(S(\vec{x}_i), y_i)$ by decresing score
2. plot each point & accumulate area



Algorithm 22.1: ROC Curve and Area under the Curve

```
ROC-CURVE(D, M):
n_1 ← |{x_i ∈ D | y_i = c_1}| // size of positive class
n_2 ← |{x_i ∈ D | y_i = c_2}| // size of negative class
// classify, score, and sort all test points
L ← sort the set {(S(x_i), y_i) : x_i ∈ D} by decreasing scores
FP ← TP ← 0
FP_prev ← TP_prev ← 0
AUC ← 0
ρ ← ∞
foreach (S(x_i), y_i) ∈ L do
    if ρ > S(x_i) then
        plot point (FP/n_2, TP/n_1)
        AUC ← AUC + TRAPEZOID-AREA((FP_prev/n_2, TP_prev/n_1), (FP/n_2, TP/n_1))
        ρ ← S(x_i)
        FP_prev ← FP
        TP_prev ← TP
    if y_i = c_1 then TP ← TP + 1
    else FP ← FP + 1
plot point (FP/n_2, TP/n_1)
AUC ← AUC + TRAPEZOID-AREA((FP_prev/n_2, TP_prev/n_1), (FP/n_2, TP/n_1))

TRAPEZOID-AREA((x_1, y_1), (x_2, y_2)):
b ← |x_2 − x_1| // base of trapezoid
h ← 1/2 (y_2 + y_1) // average height of trapezoid
return (b · h)
```

Compare Classifiers

Evaluate classifier

Split data into 2 disjoint sets
- training set: Learn M
- test set: evaluate performance measure Θ

- What if we just got lucky (or unlucky) w/ partition of D?
- We want to estimate $E[\Theta]$

K-fold cross validation helps avoids

1. Pick $K \in \mathbb{Z}^+$ (usually 5-10)
   (leave-one-out $K = n$)

2. Partition shuffle(D) K equal (ish) sized partitions called folds
   $\mathcal{D} = \{D_1, \ldots, D_K\}$

3. $\forall D_i \in \mathcal{D}$

   create M_i w/ $\mathcal{D} \setminus D_i$
   evaluate Θ_i w/ M_i and D_i as test set

4. estimate $\hat{\mu}_\Theta = E[\Theta] = \frac{1}{K}\sum_{i=1}^{K}\Theta_i$

   w/ variance:

$$\hat{\sigma}^2_\Theta = \frac{1}{K}\sum_{i=1}^{K}(\Theta_i - \hat{\mu}_i)^2$$