MACHINE LEARNING IN PHYSICS
FOUNDATIONS 5

HARRISON B. PROSPER

PHY6938

Recap

What have we learned so far?

1. Machine learning models are trained by minimizing
$$R(\omega) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_i)$$
using some variation of **S**tochastic **G**radient **D**escent. Stochasticity is introduced by using a different batch of data, with sample size $n \ll N$, at each step.

2. The loss function, $L(y_i, f_i)$, is chosen according to the task at hand.

Recap

3. The empirical risk function (which is typically referred to as the "loss" in ML circles) is an *unbiased* Monte Carlo estimate of the risk functional

$$R[f] = \int dx \int dy\, L(y, f)\, p(x, y)$$

where $p(x, y)$ is the probability density of the data.

4. Minimizing the risk functional with respect to the ML model f shows that the best-fit model satisfies

$$\int dy\, \frac{\partial L}{\partial f} p(y \mid x) = 0$$

Recap

5. We found that the binary cross entropy loss yields a best-fit model f^* that approximates the discriminant

$$D(x) \equiv \frac{p(x|1)}{p(x|1) + p(x|0)}$$

when a balanced dataset is used where the two classes of object are labeled by $y = 0$ or 1.

6. Two commonly used measures of classifier quality are the Receiver Operating Characteristic (ROC) curve and the Area Under the (ROC) Curve (AUC).

CLASSIFIER PERFORMANCE

Classifier Performance: Counts

Given a classifier, $D(x)$, the indicator function $I[*]$, the targets y, and defining "positives" as objects with $y > \frac{1}{2}$ and "negatives" as those with $y < \frac{1}{2}$, the following counts can be computed:

1. False Negatives (FN): $\sum_{y_i=1} I[D(x_i) \leq t]$
2. True Positives (TP): $\sum_{y_i=1} I[D(x_i) > t]$
3. True Negatives (TN): $\sum_{y_i=0} I[D(x_i) \leq t]$
4. False Positives (FP): $\sum_{y_i=0} I[D(x_i) > t]$

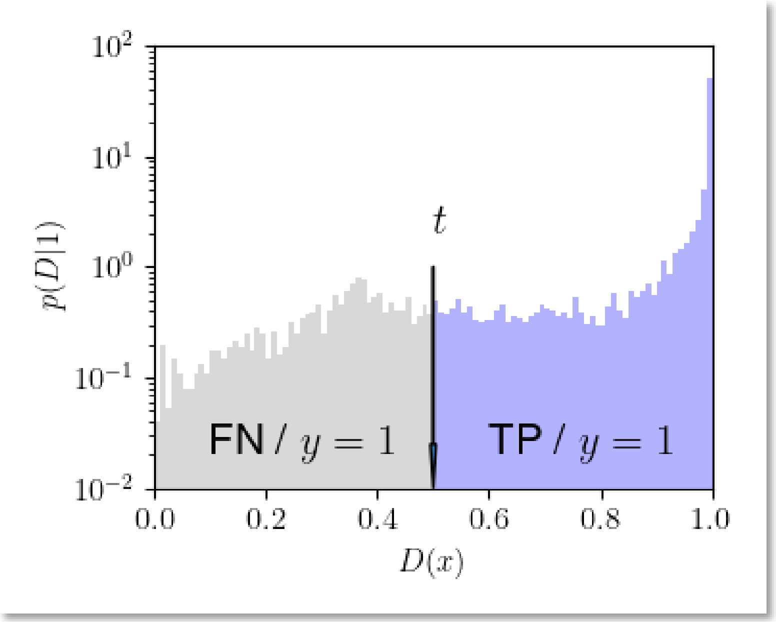
$I[z] = 1$ if z is true, 0 otherwise; t is a given threshold.

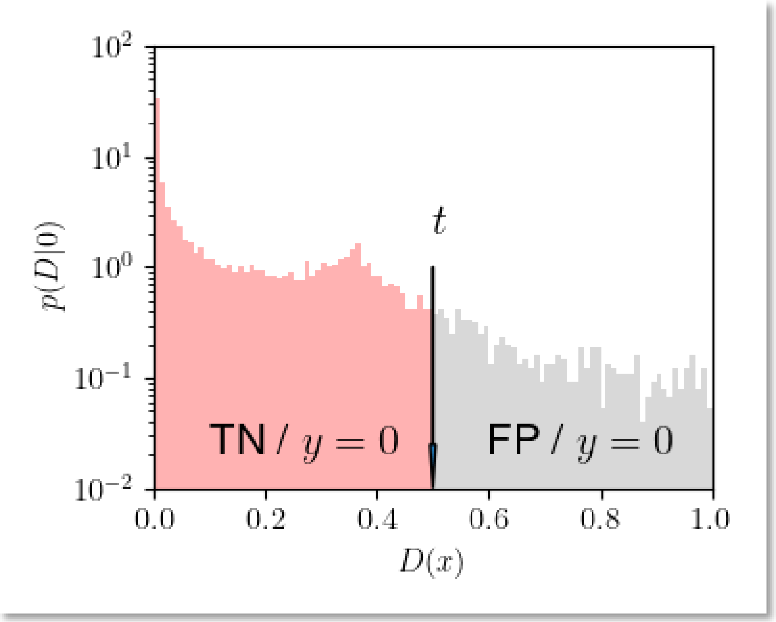
Classifier Performance: Counts

FN: $\sum_{y_i=1} I[D(x_i) \leq t]$ $t = 0.5$

TP: $\sum_{y_i=1} I[D(x_i) > t]$

$y = 0$





$y = 1$

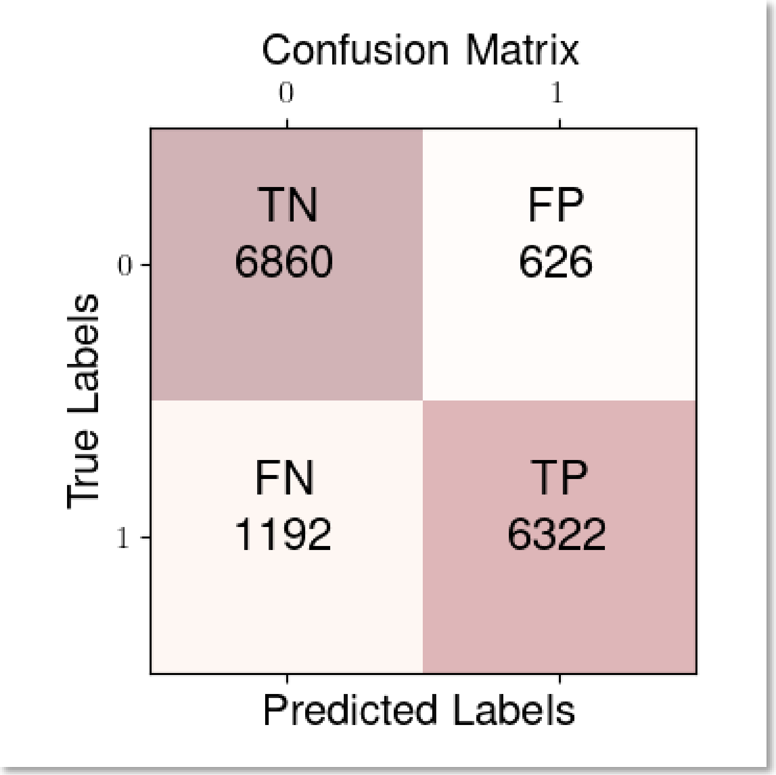
TN: $\sum_{y_i=0} I[D(x_i) \leq t]$

FP: $\sum_{y_i=0} I[D(x_i) > t]$

Confusion Matrix

FN: $\sum_{y_i=1} I[D(x_i) \leq t]$

TP: $\sum_{y_i=1} I[D(x_i) > t]$



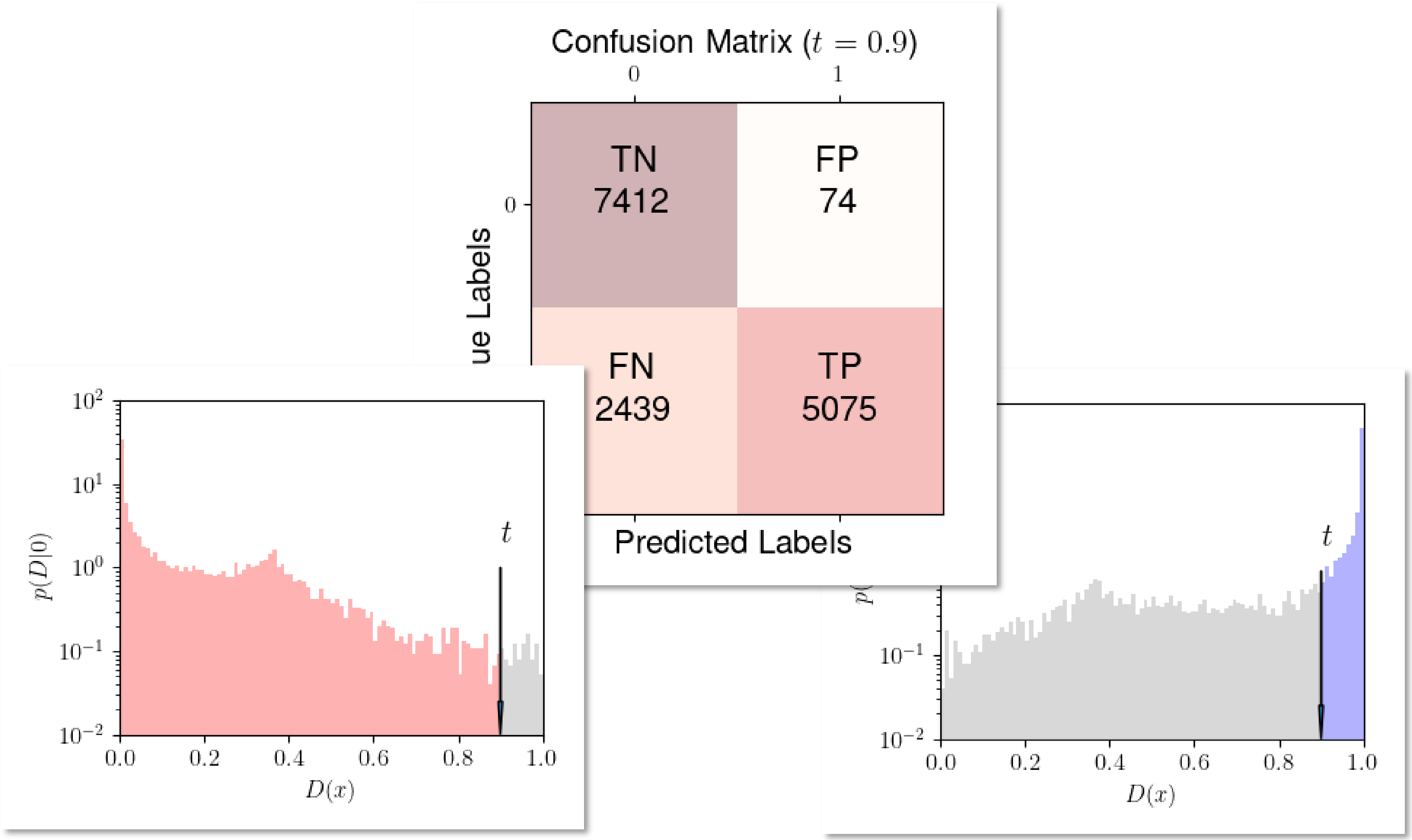
The confusion matrix is a simple way to summarize the counts in the four disjoint regions.

A perfectly separable dataset would yield a diagonal matrix.

TN: $\sum_{y_i=0} I[D(x_i) \leq t]$

FP: $\sum_{y_i=0} I[D(x_i) > t]$

Confusion Matrix ($t = 0.9$)



Accuracy, Precision, Recall

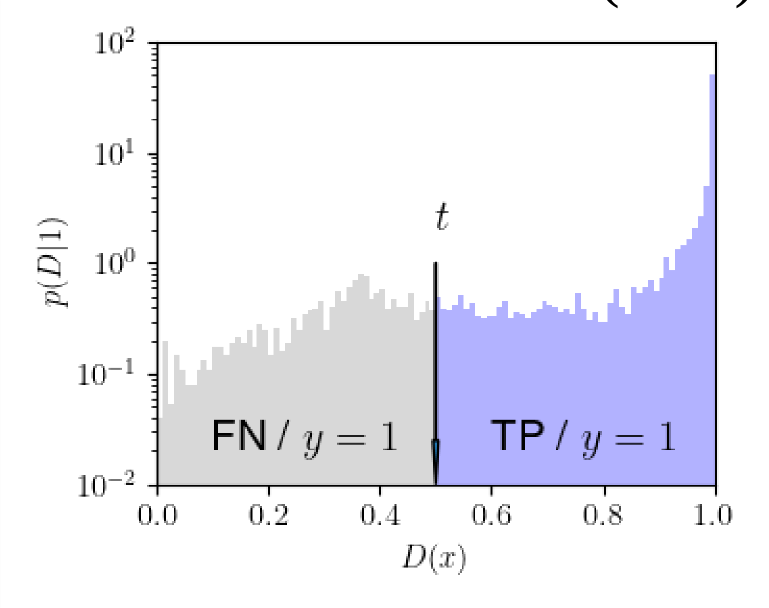
Accuracy = (TP + TN) / (FN + TP + TN + FP)

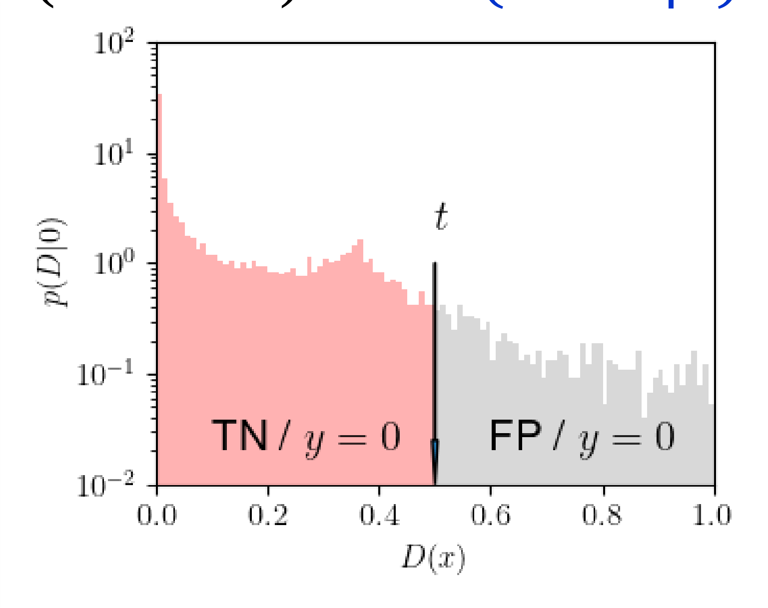
Precision = TP / (TP + FP) $P(y = 1|D > t)$

Recall = TP / (FN + TP)

True Positive Rate (TPR) = TP / (FN + TP) $P(D > t|1)$

False Positive Rate (FPR) = FP / (TN + FP) $P(D > t|0)$





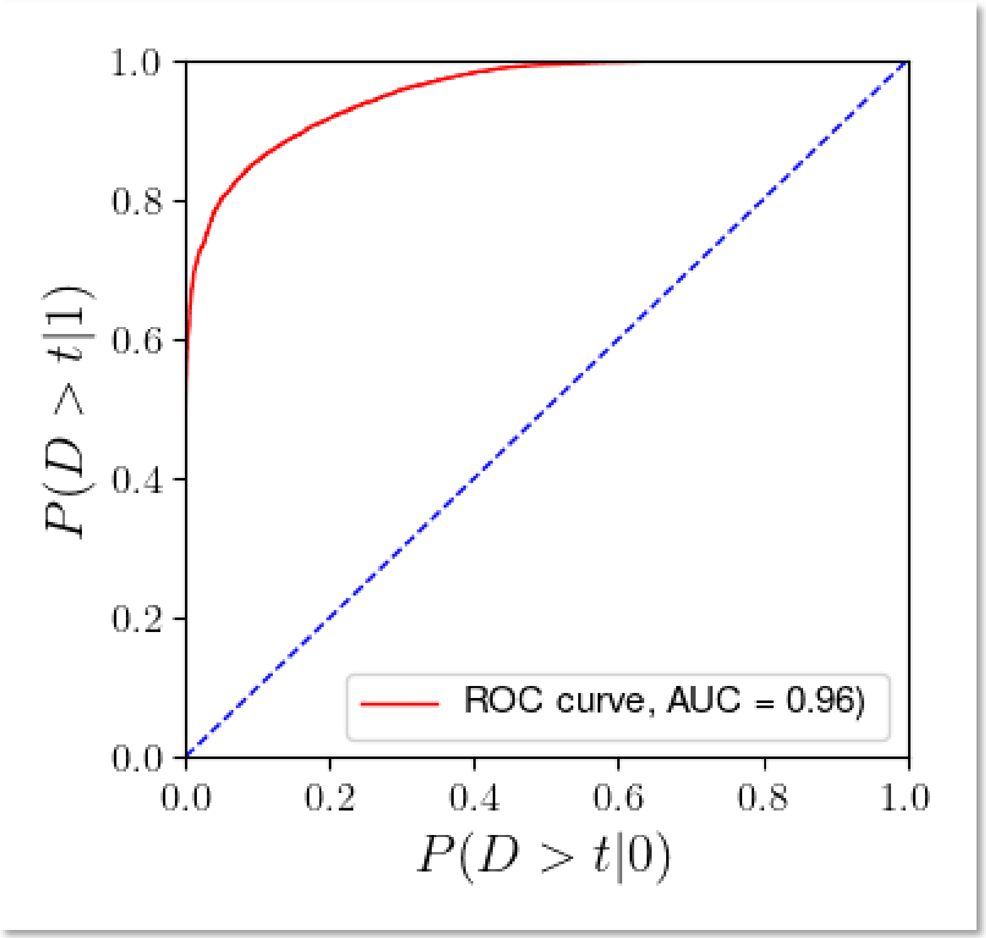
ROC and AUC

The ROC curve is a plot of TPR vs. FPR, that is,

$P(D > t|1)$ vs. $P(D > t|0)$

where (usually)

$$D(x) = \frac{p(x|1)}{p(x|1) + p(x|0)}$$



BEYOND CLASSIFICATION

Beyond Classification

By now it should be clear that *any* ML model, regardless of sophistication, when trained using binary cross entropy approximates the *same* function, namely:

$$D(x) = \frac{p(x|1)}{p(x|1) + p(x|0)}$$

Note: This function can be rewritten as follows

$$p(x|1) = p(x|0)\left[\frac{D(x)}{1 - D(x)}\right]$$

Beyond Classification

Suppose that $p(x|0)$ is a *known* d-dimensional density that approximates an *unknown* density $p(x|1)$. The expression

$$p(x|1) = p(x|0)\left[\frac{D(x)}{1 - D(x)}\right]$$

suggests a way to use machine learning to improve upon the approximation $p(x|0)$.

Density Estimation

Consider dataset $\mathrm{U}[x]$ of size N, where each data instance x, labeled $y = 1$, is sampled from an *unknown* density $u(x) \equiv p(x|1)$.

Algorithm

1. Generate another dataset $\mathrm{K}[x]$ of size N, labeled $y = 0$, where $x \sim k(x) \equiv p(x|0)$ is a *known* density.

2. Train a binary classifier $f(x, \omega)$.

3. Approximate the unknown density $u(x)$ using
$$u(x) = k(x)\left[\frac{D(x)}{1 - D(x)}\right]$$

Conditional Density Estimation

But we can go further*.

Suppose our dataset U is from a simulation that depends on a set of parameters θ.

Let's sample these parameters from a *known* prior, $\pi(\theta)$. For each point θ, sample x from the *unknown* density $u(x|\theta)$ using the simulator.

The dataset, labeled $y = 1$, comprises N pairs $\{(x, \theta)\}$ that constitute a point cloud representation of the unknown density $u(x|\theta)$.

* Baldi, P., Cranmer, K., Faucett, T. *et al.* Parameterized neural networks for high-energy physics. *Eur. Phys. J. C* **76**, 235 (2016).

Conditional Density Estimation

Algorithm

1. Generate a dataset of size N, labeled $y = 0$, where θ is sampled from the *same* prior $\pi(\theta)$. For each θ, sample $x \sim k(x|\theta)$, where $k(x|\theta)$ is a *known* density.

2. Train a classifier $f(x, \omega)$.

3. Compute $u(x, \theta) = k(x|\theta)\pi(\theta)\left[\frac{D(x,\theta)}{1-D(x,\theta)}\right]$.
   Divide by $\pi(\theta)$:
$$u(x|\theta) = k(x|\theta)\left[\frac{D(x,\theta)}{1 - D(x,\theta)}\right]$$

Summary

- In addition to the ROC curve and the AUC, the performance of a classifier can be characterized with several other numbers including:
  - False Negatives (FN)
  - True Positives (TP)
  - True Negatives (TN)
  - False Positives (FP)

- A classifier can be used to improve the approximation of a probability density by exploiting the fact that we know what a classifier approximates.