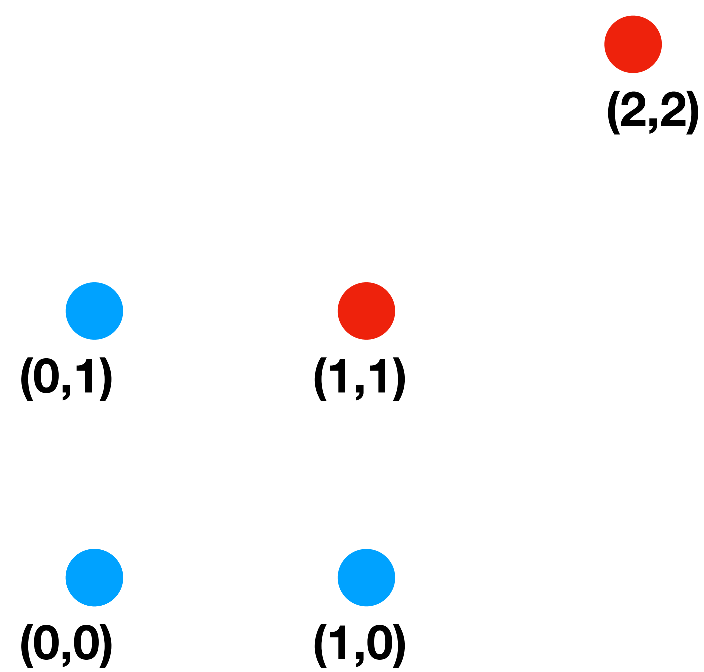
CS 4342: Class 13

Jacob Whitehill

SVMs

Exercise

- Specify a hyperplane (**w**, *b*) that would be used by an SVM for the following data:



Quadratic programming

SVM optimization problem

- Once again, we wish to:
  - Minimize: $\frac{1}{2}\mathbf{w}^\top \mathbf{w}$
  - Subject to: $y^{(i)}(\mathbf{x}^{(i)\top}\mathbf{w} + b) \geq 1 \quad \forall i$
- This is a **quadratic programming** problem: quadratic objective with linear inequality (and/or equality) constraints.
- There are many efficient solvers for quadratic programs.

Quadratic programming

- Quadratic programming is *not* a kind of computer programming.
- **Quadratic programming (QP)** problems are a kind of mathematical optimization problem:
    - Quadratic objective function (which we want to minimize or maximize).
    - Linear equality and/or inequality constraints.
- Same vein as linear programming, dynamic programming.

Quadratic programming

- Nonetheless, quadratic programs are typically *solved* using computer programs.
- As part of homework 4, you will use an off-the-shelf Python-based quadratic programming solver (**`cvxopt`**) to train an SVM.

cvxopt

`cvxopt.solvers.qp`$(P, q\,[\,, G, h\,[\,, A, b\,[\,, solver\,[\,, initvals\,]\,]\,]\,])$

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

cvxopt

Quadratic objective Linear objective

`cvxopt.solvers.qp`(*P*, *q* [, *G*, *h* [, *A*, *b* [, *solver* [, *initvals*]]]])

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

Inequality constraints Equality constraints

- To train an SVM using a QP, we need to define the appropriate matrices from our training data.
- The **x** comprises both the **w** (hyperplane) and *b* (bias) (similar to how we implemented bias in linear regression).

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

`cvxopt`

Quadratic objective

Linear objective

`cvxopt.solvers.qp`(*P*, *q* [, *G*, *h* [, *A*, *b* [, *solver* [, *initvals*]]]])

Solves the pair of primal and dual convex quadratic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + q^T x \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

Inequality constraints

Equality constraints

- **q** will just be 0 (we have no linear objective function).
- **P**: part of homework 4.

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

cvxopt

Quadratic objective

Linear objective

`cvxopt.solvers.qp`(*P*, *q* [, *G*, *h* [, *A*, *b* [, *solver* [, *initvals*]]]])

Solves the pair of primal and dual convex quadratic programs

$$
\begin{array}{ll}
\text{minimize} & (1/2)x^T Px + q^T x \\
\text{subject to} & Gx \preceq h \\
 & Ax = b
\end{array}
$$

Inequality constraints

Equality constraints

- **G** and **h** need to encode the linear inequality constraints.

- We will not use **A** or **b** (optional parameters) since we have no equality constraints.

http://cvxopt.org/userguide/coneprog.html#quadratic-programming

Defining G and h

- Suppose you have just two optimization variables — x_1 and x_2 — as well as the following constraints:
  - $2x_1 - 3x_2 \leq 2$
  - $x_1 + x_2 \geq 0$
- We need to express these both as linear inequality constraints (≤ 0).

Defining G and h

- Suppose you have just two optimization variables — x_1 and x_2 — as well as the following constraints:
  - $2x_1 - 3x_2 \leq 2$
  - $x_1 + x_2 \geq 0 \quad \Rightarrow \quad -x_1 - x_2 \leq 0$
- We need to express these both as linear inequality constraints (≤ 0).

Defining G and h

- Suppose you have just two optimization variables — x_1 and x_2 — as well as the following constraints:
  - $2x_1 - 3x_2 \leq 2$
  - $x_1 + x_2 \geq 0 \quad \Rightarrow \quad -x_1 - x_2 \leq 0$
- We need to express these both as linear inequality constraints (≤ 0).

$$\underbrace{\left[\begin{array}{cc} 2 & -3 \\ -1 & -1 \end{array}\right]}_{\mathbf{G}} \underbrace{\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]}_{\mathbf{x}} \leq \underbrace{\left[\begin{array}{c} 2 \\ 0 \end{array}\right]}_{\mathbf{h}}$$

More on accuracy metrics

True positives, false positives; true negatives, false negatives

- After training a classifier, we can apply the machine to new data to estimate the class label as $\hat{y}$.

- Examples:

  - Step-wise classification:

$$\begin{aligned} g^{(j)}(\mathbf{x}) &= \mathbb{I}[\mathbf{x}_{r_1,c_1} > \mathbf{x}_{r_2,c_2}] \\ \hat{y} = g(\mathbf{x}) &= \mathbb{I}\left[\left(\frac{1}{m}\sum_{j=1}^{m} g^{(j)}(\mathbf{x})\right) > 0.5\right] \end{aligned}$$

  - The machine always outputs either 1 or 0.

True positives, false positives; true negatives, false negatives

- After training a classifier, we can apply the machine to new data to estimate the class label as $\hat{y}$.
- Examples:
  - Logistic regression: $\hat{y} = \sigma(\mathbf{x}^\top \mathbf{w})$
  - Although $\hat{y}$ is probabilistic (in (0,1)), we can apply a **threshold** (e.g., τ=0.5) to convert to a hard label:

$$\hat{y} = \mathbb{I}[\sigma(\mathbf{x}^\top \mathbf{w}) > \tau]$$

True positives, false positives; true negatives, false negatives

- Once both $y, \hat{y} \in \{0, 1\}$, we can compute the number of:
  - **True positives (TP)**: The number of positive examples ($y^{(i)}=1$) estimated by the machine to be positive ($\hat{y}^{(i)}=1$).
  - **False positives (FP)**: The number of negative examples ($y^{(i)}=0$) estimated by the machine to be positive ($\hat{y}^{(i)}=1$).
  - **True negatives (TN)**: The number of negative examples ($y^{(i)}=0$) estimated by the machine to be negative ($\hat{y}^{(i)}=0$).
  - **False negatives (FN)**: The number of positive examples ($y^{(i)}=1$) estimated by the machine to be positive ($\hat{y}^{(i)}=0$).

True positives, false positives; true negatives, false negatives

- We can also compute the *rate* of TP, FP, TN, FN:
  - **True positive rate**: The fraction of positive examples ($y^{(i)}=1$) estimated by the machine to be positive ($\hat{y}^{(i)}=1$).
  - **False positive rate**: The fraction of negative examples ($y^{(i)}=0$) estimated by the machine to be positive ($\hat{y}^{(i)}=1$).
  - **True negative rate**: The fraction of negative examples ($y^{(i)}=0$) estimated by the machine to be negative ($\hat{y}^{(i)}=0$).
  - **False negative rate**: The fraction of positive examples ($y^{(i)}=1$) estimated by the machine to be positive ($\hat{y}^{(i)}=0$).

True positives, false positives; true negatives, false negatives

- We can also compute the *rate* of TP, FP, TN, FN:
  - **True positive rate**: The fraction of positive examples ($y^{(i)}$=1) estimated by the machine to be positive ($\hat{y}^{(i)}$=1).
  - **False positive rate**: The fraction of negative examples ($y^{(i)}$=0) estimated by the machine to be positive ($\hat{y}^{(i)}$=1).
  - **True negative rate**: 1 - FPR
  - **False negative rate**: 1 - TPR

Example

- Suppose $\mathbf{y} = [\ 1, 1, 0, 0, 1\ ]$ and $\hat{\mathbf{y}} = [\ 0.7, 0.3, 0.2, 0.9, 0\ ]$.

Example

- Suppose $\mathbf{y}$ = [1, 1, 0, 0, 1] and $\hat{\mathbf{y}}$ = [1, 0, 0, 1, 0].

Apply threshold (0.5) to obtain labels in { 0, 1 }.

Example

- Suppose $\mathbf{y}$ = [1, 1, 0, 0, 1] and $\hat{\mathbf{y}}$ = [1, 0, 0, 1, 0].
- Then:
  - TPR =
  - FPR =
  - TNR =
  - FNR =

Different $\tau \Rightarrow$ Different TPR, TNR

- If you choose a different threshold τ, you will obtain different binary labels.

- Suppose $\mathbf{y}$ = [1, 1, 0, 0, 1]. If the machine's real-valued outputs are [0.7, 0.3, 0.2, 0.9, 0], then:
  - τ = -1 $\Rightarrow$ $\hat{\mathbf{y}}$ = [1, 1, 1, 1, 1] $\Rightarrow$ TPR=1, FPR=1.
  - τ = 0.19 $\Rightarrow$ $\hat{\mathbf{y}}$ = [1, 1, 1, 1, 0] $\Rightarrow$ TPR=2/3, FPR=1.
  - τ = 0.5 $\Rightarrow$ $\hat{\mathbf{y}}$ = [1, 0, 0, 1, 0] $\Rightarrow$ TPR=1/3, FPR=1/2.
  - τ = 0.7 $\Rightarrow$ $\hat{\mathbf{y}}$ = [0, 0, 0, 1, 0] $\Rightarrow$ TPR=0, FPR=1/2.

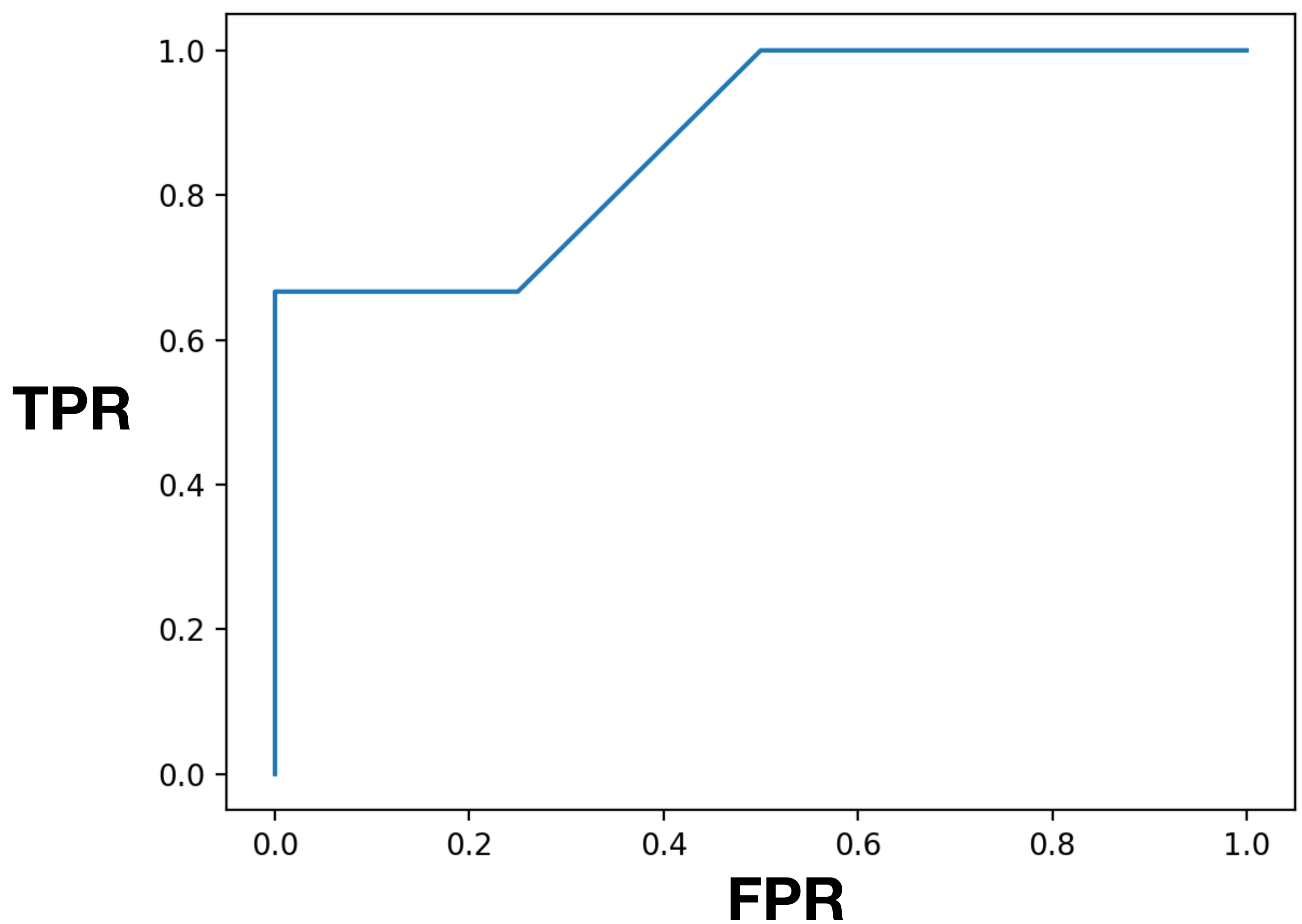
Different $\tau \Rightarrow$ Different TPR, TNR

- Higher threshold $\tau \Rightarrow$ lower TPR, lower FPR.

- Suppose $\mathbf{y}$ = [1, 1, 0, 0, 1]. If the machine's real-valued outputs are [0.7, 0.3, 0.2, 0.9, 0], then:
  - $\tau = -1 \Rightarrow \hat{\mathbf{y}} = [\ 1, 1, 1, 1, 1\ ] \Rightarrow$ TPR=1, FPR=1.
  - $\tau = 0.19 \Rightarrow \hat{\mathbf{y}} = [\ 1, 1, 1, 1, 0\ ] \Rightarrow$ TPR=2/3, FPR=1.
  - $\tau = 0.5 \Rightarrow \hat{\mathbf{y}} = [\ 1, 0, 0, 1, 0\ ] \Rightarrow$ TPR=1/3, FPR=1/2.
  - $\tau = 0.7 \Rightarrow \hat{\mathbf{y}} = [\ 0, 0, 0, 1, 0\ ] \Rightarrow$ TPR=0, FPR=1/2.

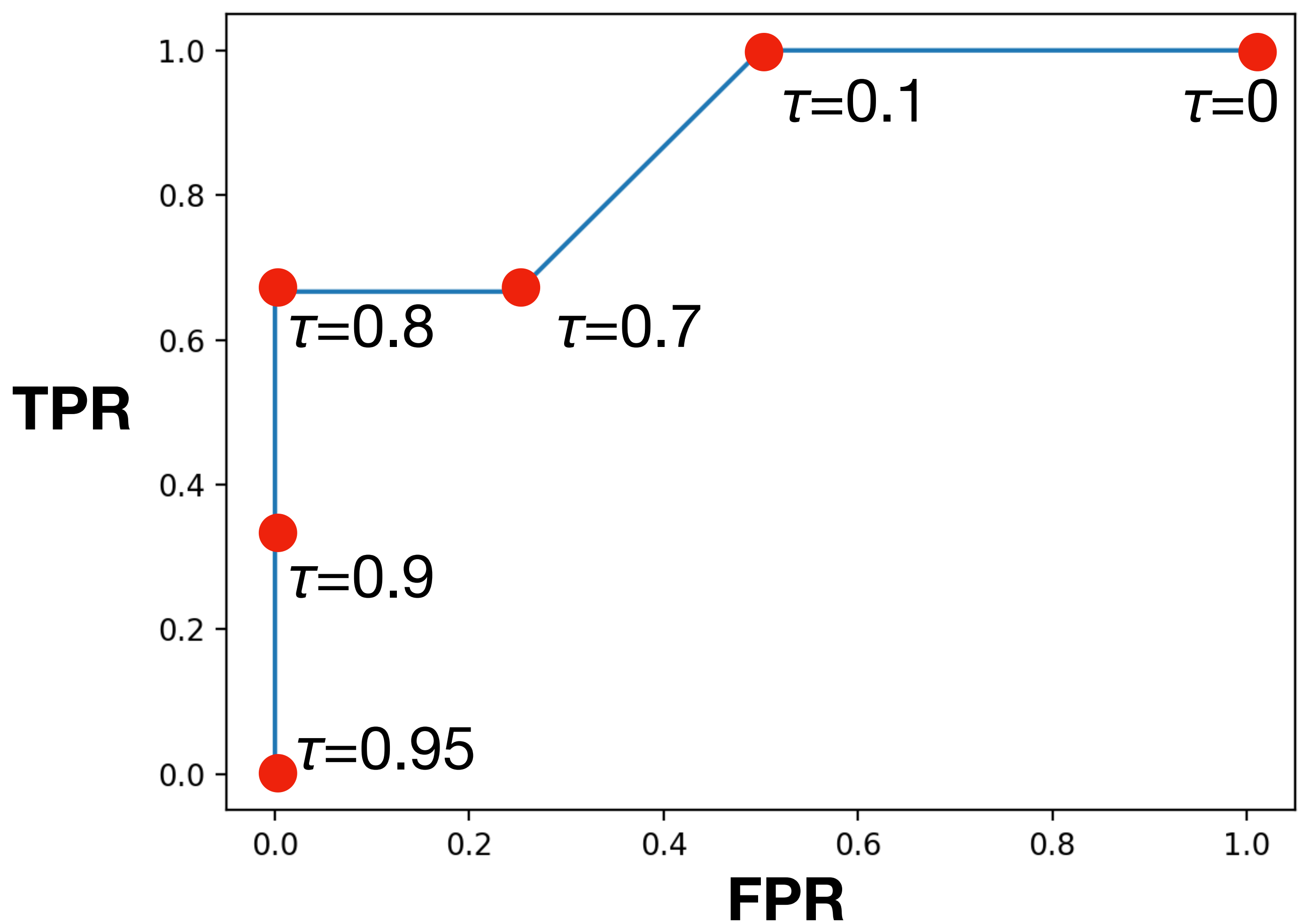
Receiver operating characteristics (ROC) curve

- If you plot the TPR v. FPR for *all possible thresholds*, you obtain the ROC curve of the classifier w.r.t. ground-truth:
    - $\mathbf{y}$ = [0, 0, 0, 0, 1, 1, 1] and $\hat{\mathbf{y}}$ = [0.1, 0.5, 0.8, 0.7, 0.9, 0.7, 0.95].



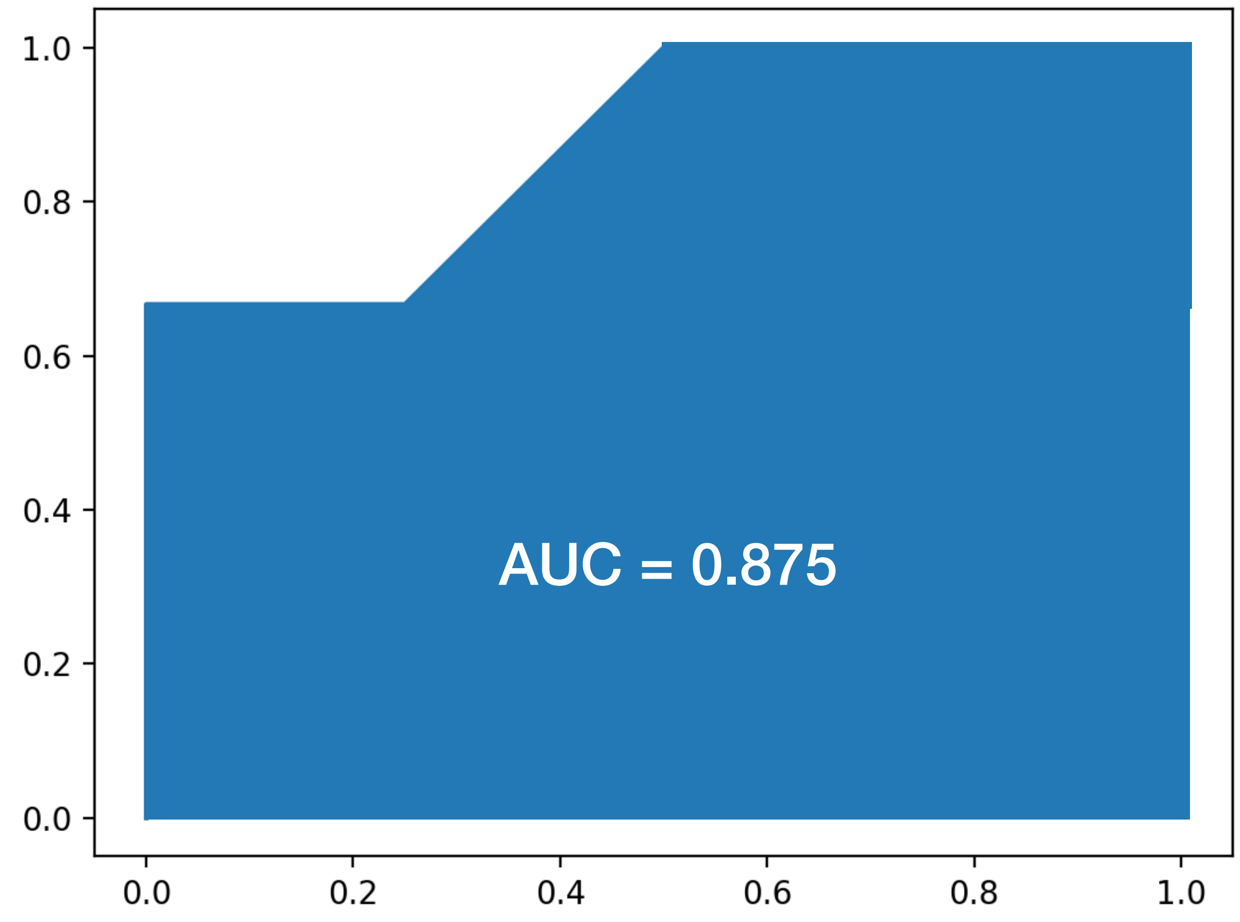
Receiver operating characteristics (ROC) curve

- If you plot the TPR v. FPR for *all possible thresholds*, you obtain the ROC curve of the classifier w.r.t. ground-truth:
  - $\mathbf{y}$ = [0, 0, 0, 0, 1, 1, 1] and $\hat{\mathbf{y}}$ = [0.1, 0.5, 0.8, 0.7, 0.9, 0.7, 0.95].



Area Under the ROC Curve (AUC)

- We can compute an aggregate metric over all possible thresholds by **integrating** the ROC curve:
    - **y** = [0, 0, 0, 0, 1, 1, 1] and **ŷ** = [0.1, 0.5, 0.8, 0.7, 0.9, 0.7, 0.95].



Area Under the ROC Curve (AUC)

- The AUC is a **threshold-independent** metric of how well the classifier discriminates between the 2 classes.
- The AUC of a classifier that always guesses correctly: 1
- The AUC of a classifier that always guesses incorrectly: 0
- The AUC of a classifier that always guesses the same value: 0.5
- The (expected) AUC of a classifier that guesses randomly: 0.5

Area Under the ROC Curve (AUC)

- The (expected) AUC is **not affected** by the ratio of positive to negative classes.
- AUC expresses how well a classifier can discriminate between two classes.
- Note that a classifier can have excellent *discriminability* but still make many *mistakes* in classification.

Area Under the ROC Curve (AUC)

- The AUC is also equivalent to the following:
  - Let (i,j) represent a randomly chosen pair of examples such that y_i=1 and y_j=0.
  - The AUC is the probability that $\hat{y}_i > \hat{y}_j$, i.e., the probability that the machine's output can correctly distinguish which example in the pair is positive.