Data 100, Discussion 13

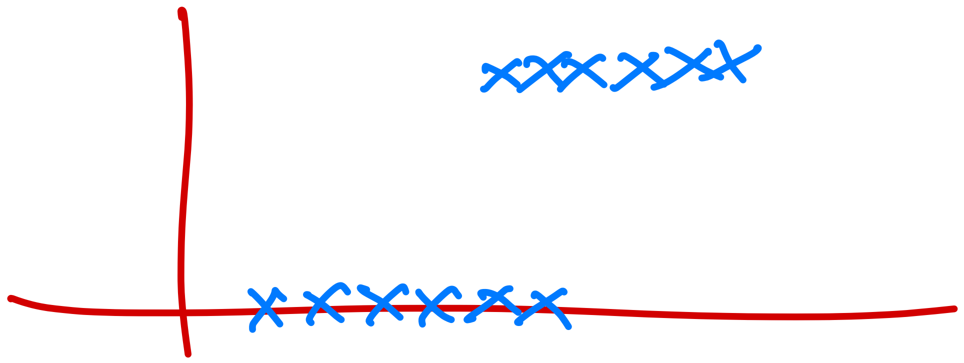
Suraj Rampure

Wednesday, November 20th, 2019

Agenda

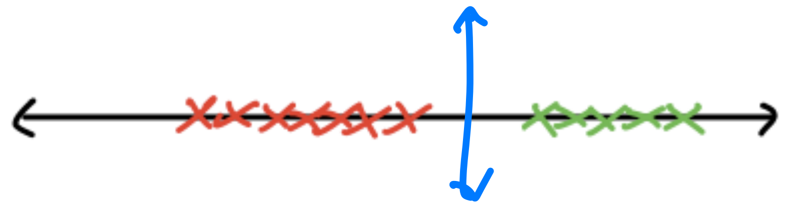
- Linear Separability
- Evaluating Classifiers
- SQL

Linear Separability

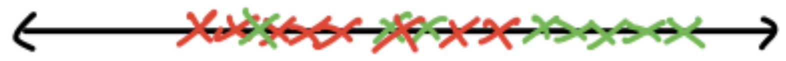


The goal of linear regression is to model the probability of a point belonging to a class. Typically, we do this when there is some uncertainty, i.e. overlap, in our training set.

In some cases, we are able to draw a *linear decision boundary* to separate our data.

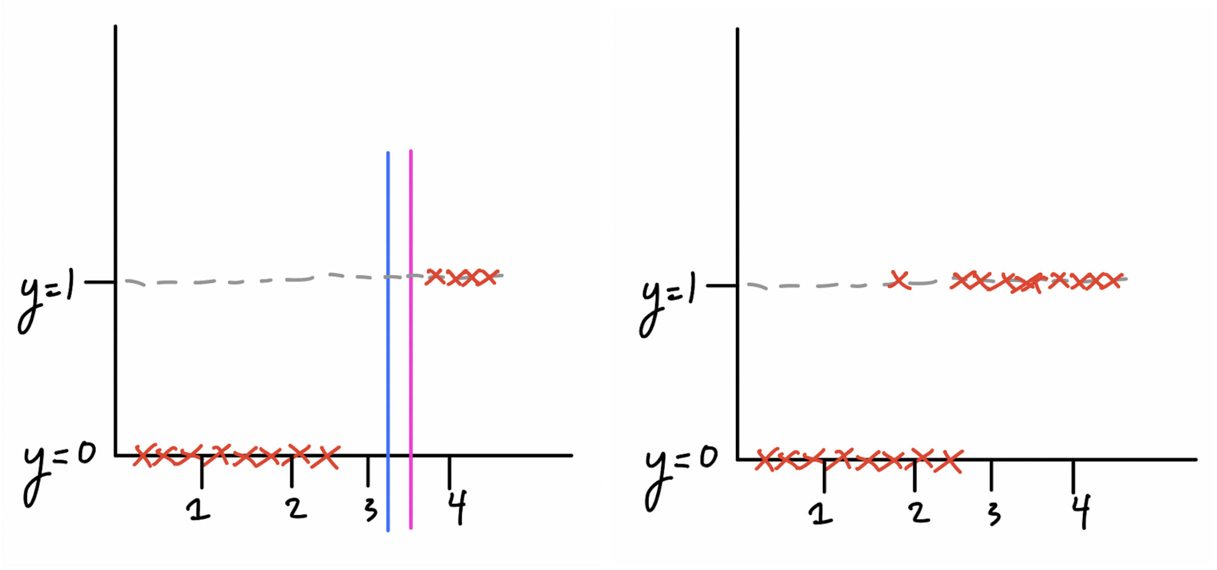


Above, we see that our data is **linearly separable**. We can draw a line (infinitely many lines, in fact) between the clusters of red dots and green dots.



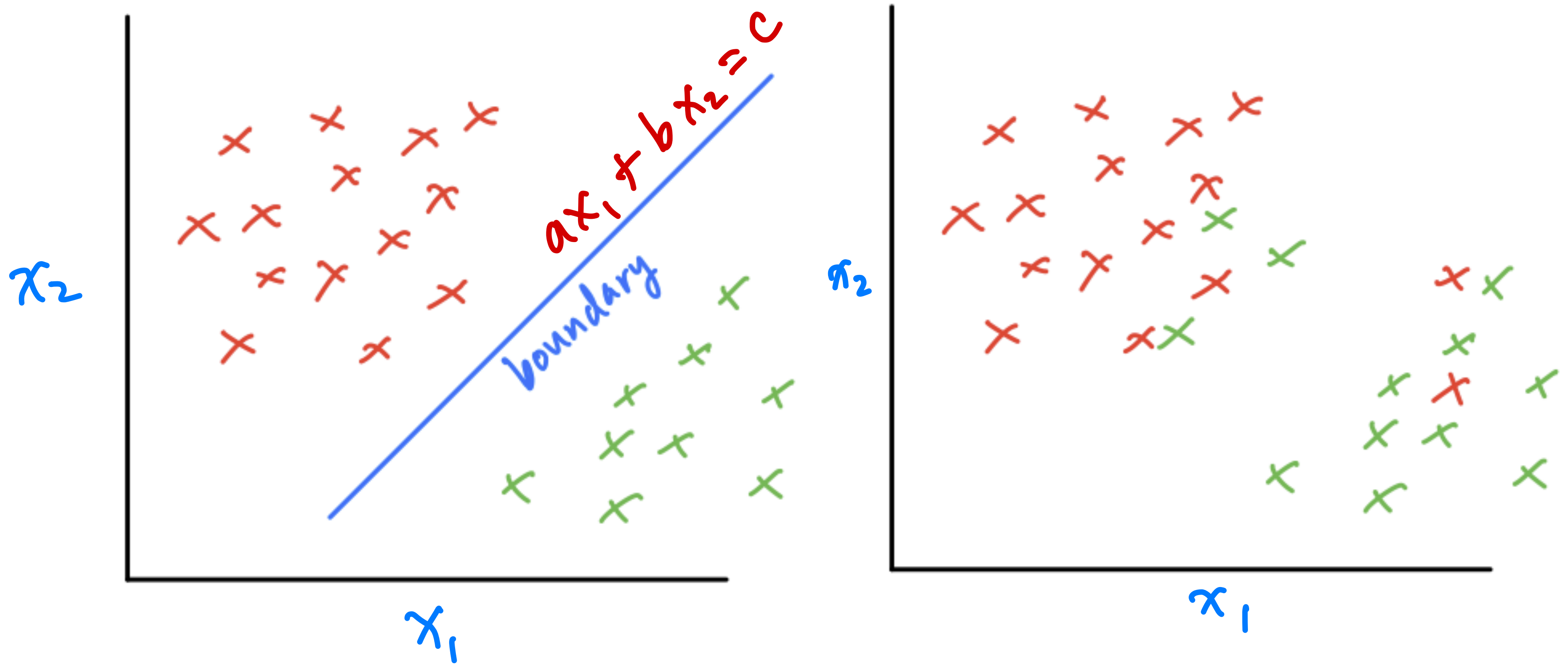
This is not the case in the second example.

Again, let's look at data in 1D, but plotted in 2D (one dimension is the value of our variable, the second dimension is the label, 0 or 1 --- this is equivalent to the drawings on the previous slide).



- On the left, we have an example of linearly separable data. In the blue and purple are two possible hyperplanes that can separate our data.
- However, on the right, we have non-linearly separable data. In this case we would use a tool like logistic regression to model probabilities.

Linear Separability in Two Dimensions



Here, we have examples of both linearly separable and non-linearly separable data in two dimensions. Here, our data is truly two dimensional, as our feature space has two components --- an x_1 and a x_2. The class is represented by the color (Y).

$d=1$: $x=a$

$d=2$: $ax_1+bx_2=c$

$\Leftrightarrow x_2=mx_1+n$

Formal Definition of Linear Separability

We say data with d dimensions is linearly separable iff you can draw a degree $d-1$ $\underbrace{\text{hyperplane}}_{\text{line}}$ that completely separates the points.

An equivalent definition: our data is linearly separable iff the following inequality **perfectly classifies it** (i.e. with 100% accuracy), for some c:

$$\underbrace{x^T\beta}_{\text{hyperplane}} \geq c$$

if above line, class 1

if below line, class 0

$\beta_1x_1+\beta_2x_2+\dots+\beta_px_p$

For example...

- If we have a single feature x, our data is linearly separable iff we can find some β, c s.t. $\beta x \geq c$ is a perfect classifier
- If we have three features x_1, x_2, x_3, our data is linearly separable iff we can find some $\beta_1, \beta_2, \beta_3, c$ s.t. $\beta_1x_1+\beta_2x_2+\beta_3x_3 \geq c$ is a perfect classifier
- Note: One or more of these β may be 0!

Cross Entropy Loss and Linear Separability

$\hat{y} = \sigma(x^T\beta)$

Recall, when using cross entropy loss, our empirical risk is of the form

$$R(\vec{\beta}) = -\frac{1}{n}\left(y\log\sigma(x^T\vec{\beta}) + (1-y)\log(1-\sigma(x^T\vec{\beta}))\right)$$

If our data is linearly separable, we don't really need to perform logistic regression, since there's no "uncertainty" – we can just skip straight to finding a separating hyperplane. However, we don't typically know this beforehand, and so we may end up trying to perform logistic regression on linearly separable data.

$\underset{t\to\infty}{\sigma(t)} \to 1$

$\underset{t\to-\infty}{\sigma(t)} \to 0$

$\sigma(t) = \frac{1}{1+e^{-t}}$

Questions:

- If our data is linearly separable, what's the minimum value of our average cross entropy loss? 0
- What value of $\vec{\beta}$ provides this? $\pm\infty$
- What problems does this cause? How can we account for this? regularize!

*Note: the optimal CE loss is never actually 0, just approaches

(b) Suppose you use gradient descent to train a logistic regression model on two design matrices $\mathbb{X}_a$ and $\mathbb{X}_b$ and use some arbitrary threshold T. After training, you find that the training accuracy for $\mathbb{X}_a$ is 100% and the training accuracy for $\mathbb{X}_b$ is 98%. What can you say about whether the data is linearly separable for the two design matrices?

$\boxed{\sigma(x^T\beta) \geq T}$ ← perfect classifier

$x^T\beta \geq \underset{\uparrow}{\sigma^{-1}(T)}$ ← eq'n of a hyperplane

"c"

X_a: ✓

X_b: ?

Evaluating the Effectiveness of a Classifier

Suppose we train a binary classifer, and suppose `y` represents actual values, and `y_pred` represents predicted values.

Recall (no pun intended) the following definitions:

- True Positives: `TP = np.count_nonzero((y == y_pred) & (y_pred == 1))`
- True Negatives: `TN = np.count_nonzero((y == y_pred) & (y_pred == 0))`
- False Positives: `FP = np.count_nonzero((y != y_pred) & (y_pred == 1))`
- False Negatives: `FN = np.count_nonzero((y != y_pred) & (y_pred == 0))`

Then, we have the following definitions:

- Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$

(b) Suppose you use gradient descent to train a logistic regression model on two design matrices $\mathbb{X}_a$ and $\mathbb{X}_b$ and use some arbitrary threshold T. After training, you find that the training accuracy for $\mathbb{X}_a$ is 100% and the training accuracy for $\mathbb{X}_b$ is 98%. What can you say about whether the data is linearly separable for the two design matrices?

$$\boxed{\sigma(x^T\beta) \geq T} \leftarrow \text{perfect classifier}$$

$$x^T\beta \geq \underset{\uparrow}{\sigma^{-1}(T)} \leftarrow \text{eq'n of a hyperplane}$$

"c"

X_a : ✓

X_b : ?

Evaluating the Effectiveness of a Classifier

We now introduce a new metric:

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN} = \frac{FP}{\text{all negatives}}$$

This tells us **the proportion of observations that were actually negative that we classified as positive**. You can think of this as being the "false alarm rate."

Similarly, we have

$$\text{True Positive Rate (TPR)} = \text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{all positives}}$$

which is the **proportion of observations that were actually positive that we classified as positive.** You can think of this as being the "~~miss~~ detection rate". (Note, this is the same as recall.)

Question: As our classification threshold increases, how do both of these quantities change?

(b) Suppose you use gradient descent to train a logistic regression model on two design matrices $\mathbb{X}_a$ and $\mathbb{X}_b$ and use some arbitrary threshold T. After training, you find that the training accuracy for $\mathbb{X}_a$ is 100% and the training accuracy for $\mathbb{X}_b$ is 98%. What can you say about whether the data is linearly separable for the two design matrices?

X_a : ✓

X_b : ?

$$\boxed{\sigma(x^T\beta) \geq T} \leftarrow \text{perfect classifier}$$

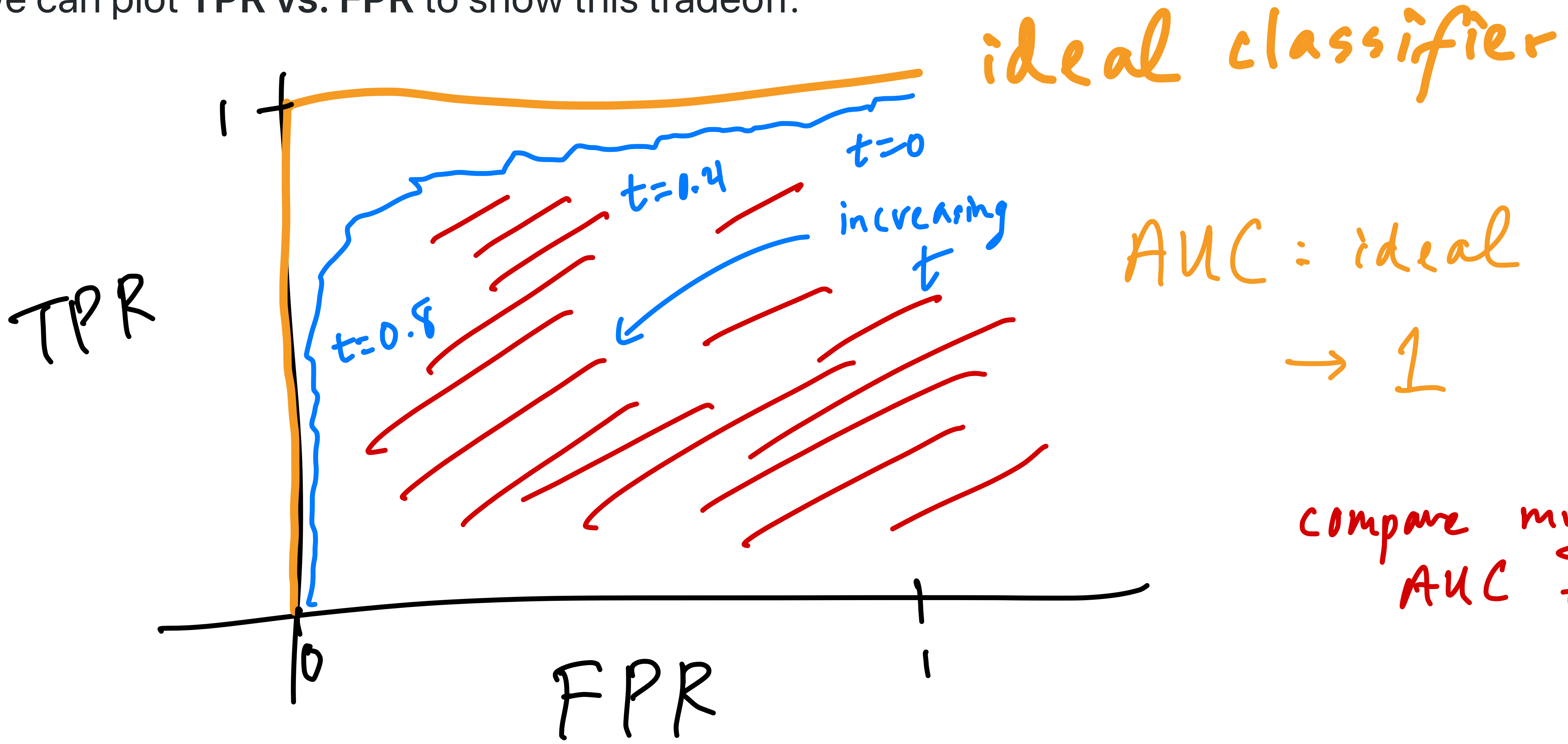
$$x^T\beta \geq \underset{\substack{\uparrow \\ \text{"c"}}}{\sigma^{-1}(T)} \leftarrow \text{eq'n of a hyperplane}$$

ROC Curves

AUC: "area under curve"

- Ideally, we would have a TPR (recall) of 1 and a FPR of 0.
- As our threshold increases, **TPR and FPR both decrease**.

We can plot **TPR vs. FPR** to show this tradeoff.



AUC: ideal → 1

compare my AUC to 1

(b) Suppose you use gradient descent to train a logistic regression model on two design matrices $\mathbb{X}_a$ and $\mathbb{X}_b$ and use some arbitrary threshold T. After training, you find that the training accuracy for $\mathbb{X}_a$ is 100% and the training accuracy for $\mathbb{X}_b$ is 98%. What can you say about whether the data is linearly separable for the two design matrices?

$$\boxed{\sigma(x^T\beta) \geq T} \leftarrow \text{perfect classifier}$$

$$x^T\beta \geq \underset{\substack{\uparrow \\ \text{"c"}}}{\sigma^{-1}(T)} \leftarrow \text{eq'n of a hyperplane}$$

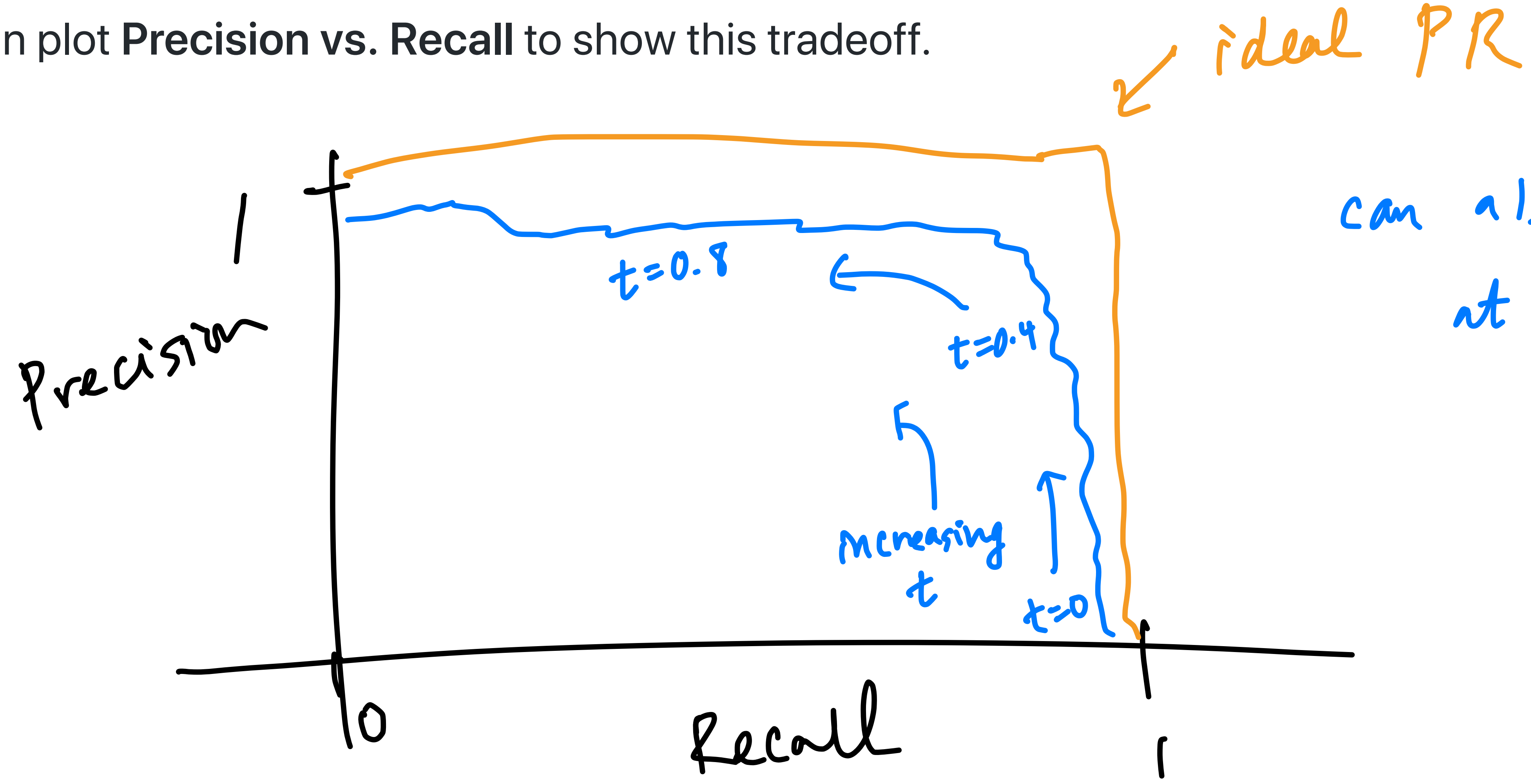
X_a : ✓

X_b : ?

PR Curves

- Ideally, we would have a precision of 1 and recall of 1.
- As our threshold increases, **precision tends to increase**, while **recall decreases**.

We can plot **Precision vs. Recall** to show this tradeoff.



can also look at AUC

Area Under Curve

- The AUC (area under curve) metric for a specific curve (ROC or PR) gives you a single number to evaluate your classifier's performance.
- It **does not** tell you how to choose your threshold. To do that, you should look at the ROC / PR curve and make a decision based off of your classification goal.

$\hookrightarrow$ 2 different AUCs: 1 for ROC, 1 for PR

SQL

Let's switch to code.