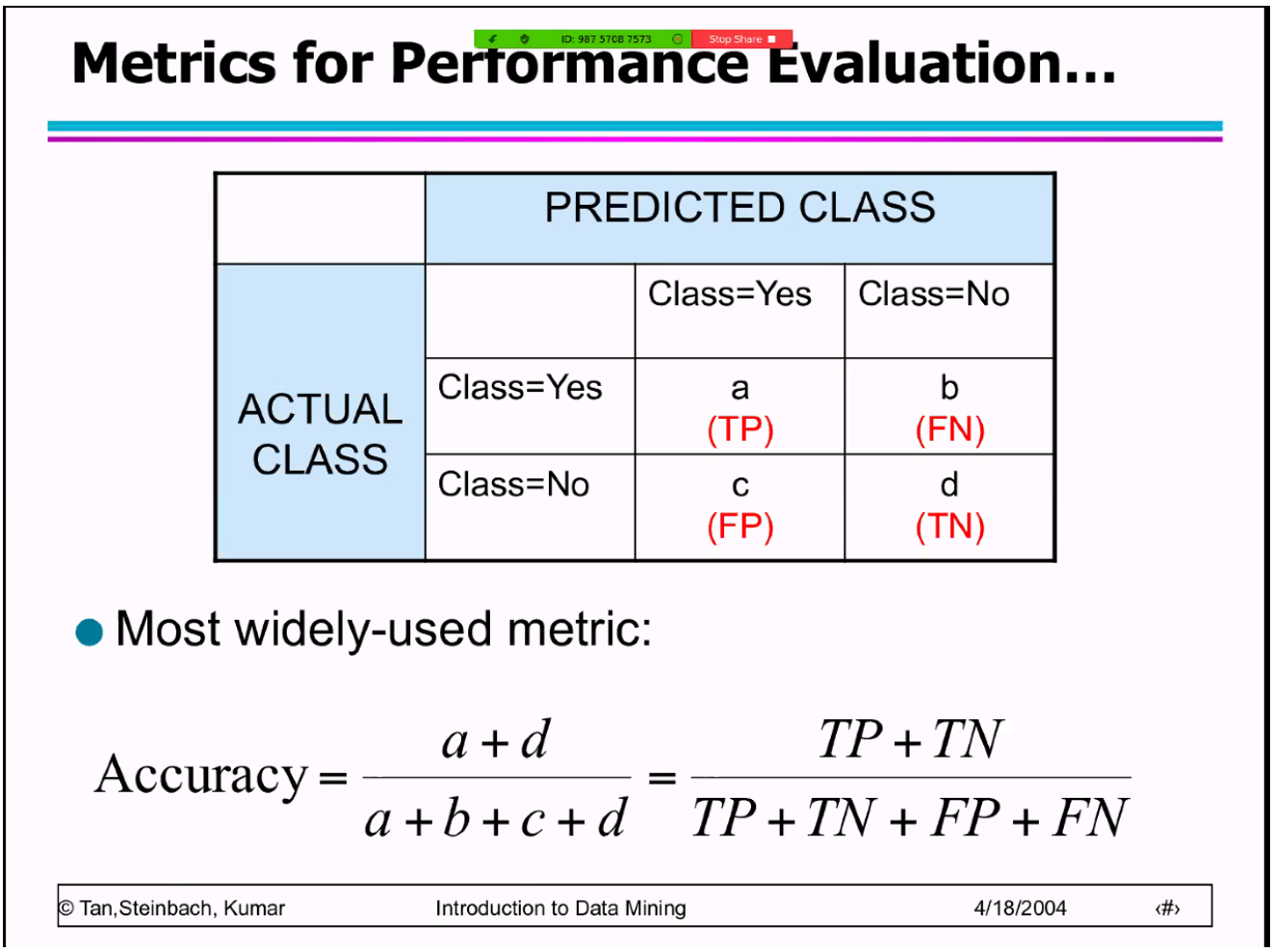
Lecture 12

Model evaluation: how can we evaluate the performance of a model? How can we obtain reliable estimates? Finally, how can we compare the relative performance among competing models?

Initially, we'll focus on the predictive capability of a model, rather than how quick it takes to classify or build the model, or scalability, etc.

Confusion matrix: a 2x2 matrix where we have predicted class as the columns, the actual class as the rows. For example, let's say we have class yes vs. no. Then, in the upper left corner (a), we have true positive, upper right (b) we have false negative (model predicted no, but actual was yes), bottom left (c) is false positive (model predicted yes, but actual is no), and then bottom right (d) is a true negative.

So, given this, what is accuracy?



Basically, how many times did we make the correct prediction vs how many total predictions did we make?

There's an inherent problem here though: if our data is not distributed evenly. For exaple, imagine class 0 haas 9990 examples, but class 1 has 10 data points. If the model predicts everything to be 0, then the model has an accuracy of 99.9%. Which is of course absurd, since the model hasn't learned anything. In this case, accuracy is misleading.

Are there alternatives? One alternative is the cost matrix: we can attribute different weights/costs to each of the different entries in the confusion matrix. Then $C(i|j)$ is the cost of misclassifying class j as class i. Now, using this cost matrix, we can list both the accuracy of a model and the cost of a model.

In general, cost can be proporational to accuracy if the cost of false negatives and false positives are the same, and the cost of true predictions are the same.

There are other metrics that we can use instead of accuracy to demonstrate the performance of a model:

- Precision: $a/(a+c)$ - how well are we doing in predicting class yes?
- Recall: $a/(a+b)$
- F-measure: combination of recall and precision.
- weighted accuracy: similar to cost function, where we weight each of a, b, c, and d.

Naive Bayes and SVM

We have so far talked about k-nearest-neighbors and decision trees. These are other tools we can use for the midterm.

In order to understand bayes, we need to know bayes theorem:

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} \tag{1}$$

Why is this important? Consider:

- A doctor knows that meningitis causes stiff neck 50% of the time
- Prior probability of any patient having meningitis is 1/50,000
- Prior probability of any patient having stiff neck is 1/20

If a patient comes in with a stiff neck, what are the chances that said patient has meningitis? It's 0.0002, not 0.5 as you might have thought. Effectively, we have $P(M|S) = \frac{0.5*1/50000}{1/20} = 0.0002$

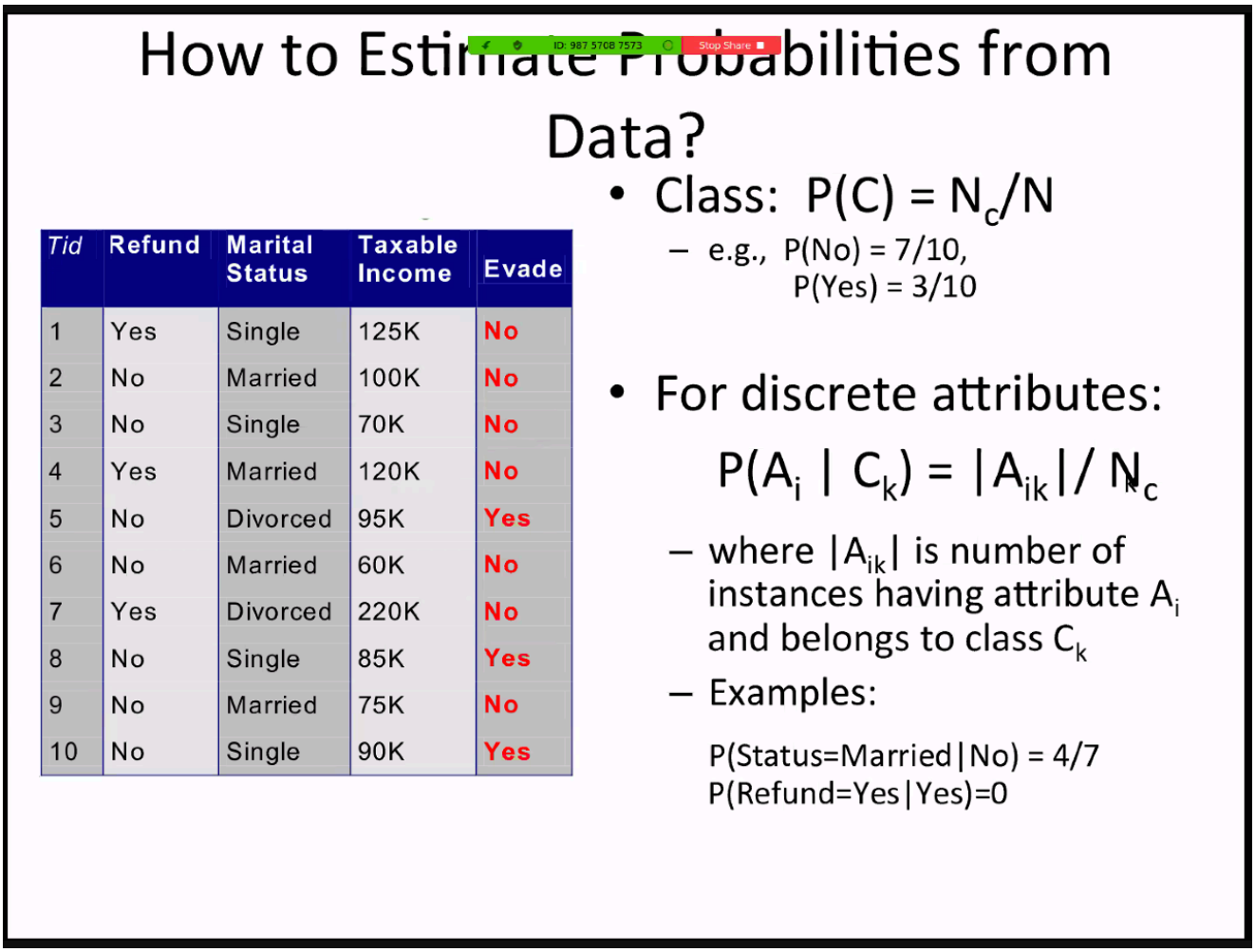
Bayesian classifiers: what we want to do here, is understand: given a set of attributes, what class do you belong to. Given a record with attributes $(A_1, A_2, \ldots, A_n)$ predict the class C. Specifically, we want to find the value of C that maximizes $P(C|A_1, A_2, \ldots, A_n)$

Is it possible to estimate this $P(C|A_1, A_2, \ldots, A_n)$ directly from the data? Well, it's not easy to compute this directly. Instead, we can use Bayes' Theorem. Here, note that the denominator won't reall change with the class, so in this analysis we can basically just ignore it. We want to maximize the numerator.

How do we compute $P(C)$? Here, we can just look at the proportion in the data set.

How do we compute $P(A_1, A_2, \ldots, A_n|C)$? This is trickier - here, we basically have to assume that each of these attributes are independent, and we can then compute this as $P(A_1, A_2, \ldots, A_n|C) = P(A_1|C)P(A_2|C)\ldots p(A_n|C)$. We can estimate each $P(A_i|C_j)$ for all A_i and C_j.

Finally, we can classify a new point to C_j if $P(C_j)\ \Pi\ P(A_i|C_j)$ is maximal.



How can we handle continuous attributes? There are a few options:

- Discretize the range into gins.
- Can do a two way split
- Probability density estimation: assume attribute follows normal distribution. Use the data to compute the mean and standard deviation, and then once the distribution is known, estimate the conditional probability.

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair
- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Naive Bayes Classifier: if we have a small dataset, we may never actually see a certain set of attributes! This is problematic because a single datapoint that we are missing can cause an entire computation to be zero, and we've lost quite a bit of information. This is relatively easy to resolve - if we've never seen the class before, then just use a constant, or possibly add one in the numerator (so we never get 0). Note that if we do this, we'd need to scale by a specific constant so that proportions still add to 1.

In general, the naive bayes classifier:

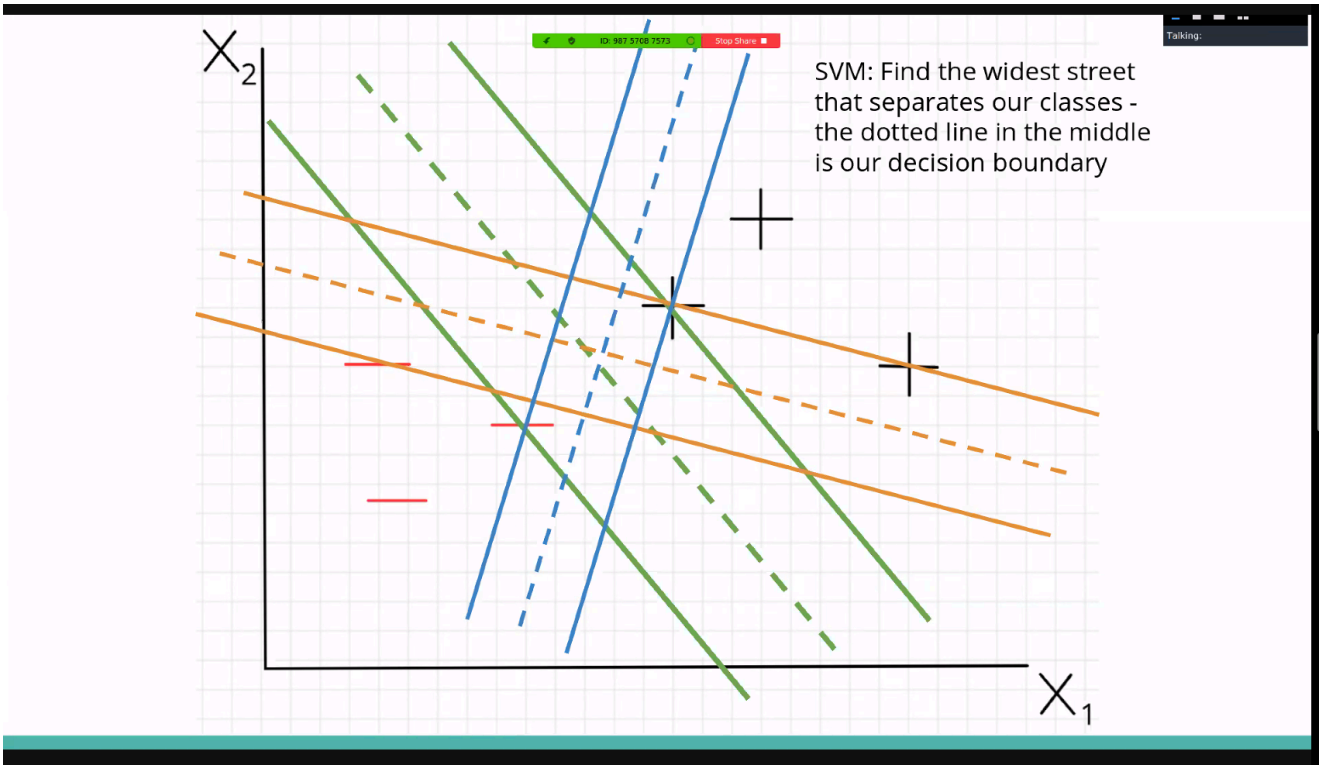
- Robust to isolated noise points
- handle missing values by ignoeing the instance during probability estimate calculations
- robust to irrelevant attributes
- Independence assumtions may not hold for some attributes. (Can use other techniques such as bayeisn belief networks.)

SVM: Support Vector Machine

For a given dataset, we may have different boundaries for decisions depending on the type of classifier used. For example, a decision tree and nearest neighbot classifier may have different decision boundaries.

Goal: How can we find the best line that separates our classes? We find this quite often: there are many different possible lines, and it isn't clear what "best" even means, and we end up having to constrain the problem space.

Let's rephrase: our goal is to find the widest street that separates our classes - the dottet line in the midde is the decision boundary, and the lines next to each dotted line define the width of the street. In some sense, we want to find the thing that maximally boundaries the different classes.



In this case, we'd be looking for the green line.

Suppose we did indeed find this decision boundary, how would we classify an unknown input $\mathbf{u}$? Let w be a vector that is perpendicular to the decision boundary. Then, in the projection of w onto u is greater than some constant, then classify it as positive, otherwise negative. In math terms, that's $\overrightarrow{w} \cdot \overrightarrow{u} + b \geq 0$. This is our decision rule.

What we need to do is find $\overrightarrow{w}$ and b. Some questions:

1. There are lots of $\overrightarrow{w}$ vectors that are perpendicular to the street - how do we know which one?
2. What is b?

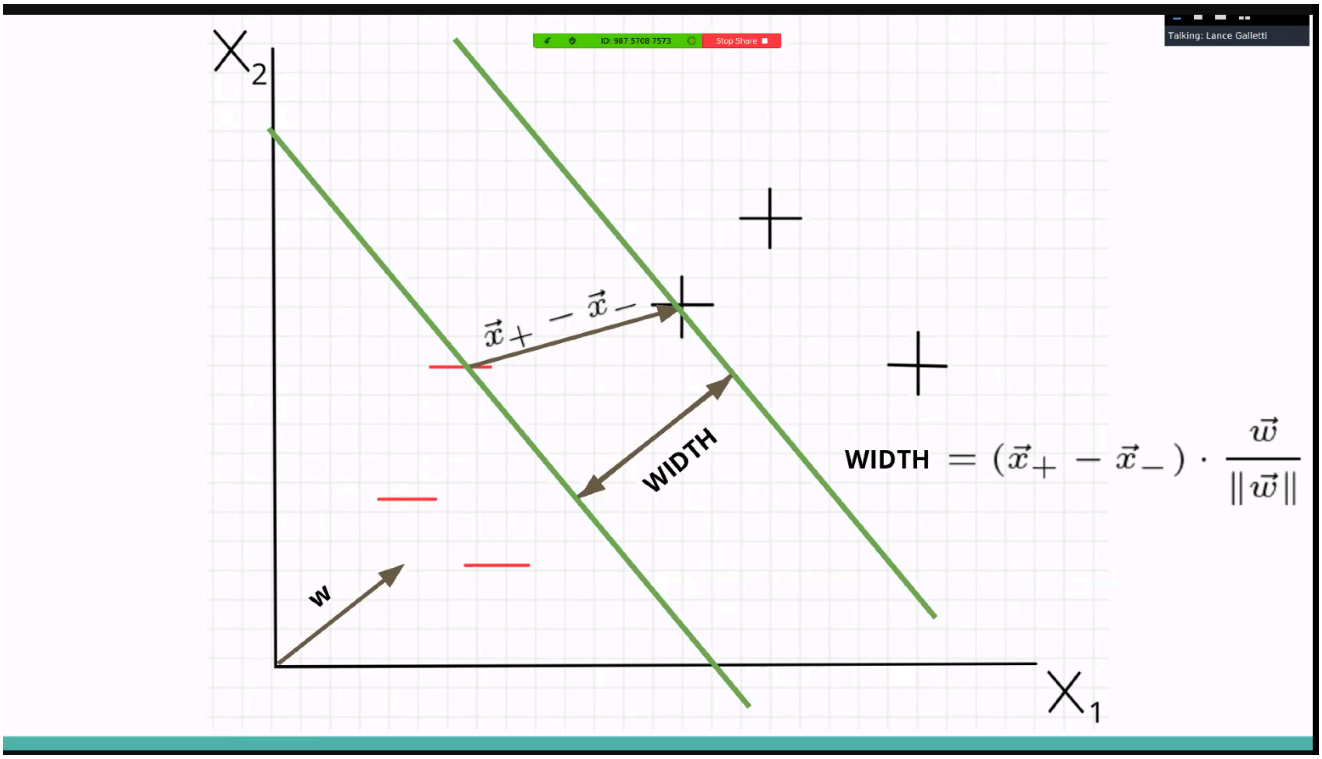
Let's state our problem a bit more mathematically. What does it mean to find the widest street? Well, we want all of our data points to not be inside the street. That means that the decision rule must have some space around it. That is, if we project a positive data point onto $\overrightarrow{w}$ and add b, then it must be greater than or equal to one, and then similarly for negative points, that need to be less than or equal to negative 1. Note that this does not apply to some unknown point u, but it does apply for everything in the data set.

How do we find the widest street? It's a bit frustrating that we have two formulas, let's combine into one by defining a new term y_i which is +1 for a positive point, and -1 for a negative point. Now, if we multiply the equation by y_i, we have a single equation: $y_i(\overrightarrow{w} \cdot \overrightarrow{x}_i) \geq 1$

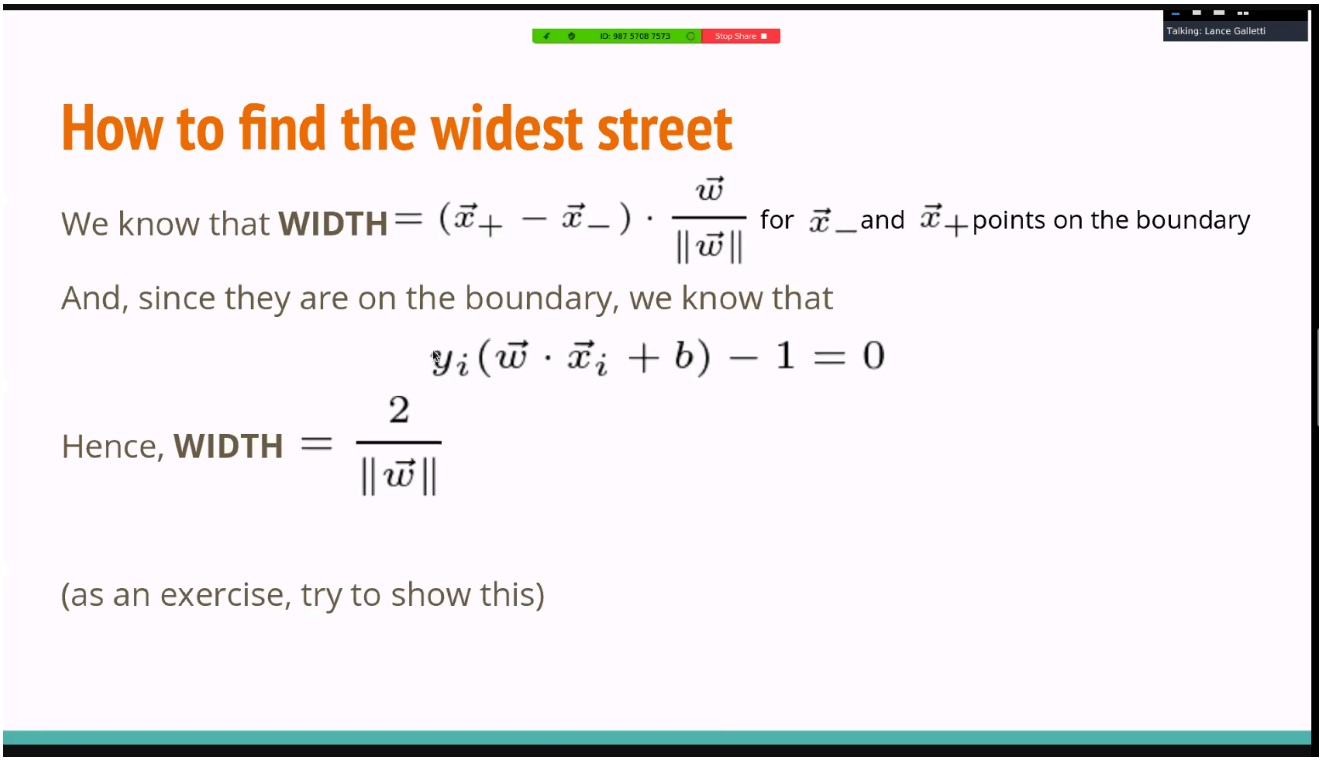
Meaning, for x_i on the decision boundary, we want: $y_i(\overrightarrow{w} \cdot \overrightarrow{x}_i + b) - 1 = 0$.

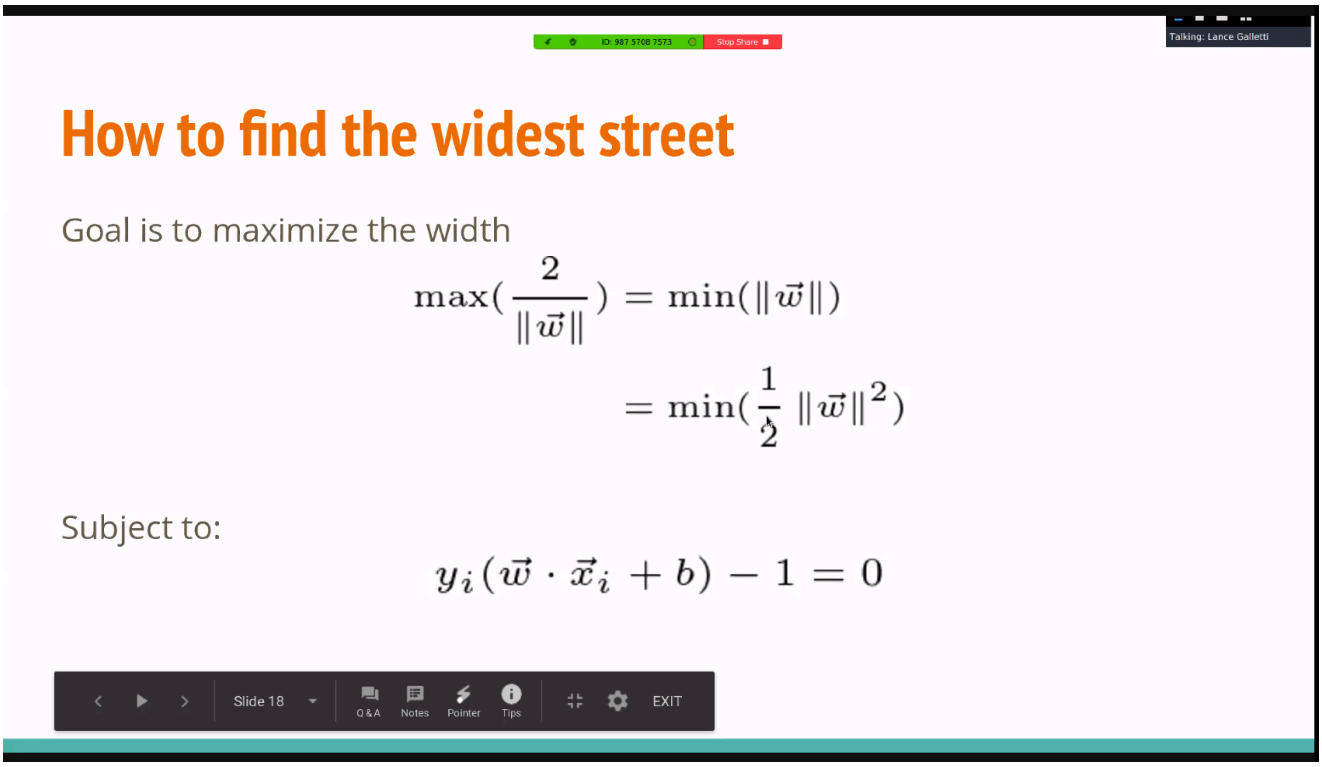
To enforce this, we need to learn $\overrightarrow{w}$ and b in order to have these constraints.

Ok, so how do we maximize the width of the street? Let's imagine that we have points $x+$ and $x-$ on the boundary. How do we calculate the width? We can calculate the vector difference, and then project that onto $\overrightarrow{w}$. Note that we also divide by the magnitude of $\overrightarrow{w}$, so that we have a uniform width on $\overrightarrow{w}$.



We can now state the full problem in a single slide:





We have now turned our maximation problem into a minimization problem. We can then use lagrange multipliers to form a single expression, and take the derivatives to find the extremum of $\mathbf{L}$

After some math, the key point is that it all depends on the inner product between different data points.

Ok, now, what happens if our data is not linearly separable? Then, how can we find the widest street? One thing we can do is to apply a mapping of the space (a nonlinear mapping), such that now we can find a linearly separable line.

Unfortunately, this is quite difficult, because there are infinite possible spaces (transformations), and each of these has different streets that split the data.

However, as we saw before, all we need to do is be able to define the distance (inner product) between data points. Then, if we can define that inner product, we actually don't even need to know the transformation itself.