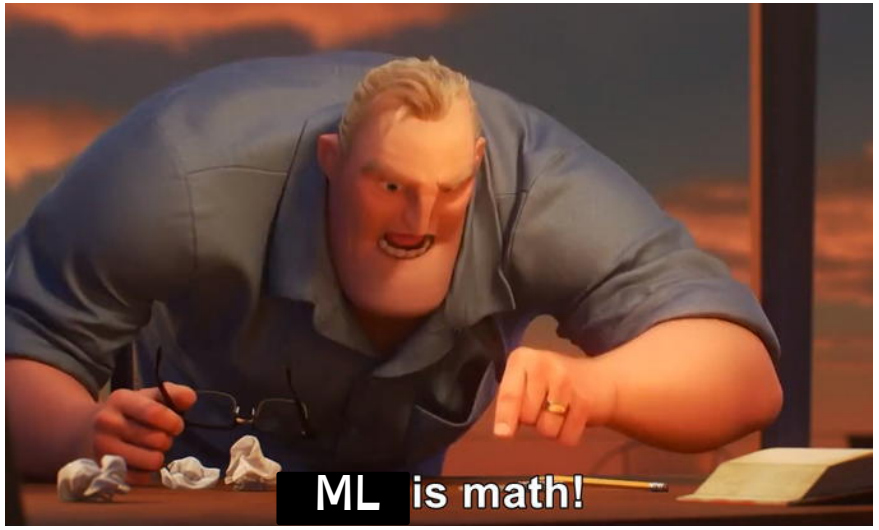
CPSC 340:
Machine Learning and Data Mining

Linear Classifiers

Summer 2021

In This Lecture

1. SVM and Logistic Regression (20 minutes)
2. Linear Probabilistic Classifier (10 minutes)
3. Multi-class Logistic Regression (25 minutes)



Perceptron Algorithm for Linearly-Separable Data

- One of the first "learning" algorithms was the "perceptron" (1957).
  - Searches for a 'w' such that $\text{sign}(w^T x_i) = y_i$ for all i.

- Perceptron algorithm:
  - Start with $w^0 = 0$.
  - Go through examples in any order until you make a mistake predicting y_i.
    - Set $w^{t+1} = w^t + y_i x_i$.
  - Keep going through examples until you make no errors on training data.

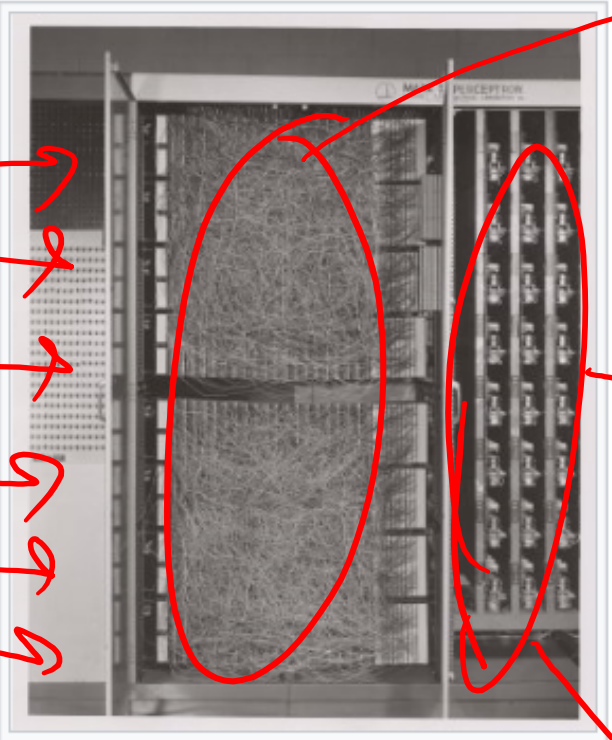
- If a perfect classifier exists, this algorithm finds one in finite number of steps.

- Intuition:
  - Consider a case where $w^T x_i < 0$ but $y_i = +1$.
  - In this case the update "adds more of x_i to w" so that $w^T x_i$ is larger.

$$(w^{t+1})^T x_i = (w^t + x_i)^T x_i = (w^t)^T x_i + x_i^T x_i = (\text{old prediction}) + \|x_i\|^2$$

  - If y_i = -1, you would be subtracting the squared norm.

History [edit]



The Mark I Perceptron machine was the first implementation of the perceptron algorithm. The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image. The main visible feature is a patchboard that allowed experimentation with different combinations of input features. To the right of that are arrays of potentiometers that implemented the adaptive weights.[2]:213

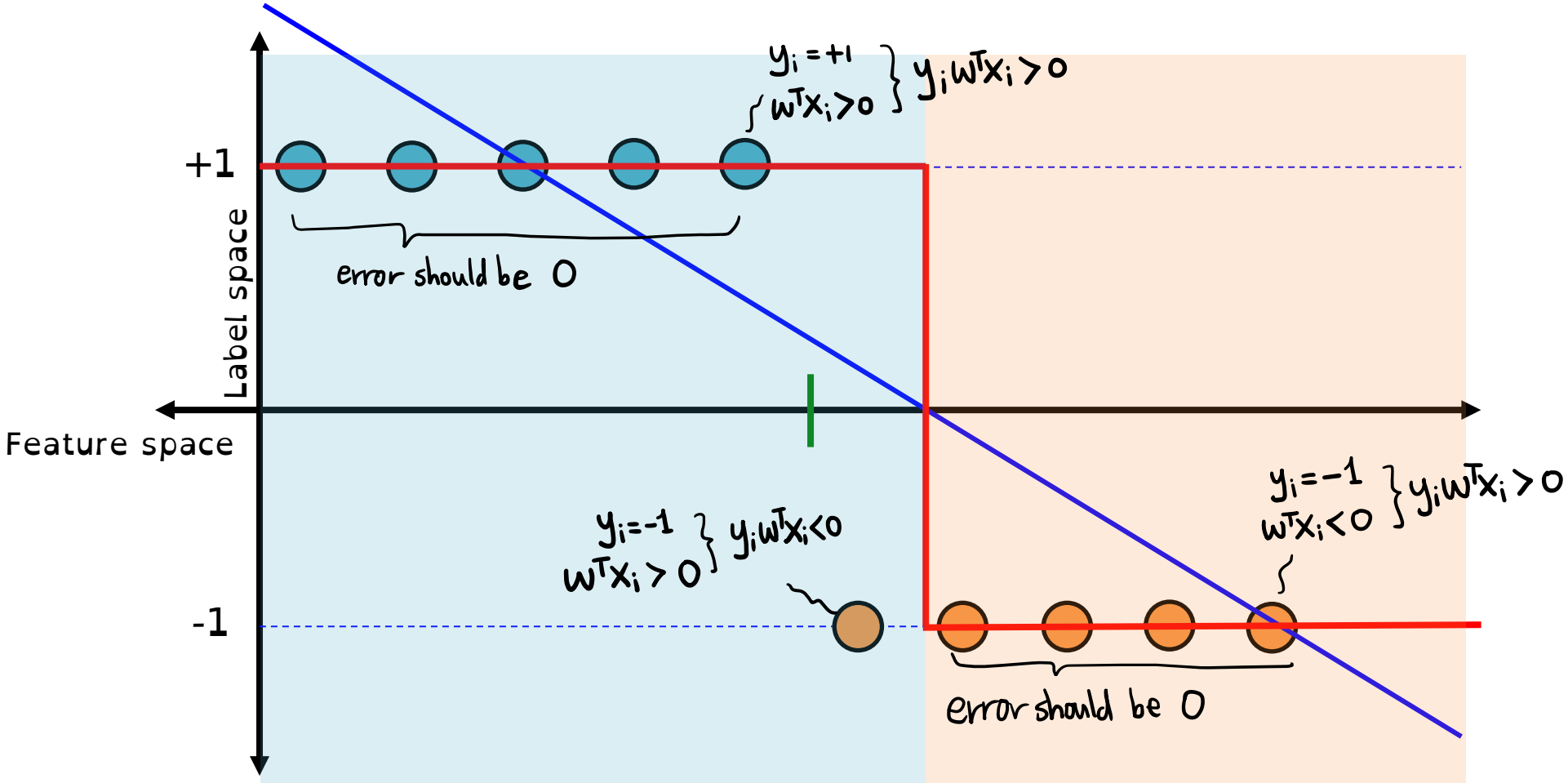
https://en.wikipedia.org/wiki/Perceptron

Coming Up Next

SUPPORT VECTOR MACHINE AND LOGISTIC REGRESSION

Hinge Loss

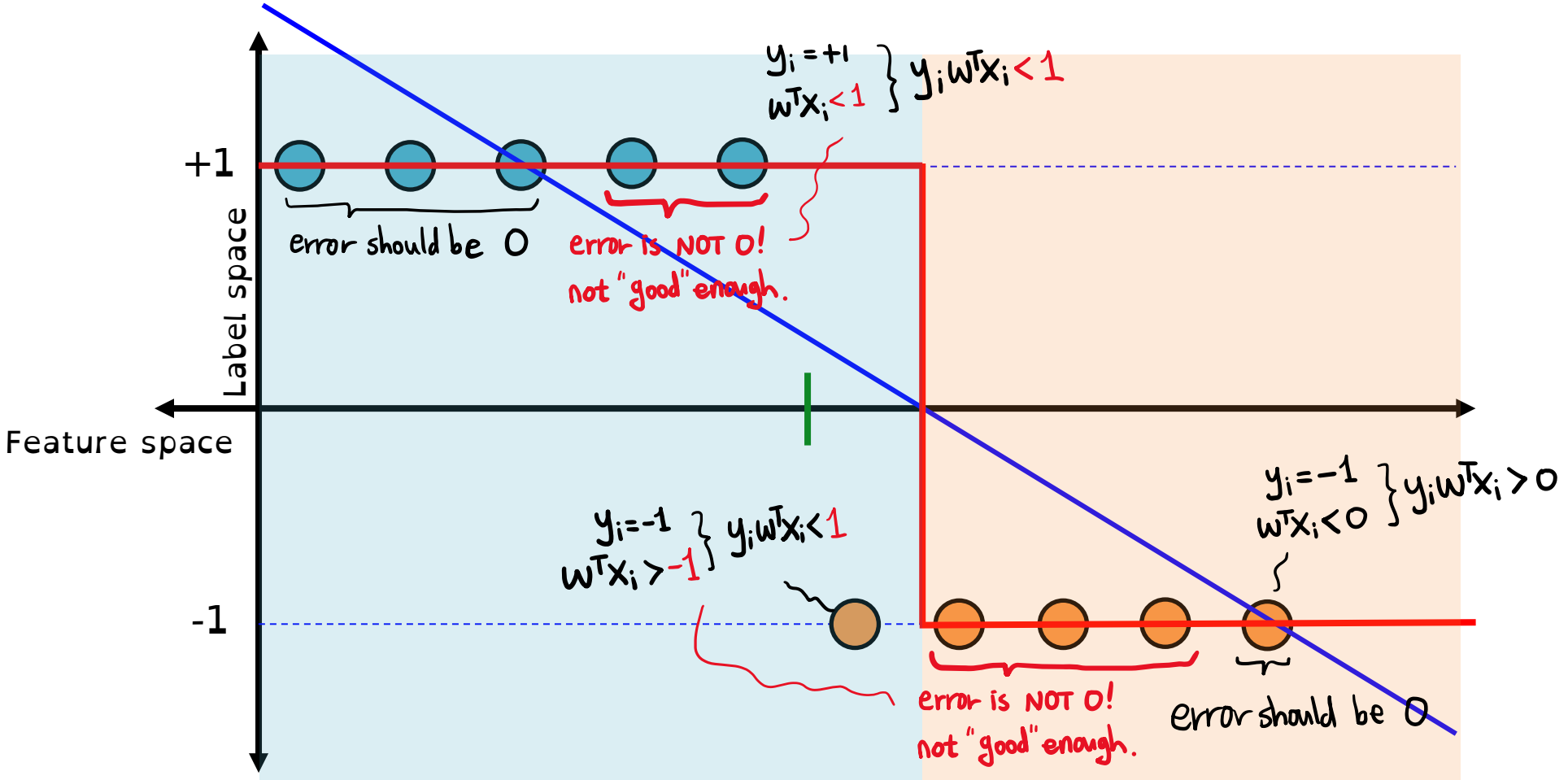
- We saw that we classify examples 'i' correctly if $y_i w^T x_i > 0$.
  - Our convex approximation is the amount this inequality is violated.



Hinge Loss

- Consider replacing $y_i w^T x_i > 0$ with $y_i w^T x_i \geq 1$.

  (the "1" is arbitrary: we could make ||w|| bigger/smaller to use any positive constant)



Hinge Loss

Let's count # times $y_i w^T x_i < 1$

[1] $$f(w) = \sum_{i=1}^{n} \mathbf{I}(y_i w^T x_i < 1) = \sum_{i=1}^{n} \mathbf{I}(0 < 1 - y_i w^T x_i)$$

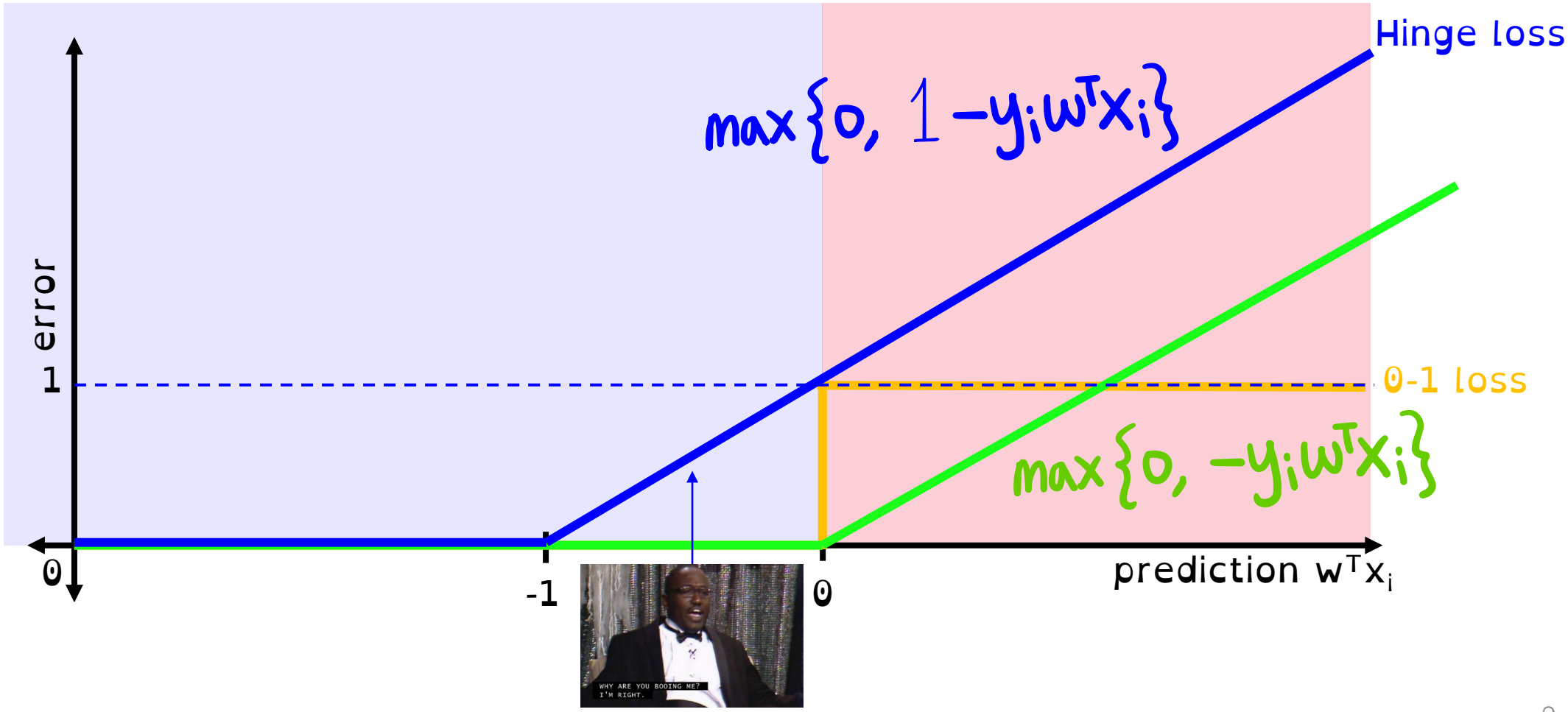
[2] $$\approx \max\{0, 1 - y_i w^T x_i\}$$

Training examples when they're too close to the decision boundary

- This is the called hinge loss.
  - It's convex: max(constant,linear).
  - It's not degenerate: f(0) = 1 instead of 0.

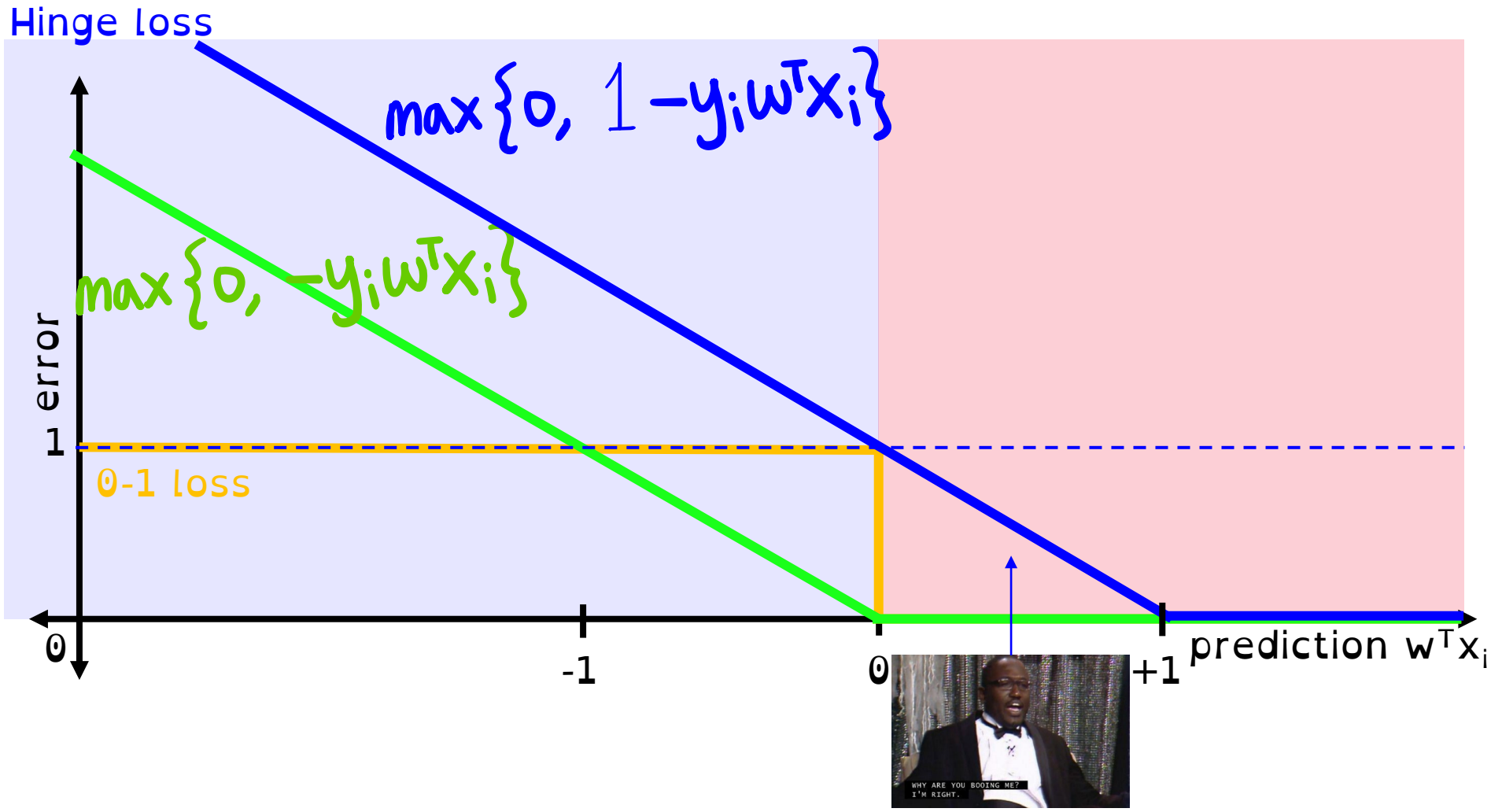
Visualizing Hinge Loss

Given example (x_i, -1)



Visualizing Hinge Loss

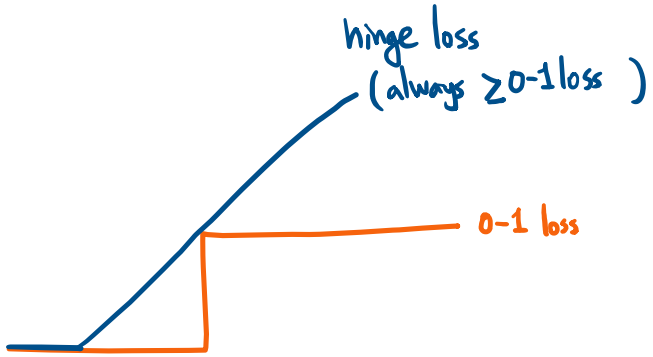
Given example (x_i, +1)



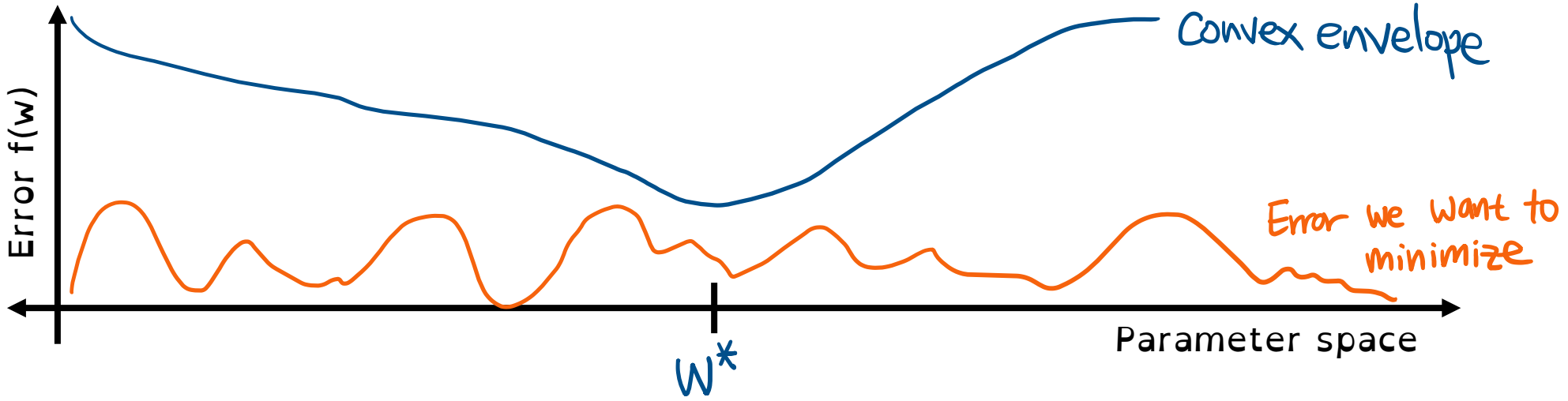
Hinge Loss

- Hinge loss for all 'n' training examples is given by:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\}$$



  - Upper convex envelope on 0-1 loss.
    - So minimizing hinge loss indirectly tries to minimize training error.
  - Like perceptron, finds a perfect linear classifier if one exists.



Support Vector Machine

- Support vector machine (SVM) is hinge loss with L2-regularization.

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$

  - There exist specialized optimization algorithm for this problem.
  - SVMs can also be viewed as "maximizing the margin" (later).

Logistic Loss

- We can smooth max the degenerate loss with log-sum-exp:

$$\max\{0, -y_i w^T x_i\} \approx \log(\exp(0) + \exp(-y_i w^T x_i))$$

- Summing over all examples gives:

$$f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$$

- This is the "logistic loss" and model is called "logistic regression".
  - It's not degenerate: f(0) = _____ instead of 0.
  - Convex and differentiable: minimize this with gradient descent.
  - You should also add regularization.
  - We'll see later that it has a probabilistic interpretation.

Convex Approximations to 0-1 Loss

"Error" or "loss" for predicting $w^T x_i$ when true label y_i is -1.

"hinge" loss

"logistic" loss (similar but smooth)

What we want is the "0-1 loss."

Prediction $w^T x_i$

$y_i = -1$

0

They don't penalize for being "very right."

they still penalize you for being "very wrong"

Logistic Regression and SVMs

- Logistic regression and SVMs are used EVERYWHERE!
  - Fast training and testing.
    - Training on huge datasets using "stochastic" gradient descent (next week).
    - Prediction is just computing $w^T x_i$.
  - Weights w_j are easy to understand.
    - It's how much w_j changes the prediction and in what direction.
  - We can often get a good test error.
    - With low-dimensional features using RBFs and regularization.
    - With high-dimensional features and regularization.
  - Smoother predictions than random forests.

Comparison of "Black Box" Classifiers

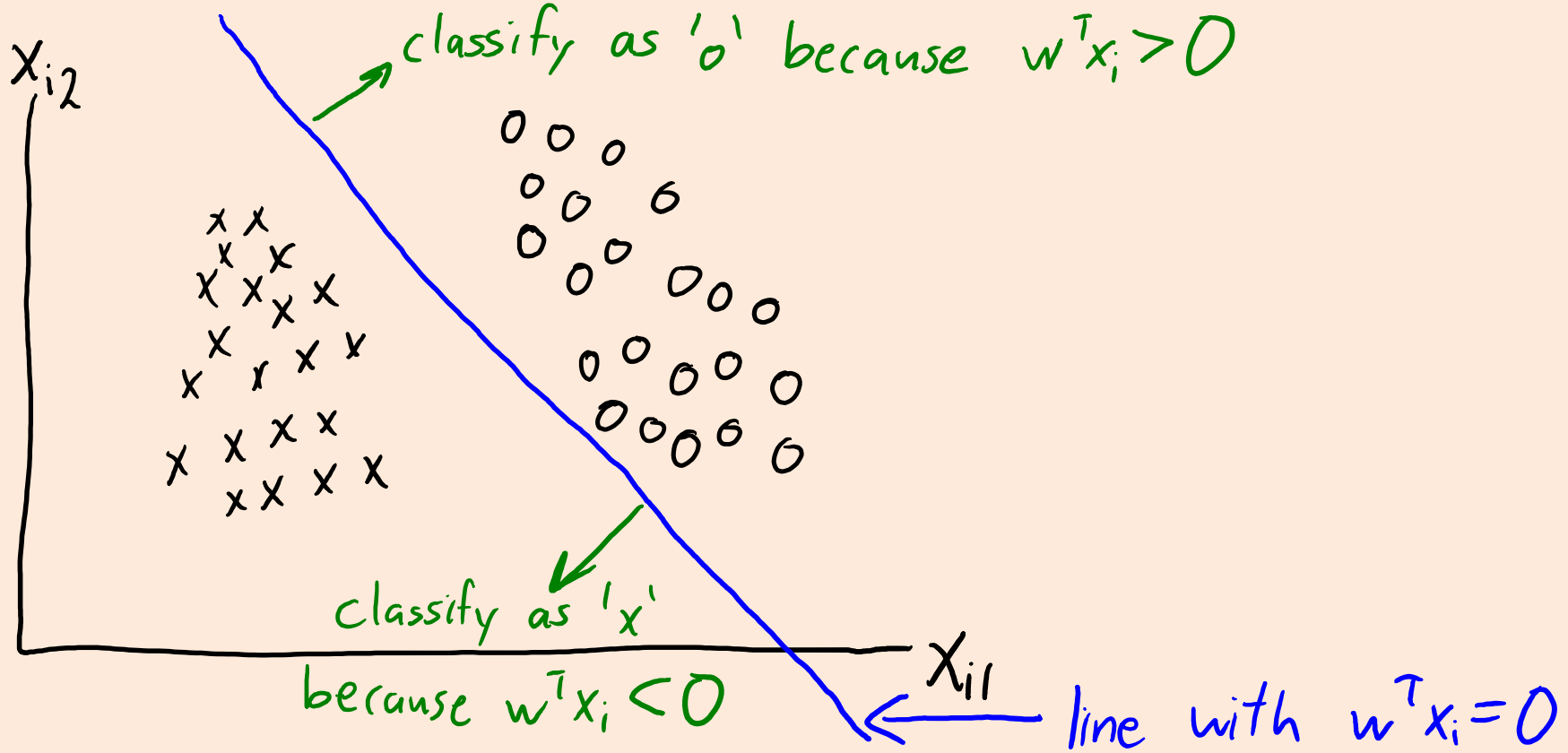
- Fernandez-Delgado et al. [2014]:
  - "Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?"

- Compared 179 classifiers on 121 datasets.
- Random forests are most likely to be the best classifier.
- Next best class of methods was SVMs (L2-regularization, RBFs).

- "Why should I care about logistic regression if I know about deep learning?"

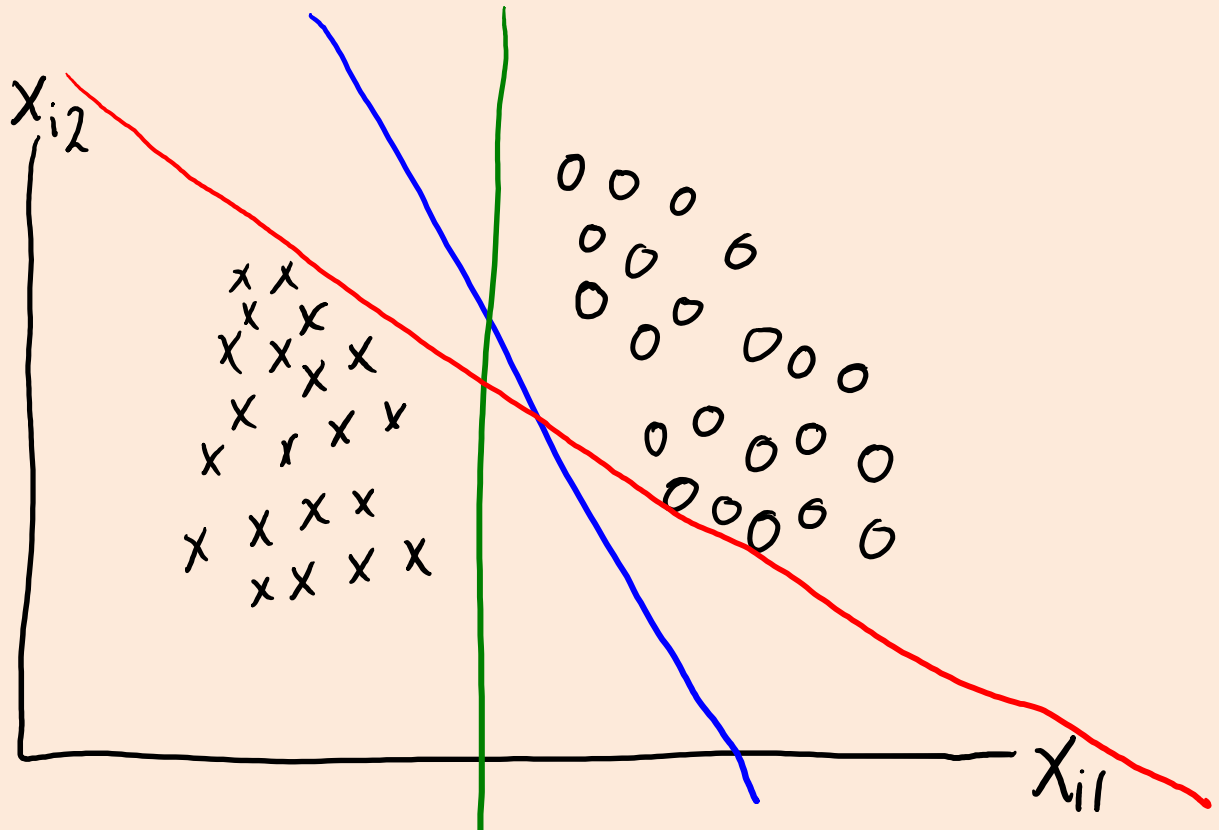
Maximum-Margin Perspective

- Consider a linearly-separable dataset.



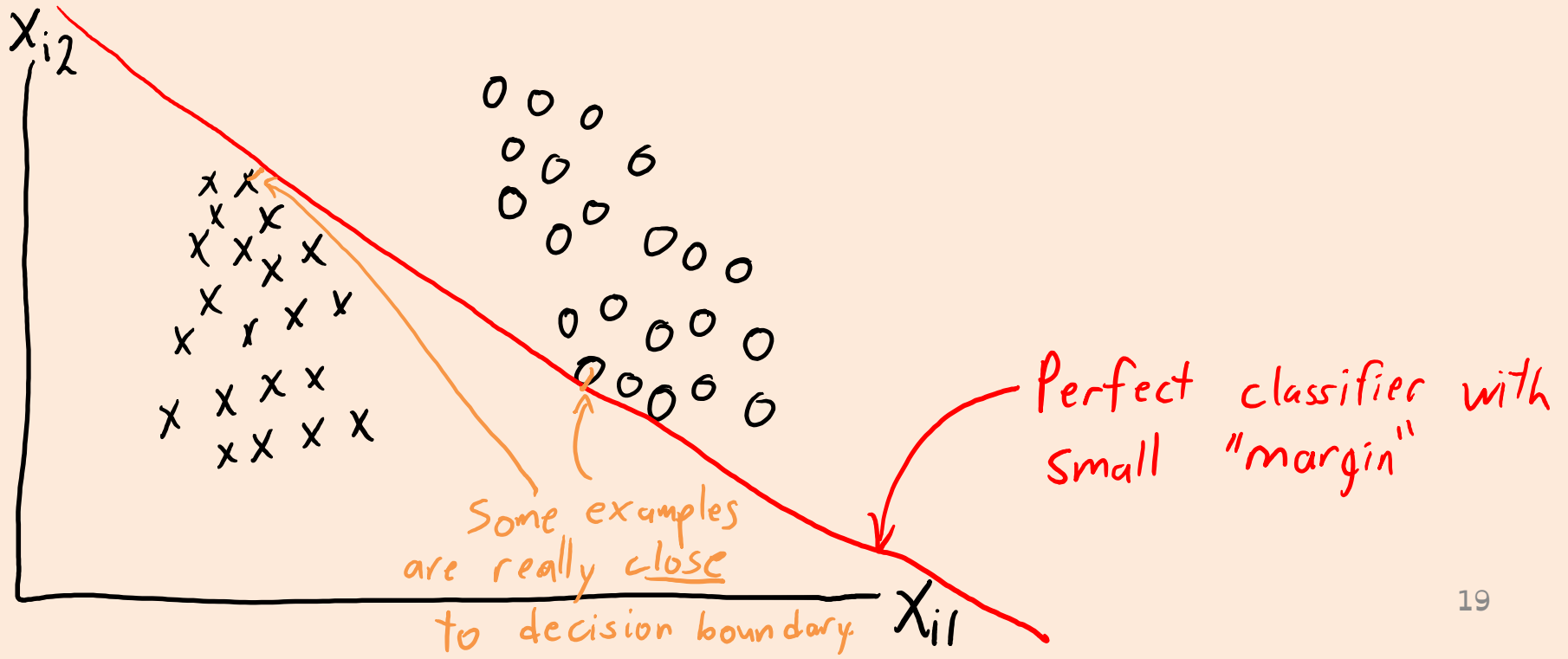
Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Perceptron algorithm finds *some* classifier with zero error.
  - But are all zero-error classifiers equally good?



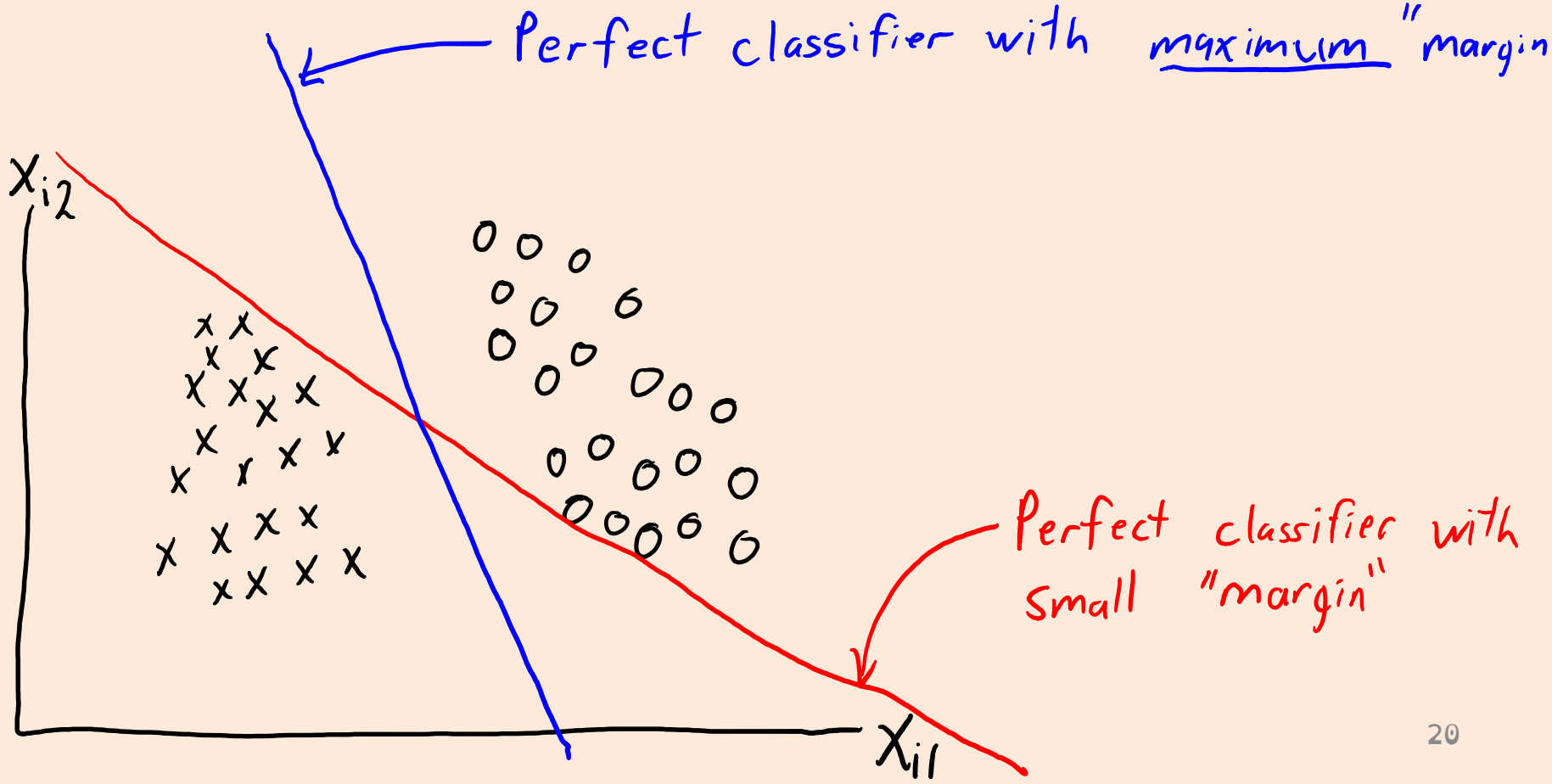
Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

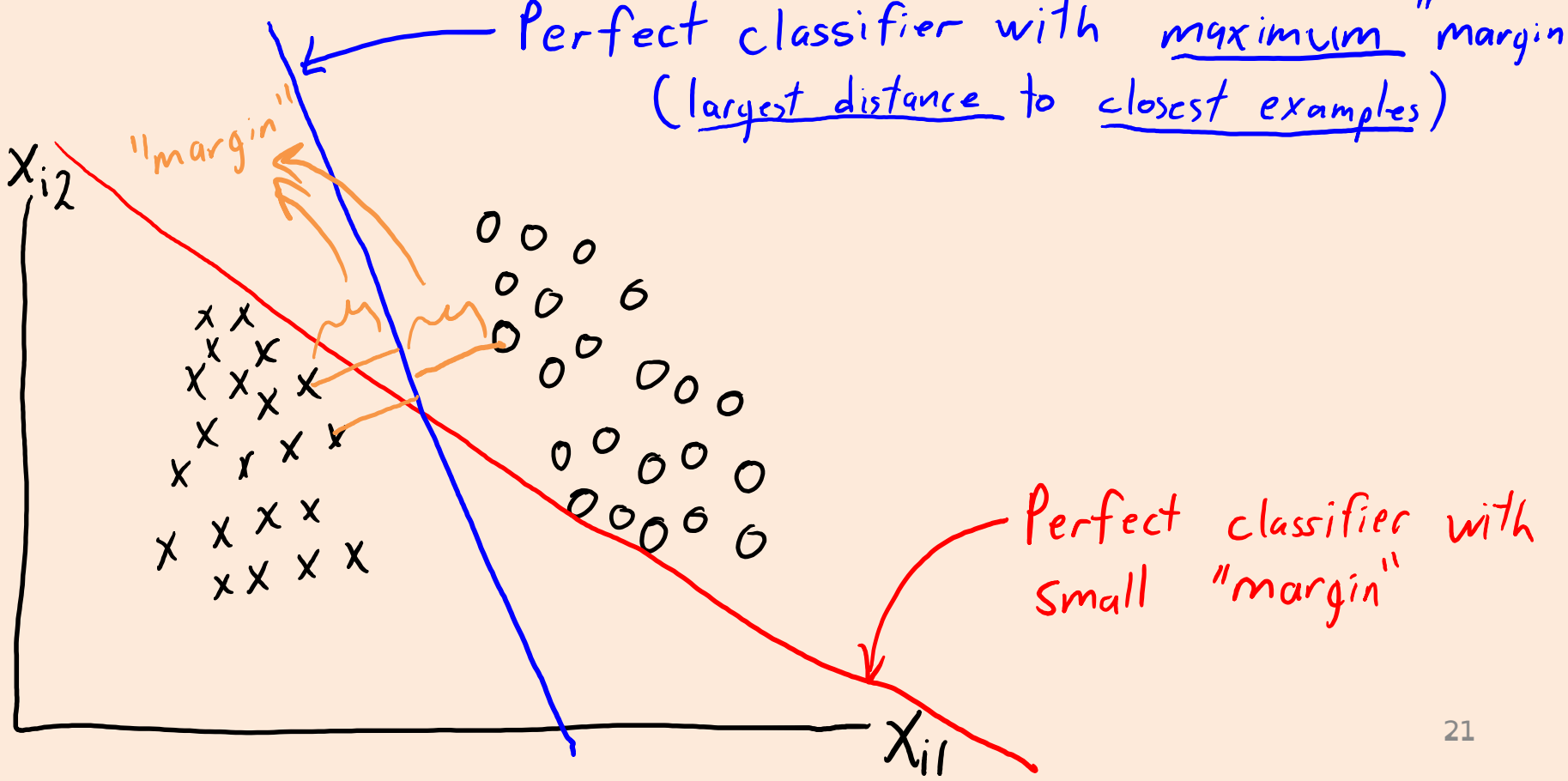


Maximum-Margin Perspective

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.

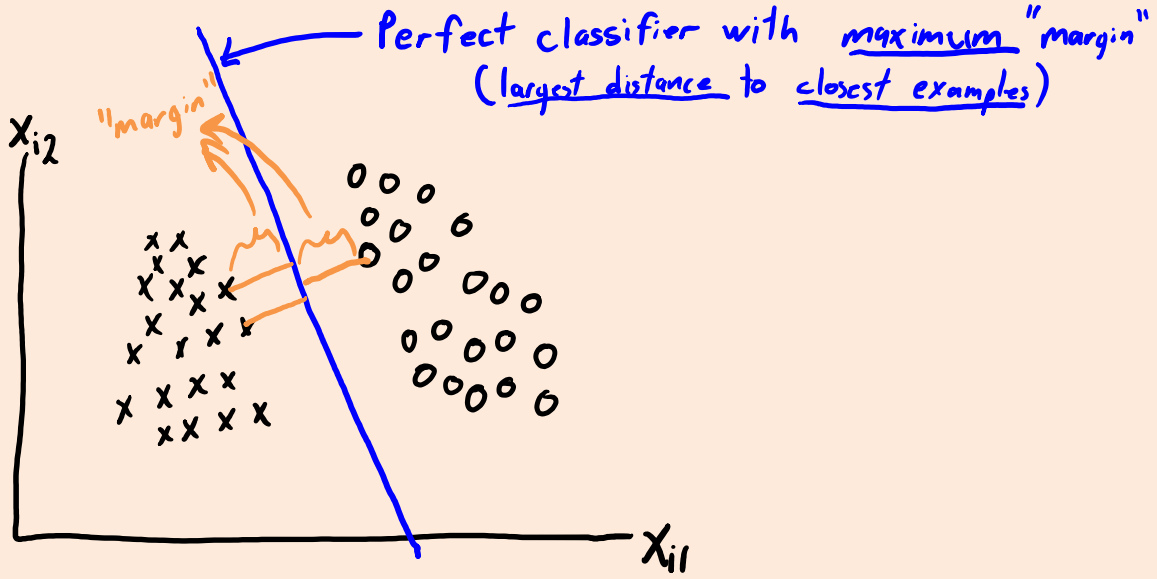
Why maximize margin?

If test data is close to training data, then max margin leaves more "room" before we make an error.



Maximum-Margin Perspective

- For linearly-separable data:



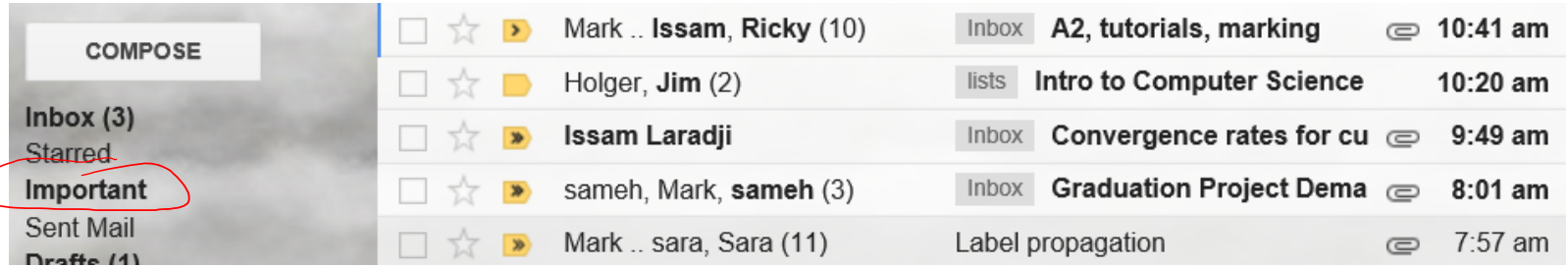
  - With small-enough $\lambda > 0$, SVMs find the maximum-margin classifier.
    - Need λ small enough that hinge loss is 0 in solution.
    - Origin of the name: the "support vectors" are the points closest to the line (see bonus).
  - Recent result: logistic regression also finds maximum-margin classifier.
    - With $\lambda=0$ and if you fit it with gradient descent (not true for many other optimizers).

Coming Up Next

LINEAR PROBABILISTIC CLASSIFIER

Previously: Identifying Important E-mails

- Recall problem of identifying 'important' e-mails:



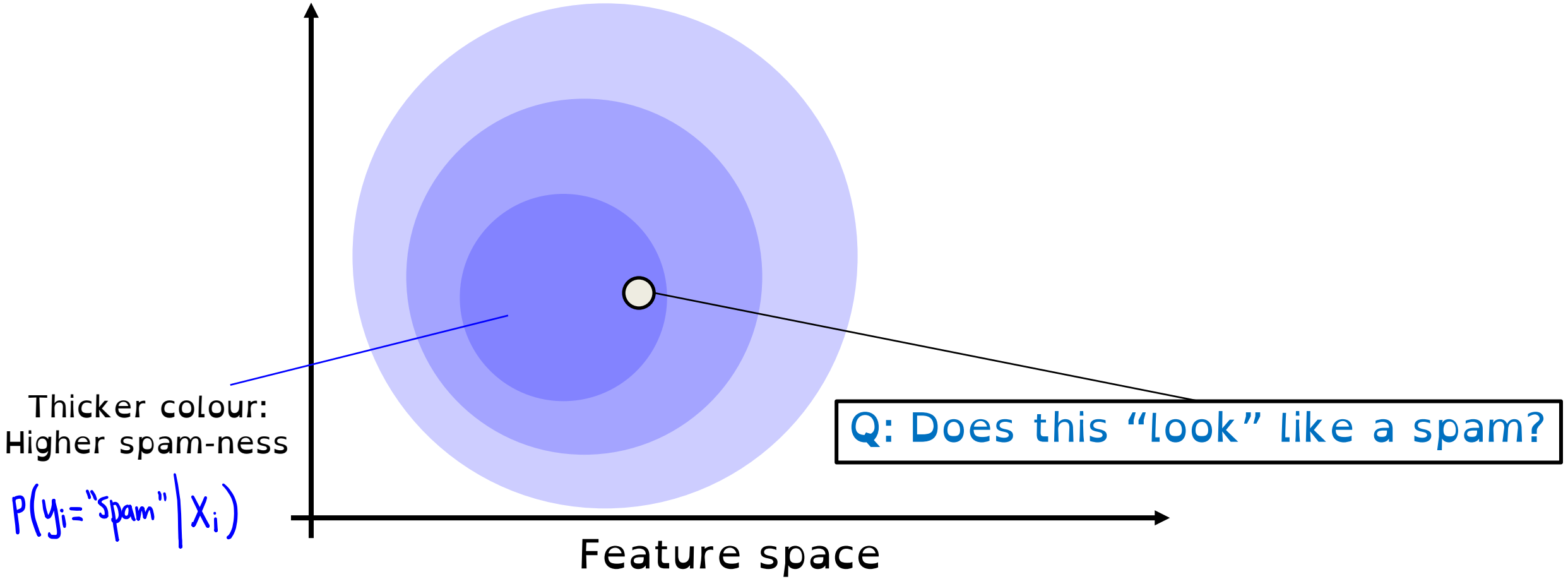
- We can do binary classification by taking sign of linear model:

$$\hat{y}_i = \text{sign}(w^T x_i)$$

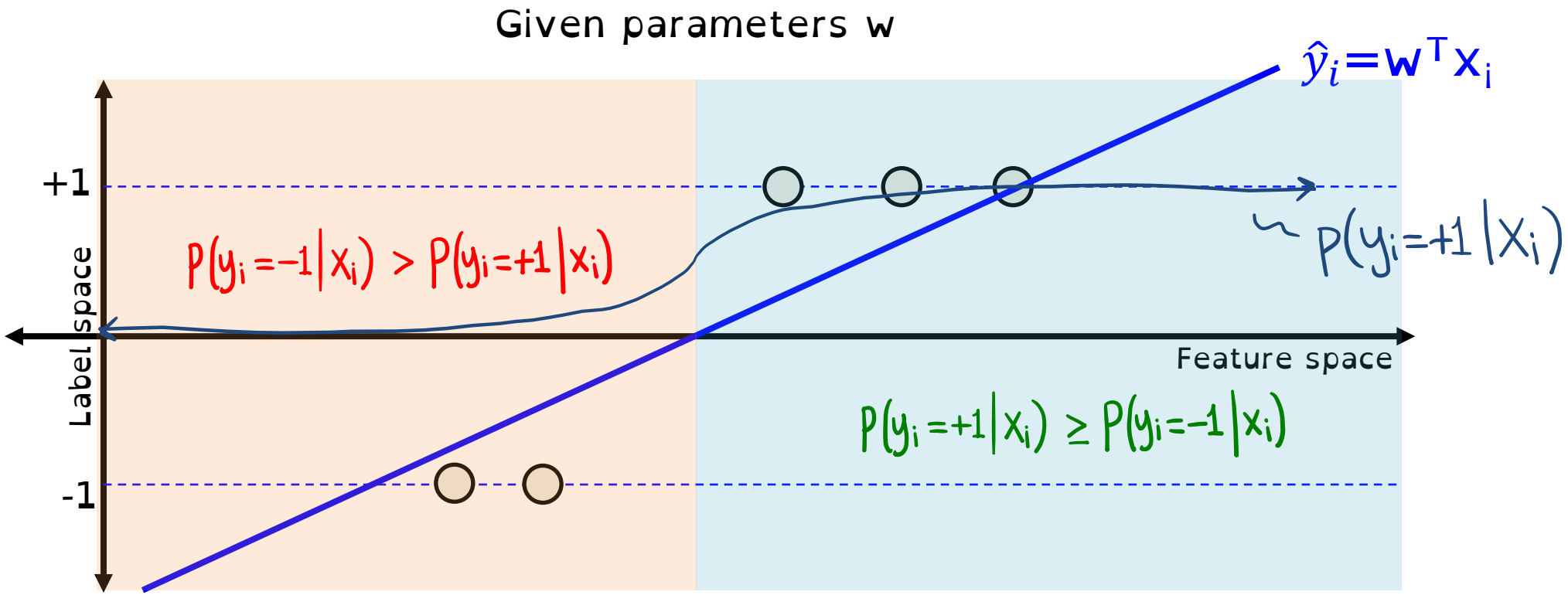
  - Convex loss functions (hinge/logistic loss) let us find an appropriate 'w'.

- But what if we want a probabilistic classifier?
  - Want a model of $p(y_i = \text{"important"} \mid x_i)$ for use in decision theory.

Recall: "Spam-ness"



Linear Prediction of "+1-ness"



Q: How should $p(y_i=+1 \mid x_i)$ behave?

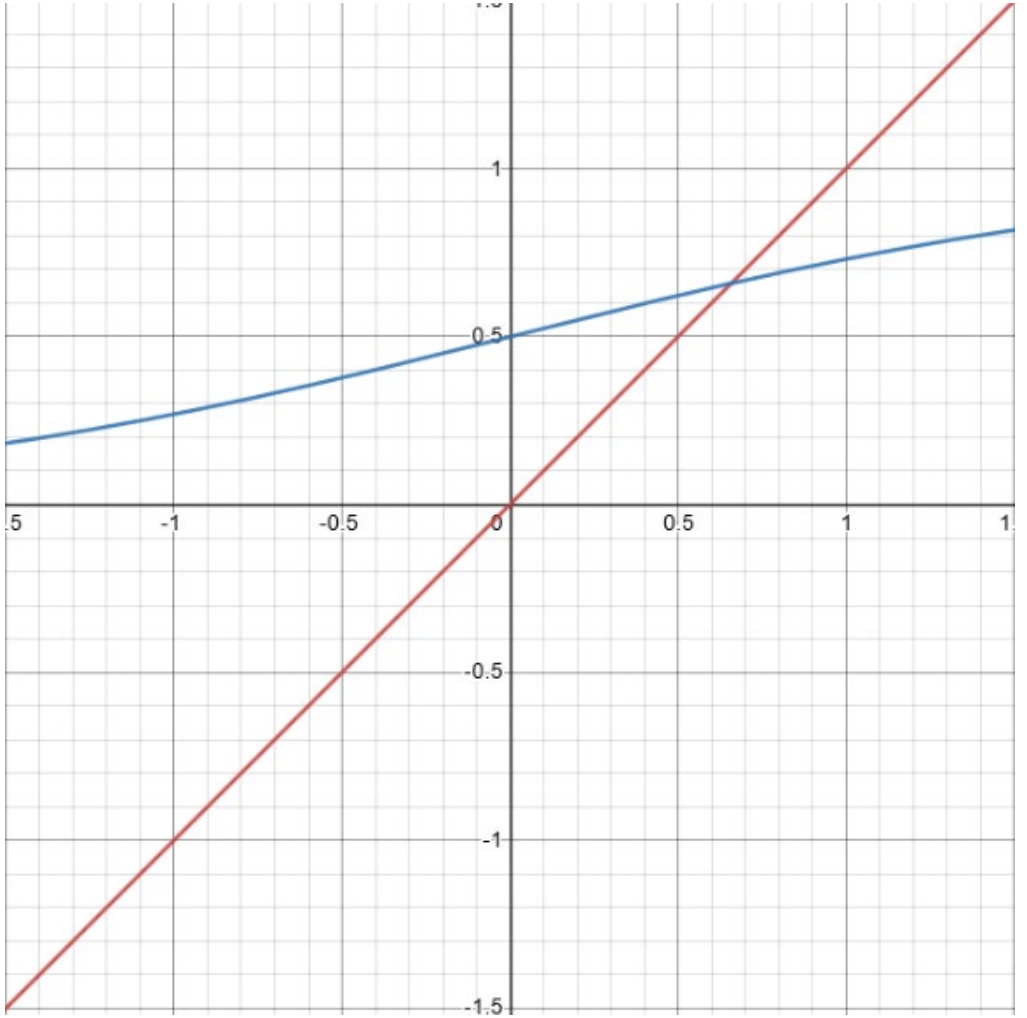
Sigmoid Function

$\text{Sigmoid}: \mathbb{R} \rightarrow (0, 1)$

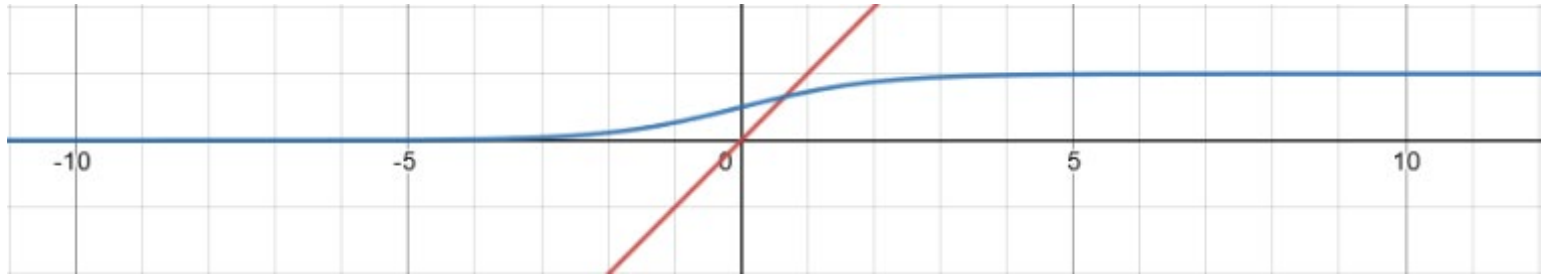
$$\text{Sigmoid}(z) = \frac{1}{1+\exp(-z)}$$

Q: What is sigmoid(z) when z is negative?

What is sigmoid(z) when z is positive?



+1-ness with Sigmoid



f(x) = sigmoid(wx)

- Idea: let's compute +1-ness with sigmoid.
- Given parameters **w**:
  1. Compute $z_i = \mathbf{w}^T x_i$
  2. Compute $p(y_i = +1 \mid \mathbf{w}, x_i) = \text{sigmoid}(z_i)$

$$P(y_i = +1 \mid w, \cdot) : \mathbb{R}^d \longrightarrow (0, 1)$$

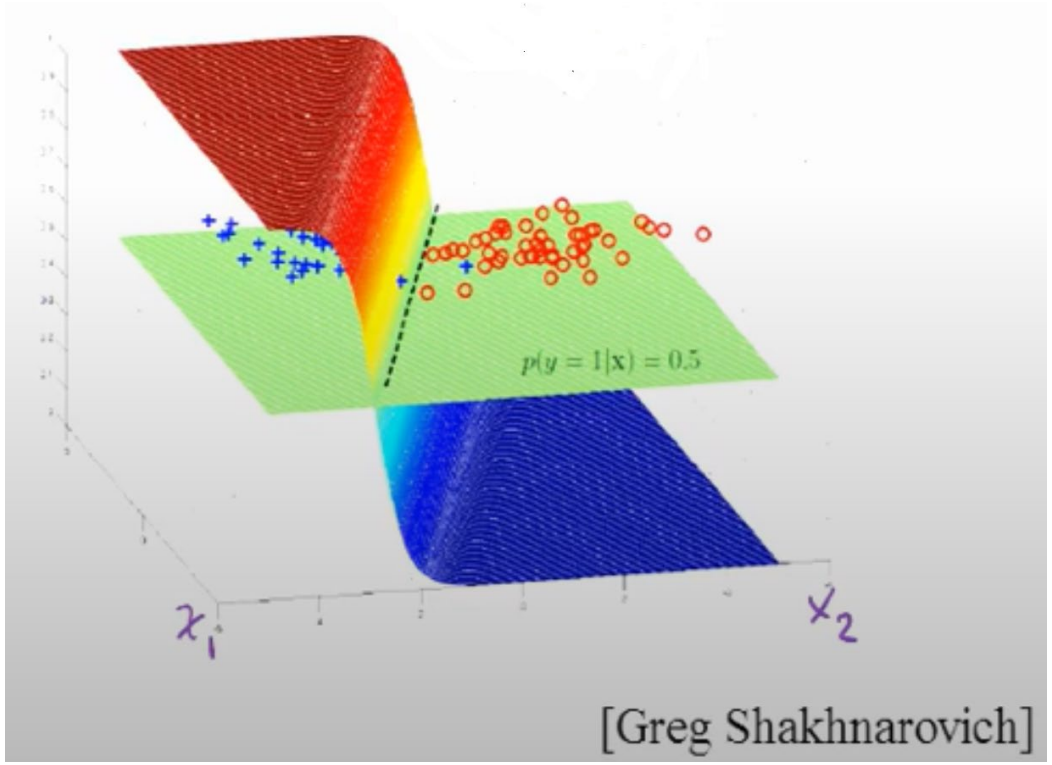
$$p(y_i = +1 \mid w, x_i) = \text{Sigmoid}(w^T x_i)$$

Probabilities for Linear Classifiers using Sigmoid

- Using sigmoid function, we output probabilities for linear models using:

$$p(y_i = +1 \mid w, x_i) = \frac{1}{1 + \exp(-w^T x_i)}$$

- Visualization for 2 features:



https://www.youtube.com/watch?v=Zc7ouSD0DEQ

What About "-1-ness"?

- Using sigmoid function, we output probabilities for linear models using:

$$p(y_i = +1 \mid w, x_i) = \frac{1}{1 + \exp(-w^T x_i)}$$

- By rules of probability:

$$p(y_i = -1 \mid w, x_i) = 1 - p(y_i = 1 \mid w, x_i)$$

$$= \frac{1}{1 + \exp(w^T x_i)}$$ (with some effort)

- We then use these for "probability that an email x_i is important".
- This may seem heuristic, but later we'll see that:
  - minimizing logistic loss does "maximum likelihood estimation" in this model.

Coming Up Next

MULTI-CLASS CLASSIFICATION INTRO

Multi-Class Linear Classification

- We've been considering linear models for binary classification:

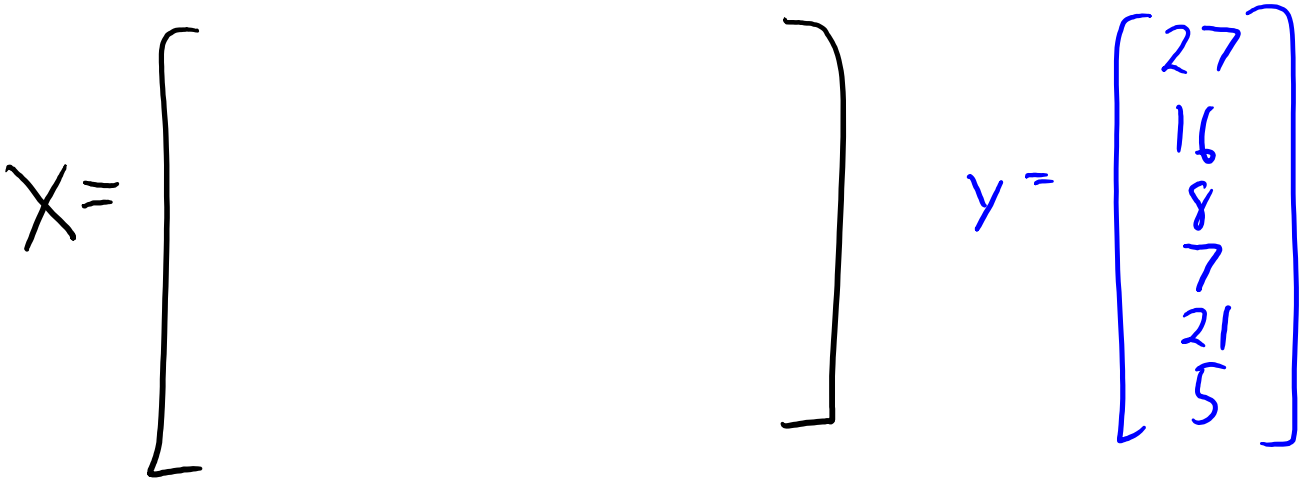
$$X = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

- E.g., is there a cat in this image or not?

https://www.youtube.com/watch?v=tntOCGkgt98

Multi-Class Linear Classification

- Today we'll discuss linear models for multi-class classification:



- For example, classify image as "cat", "dog", or "person".
  - This was natural for methods of Part 1 (decision trees, naïve Bayes, KNN).
  - For linear models, we need some new notation.

Q: Can we use binary classifiers for multi-class?

"One vs All" Classification

- Training phase:
  - For each class 'c', train binary classifier to predict whether example is a 'c'.
    - For example, train a "cat detector", a "dog detector", and a "human detector".
    - If we have 'k' classes, this gives 'k' binary classifiers .

$$X = \underset{n}{\begin{bmatrix} & \\ & \end{bmatrix}}_{d} \qquad y = \underset{n}{\begin{bmatrix} \text{"cat"} \\ \text{"dog"} \\ \text{"human"} \\ \text{"cat"} \end{bmatrix}}_{1} \rightarrow y_{cat} = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix}$$

$(X, y_{cat}) \rightarrow w_{cat}$ "cat detector" $\qquad (X, y_{dog}) \rightarrow w_{dog} \qquad (X, y_{human}) \rightarrow w_{human}$

"One vs All" Classification

$$W = \begin{bmatrix} \text{---} & w_{cat} & \text{---} \\ \text{---} & w_{dog} & \text{---} \\ & \vdots & \\ \text{---} & w_{human} & \text{---} \end{bmatrix}$$

k = #classes, d

- Prediction phase:
  - Apply the 'k' binary classifiers to get a "score" for each class 'c'.
  - Predict the 'c' with the highest score.

features x_i

$w_{cat}^T x_i = -0.1$

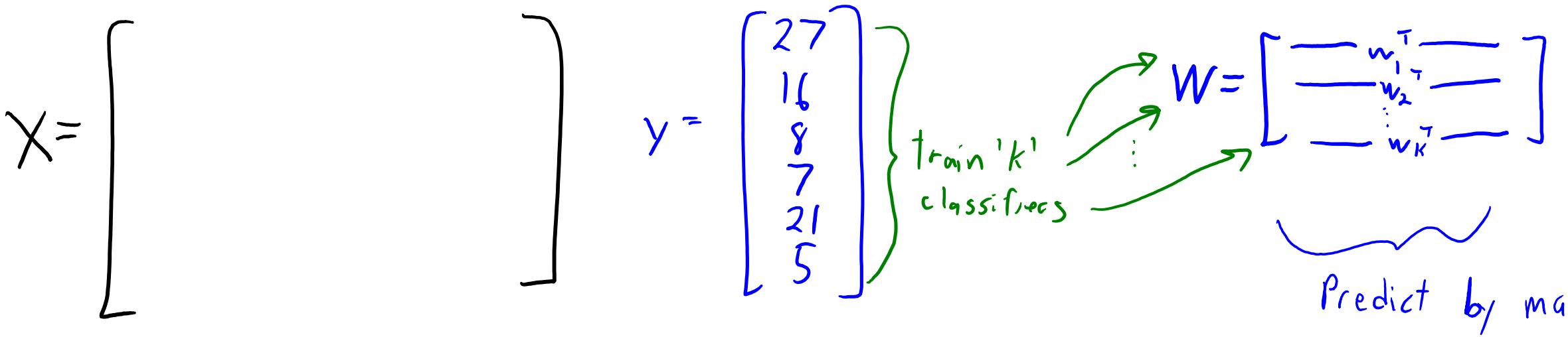
$w_{dog}^T x_i = -0.8$

$w_{human}^T x_i = 0.9$

$\hat{y}_i = \text{human}$

Multi-Class Linear Classification (MEMORIZE)

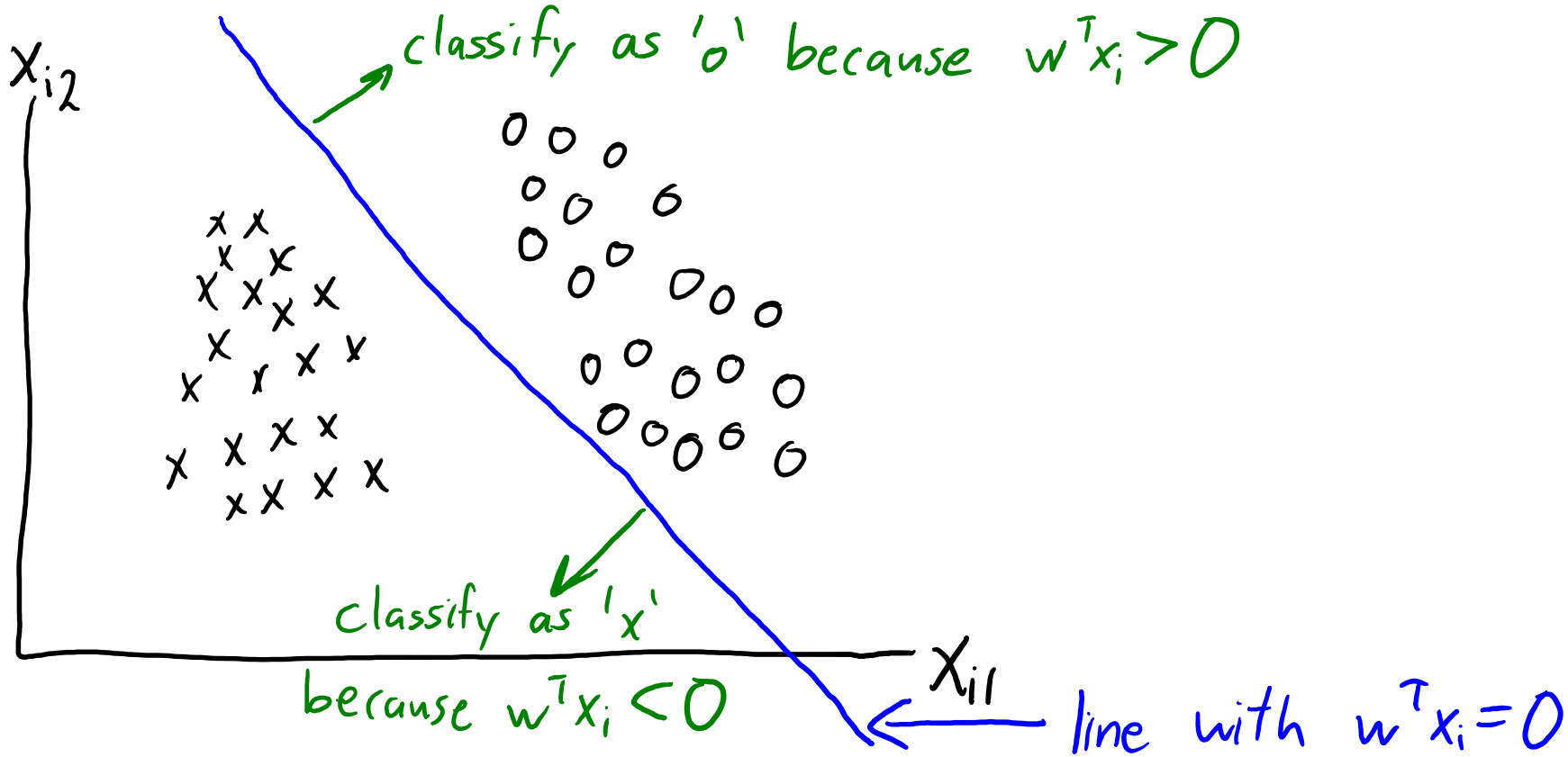
- Back to multi-class classification where we have 1 "correct" label:



- We'll use 'w_{y_i}' as classifier where $c=y_i$ (row of correct class label).
  - So if $y_i=2$ then $w_{y_i} = w_2$.

Shape of Decision Boundaries

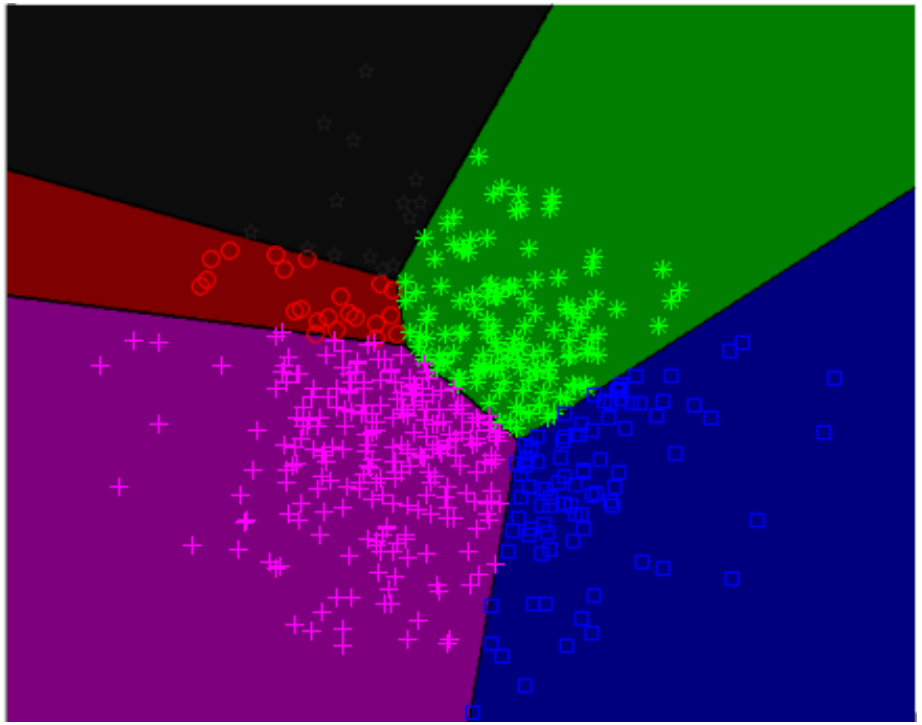
- Recall that a binary linear classifier splits space using a hyper-plane:



- Divides x_i space into 2 "half-spaces".

Shape of Decision Boundaries

- Multi-class linear classifier is intersection of these "half-spaces":
  - This divides the space into convex regions (like k-means):



"Blue" region is region where we have:

$$w_{blue}^T x_i \geq w_{green}^T x_i$$

$$w_{blue}^T x_i \geq w_{magenta}^T x_i$$

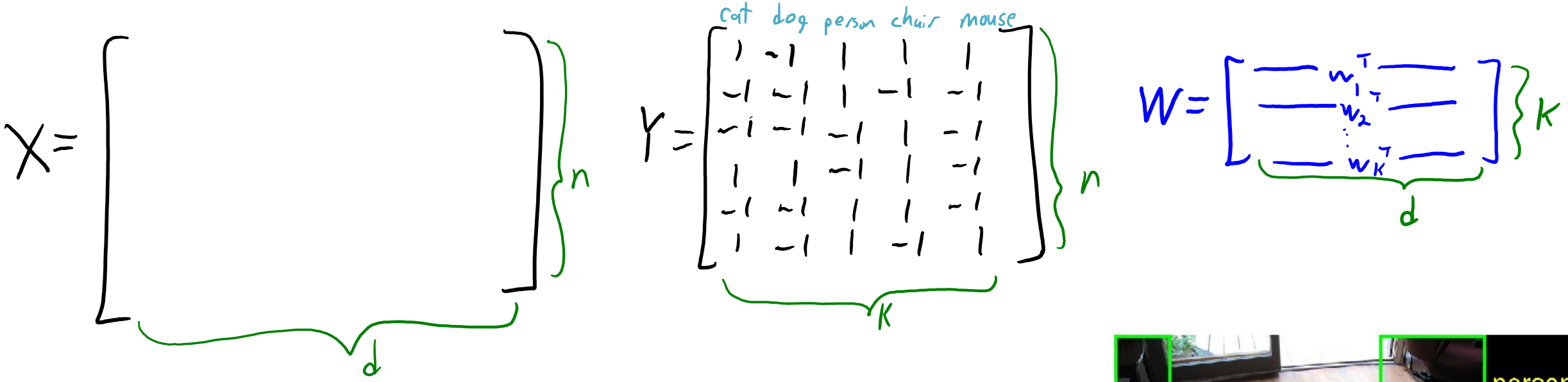
$$w_{blue}^T x_i \geq w_{red}^T x_i$$

$$w_{blue}^T x_i \geq w_{black}^T x_i$$

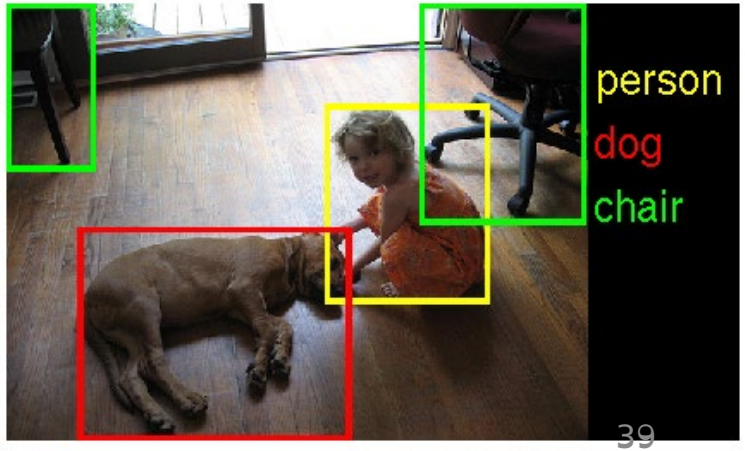
  - Could be non-convex with change of basis.

Digression: Multi-Label Classification

- A related problem is multi-label classification:
-



- Which of the 'k' objects are in this image?
  - There may be more than one "correct" class label.
  - Here we can also fit 'k' binary classifiers.
    - But we would take all the $\text{sign}(w_c^T x_i)=+1$ as the labels.



http://image-net.org/challenges/LSVRC/2013/

"One vs All" Multi-Class Linear Classification

- Problem: We didn't train the w_c so that the largest $w_c^T x_i$ would be $w_{y_i}^T x_i$.
  - Each classifier is just trying to get the sign right.

features x_i

$w_1^T x_i = -5$ ("cat" score)

$w_2^T x_i = -0.1$ ("dog" score)

$w_3^T x_i = -0.2$ ("human" score)

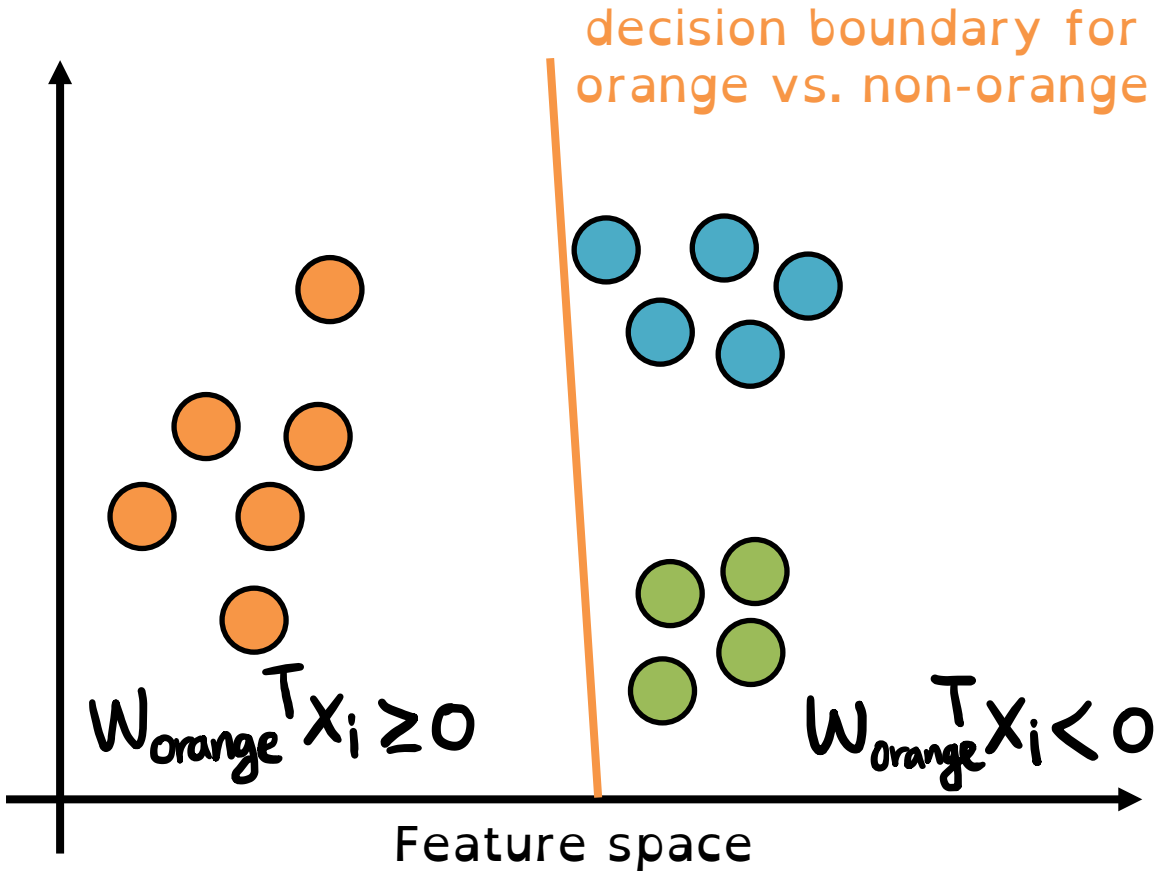
  - Here the classifier incorrectly predicts "dog".
    - "One vs All" doesn't try to put $w_2^T x_i$ and $w_3^T x_i$ on same scale for decisions like this.
    - We should try to make $w_3^T x_i$ positive and $w_2^T x_i$ negative relative to each other.
    - The multi-class hinge losses and the multi-class logistic loss do this.

https://laughingsquid.com/pug-mask-a-latex-mask-so-you-can-look-like-a-dog/

Coming Up Next

MULTI-CLASS SVM

Binary Classifiers are "Under-constrained"



Binary Classifiers are "Under-constrained"

Q: What's the prediction for this example?

What Do We Want for Multi-Class Classifiers?

- Idea: additional constraints on slopes
  - Will "force" classifier to find better boundaries

- Think of $w_{cat}^T x_i$, $w_{dog}^T x_i$, $w_{human}^T x_i$ as scores.
  - Previously, we only wanted $w_{cat}^T x_i > 0$ (underconstrained!)
  - New constraint: "cat" example x_i should have higher $w_{cat}^T x_i$ than $w_{dog}^T x_i$, $w_{human}^T x_i$
  - Now, we want: $w_{cat}^T x_i > w_{dog}^T x_i$ and $w_{cat}^T x_i > w_{human}^T x_i$

Q: How should we design the error here?

Multi-Class Loss Function

Now, we want: $w_{cat}^T x_i > w_{dog}^T x_i$ and $w_{cat}^T x_i > w_{human}^T x_i$

Let's count # times $w_{cat}^T x_i \leq w_{dog}^T x_i$ + # $w_{cat}^T x_i \leq w_{human}^T x_i$

[1] $$f_{cat}(W) = \sum_{i \in \text{cat examples}} I\left(w_{cat}^T x_i \leq w_{dog}^T x_i\right) + I\left(w_{cat}^T x_i \leq w_{human}^T x_i\right)$$

[2] $$= \sum_{i \in \text{cat examples}} I\left(0 \leq -w_{cat}^T x_i + w_{dog}^T x_i\right) + I\left(0 \leq -w_{cat}^T x_i + w_{human}^T x_i\right)$$

[3] $$\approx \sum_{i \in \text{cat examples}} \max\left\{0, -w_{cat}^T x_i + w_{dog}^T x_i\right\} + \max\left\{0, -w_{cat}^T x_i + w_{human}^T x_i\right\}$$

Multi-Class Loss Function

- Let's generalize this!

[4] $$f_{cat}(W) = \sum_{i \in \text{cat examples}} \max\{0, -W_{cat}^T x_i + W_{dog}^T x_i\} + \max\{0, -W_{cat}^T x_i + W_{human}^T x_i\}$$

[5] $$f_c(W) = \sum_{i \in \text{"c" examples}} \sum_{c' \neq c} \max\{0, -W_c^T x_i + W_{c'} x_i\}$$

[6] $$f(W) = \sum_{c=1}^{k} f_c(W) = \sum_{c=1}^{k} \sum_{i \in \text{"c" examples}} \sum_{c' \neq c} \max\{0, -W_c^T x_i + W_{c'} x_i\}$$

[7] $$= \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, -W_{y_i}^T x_i + W_{c'} x_i\}$$

Multi-Class Loss Function

$$f(W) = \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, -W_{y_i}^T X_i + W_{c'} X_i\}$$

$$X = \underset{n}{\begin{bmatrix} — x_i — \end{bmatrix}} \underset{d}{} \leftarrow i \qquad y = \underset{n}{\begin{bmatrix} \text{"cat"} \\ \text{"dog"} \\ \text{"human"} \\ \text{"cat"} \end{bmatrix}} \underset{1}{} \leftarrow i \qquad W = \underset{k}{\begin{bmatrix} — W_{cat} — \\ — W_{dog} — \\ — W_{human} — \end{bmatrix}} \underset{d}{} \quad W_{y_i} \rightarrow$$

$$= \max\{0, -W_{cat}^T X_i + W_{dog}^T X_i\} + \max\{0, -W_{cat}^T X_i + W_{human}^T X_i\}$$

Multi-Class Hinge Loss

$$f(W) = \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, -w_{y_i}^T x_i + w_{c'} x_i\}$$

- This function is degenerate: $f(0) = \_\_\_$.
- As with binary SVM, we introduce an offset

$$f(W) = \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_{c'} x_i\}$$

"sum"-rule multi-class hinge loss

$$f(W) = \sum_{i=1}^{n} \max_{c' \neq y_i} \{\max\{0, 1 - w_{y_i}^T x_i + w_{c'} x_i\}\}$$

"max"-rule multi-class hinge loss

Multi-Class SVMs

- Idea: for a cat example, we want: $w_{cat}^T x_i > w_{dog}^T x_i$ and $w_{cat}^T x_i > w_{human}^T x_i$

$$f(W) = \sum_{i=1}^{n} \sum_{c' \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_{c'} x_i\}$$

$$f(W) = \sum_{i=1}^{n} \max_{c' \neq y_i} \{\max\{0, 1 - w_{y_i}^T x_i + w_{c'} x_i\}\}$$

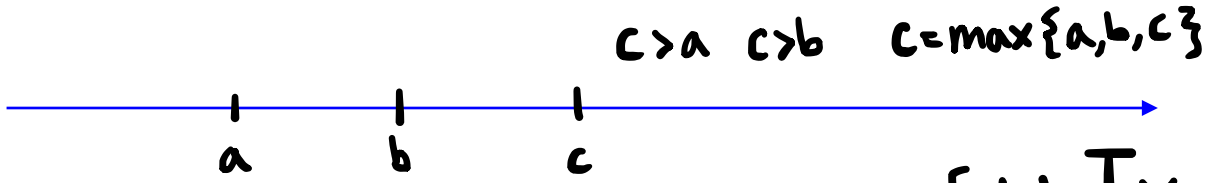
- For each training example 'i':
  - "Sum" rule penalizes for each 'c' that violates the constraint.
  - "Max" rule penalizes for one 'c' that violates the constraint the most.
    - "Sum" gives a penalty of 'k-1' for W=0, "max" gives a penalty of '1'.
- If we add L2-regularization, both are called multi-class SVMs:
  - "Max" rule is more popular, "sum" rule usually works better.
  - Both are convex upper bounds on the 0-1 loss.

Coming Up Next

MULTI-CLASS LOGISTIC REGRESSION

Multi-Class Logistic Regression

- Idea: for a cat example, we want: $w_{cat}^T x_i > w_{dog}^T x_i$ and $w_{cat}^T x_i > w_{human}^T x_i$
  - In other words: **we want $w_{cat}^T x_i$ to be max $w_c^T x_i$**



$$\left.\begin{array}{l} w_{cat}^T x_i > w_{dog}^T x_i \\ w_{cat}^T x_i > w_{human}^T x_i \end{array}\right\} \Leftrightarrow w_{cat}^T x_i = \max\left\{\begin{array}{l} w_{cat}^T x_i \\ w_{dog}^T x_i \\ w_{human}^T x_i \end{array}\right\}$$

$$\text{Count \#times } w_{cat}^T x_i < \max\left\{\begin{array}{l} w_{cat}^T x_i \\ w_{dog}^T x_i \\ w_{human}^T x_i \end{array}\right\} \text{ or } 0 < -w_{cat}^T x_i + \max\left\{\begin{array}{l} w_{cat}^T x_i \\ w_{dog}^T x_i \\ w_{human}^T x_i \end{array}\right\}$$

Q: What happens when W=0?

Multi-Class Logistic Regression

$$-w_{cat}^T x_i + \max \begin{Bmatrix} w_{cat}^T x_i \\ w_{dog}^T x_i \\ w_{human}^T x_i \end{Bmatrix} \approx -w_{cat}^T x_i + \log \begin{pmatrix} \exp(w_{cat}^T x_i) \\ + \\ \exp(w_{dog}^T x_i) \\ + \\ \exp(w_{human}^T x_i) \end{pmatrix}$$

log-sum-exp

- Ideas:
  1. use log-sum-exp to approximate max
  2. instead of counting number of times this quantity is positive, use this quantity as objective function

Q: What happens when W=0?

Multi-Class Logistic Regression

$$f(W) = \sum_{i=1}^{n} -w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right)$$

"Softmax loss"

$$f(W) = \sum_{i=1}^{n} -w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right) + \frac{\lambda}{2}\sum_{c=1}^{k}\sum_{j=1}^{d} w_{cj}^2$$

L2-regularized softmax loss

- Multi-class (multinomial) logistic regression: optimize W with L2-regularized softmax loss

Multi-Class Logistic Regression

$$f(W) = \sum_{i=1}^{n}\left[-w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k}\exp(w_c^T x_i)\right)\right] + \frac{\lambda}{2}\sum_{c=1}^{k}\sum_{j=1}^{d} w_{cj}^2$$

Tries to make $w_c^T x_i$ big for the correct label

Approximates $\max_c\{w_c^T x_i\}$ so tries to make $w_c^T x_i$ small for all labels.

Usual L_2-regularizer on elements of 'W'

- This objective is convex (should be clear for 1st and 3rd terms).
  - It's differentiable so you can use gradient descent.
- When k=2, equivalent to using binary logistic loss.
  - Not obvious at the moment.

Softmax Function

$$\text{Softmax} : \mathbb{R} \times \mathbb{R}^K \to (0, 1)$$

$$\text{Softmax}\left(z, \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_K \end{bmatrix}}_{z \in \begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix}}\right) = \frac{\exp(z)}{\sum_{c=1}^{K} \exp(z_c)}$$

- Evaluates "max-ness" of z compared to group
  - if z is big compared to every other z_c, softmax is close to 1

"Max-ness" to Find "Cat-ness"

$$P(y_i = \text{"cat"} \mid W, \cdot) : \mathbb{R}^d \rightarrow (0, 1)$$

$$P(y_i = \text{"cat"} \mid W, x_i) = \text{Softmax}\left(W_{cat}^T x_i, \begin{bmatrix} W_{cat}^T x_i \\ W_{dog}^T x_i \\ W_{human}^T x_i \end{bmatrix}\right)$$

"cat-ness"

"How big is my cat score compared to my dog score and human score?"

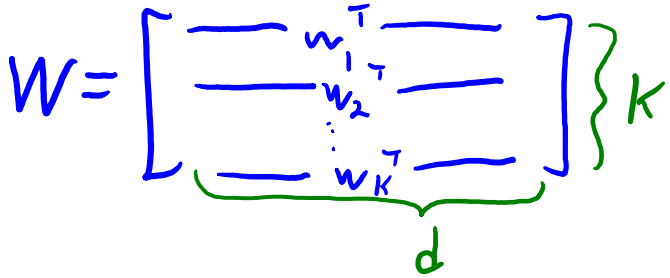
$W_{cat}^T x_i = 1.83$ → cat-ness = ~93.6%

$W_{dog}^T x_i = -1.17$ → dog-ness = ~0.04%

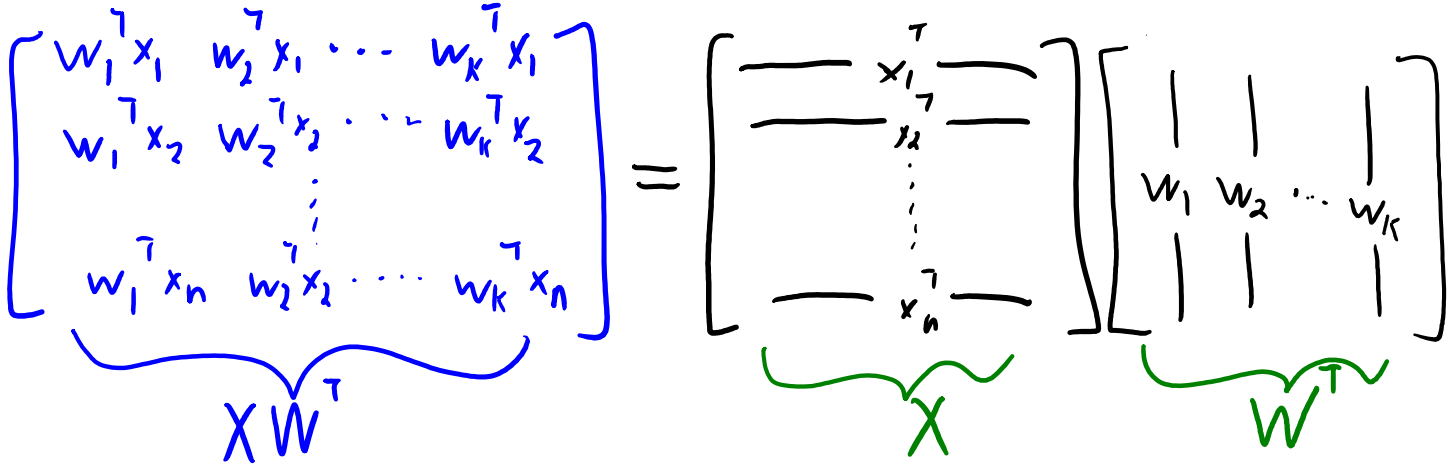
$W_{human}^T x_i = -2.20$ → human-ness = ~0.02%

Multi-Class Linear Prediction in Matrix Notation

- In multi-class linear classifiers our weights are:



- To predict on all training examples, we first compute all $w_c^T x_i$.
  - Or in matrix notation:



  - So predictions are maximum column indices of XW^T (which is 'n' by 'k').

How Do I Regularize W?

- The Frobenius norm of a ('k' by 'd') matrix 'W' is defined by:

$$\|W\|_F = \sqrt{\sum_{c=1}^{k}\sum_{j=1}^{d} w_{jc}^2}$$

(L_2-norm if you "stack" elements into one big vector)

- We can use this to write regularizer in matrix notation:

$$\frac{\lambda}{2}\sum_{c=1}^{k}\sum_{j=1}^{d} w_{cj}^2 = \frac{\lambda}{2}\sum_{c=1}^{k}\|w_c\|^2 \quad \text{("}L_2\text{-regularizer on each vector")}$$

$$= \frac{\lambda}{2}\|W\|_F^2 \quad \text{("Frobenius-regularizer on matrix")}$$

Summary

- Logistic loss uses a smooth convex approximation to the 0-1 loss.
- SVMs and logistic regression are very widely-used.
  - A lot of ML consulting: "find good features, use L2-regularized logistic/SVM".
  - Under certain conditions, can be viewed as "maximizing the margin".
  - Both are just linear classifiers (a hyperplane dividing into two halfspaces).
- Sigmoid function is a way to turn linear predictions into probabilities.

- One vs all turns a binary classifier into a multi-class classifier.
- Multi-class SVMs measure violation of classification constraints.
- Multi-class logistics regression: uses softmax loss

- Next time: kernels?

Online Classification with Perceptron

- Perceptron for online linear binary classification [Rosenblatt, 1957]
  - Start with $w_0 = 0$.
  - At time 't' we receive features x_t.
  - We predict $\hat{y}_t = \text{sign}(w_t^T x_t)$.
  - If $\hat{y}_t \neq y_t$, then set $w_{t+1} = w_t + y_t x_t$.
    - Otherwise, set $w_{t+1} = w_t$.

(Slides are old so above I'm using subscripts of 't' instead of superscripts.)

- Perceptron mistake bound [Novikoff, 1962]:
  - Assume data is linearly-separable with a "margin":
    - There exists w^* with $\|w^*\|=1$ such that $\text{sign}(x_t^T w^*) = \text{sign}(y_t)$ for all 't' and $|x^T w^*| \geq \gamma$.
  - Then the number of total mistakes is bounded.
    - No requirement that data is IID.

> 0

Perceptron Mistake Bound

- Let's normalize each x_t so that $\|x_t\| = 1$.
  - Length doesn't change label.
- Whenever we make a mistake, we have $\text{sign}(y_t) \neq \text{sign}(w_t^T x_t)$ and

$$\begin{aligned}
\|w_{t+1}\|^2 &= \|w_t + y x_t\|^2 \\
&= \|w_t\|^2 + 2\underbrace{y_t w_t^T x_t}_{<0} + 1 \\
&\leq \|w_t\|^2 + 1 \\
&\leq \|w_{t-1}\|^2 + 2 \\
&\leq \|w_{t-2}\|^2 + 3.
\end{aligned}$$

- So after 'k' errors we have $\|w_t\|^2 \leq k$.

Perceptron Mistake Bound

- Let's consider a solution w^*, so $\text{sign}(y_t) = \text{sign}(x_t^T w^*)$.
  - And let's choose a w^* with $\|w^*\| = 1$,
- Whenever we make a mistake, we have:

$$\begin{aligned}
\|w_{t+1}\| &= \|w_{t+1}\|\|w_*\| \\
&\geq w_{t+1}^T w_* \\
&= (w_t + y_t x_t)^T w_* \\
&= w_t^T w_* + y_t x_t^T w_* \\
&= w_t^T w_* + |x_t^T w_*| \\
&\geq w_t^T w_* + \gamma.
\end{aligned}$$

  - Note: $w_t^T w_* \geq 0$ by induction (starts at 0, then at least as big as old value plus γ).
- So after 'k' mistakes we have $\|w_t\| \geq \gamma k$.

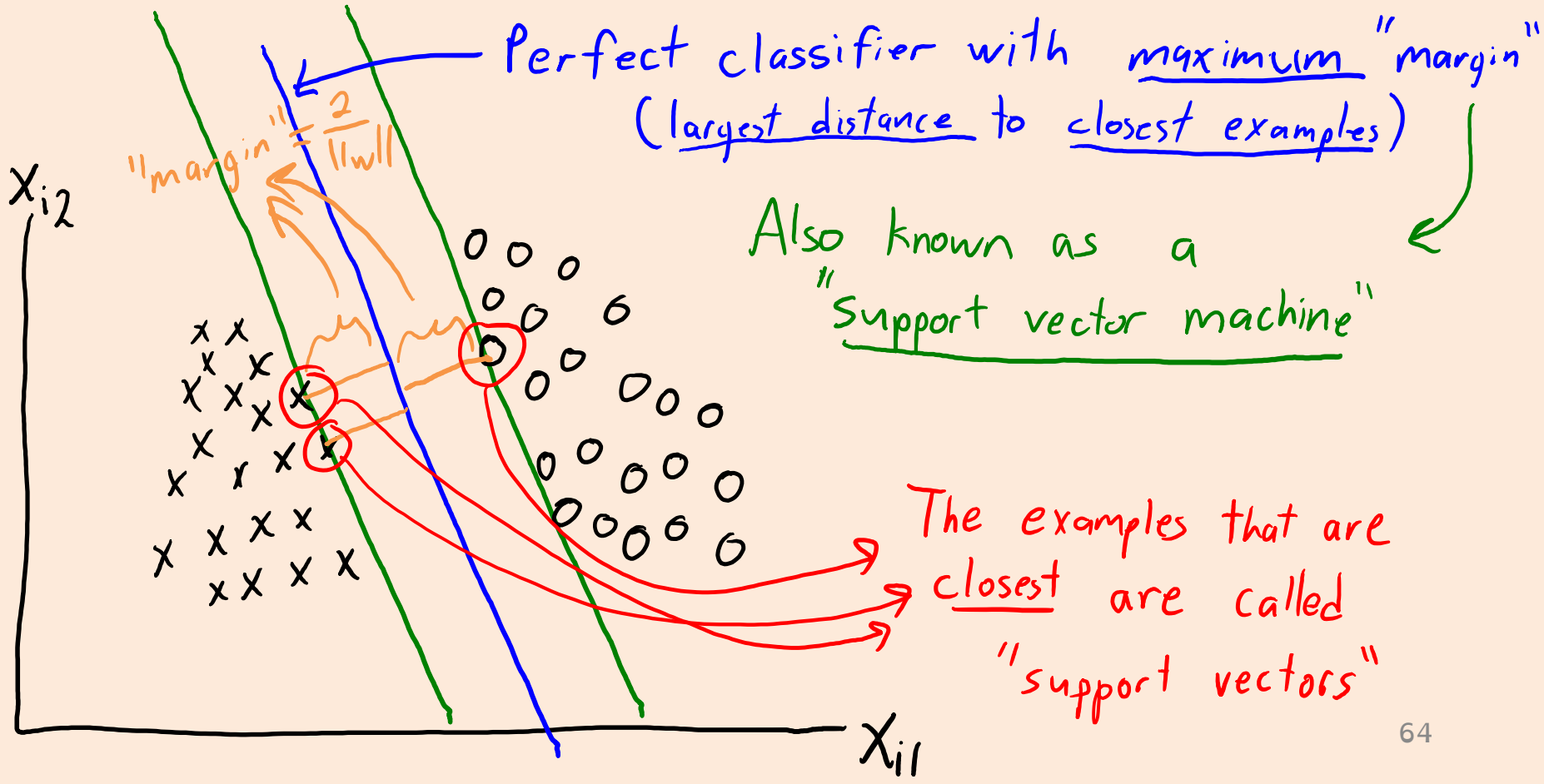
Perceptron Mistake Bound

- So our two bounds are $\|w_t\| \leq \text{sqrt}(k)$ and $\|w_t\| \geq \gamma k$.

- This gives $\gamma k \leq \text{sqrt}(k)$, or a maximum of $1/\gamma^2$ mistakes.
  - Note that $\gamma > 0$ by assumption and is upper-bounded by one by $\|x\| \leq 1$.
  - After this 'k', under our assumptions we're guaranteed to have a perfect classifier.

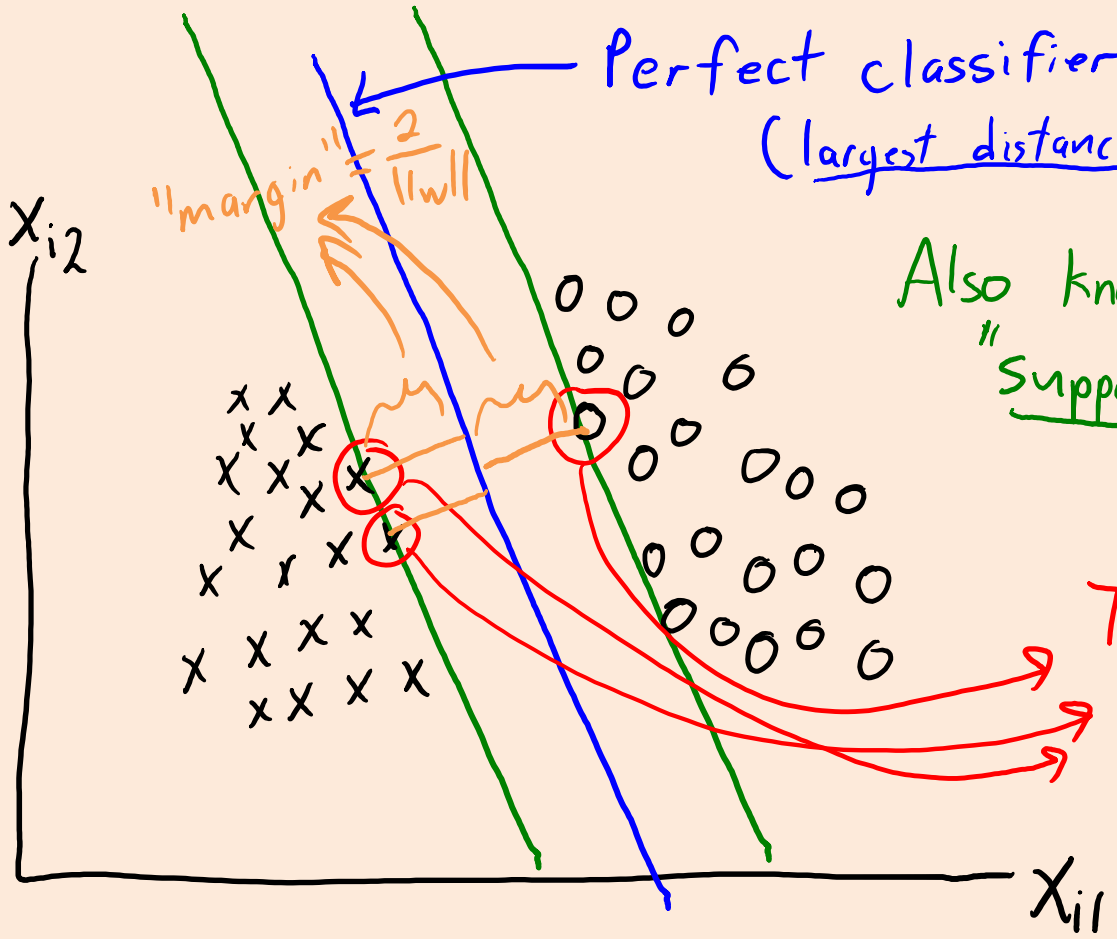
Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Perfect classifier with maximum "margin"
(largest distance to closest examples)

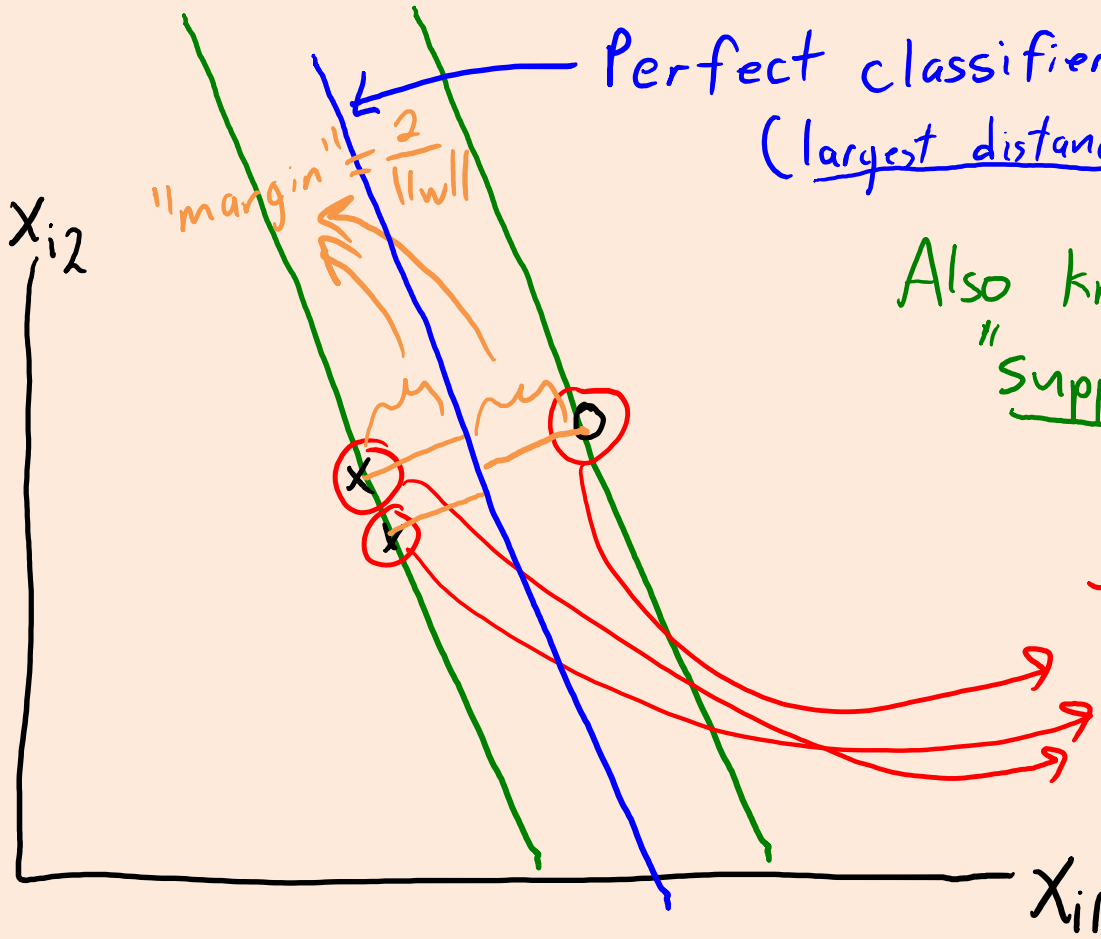
Also known as a "support vector machine"

Final classifier only depends on support vectors

The examples that are closest are called "support vectors"

Maximum-Margin Classifier

- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Perfect classifier with maximum "margin"
(largest distance to closest examples)

Also known as a "support vector machine"

Final classifier only depends on support vectors

You could throw away the other examples and get the same classifier.

The examples that are closest are called "support vectors"

Support Vector Machines

- For linearly-separable data, SVM minimizes:

$$f(w) = \frac{1}{2}\|w\|^2$$ (equivalent to maximizing margin $\frac{2}{\|w\|}$)

  - Subject to the constraints that: (see Wikipedia/textbooks)

$$w^T x_i \geq 1 \quad \text{for } y_i = 1$$
$$w^T x_i \leq -1 \quad \text{for } y_i = -1$$

(classify all examples correctly)

- But most data is not linearly separable.
- For non-separable data, try to minimize violation of constraints:

If $w^T x_i \leq -1$ and $y_i = -1$ then "violation" should be zero.

If $w^T x_i \geq -1$ and $y_i = -1$ then we "violate constraint" by $1 + w^T x_i$

⋮

→ Constraint violation is the hinge loss.

Support Vector Machines

- Try to maximizing margin and also minimizing constraint violation:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{1}{2}\|w\|^2$$

Hinge loss for example 'i':

if's the amount we violate $y_i w^T x_i \geq 1$

"slack"

Original SVM objective: encourages large margin.

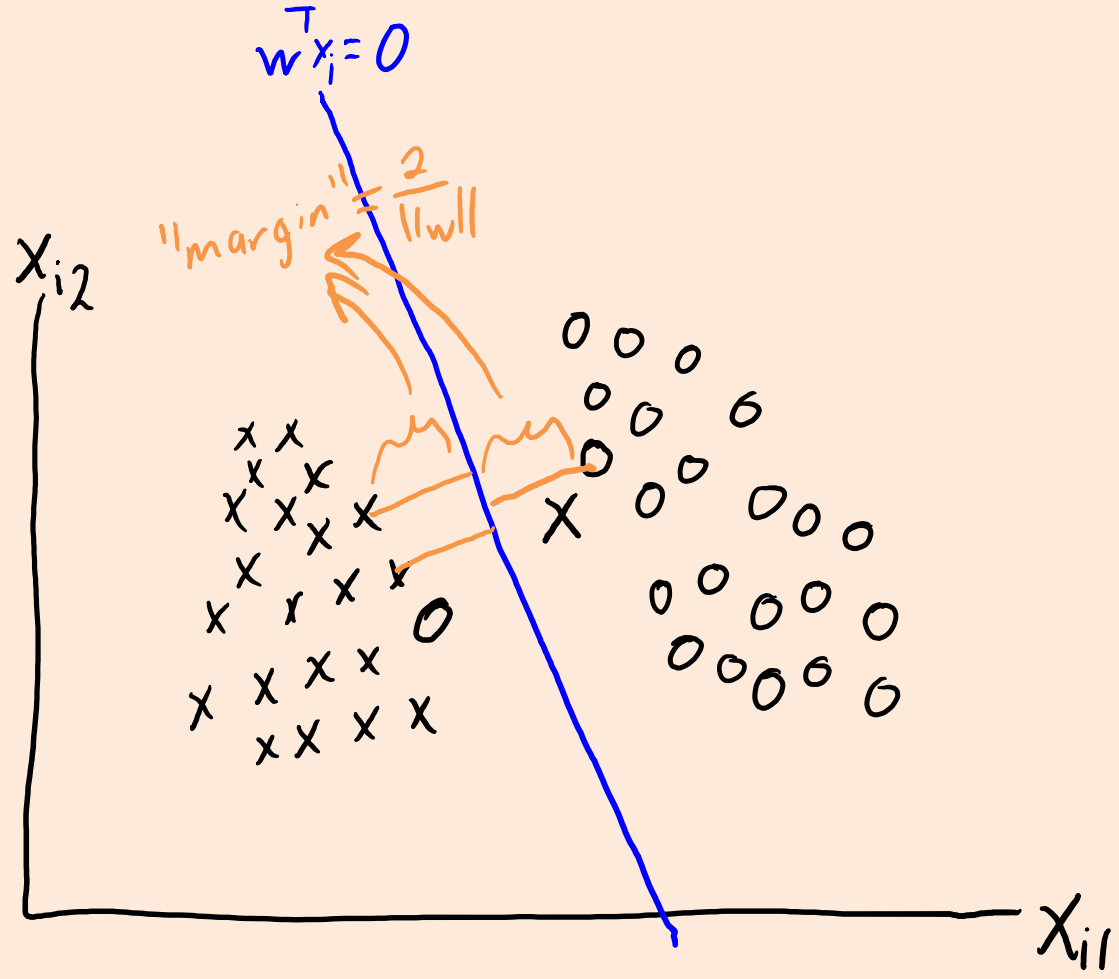
- We typically control margin/violation trade-off with parameter "λ":

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$

- This is the standard SVM formulation (L2-regularized hinge).
  - Some formulations use $\lambda = 1$ and multiply hinge by 'C' (equivalent).

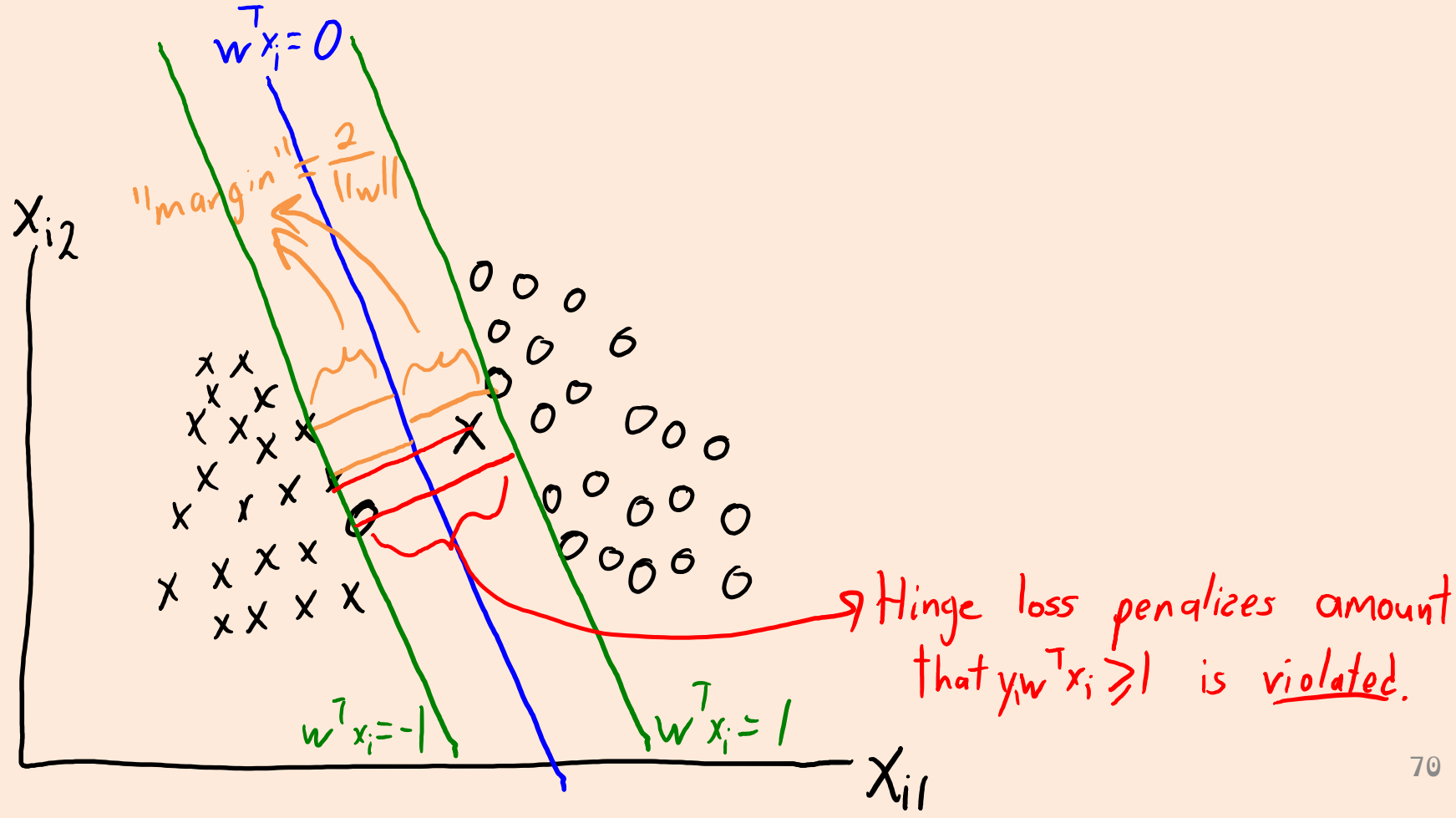
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

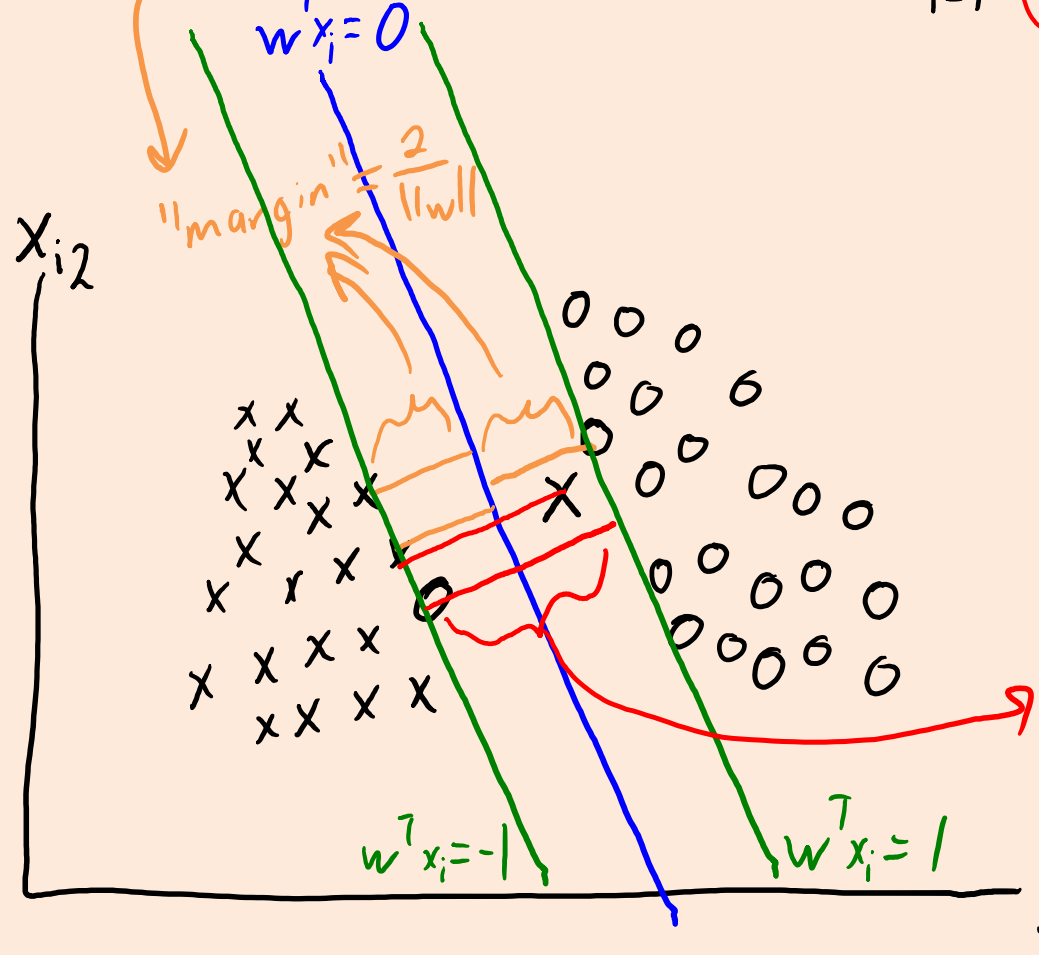
- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$



Logistic regression can be viewed as smooth approximation to SVMs.

But, no concept of "support vectors" with logistic loss.

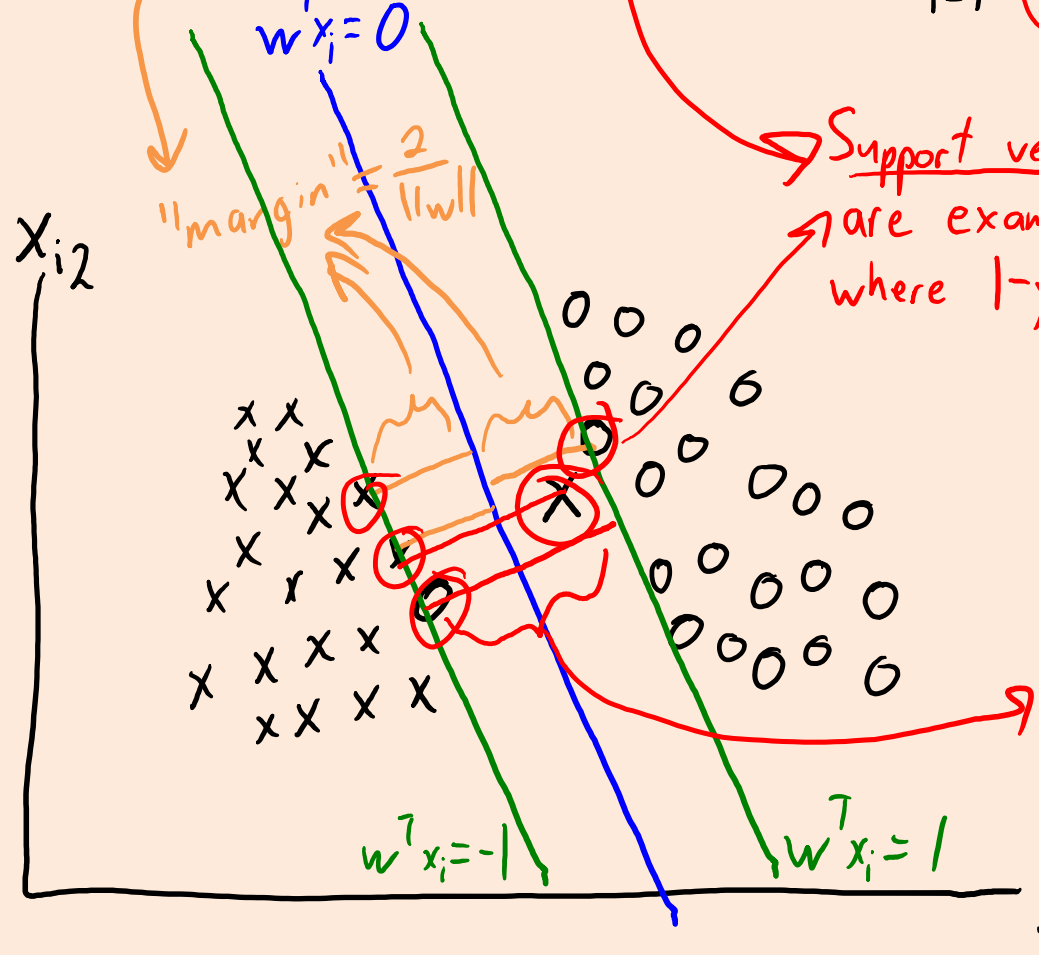
λ controls trade-off between having large margin and classifying examples correctly.

Hinge loss penalizes amount that $y_i w^T x_i \geq 1$ is violated.

Support Vector Machines for Non-Separable

- Non-separable case:

$$f(w) = \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2}\|w\|^2$$



Logistic regression can be viewed as smooth approximation to SVMs.

But, no concept of "support vectors" with logistic loss.

Support vectors are examples 'i' where $1 - y_i w^T x_i \geq 0$

λ controls trade-off between having large margin and classifying examples correctly.

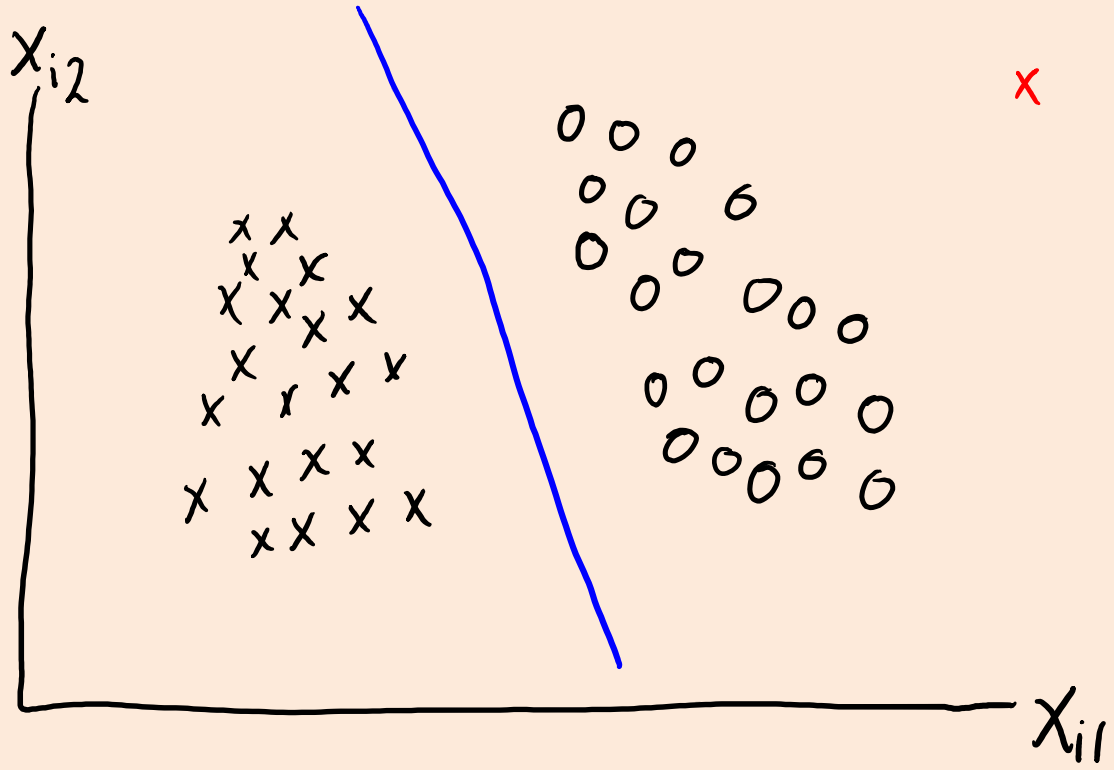
Hinge loss penalizes amount that $y_i w^T x_i \geq 1$ is violated.

Discussion of Various Linear Classifiers

- Perceptron vs. logistic vs. SVM:
  - These linear classifiers are all extremely similar. They are basically just variations on reasonable methods to learn a classifier that uses the rule $$\hat{y}_i = \text{sign}(w^Tx_i)$$. (The online vs. offline issue is a red herring, you can train logistic/SVMs online using stochastic gradient and you can write a linear program that will give you a minimizer of the perceptron objective).
  - If you want to explore the small differences, these are some of the usual arguments:
    - The perceptron has largely been replaced by logistic/SVM, except in certain subfields like theory (it is easy to prove things about perceptrons) and natural language processing (mostly historical reasons). Perceptrons have the potential disadvantages of non-regularized models (non-uniqueness and potential non-existence of the solution, potential high sensitivity to small changes in the data, and non-robustness to irrelevant features). However, perceptrons do not interact well with regularization: if you add L2-regularization and the dataset is linearly-separable, then the solution only exists as a limit and it is actually $$w=0$$ (although it may still work in practice).
    - A usual criticism of logistic regression by people that favour SVMs is that, if the data is linearly separable, then the solution only exists as a limit as some elements $$w$$ go to plus or minus $$\infty$$. However, this argument disappears if you add regularization. A second argument traditionally made by SVM people is that you can't kernelize logistic regression, but this is now known to be incorrect (we'll cover a general kernelization strategy for L2-regularized linear classifiers in one of the next two classes).
    - The remaining differences between logistic and SVMs is that logistic regression is smooth while SVMs have support vectors. This means that the logistic regression training problem is easier from an optimization perspective (we'll get to this next class). But if you have very few support vectors, you can only take advantage of this with SVMs (or perceptrons), and this is especially important if you are using kernels.
- Regarding other linear predictors for binary classification, there are a few more:
  - Probit regression uses the Gaussian CDF in place of the logistic sigmoid function. This has very similar properties to logistic regression, but it's harder to generalize to the multi-class case (while probit regression is better if you are using a "Bayesian" estimator). You could actually use any CDF as your sigmoid function, and if there is some asymmetry between the classes using an extreme value distribution is sometimes advocated in statistics.
  - In neural networks, they sometimes use tanh in place of the logistic sigmoid function, and the reason to do this is to get values into the interval [-1,1] instead of [0,1].
  - If you want to keep support vectors but get a smooth optimization problem, you can square the hinge loss (making it once but not twice differentiable), and this is called smooth SVMs. Alternately, you could replace the non-differentiable kink with a small smooth part, and this is called Huberized SVMs.
  - Finally, some people actually just apply least squares to classification problems. If you use a flexible enough basis/kernel, then the 'bad' errors may not actually be that harmful.

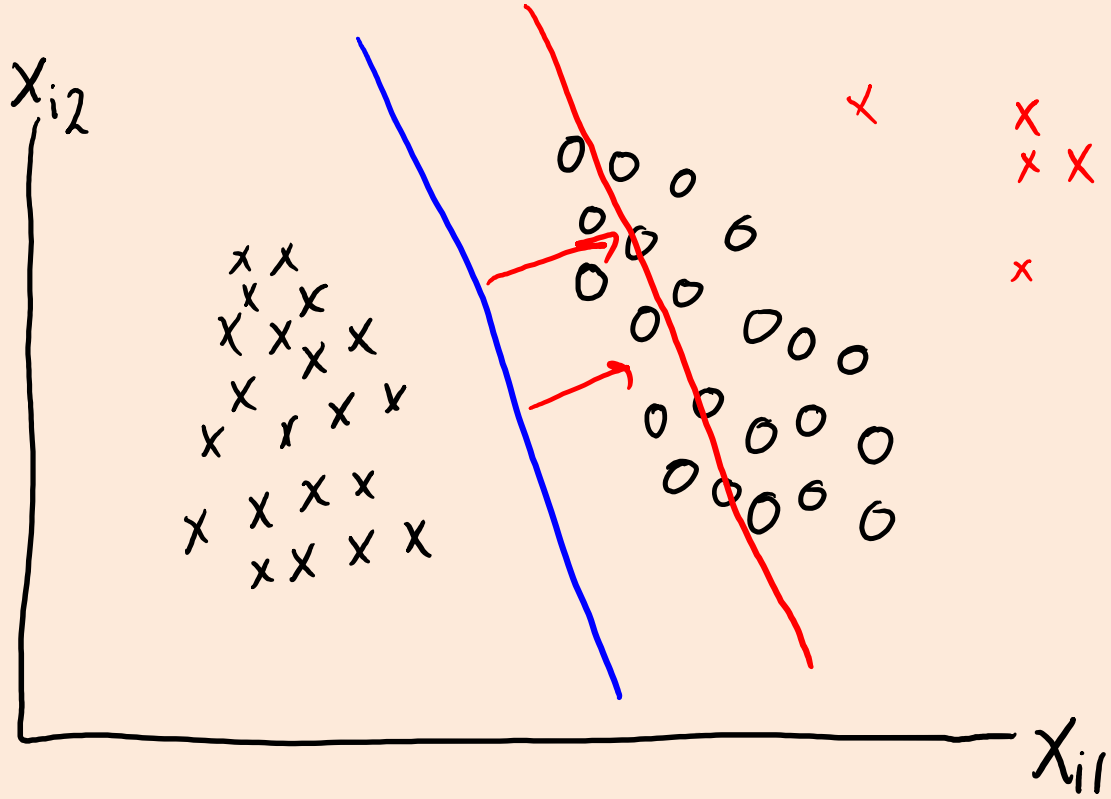
Robustness and Convex Approximations

- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



Robustness and Convex Approximations

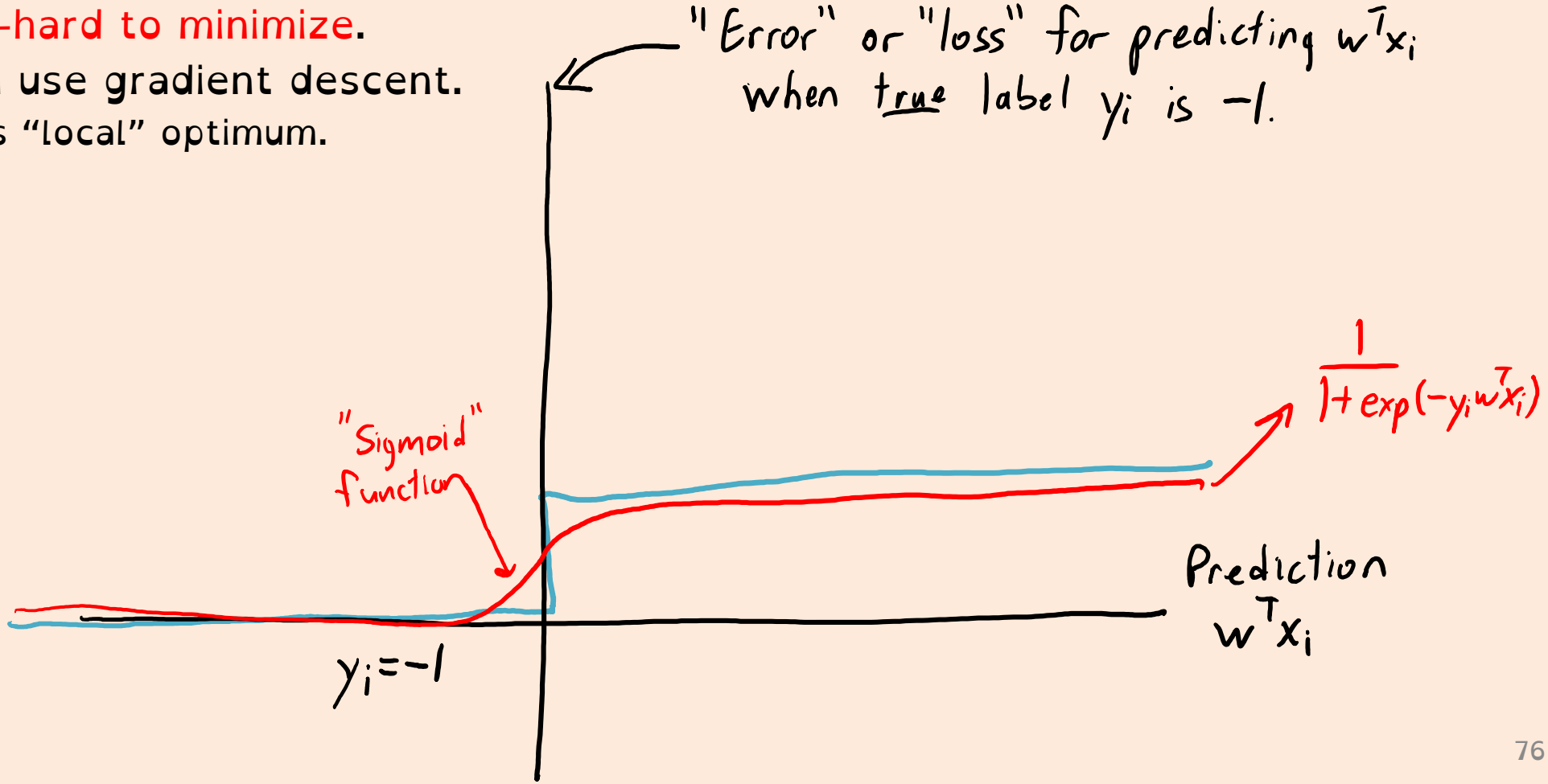
- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



- But performance degrades if we have many outliers.

Non-Convex 0-1 Approximations

- There exists some smooth non-convex 0-1 approximations.
  - Robust to many/extreme outliers.
  - Still NP-hard to minimize.
  - But can use gradient descent.
    - Finds "local" optimum.



"Robust" Logistic Regression

- A recent idea: add a "fudge factor" v_i for each example.

$$f(w,v) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i w^T x_i + v_i\right)\right)$$

- If $w^T x_i$ gets the sign wrong, we can "correct" the mis-classification by modifying v_i.
  - This makes the training error lower but doesn't directly help with test data, because we won't have the v_i for test data.
  - But having the v_i means the 'w' parameters don't need to focus as much on outliers (they can make $|v_i|$ big if sign($w^T x_i$) is very wrong).

"Robust" Logistic Regression

- A recent idea: add a "fudge factor" v_i for each example.

$$f(w,v) = \sum_{i=1}^{n} \log\left(1+\exp\left(-y_i w^T x_i + v_i\right)\right)$$

- If $w^T x_i$ gets the sign wrong, we can "correct" the mis-classification by modifying v_i.
- A problem is that we can ignore the 'w' and get a tiny training error by just updating the v_i variables.
- But we want most v_i to be zero, so "robust logistic regression" puts an L1-regularizer on the v_i values:

- You would probably also want to regularize the 'w' with different λ.

$$f(w,v) = \sum_{i=1}^{n} \log\left(1+\exp\left(-y_i w^T x_i + v_i\right)\right) + \lambda\|v\|_1$$

"All-Pairs" and ECOC Classification

- Alternative to "one vs. all" to convert binary classifier to multi-class is "all pairs".
  - For each pair of labels 'c' and 'd', fit a classifier that predicts +1 for examples of class 'c' and -1 for examples of class 'd' (so each classifier only trains on examples from two classes).
  - To make prediction, take a vote of how many of the (k-1) classifiers for class 'c' predict +1.
  - Often works better than "one vs. all", but not so fun for large 'k'.
- A variation on this is using "error correcting output codes" from information theory (see Math 342).
  - Each classifier trains to predict +1 for some of the classes and -1 for others.
  - You setup the +1/-1 code so that it has an "error correcting" property.
    - It will make the right decision even if some of the classifiers are wrong.

Motivation: Dog Image Classification

- Suppose we're classifying images of dogs into breeds:

- What if we have images where class label isn't obvious?
  - Siberian husky vs. Inuit dog?

https://www.slideshare.net/angjoo/dog-breed-classification-using-part-localization
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Preferences

- Do we need to throw out images where label is ambiguous?
  - We don't have the y_i.

  - We want classifier to prefer Siberian husky over bulldog, Chihuahua, etc.
    - Even though we don't know if these are Siberian huskies or Inuit dogs.

  - Can we design a loss that enforces preferences rather than "true" labels?

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Pairwise Preferences (Ranking)

- Instead of y_i, we're given list of (c_1,c_2) preferences for each 'i':

We want $w_{c_1}^T x_i > w_{c_2}^T x_i$ for these particular (c_1,c_2) values

- Multi-class classification is special case of choosing (y_i,c) for all 'c'.

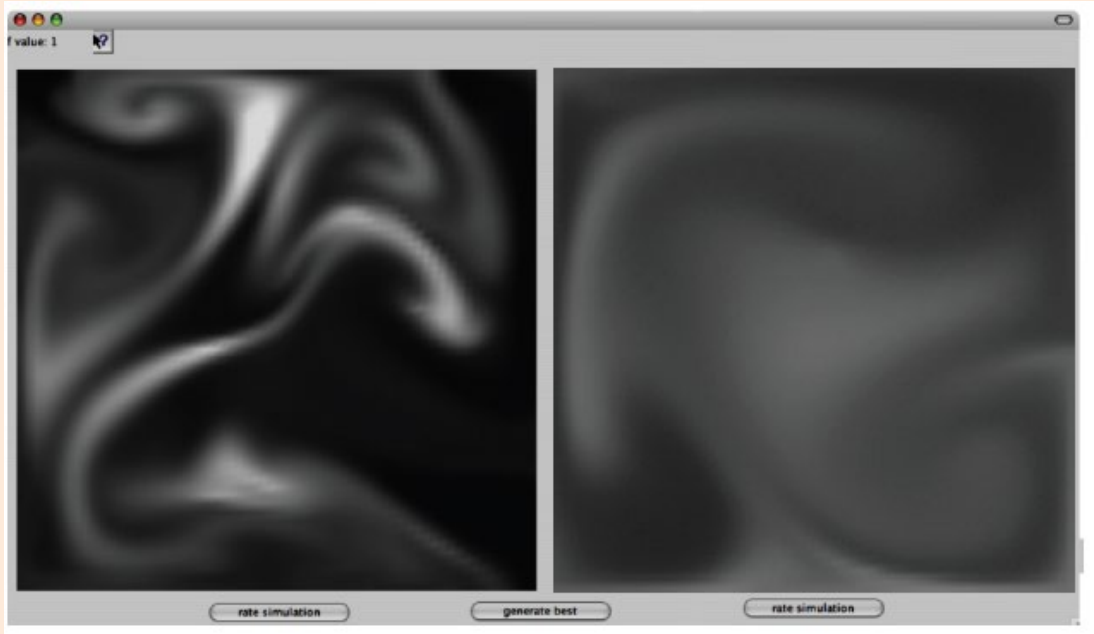
- By following the earlier steps, we can get objectives for this setting:

$$\sum_{i=1}^{n} \sum_{(c_1,c_2)} \max\{0, 1 - w_{c_1}^T x_i + w_{c_2}^T x_i\} + \frac{\lambda}{2}\|W\|_F^2$$

"sum" version of multi-class SVM

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for computer graphics:
  - We have a smoke simulator, with several parameters:

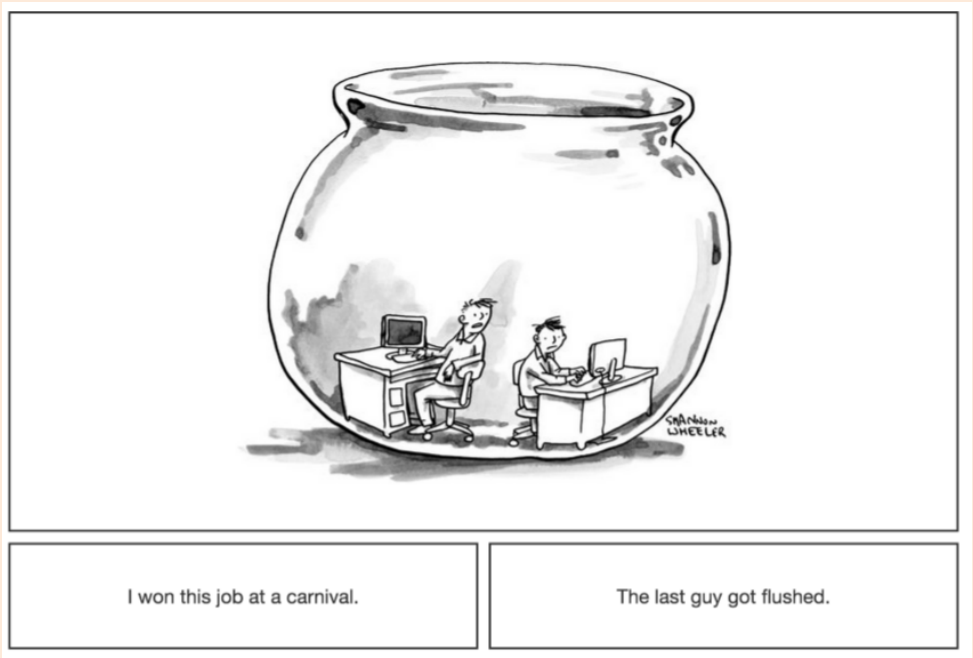


  - Don't know what the optimal parameters are, but we can ask the artist:
    - "Which one looks more like smoke"?

https://circle.ubc.ca/bitstream/handle/2429/30519/ubc_2011_spring_brochu_eric.pdf?sequence=3

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for humour:
  - New Yorker caption contest:



  - "Which one is funnier"?

https://homes.cs.washington.edu/~jamieson/resources/next.pdf

Risk Scores

- In medicine/law/finance, risk scores are sometimes used to give probabilities:

1.	**C**ongestive Heart Failure	1 point	...
2.	**H**ypertension	1 point	+ ...
3.	**A**ge $\geq$ 75	1 point	+ ...
4.	**D**iabetes Mellitus	1 point	+ ...
5.	Prior **S**troke or Transient Ischemic Attack	**2** points	+
		SCORE	=

SCORE	0	1	2	3	4	5	6
RISK	1.9%	2.8%	4.0%	5.9%	8.5%	12.5%	18.2%

Figure 1: $CHADS_2$ risk score of Gage et al. (2001) to assess stroke risk (see www.mdcalc.com for other medical scoring systems). The variables and points of this model were determined by a panel of experts, and the risk estimates were computed empirically from data.

- Get integer-valued "points" for each "risk factor", and probability is computed from data based on people with same number of points.
- Less accurate than fancy models, but interpretable and can be done by hand.
  - Some work on trying to "learn" the whole thing (like doing feature selection then rounding).

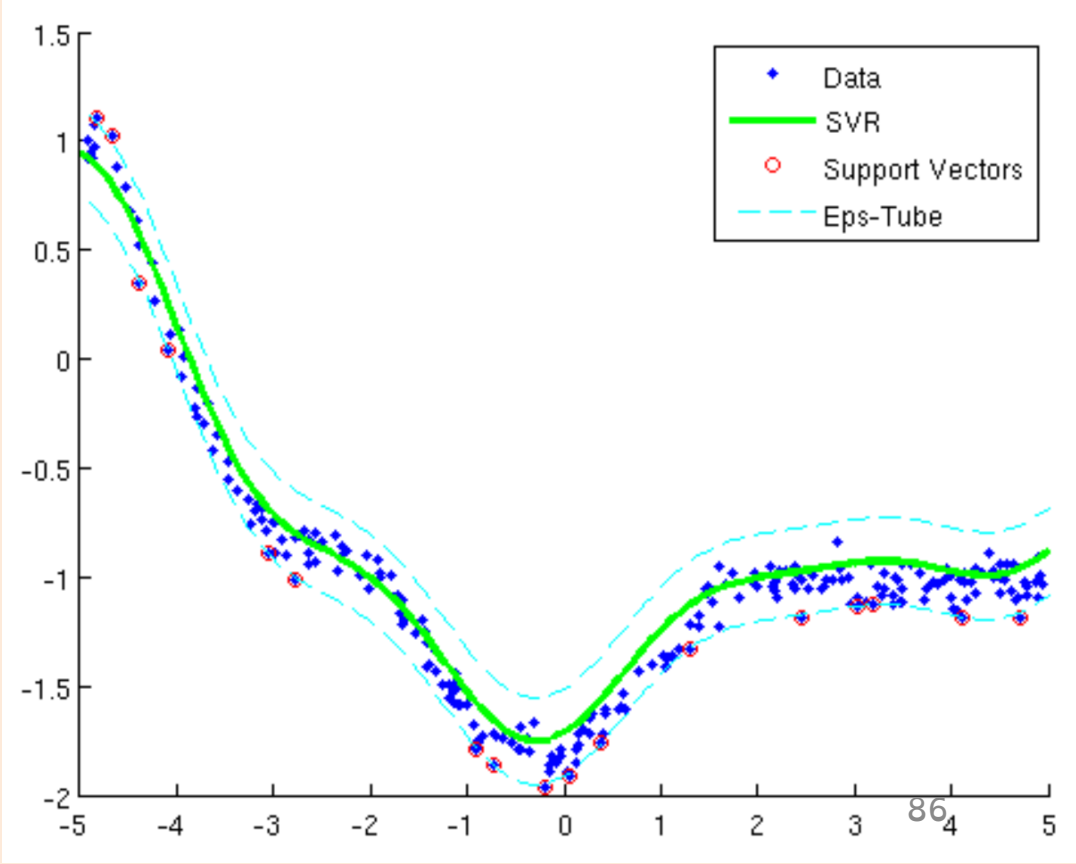
https://arxiv.org/pdf/1610.00168.pdf

Support Vector Regression

- Support vector regression objective (with hyper-parameter ϵ):

$$f(w) = \sum_{i=1}^{n} \max\{0, |w^T x_i - y_i| - \epsilon\} + \frac{\lambda}{2}\|w\|^2$$

  - Looks like L2-regularized robust regression with the L1-loss.
  - But have loss of 0 if $\hat{y}_i$ within ϵ of $\tilde{y}_i$.
    - So doesn't try to fit data exactly.
      - This can help fight overfitting.
  - Support vectors are points with loss>0.
    - Points outside the "epsilon-tube".
  - Example with Gaussian-RBFs as features:

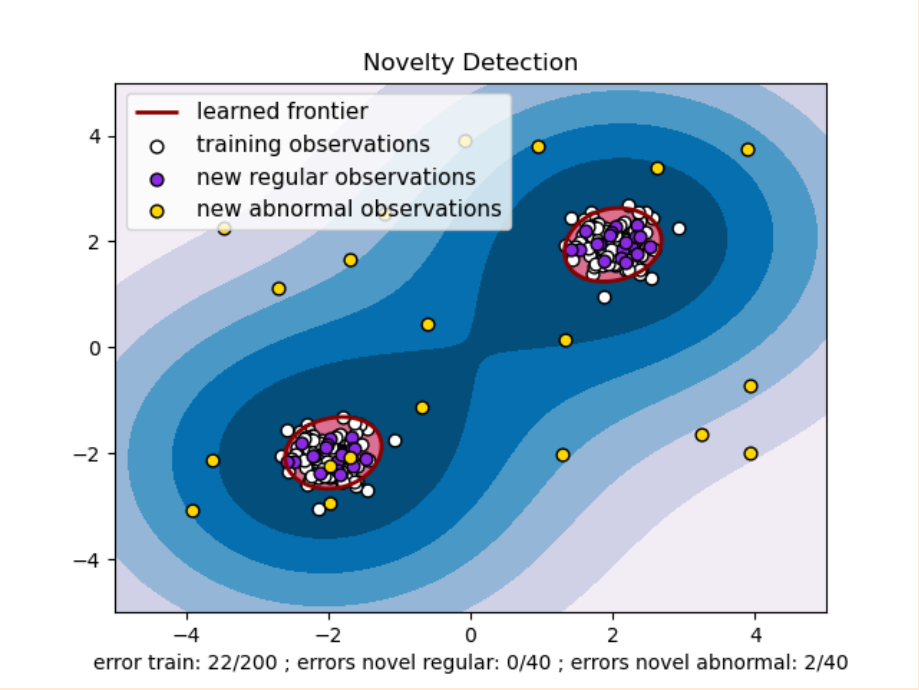


1-Class SVMs

- 1-class SVMs for outlier detection.

$$f(w, w_0) = \sum_{i=1}^{N} [\max\{0, w_0 - w^T x_i\} - w_0] + \frac{\lambda}{2}\|w\|_2^2$$

  - Variables are 'w' (vector) and 'w_0' (scalar).
  - Only trains on "inliers".
    - Tries to make $w^T x_i$ bigger than w_0 for inliers.
    - At test time: says "outlier" if $w^T x_i < w_0$.
    - Usually used with RBFs.



https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html