

CPSC 340: Machine Learning and Data Mining

More Linear Classifiers

Fall 2019

Last Time: Classification using Regression and SVMs

- Binary classification using sign of linear models:

Fit model $y_i = w^T x_i$ and predict using $\text{sign}(w^T x_i)$

$\begin{cases} +1 & \text{if } w^T x_i > 0 \\ -1 & \text{otherwise} \end{cases}$

- We considered different training “error” functions:

- Squared error: $(w^T x_i - y_i)^2$.

- If $y_i = +1$ and $w^T x_i = +100$, then squared error $(w^T x_i - y_i)^2$ is huge.

- 0-1 classification error: $(\text{sign}(w^T x_i) = y_i)?$

- Non-convex and hard to minimize in terms of ‘w’ (unless optimal error is 0).

- Degenerate convex approximation to 0-1 error: $\max\{0, -y_i w^T x_i\}$.

- Has a degenerate solution of 0.

- Hinge loss: $\max\{0, 1 - y_i w^T x_i\}$.

- Convex upper bound on number of classification errors.

- With L2-regularization, it’s called a support vector machine (SVM).

'λ' vs 'C' as SVM Hyper-Parameter

- We've written SVM in terms of regularization parameter 'λ':

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

- Some software packages instead have regularization parameter 'C':

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

- In our notation, this corresponds to using $\lambda = 1/C$.
 - Equivalent to just multiplying $f(w)$ by constant.
 - Note interpretation of 'C' is different: **high regularization for small 'C'**.
 - You can think of 'C' as "how much to focus on the classification error".

Logistic Loss

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

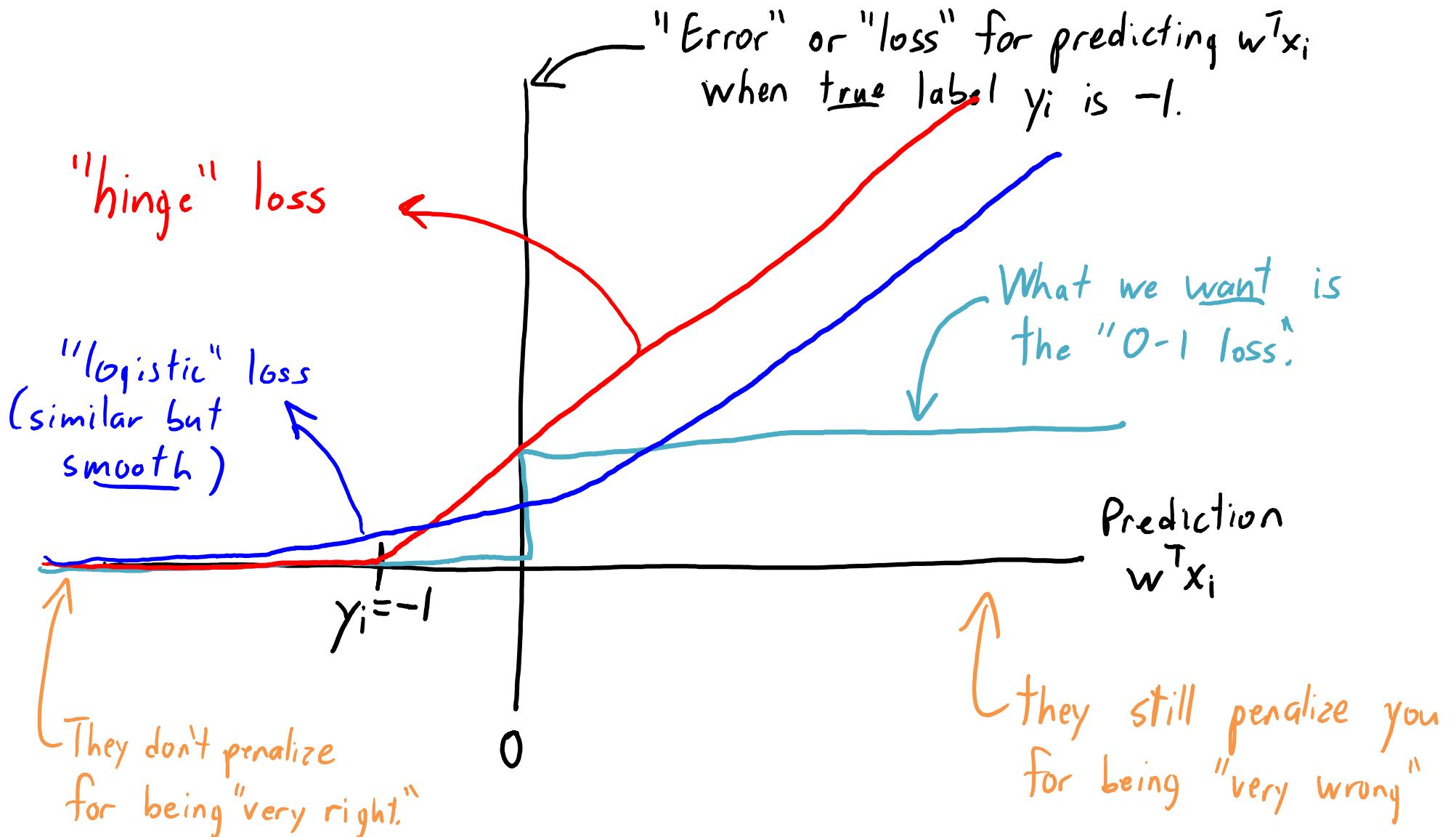
$$\max\{0, -y_i w^T x_i\} \approx \log(\underbrace{\exp(0) + \exp(-y_i w^T x_i)}_1)$$

- 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: $w=0$ now gives an error of $\log(2)$ 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



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 good test error.
 - With low-dimensional features using RBF basis and regularization.
 - With high-dimensional features and regularization.
 - Smoother predictions than random forests.

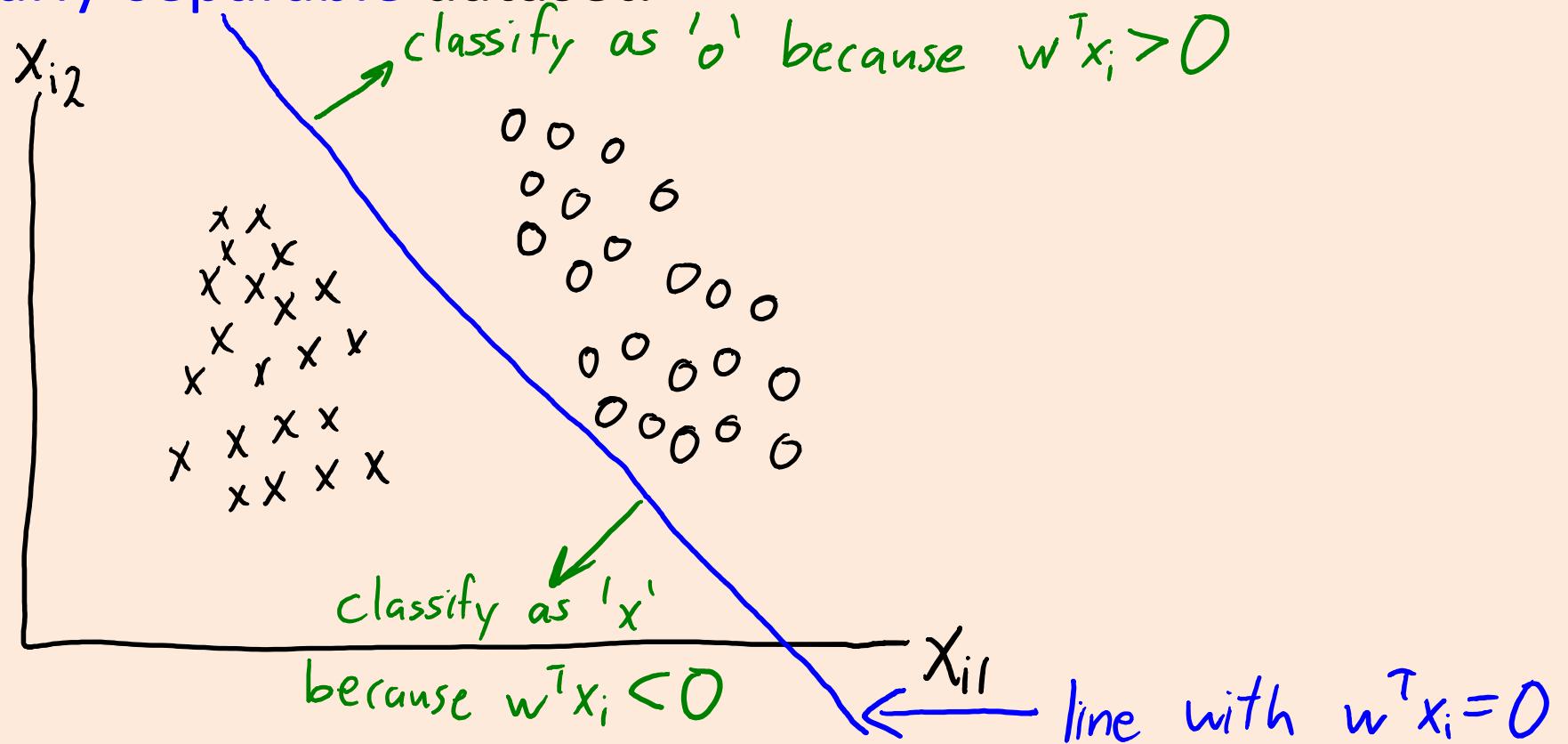
Comparison of “Black Box” Classifiers

- 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?”

(pause)

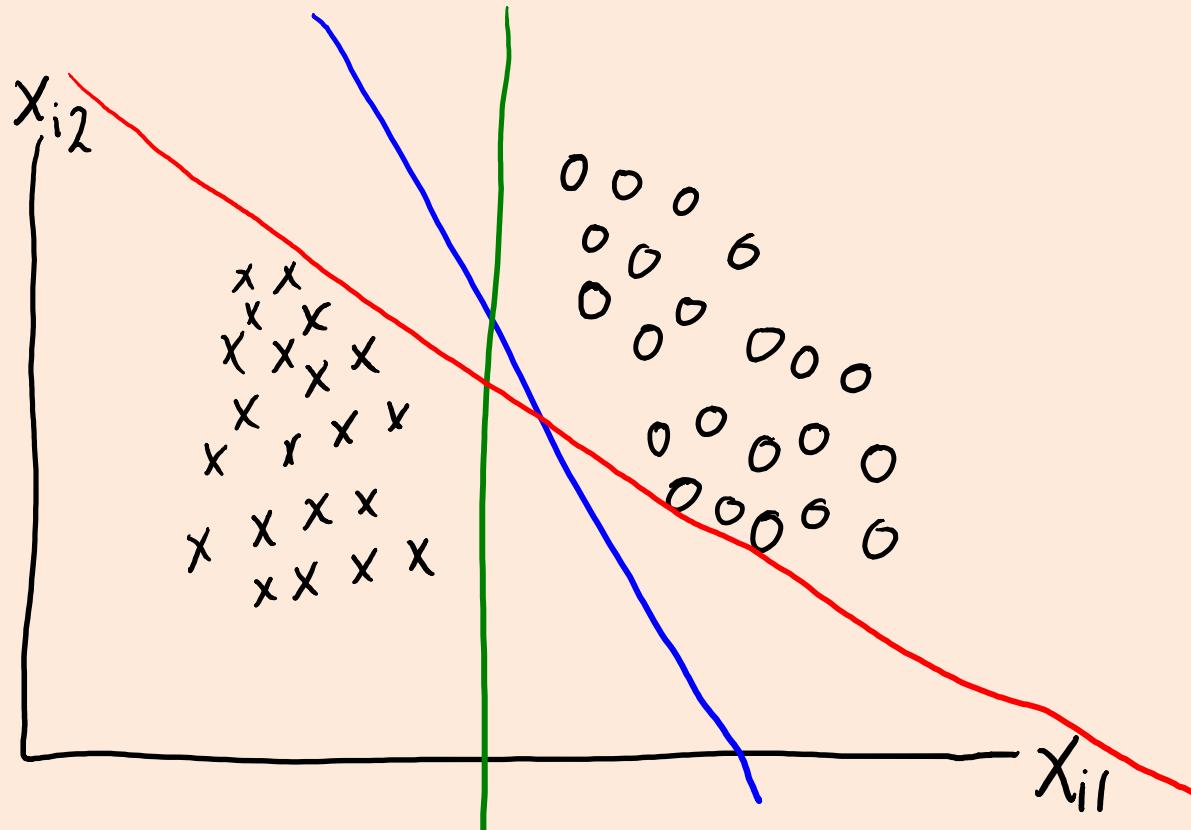
Maximum-Margin Perspective

- Consider a **linearly-separable** dataset.



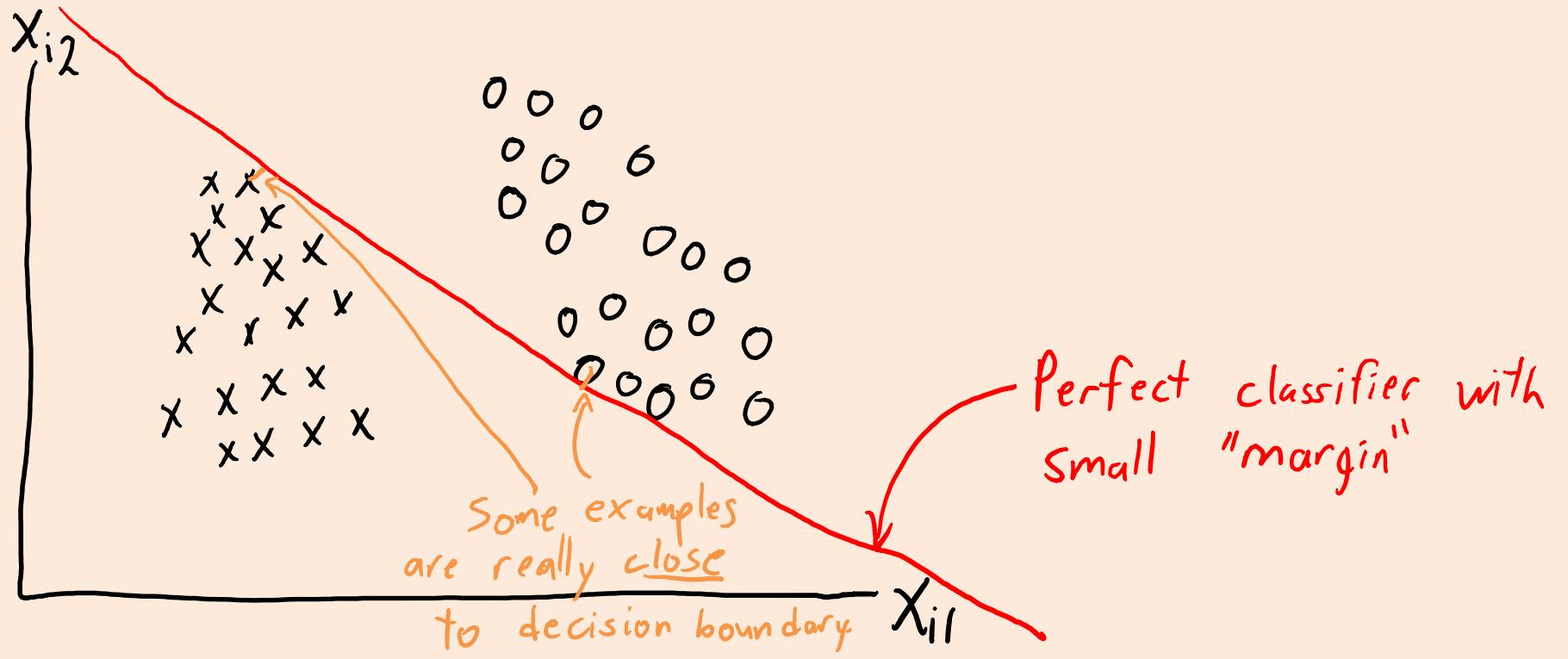
Maximum-Margin Perspective

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



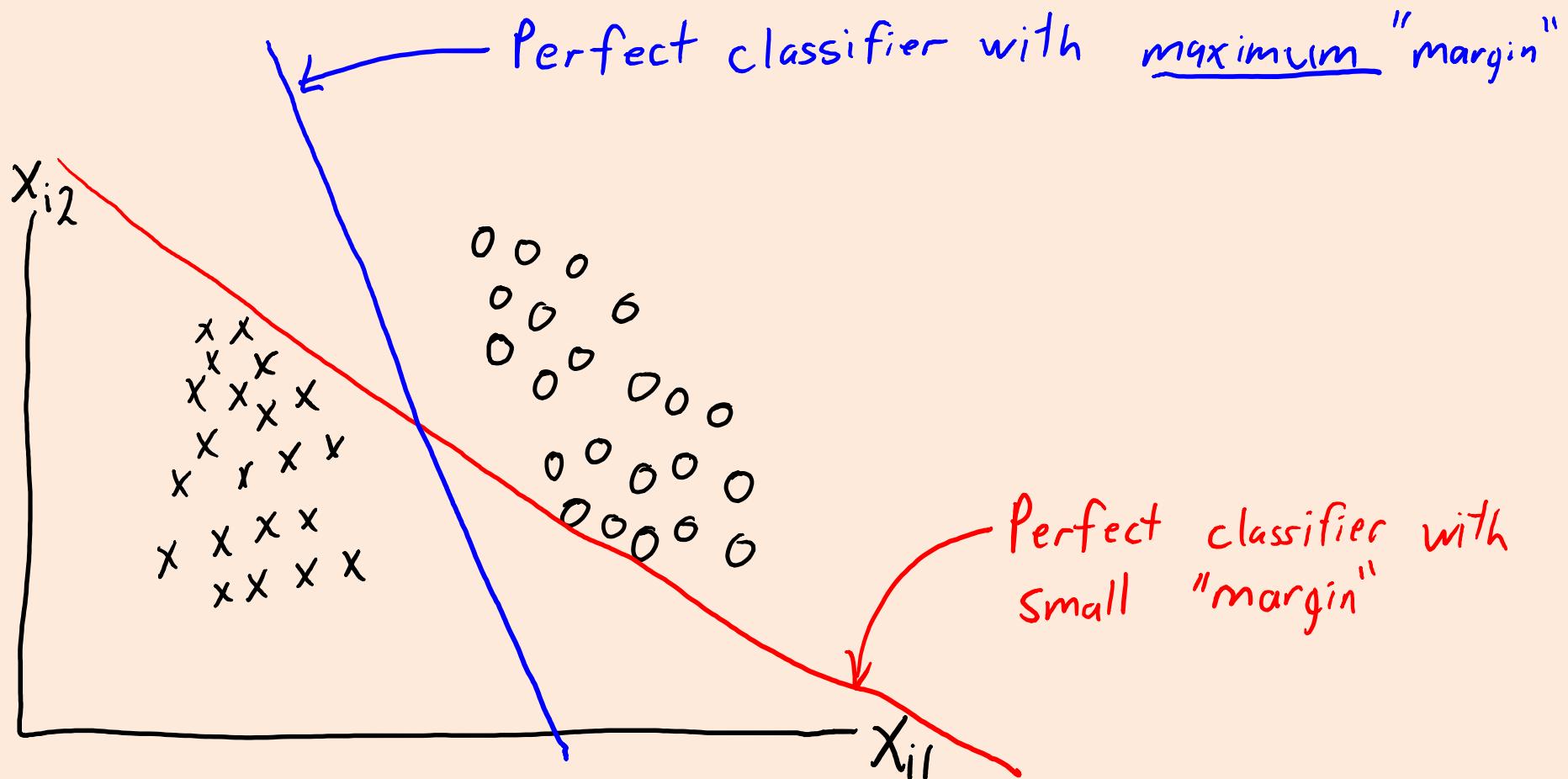
Maximum-Margin Perspective

- 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.

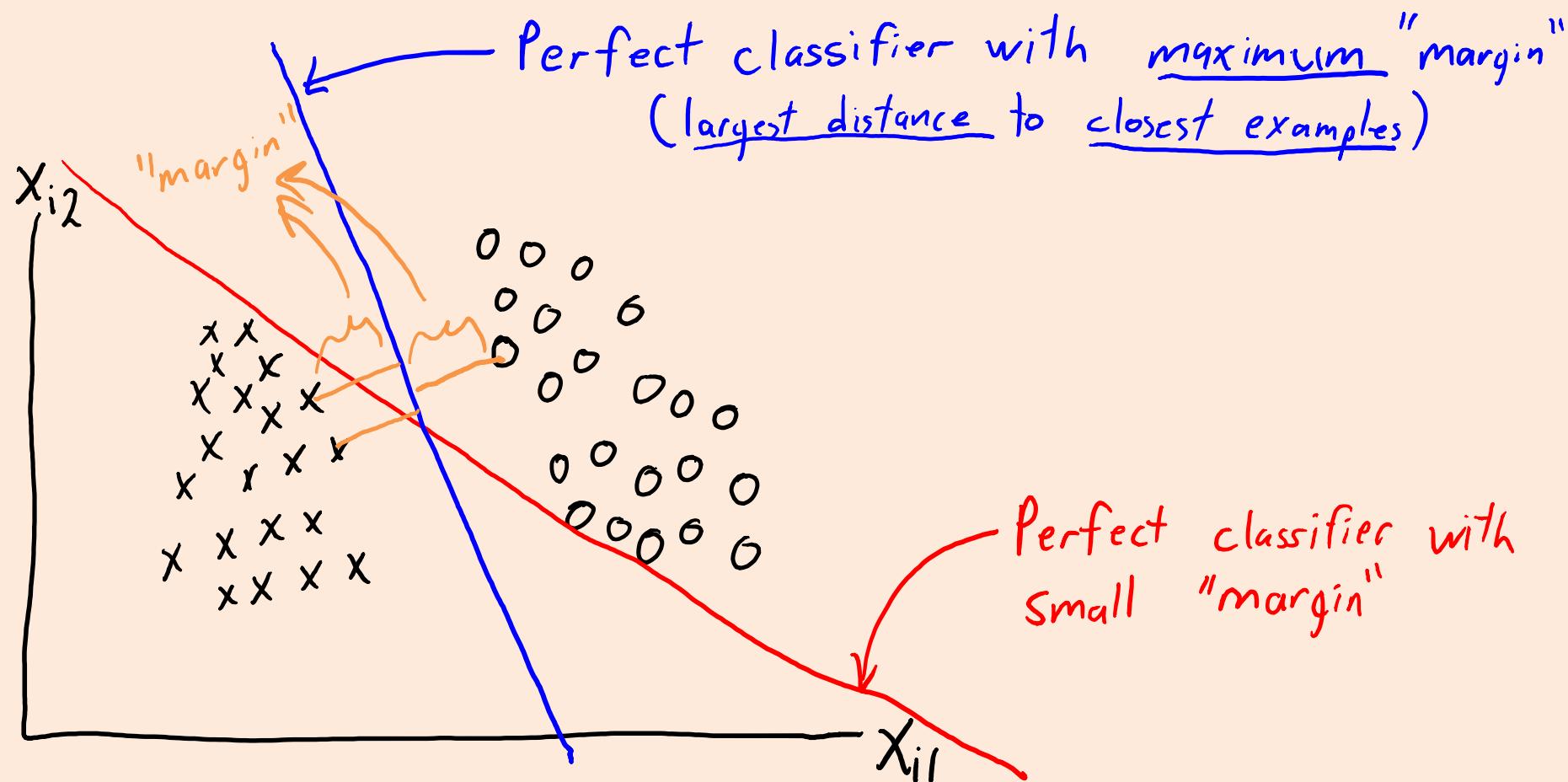


Maximum-Margin Perspective

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

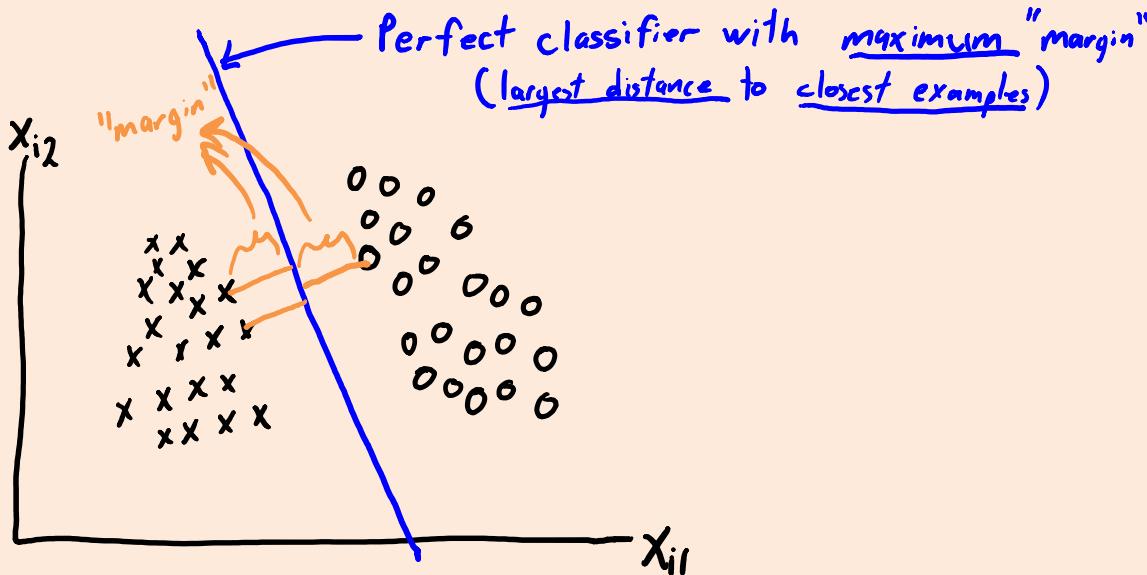
Why maximize margin?

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).

(pause)

Previously: Identifying Important E-mails

- Recall problem of identifying ‘important’ e-mails:



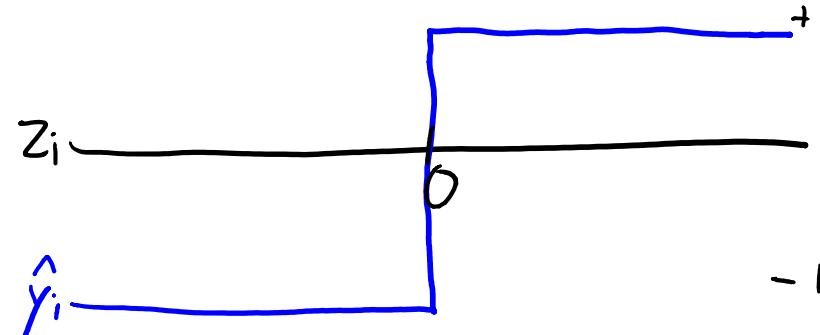
- We can do binary classification by taking **sign of linear model**:

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

- Convex loss functions (hinge/logistic loss) let us find an appropriate ‘w’.
- But what if we want a **probabilistic classifier**?
 - Want a model of $p(y_i = \text{"important"} | x_i)$ for use in decision theory.

Predictions vs. Probabilities

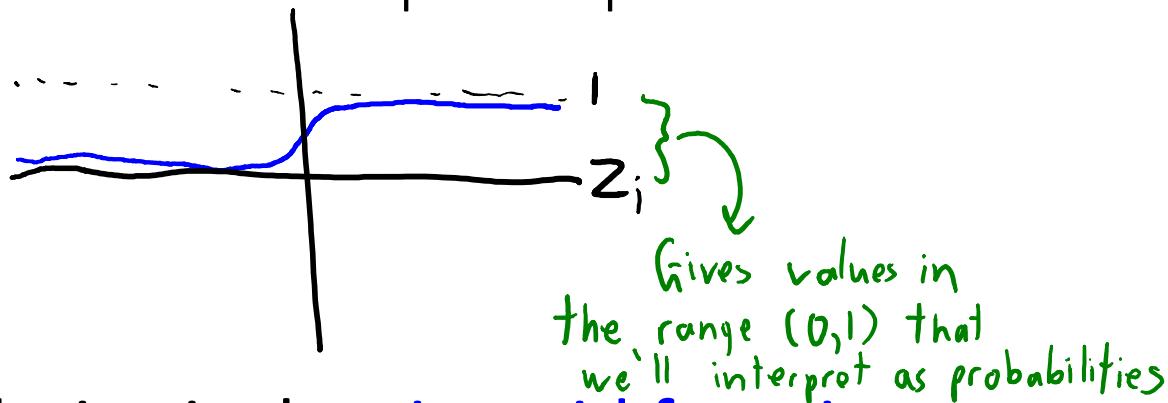
- With $z_i = w^T x_i$, linear classifiers make prediction using $\text{sign}(z_i)$:



- For predictions, “sign” maps from $w^T x_i$ to the elements $\{-1, +1\}$.
 - If $w^T x_i$ is positive we predict $+1$, if it’s negative we predict -1 .
- For probabilities, we want to map from $w^T x_i$ to the range $[0, 1]$.
 - If $w^T x_i$ is very positive, we output a value close to $+1$.
 - If $w^T x_i$ is very negative, we output a value close to 0 .
 - If $w^T x_i$ is close to 0 , we output a value close to 0.5 .

Sigmoid Function

- So we want a transformation of $z_i = w^T x_i$ that looks like this:



- The most common choice is the **sigmoid function**:

$$h(z_i) = \frac{1}{1 + \exp(-z_i)}$$

- Values of $h(z_i)$ match what we want:

$$h(-\infty) = 0 \quad h(-1) \approx 0.27 \quad h(0) = 0.5 \quad h(0.5) \approx 0.62 \quad h(+1) \approx 0.73 \quad h(+\infty) = 1$$

Probabilities for Linear Classifiers using Sigmoid

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

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

- By rules of probability:

$$\begin{aligned} p(y_i = -1 \mid w, x_i) &= 1 - p(y_i = 1 \mid w, x_i) \\ &= \frac{1}{1 + \exp(w^T x_i)} \quad (\text{with some effort}) \end{aligned}$$

- We then use these for “**probability that e-mail is important**”.
- This may seem heuristic, but later we’ll see that:
 - minimizing logistic loss does “maximum likelihood estimation” in this model.

(pause)

Multi-Class Linear Classification

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

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

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



Multi-Class Linear Classification

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

$$X = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \quad y = \begin{bmatrix} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{bmatrix}$$

- 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.

“One vs All” Classification

- Suppose you **only know how to do binary classification**:
 - “One vs all” is a way to turn a **binary classifier** into a **multi-class** method.
- **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** .
- **Prediction phase**:
 - **Apply the ‘k’ binary classifiers** to get a “score” for each class ‘c’.
 - Return the ‘c’ with the highest score.
 - (Draw picture.)

“One vs All” Linear Classification

- “One vs all” logistic regression for classifying as cat/dog/person.
 - Train a separate classifier for each class.
 - Classifier 1 tries to predict +1 for “cat” images and -1 for “dog” and “person” images.
 - Classifier 2 tries to predict +1 for “dog” images and -1 for “cat” and “person” images.
 - Classifier 3 tries to predict +1 for “person” images and -1 for “cat” and “dog” images.
 - This gives us a weight vector w_c for each class ‘c’:
 - Weights w_c try to predict +1 for class ‘c’ and -1 for all others.
 - We’ll use ‘W’ as a matrix with the w_c as rows:

$$W = \left[\begin{array}{c} w_1^T \\ w_2^T \\ \vdots \\ w_K^T \end{array} \right]^T \quad d \quad K$$

Each row ‘c’ gives weights w_c for a binary logistic regression model to predict class ‘c’.

“One vs All” Linear Classification

- “One vs all” logistic regression for classifying as cat/dog/person.
 - Prediction on example x_i given parameters ‘W’ :

$$W = \left[\begin{array}{c} w_1^T \\ w_2^T \\ \vdots \\ w_K^T \end{array} \right] \underbrace{}_{d} \Big\} K$$

- For each class ‘c’, compute $w_c^T x_i$.



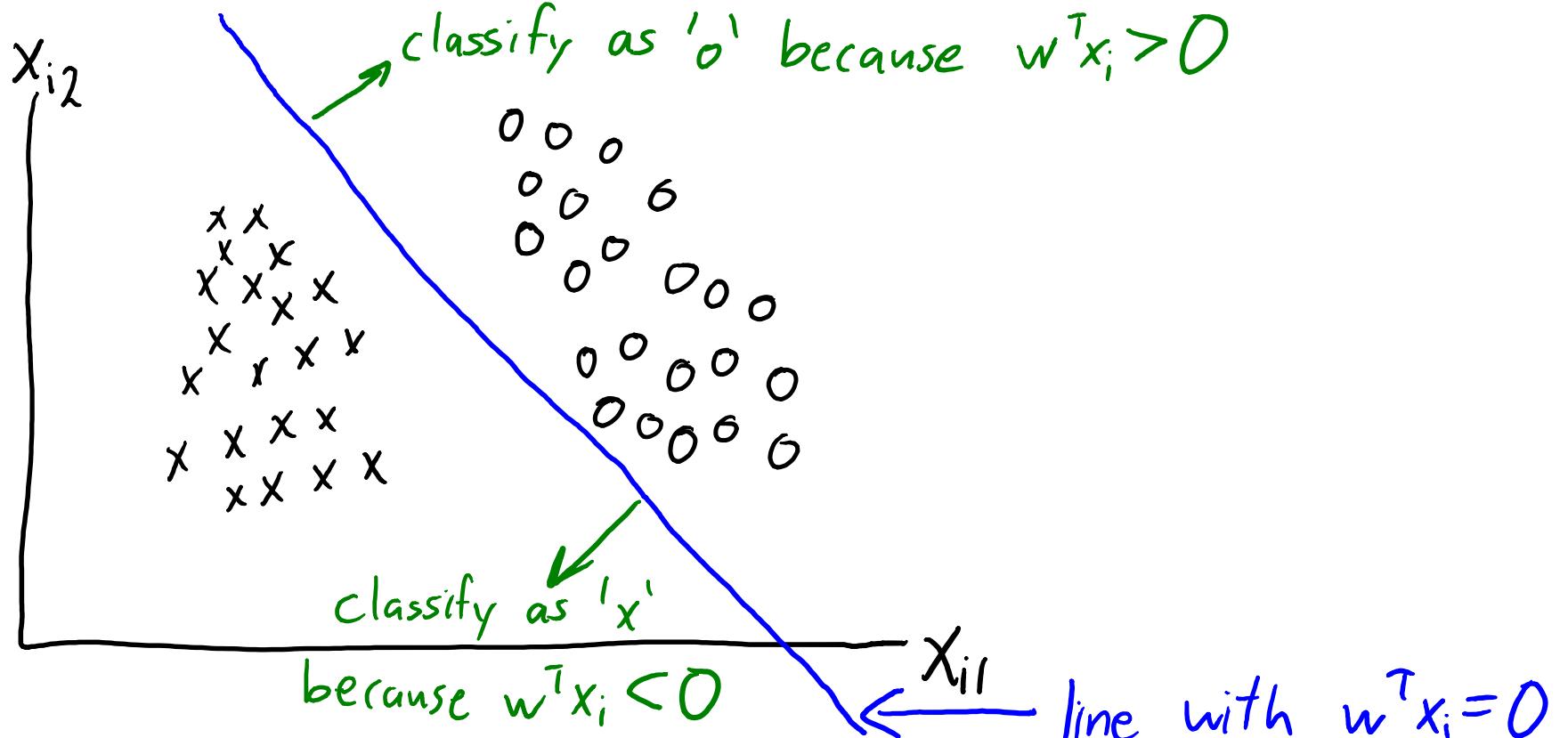
features x_i

$$\begin{aligned} w_1^T x_i &= -0.1 && ("cat" \text{ score}) \\ w_2^T x_i &= -0.8 && ("dog" \text{ score}) \\ w_3^T x_i &= 0.9 && ("human" \text{ score}) \end{aligned}$$

- Ideally, we'll get $\text{sign}(w_c^T x_i) = +1$ for one class and $\text{sign}(w_c^T x_i) = -1$ for all others.
- In practice, it **might be +1 for multiple classes or no class**.
- To predict class, we take **maximum value of $w_c^T x_i$ (“highest score”)**.
 - In the example above, predict “human” (0.9 is higher than -0.8 and -0.1).

Shape of Decision Boundaries

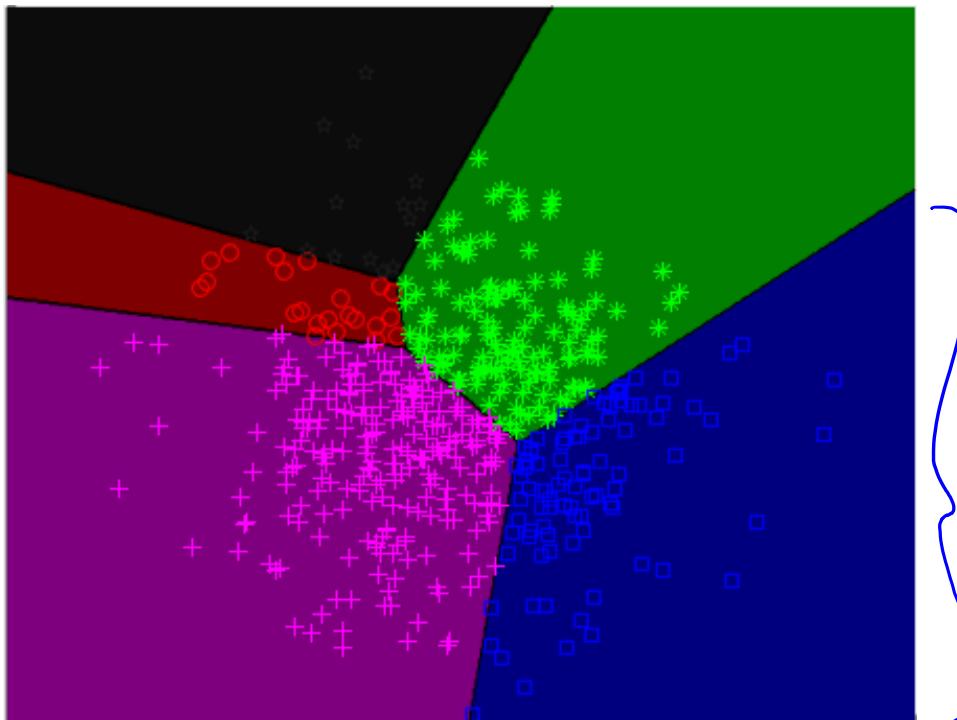
- Recall that a **binary linear classifier** splits space using a hyper-plane:



- 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:

$$\mathbf{w}_{\text{blue}}^T \mathbf{x}_i \geq \mathbf{w}_{\text{green}}^T \mathbf{x}_i$$

$$\mathbf{w}_{\text{blue}}^T \mathbf{x}_i \geq \mathbf{w}_{\text{magenta}}^T \mathbf{x}_i$$

$$\mathbf{w}_{\text{blue}}^T \mathbf{x}_i \geq \mathbf{w}_{\text{red}}^T \mathbf{x}_i$$

$$\mathbf{w}_{\text{blue}}^T \mathbf{x}_i \geq \mathbf{w}_{\text{black}}^T \mathbf{x}_i$$

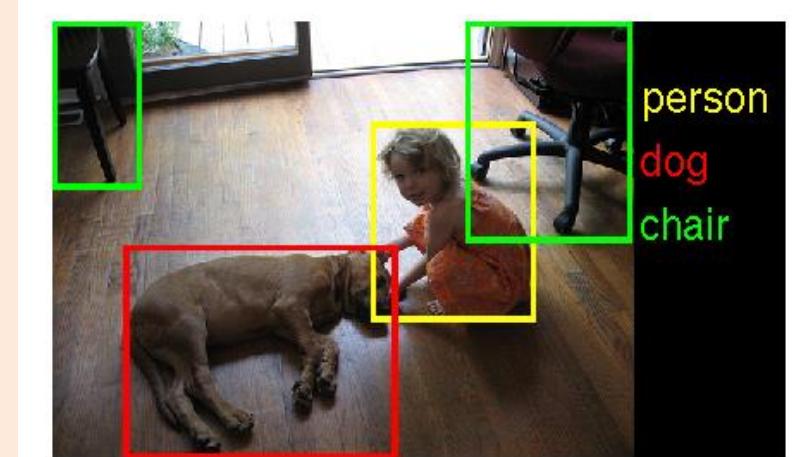
- Could be non-convex with change of basis.

Digression: Multi-Label Classification

- A related problem is **multi-label classification**:

$$X = \left[\begin{array}{c} \text{Object 1} \\ \text{Object 2} \\ \vdots \\ \text{Object n} \end{array} \right] \quad Y = \left[\begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{array} \right]_n^k$$

- 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.



“One vs All” Multi-Class Classification (MEMORIZE)

- Back to **multi-class classification** where we have 1 “correct” label:

$$X = \left[\begin{array}{c} \\ \\ \\ \\ \end{array} \right] \quad y = \left[\begin{array}{c} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{array} \right] \quad \left\{ \begin{array}{l} \text{train } 'k' \\ \text{classifiers} \end{array} \right.$$
$$W = \left[\begin{array}{c} w_1^T \\ w_2^T \\ \vdots \\ w_k^T \end{array} \right]$$

Predict by maximizing $w_c^T x_i$

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

“One vs All” Multi-Class 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

$$\begin{aligned} w_1^T x_i &= -5 && ("cat" \text{ score}) \\ w_2^T x_i &= -0.1 && ("dog" \text{ score}) \\ w_3^T x_i &= -0.2 && ("human" \text{ score}) \end{aligned}$$

- 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.

Multi-Class SVMs

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?
 - So when we maximizing over $w_c^T x_i$, we choose correct label y_i .
- Recall our derivation of the hinge loss (SVMs):
 - We wanted $y_i w^T x_i > 0$ for all ‘i’ to classify correctly.
 - We avoided non-degeneracy by aiming for $y_i w^T x_i \geq 1$.
 - We used the constraint violation as our loss: $\max\{0, 1 - y_i w^T x_i\}$.
- We can derive multi-class SVMs using the same steps...

Multi-Class SVMs

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?
We want $w_{y_i}^T x_i > w_c^T x_i$ for all 'c' that are not correct label y_i

If we penalize violation of this constraint it's degenerate.
We use $w_{y_i}^T x_i \geq w_c^T x_i + 1$ for all $c \neq y_i$ to avoid strict inequality
Equivalently: $0 \geq 1 - w_{y_i}^T x_i + w_c^T x_i$
- For here, there are two ways to measure constraint violation:

"Sum"

$$\sum_{c \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}$$

"Max"

$$\max_{c \neq y_i} \left\{ \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\} \right\}$$

Multi-Class SVMs

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?

"Sum"

$$\sum_{c \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}$$

"Max"

$$\max_{c \neq y_i} \left\{ \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\} \right\}$$

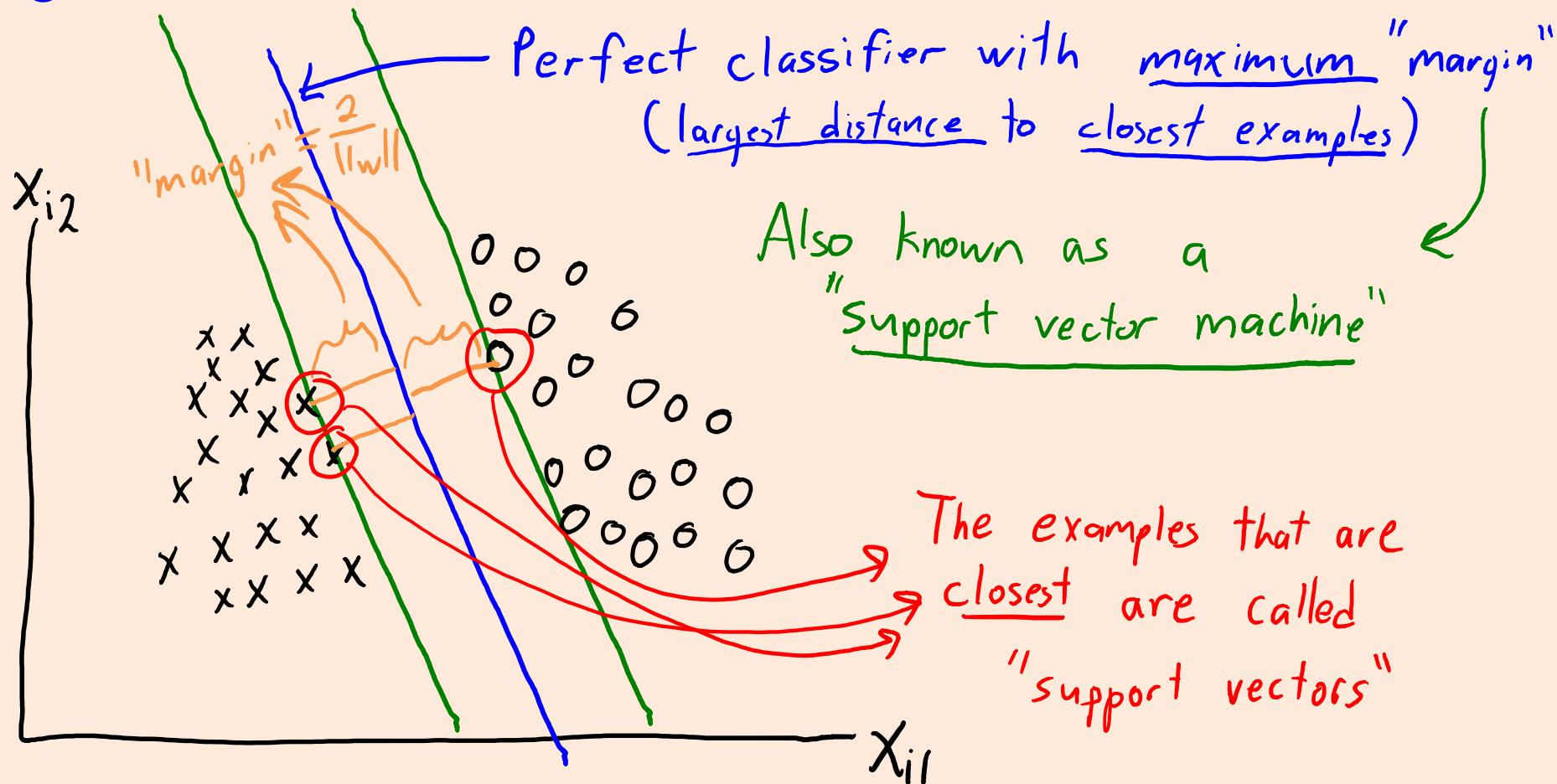
- For each training example ‘i’:
 - “Sum” rule penalizes for each ‘c’ that violates the constraint.
 - “Max” rule penalizes for one ‘c’ that violates the constraint the most.
 - “Sum” gives a penalty of ‘k-1’ for W=0, “max” gives a penalty of ‘1’.
- If we add L2-regularization, both are called **multi-class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

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.
- Next time: what makes good features?

Maximum-Margin Classifier

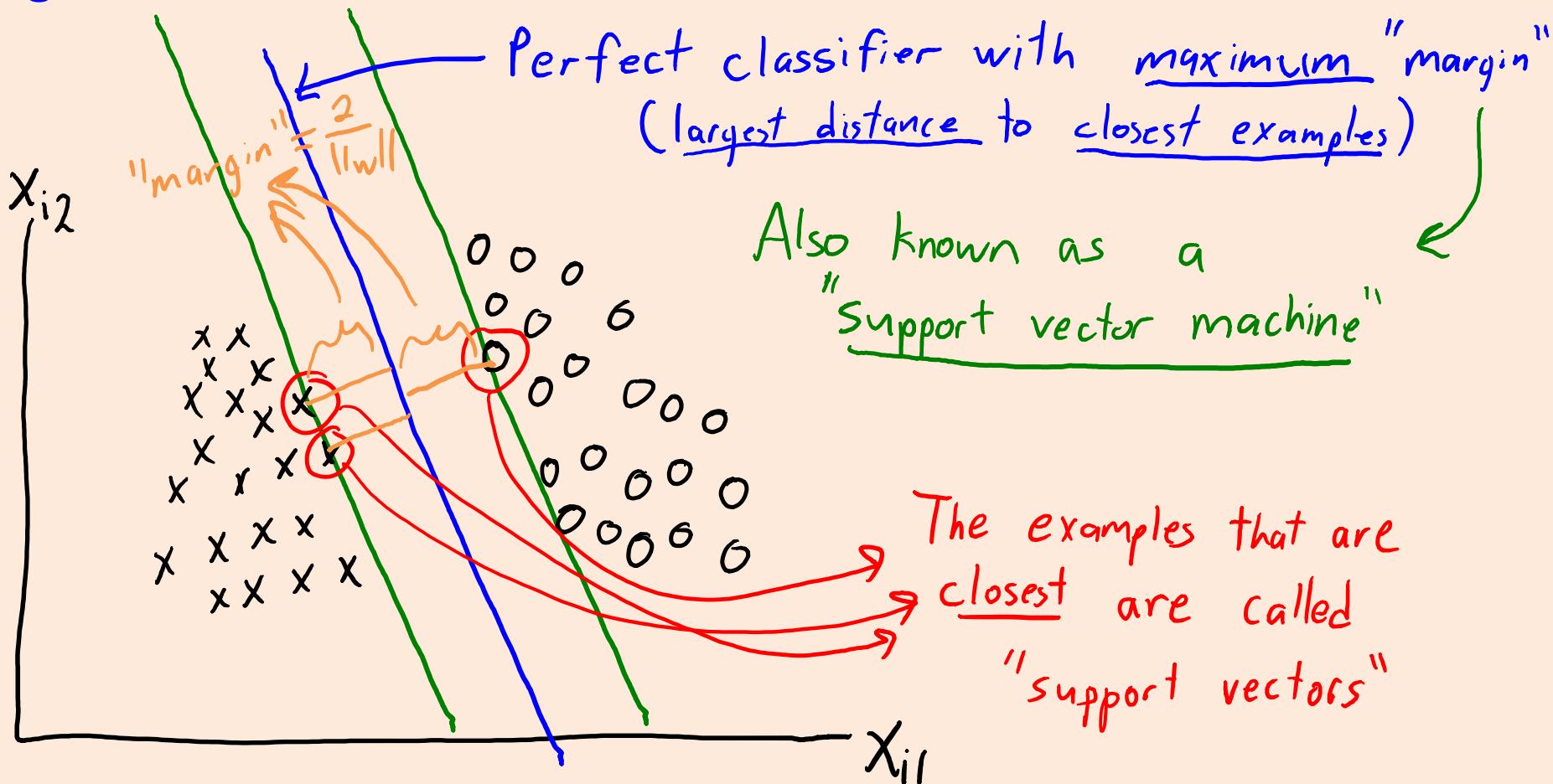
- Consider a linearly-separable dataset.
 - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Classifier

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

Final classifier only
depends on support
vectors

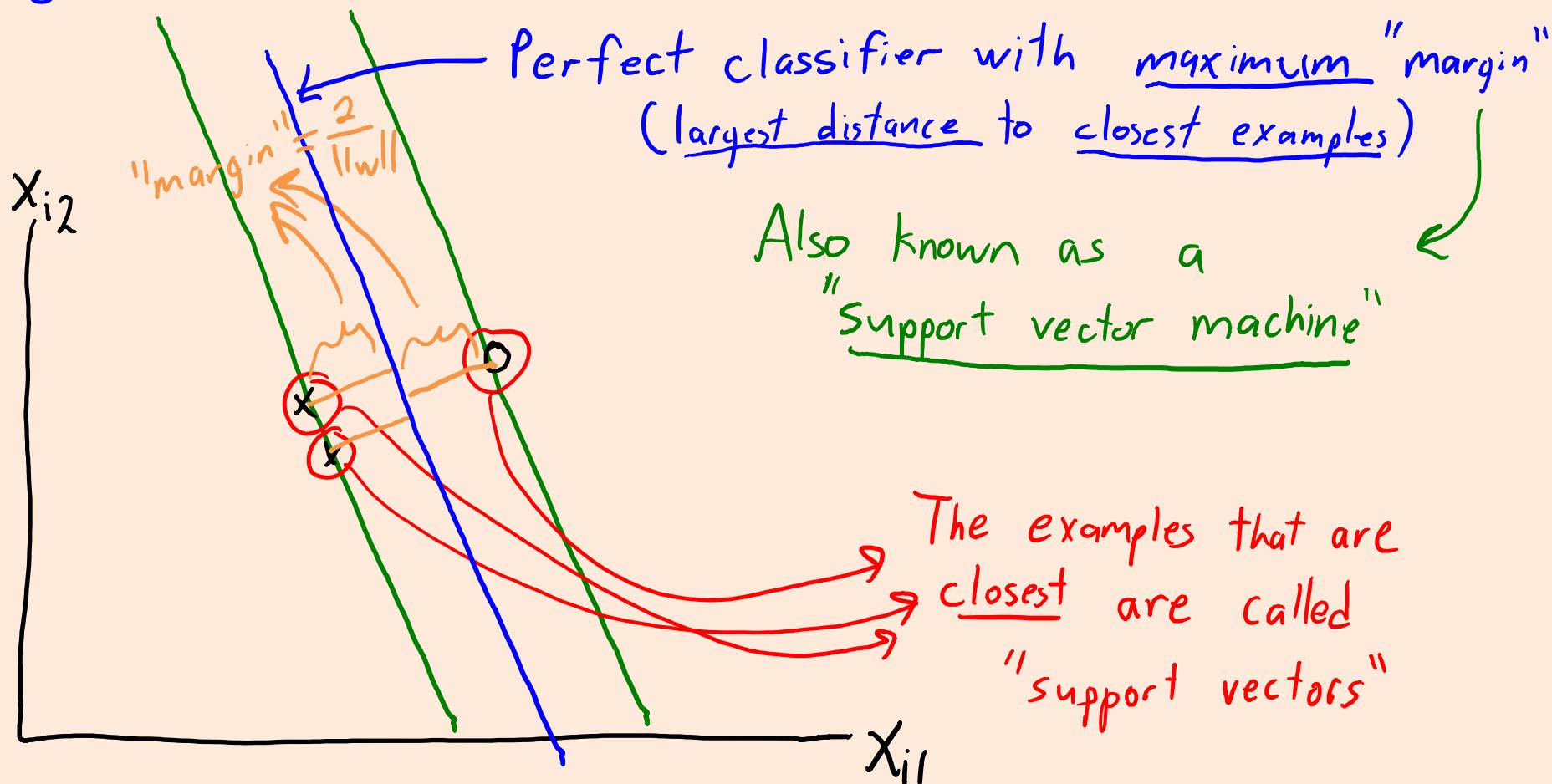


Maximum-Margin Classifier

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

Final classifier only
depends on support
vectors

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



Support Vector Machines

- For linearly-separable data, SVM minimizes:

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

- Subject to the constraints that:
(see Wikipedia/textbooks)
- $$\begin{aligned} w^T x_i &\geq 1 & \text{for } y_i = 1 & (\text{classify all examples correctly}) \\ w^T x_i &\leq -1 & \text{for } y_i = -1 & \end{aligned}$$

- 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 ':

it's the amount we violate $y_i w^T x_i \geq 1$
"slack"

Original SVM objective:
encourages large
margin.

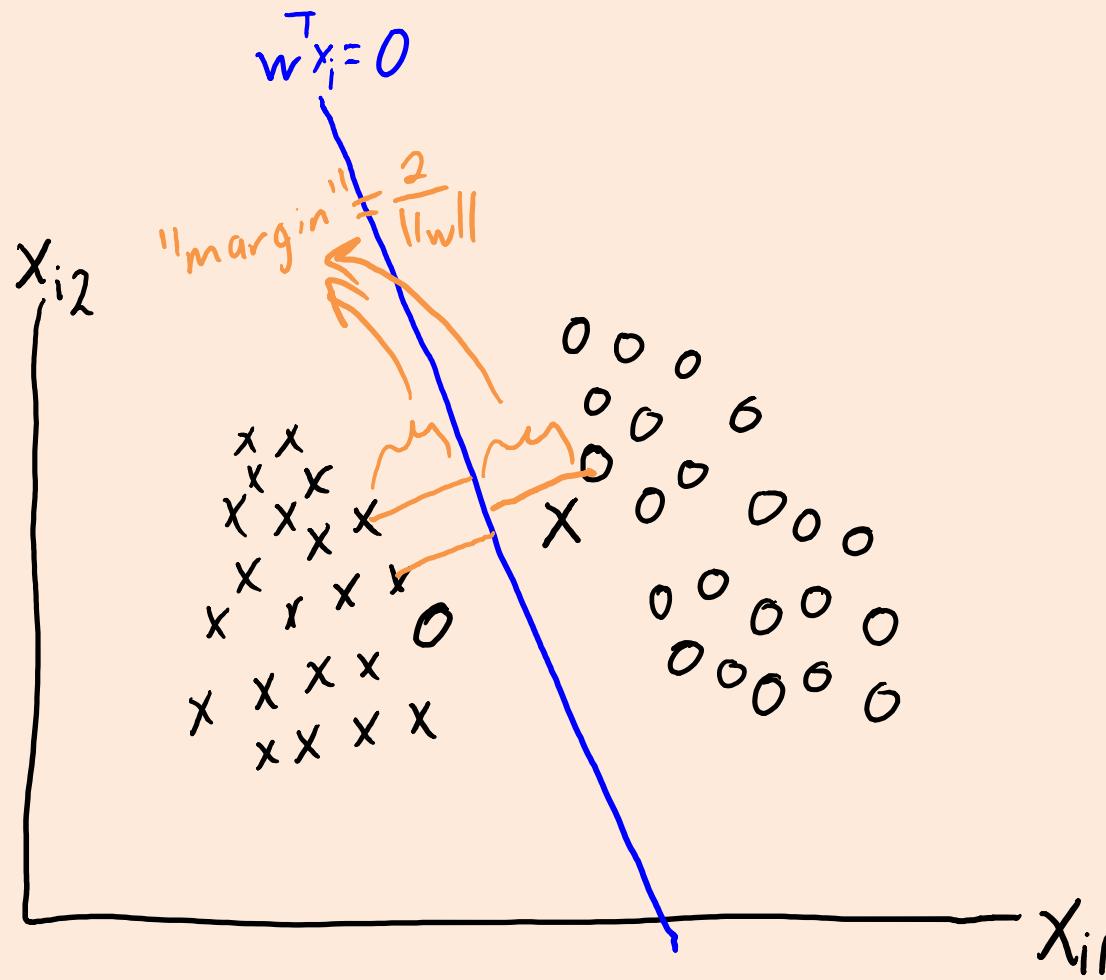
- We typically control margin/violation trade-off with parameter " λ ":

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

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

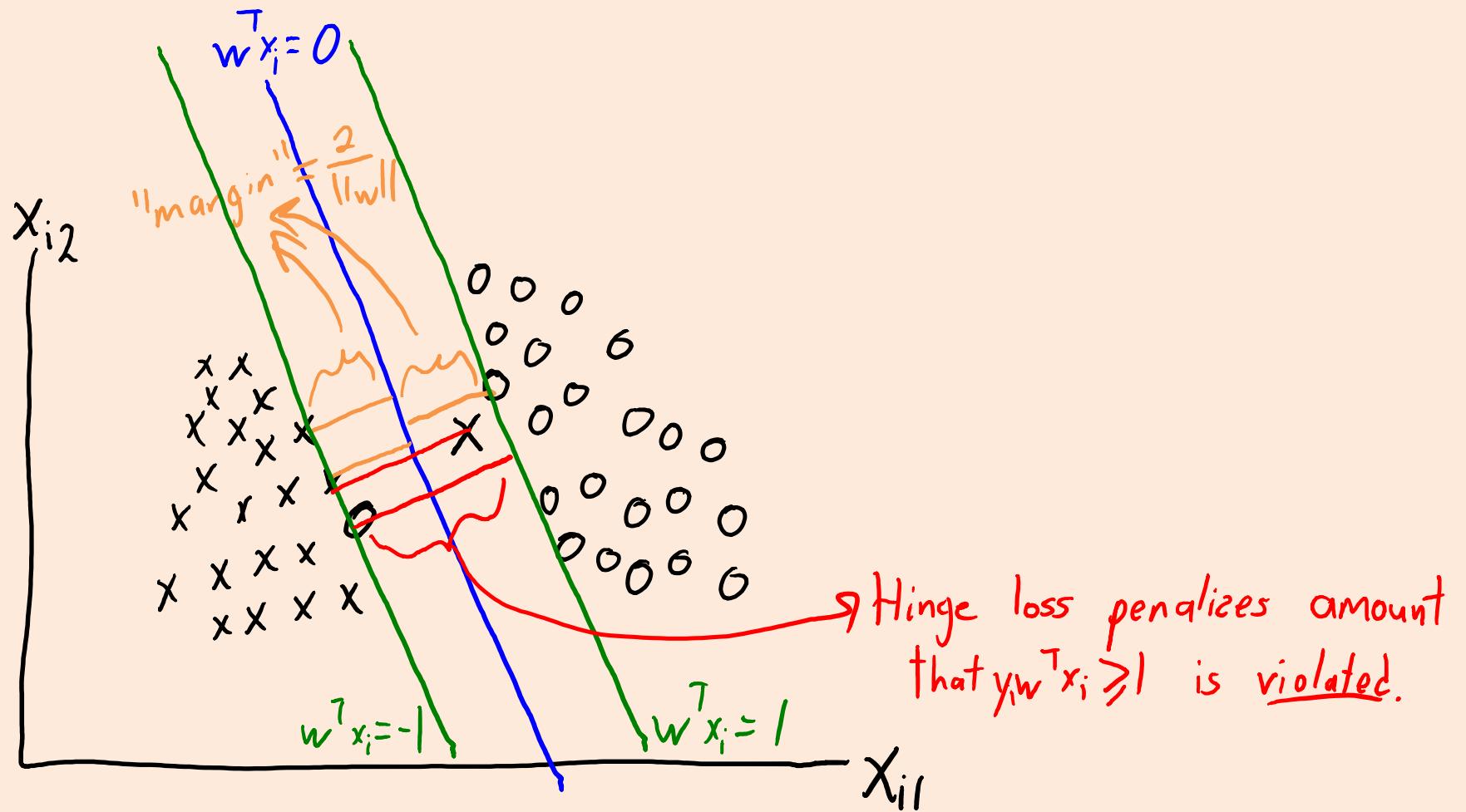
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:

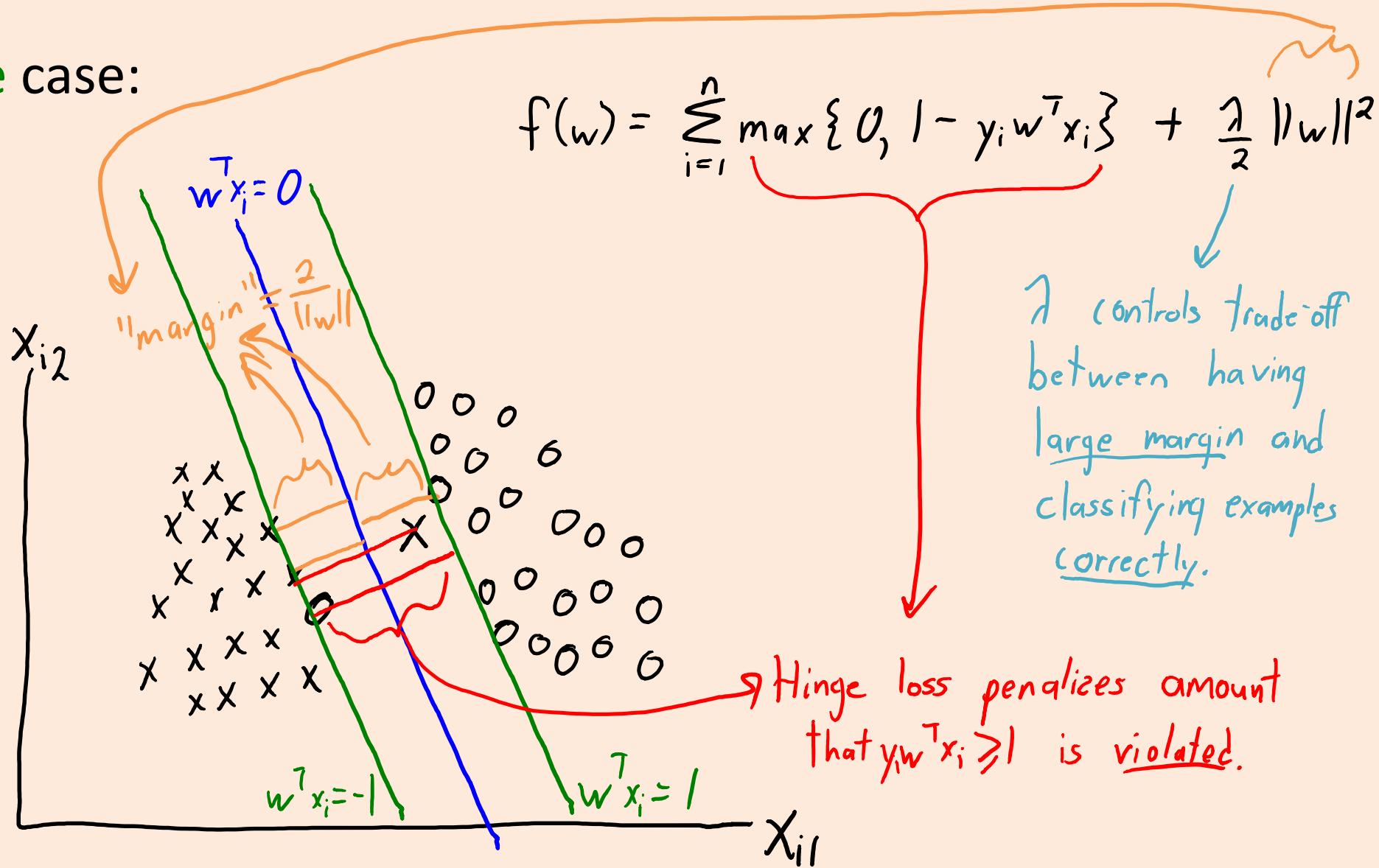


Support Vector Machines for Non-Separable

- Non-separable case:

Logistic regression can be viewed as smooth approximation to SVMs.

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

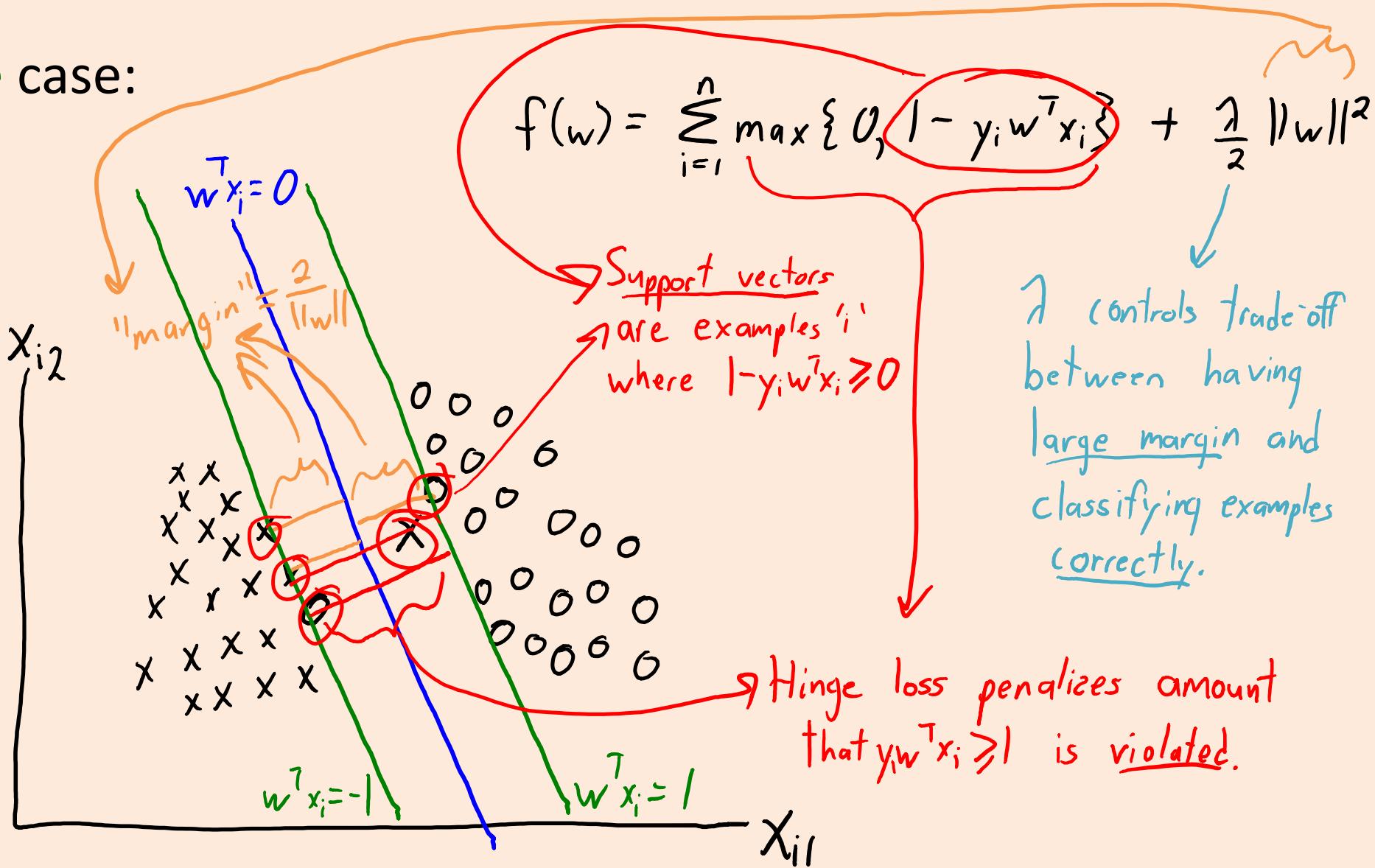


Support Vector Machines for Non-Separable

- Non-separable case:

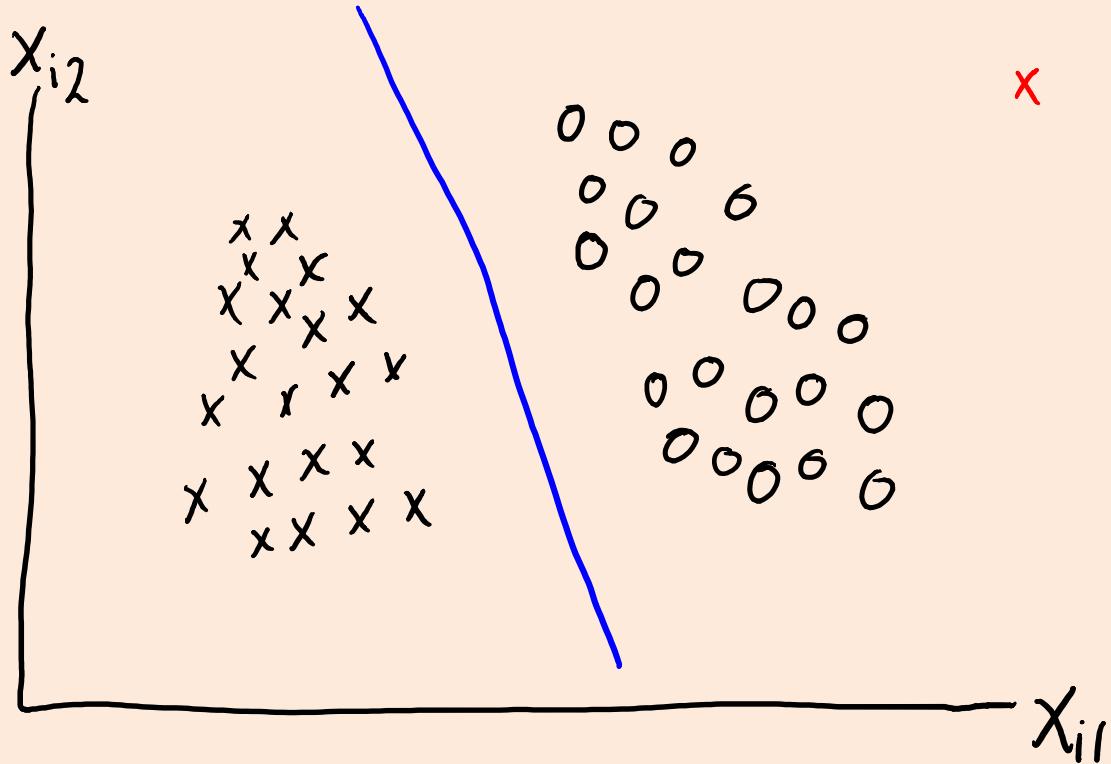
Logistic regression can be viewed as smooth approximation to SVMs.

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



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

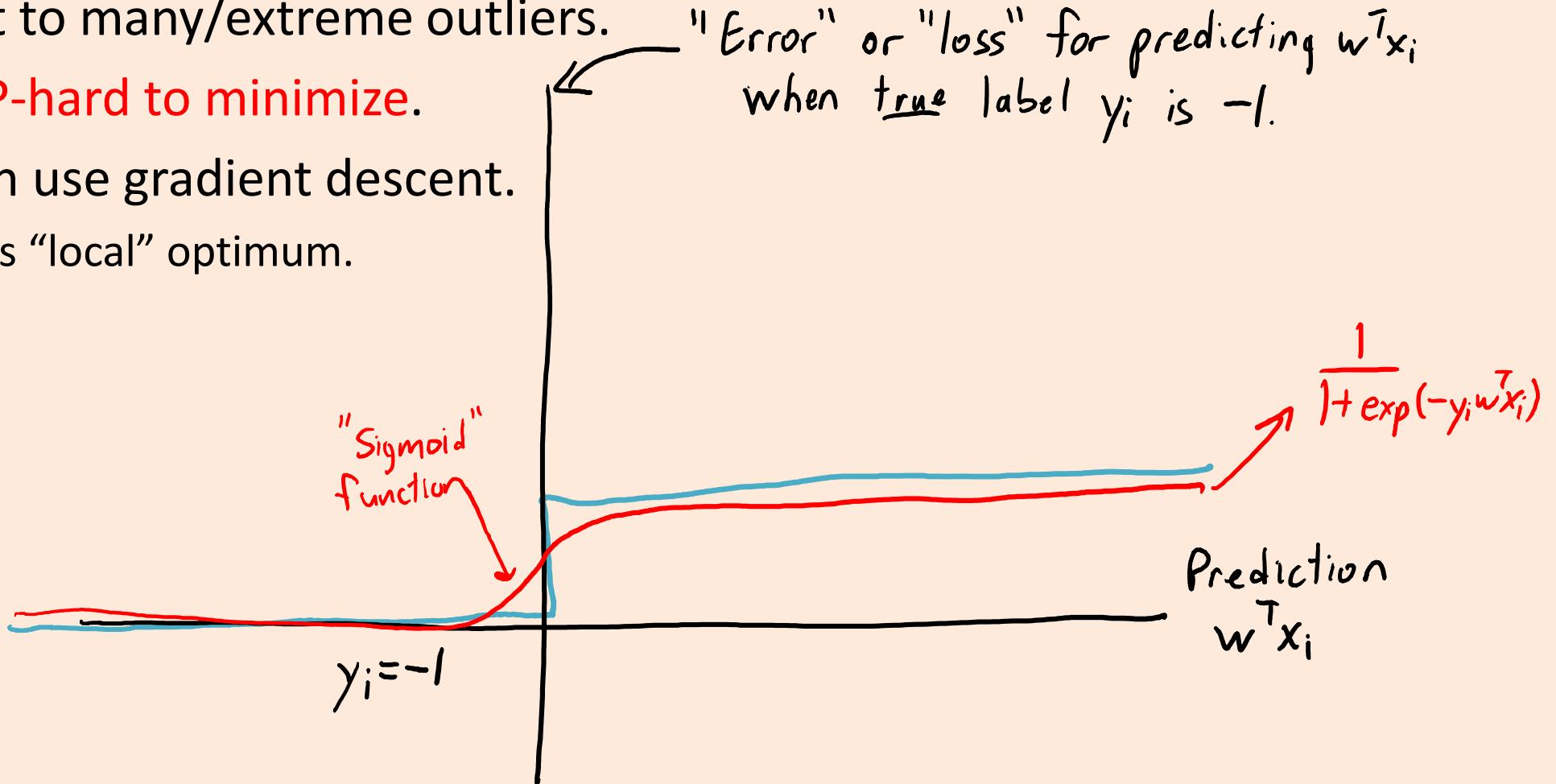
- Because the hinge/logistic grow like absolute value for mistakes, they tend **not to be affected by a small number of outliers**.



- 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.

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

- 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 $\text{sign}(w^T x_i)$ is very wrong).

“Robust” Logistic Regression

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

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

- 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:

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

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

“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.