# 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 $\text{sign}(w^T x_i)$

  $y_i \rightarrow +1, -1$

- 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 λ = 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:

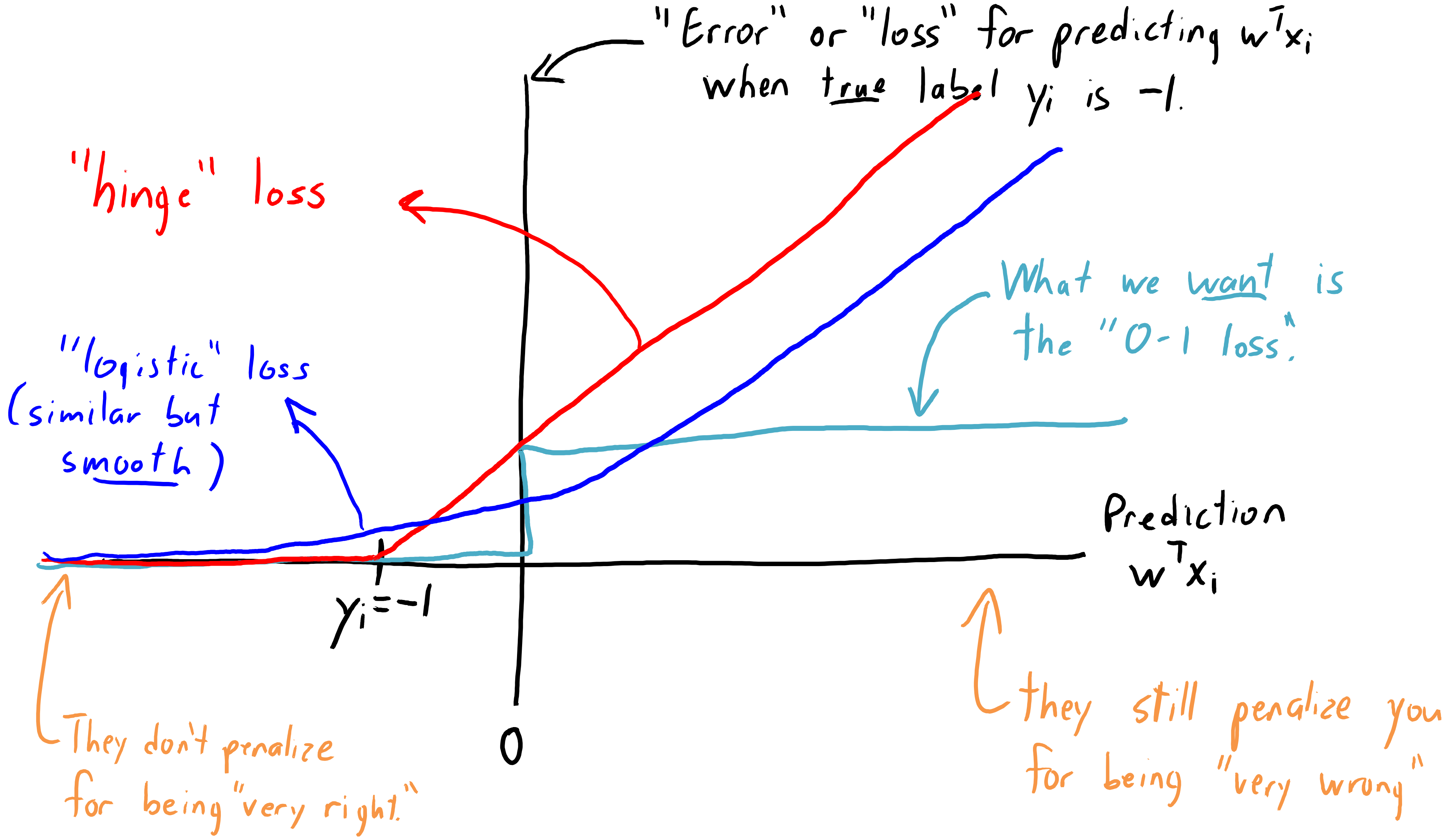
$$\max\{0, -y_i w^T x_i\} \approx \log(\underbrace{\exp(0)}_{1} + \exp(-y_i w^T x_i))$$

- 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: 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 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?"

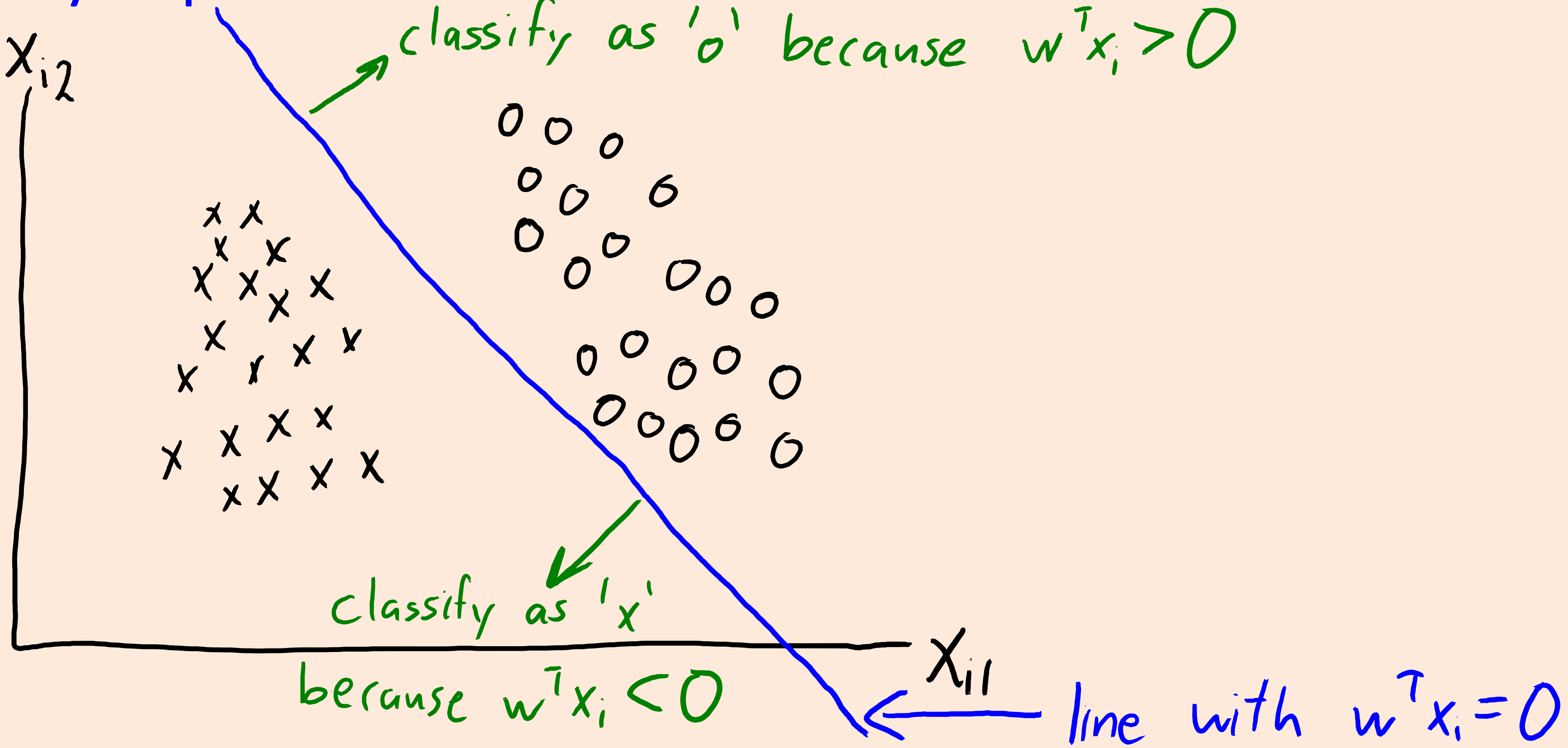
- Compared 179 classifiers on 121 datasets.
- Random forests are most likely to be the best classifier.
- Next best class of methods was SVMs (L2-regularization, RBFs).

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

(pause)

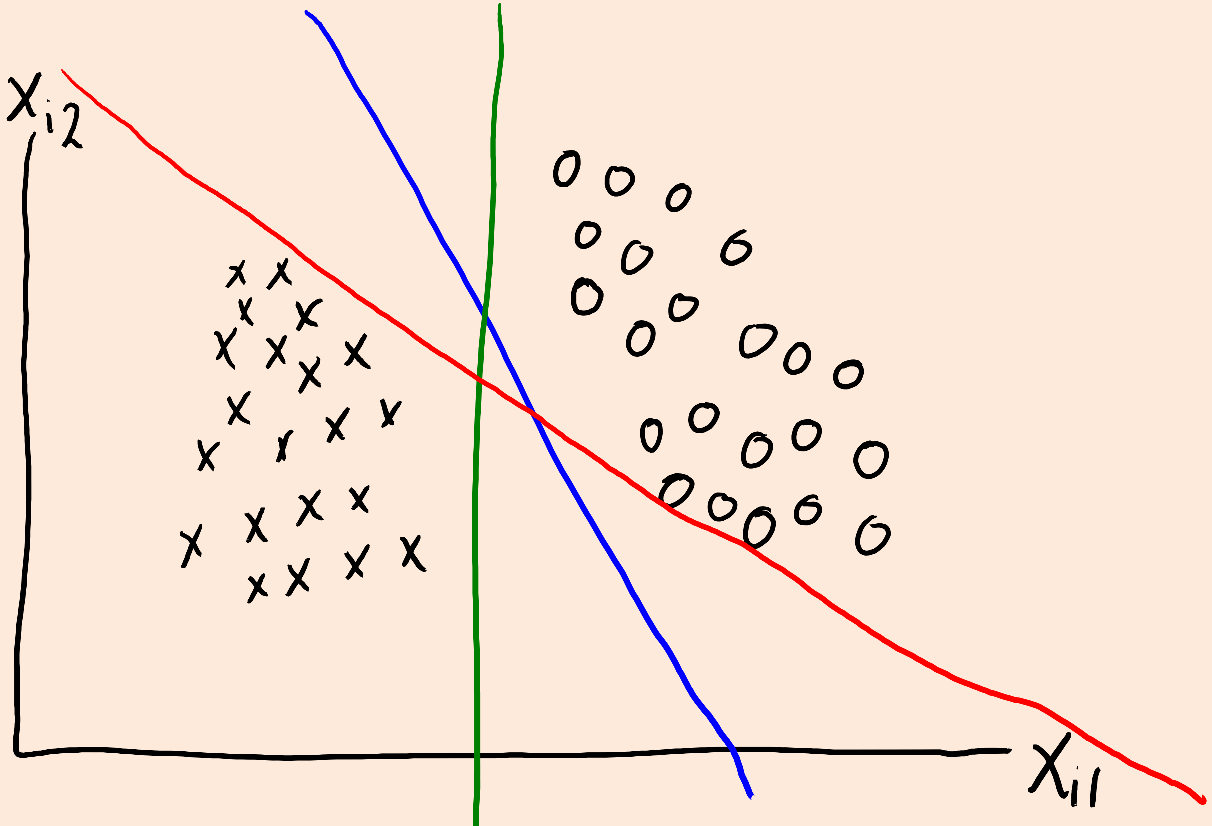
# Maximum-Margin Perspective

- Consider a linearly-separable dataset.



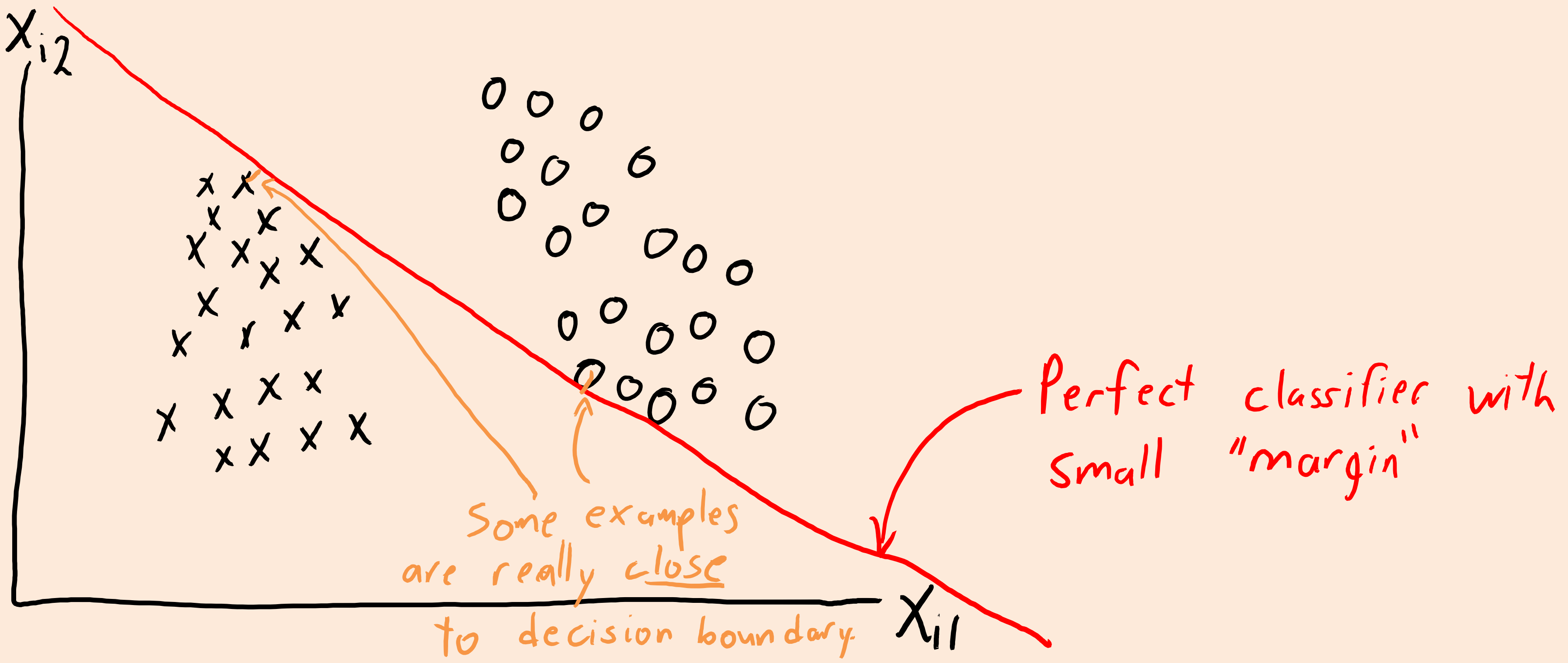
# Maximum-Margin Perspective

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



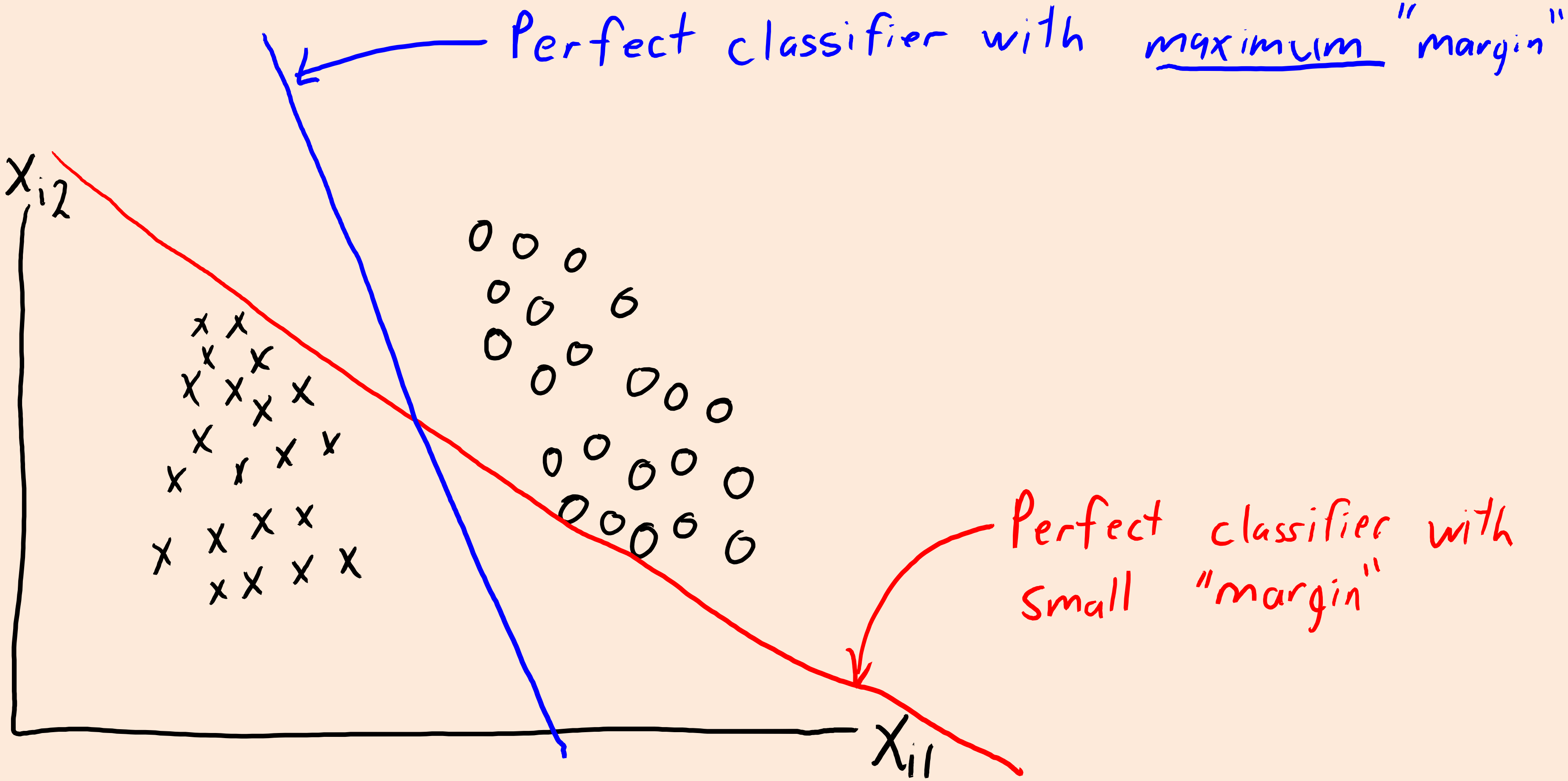
# Maximum-Margin Perspective

- 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.

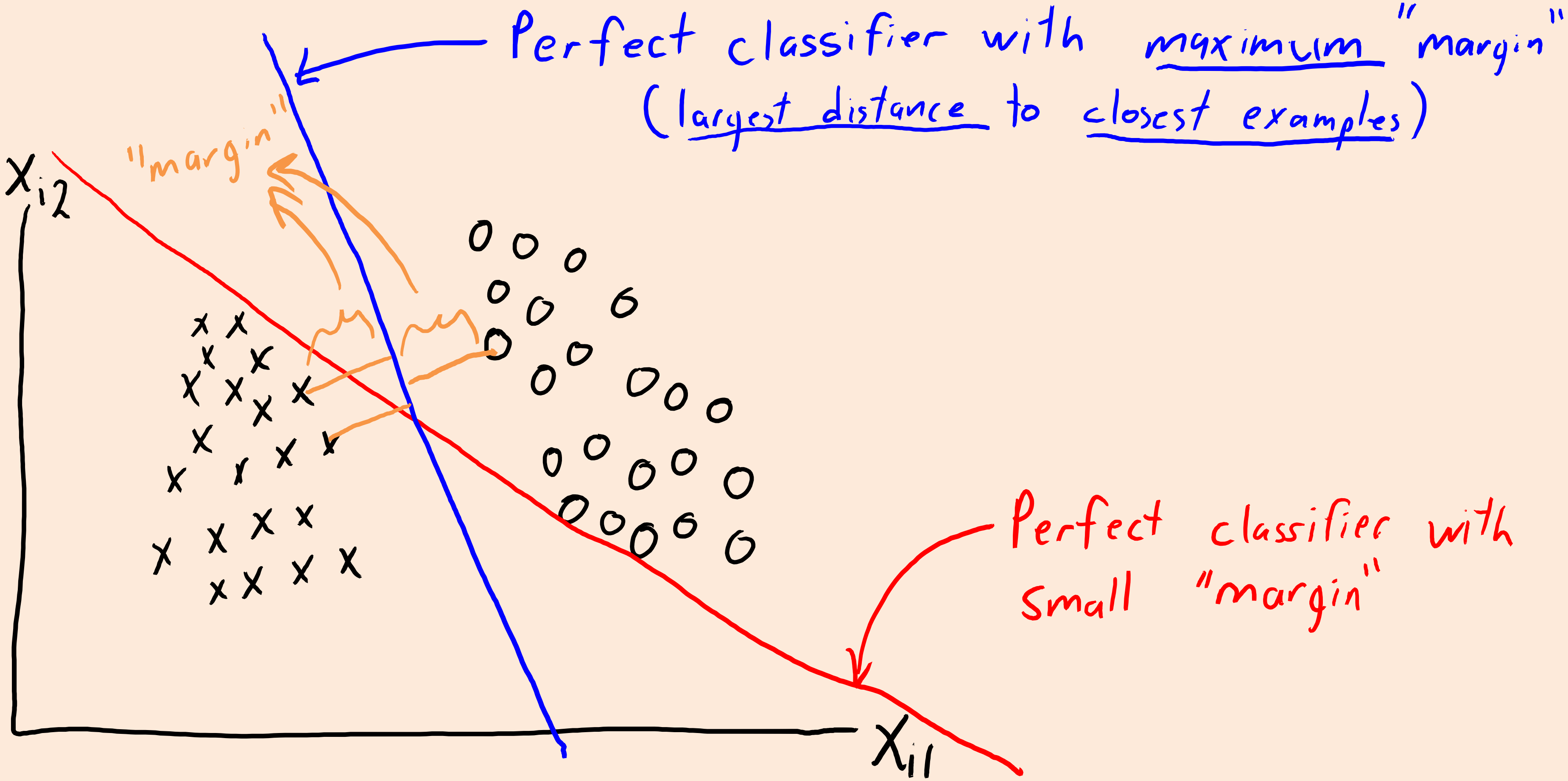


# Maximum-Margin Perspective

- 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

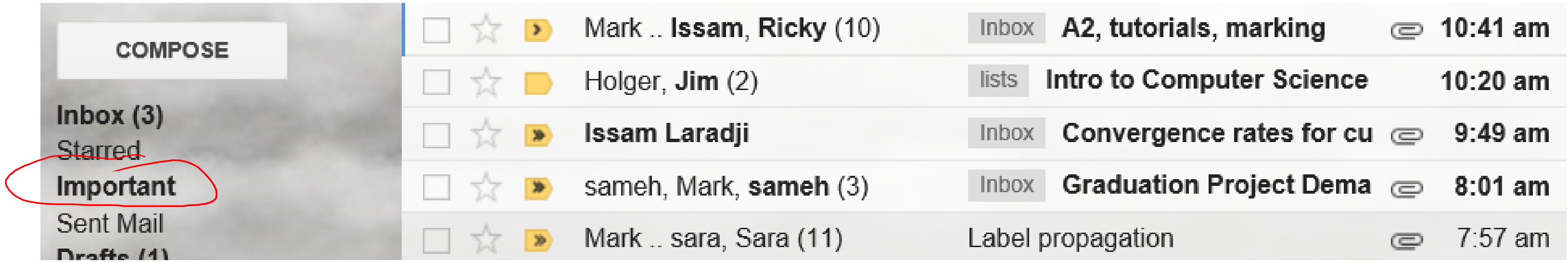
- For linearly-separable data:


- With small-enough λ > 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 λ=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:

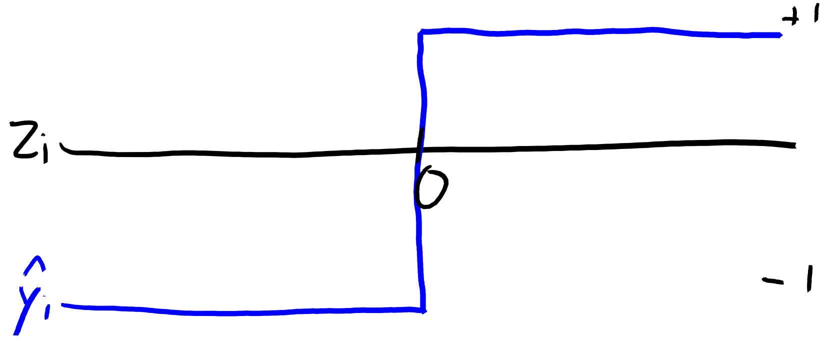
$$\hat{y}_i = \text{sign}(w^T x_i)$$

  - 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.

# Predictions vs. Probabilities

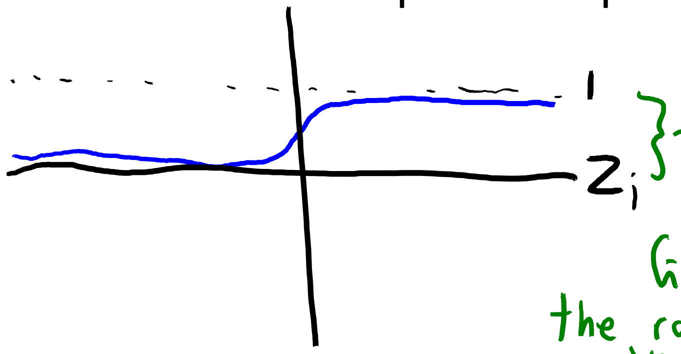
- With $z_i = w^T x_i$, linear classifiers make prediction using $\text{sign}(z_i)$:




- 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:




Gives values in the range (0,1) that we'll interpret as probabilities

- 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:

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

# 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:

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

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

- 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:

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

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

# Multi-Class Linear Classification

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

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

- 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: 

features $x_i$

"cat detector" → 2 "cat score"

"dog detector" → 3 "dog score"

"human detector" → 10 "human score"

predict "human"

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

# “One vs All” <u>Linear</u> 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:

$$W = \left.\underbrace{\begin{bmatrix} \text{——} \; w_1^T \; \text{——} \\ \text{——} \; w_2^T \; \text{——} \\ \vdots \\ \text{——} \; w_K^T \; \text{——} \end{bmatrix}}_{d}\right\} k$$

Each row ‘c’ gives weights $w_c$ for a binary logistic regression model to <u>predict class ‘c’</u>.

# "One vs All" Linear Classification

- "One vs all" logistic regression for classifying as cat/dog/person.
  - Prediction on example $x_i$ given parameters 'W' :

$$W = \left.\underbrace{\begin{bmatrix} — & w_1^T & — \\ — & w_2^T & — \\ & \vdots & \\ — & w_k^T & — \end{bmatrix}}_{d}\right\} k$$

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

features $x_i$

$w_1^T x_i = -0.1$ ("cat" score)

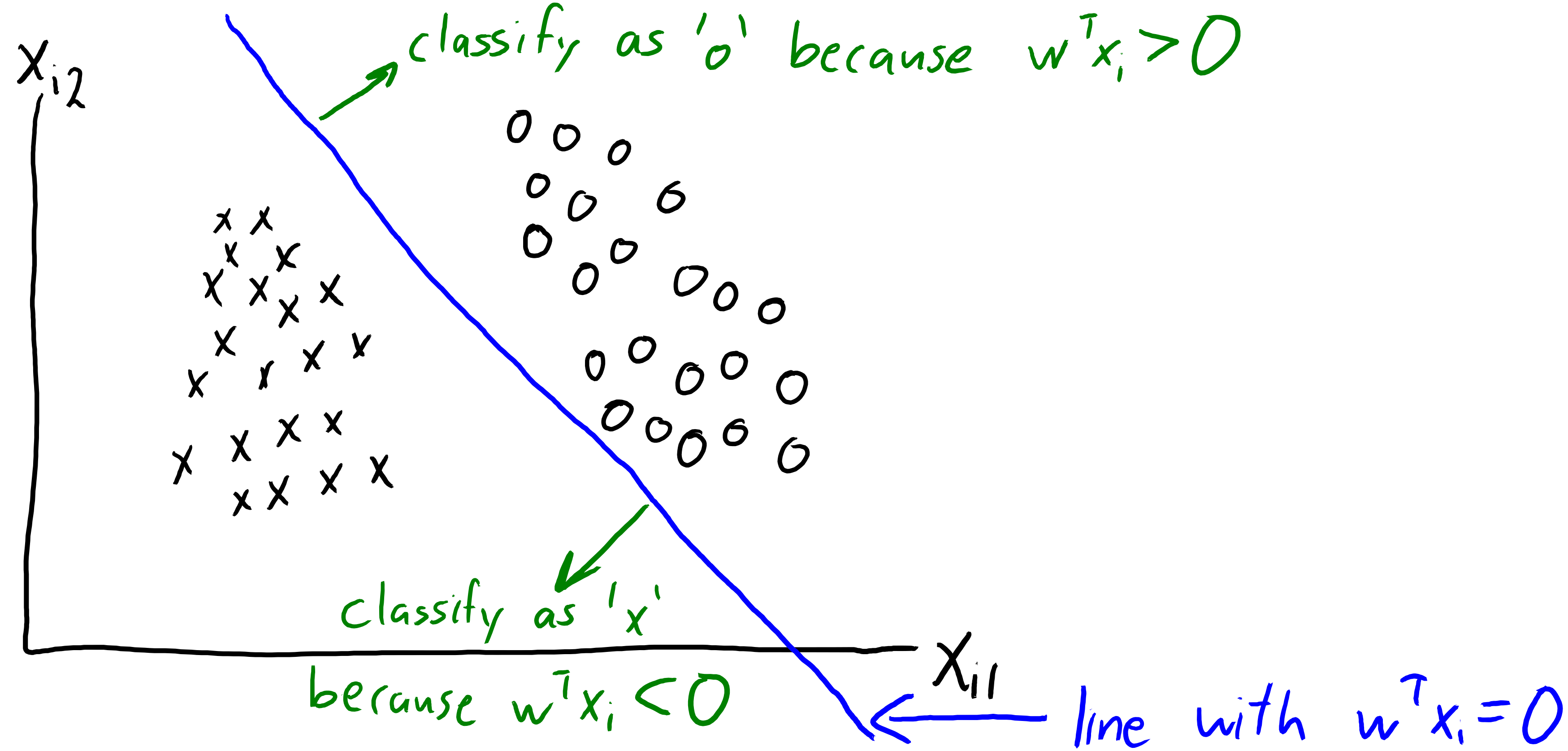
$w_2^T x_i = -0.8$ ("dog" score)

$w_3^T x_i = 0.9$ ("human" score)

    - 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

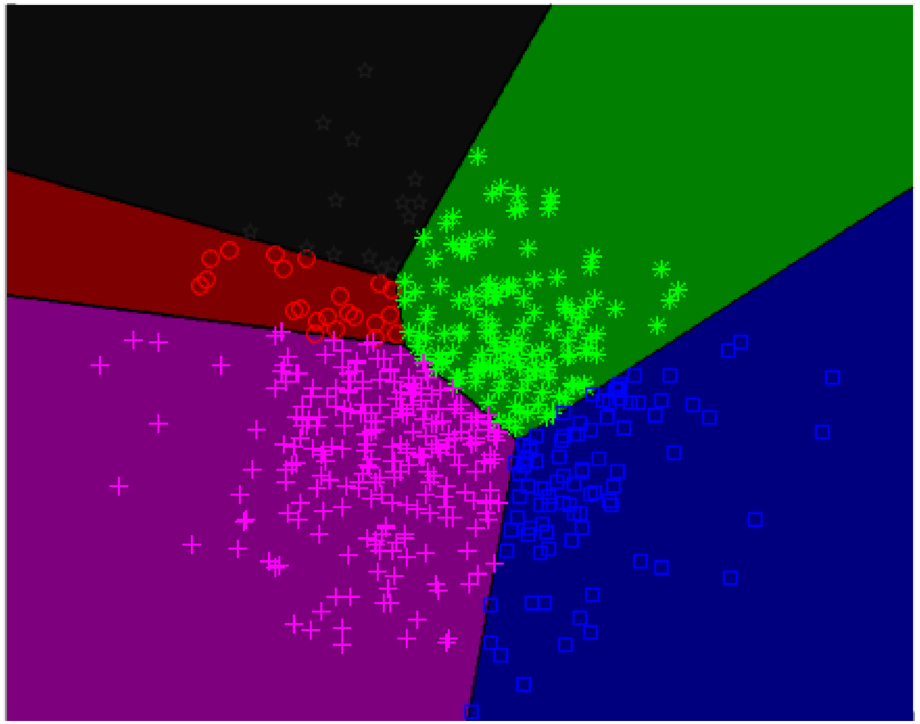
- Recall that a binary linear classifier splits space using a hyper-plane:




- 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:

$$w_{blue}^T x_i \geq w_{green}^T x_i$$
$$w_{blue}^T x_i \geq w_{magenta}^T x_i$$
$$w_{blue}^T x_i \geq w_{red}^T x_i$$
$$w_{blue}^T x_i \geq w_{black}^T x_i$$

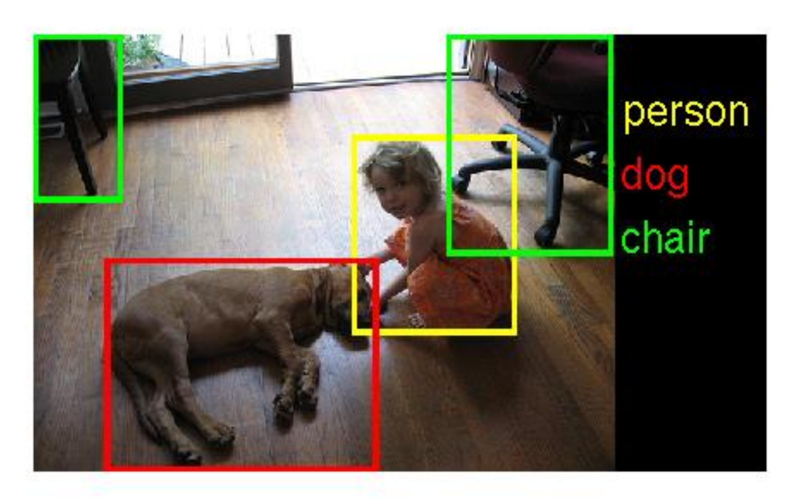
  - Could be non-convex with change of basis.

# Digression: Multi-Label Classification

- A related problem is multi-label classification:

$$X = \underbrace{\left[\begin{array}{c} \\ \\ \qquad\qquad\qquad \\ \\ \\ \end{array}\right]}_{d}\Bigg\}\,n \qquad Y = \underbrace{\begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \\ \hline 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{array}}_{k}\Bigg\}\,n \qquad W = \underbrace{\begin{bmatrix} \text{---} & w_1^T & \text{---} \\ \text{---} & w_2^T & \text{---} \\ & \vdots & \\ \text{---} & w_k^T & \text{---} \end{bmatrix}}_{d}\Bigg\}\,k$$

- 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.




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

# Multi-Class Linear Classification (MEMORIZE)

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

$$X = \begin{bmatrix} \; \end{bmatrix} \qquad y = \begin{bmatrix} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{bmatrix} \qquad W = \begin{bmatrix} \text{---} \; w_1^T \; \text{---} \\ \text{---} \; w_2^T \; \text{---} \\ \vdots \\ \text{---} \; w_k^T \; \text{---} \end{bmatrix}$$

train 'k' classifiers

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 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/

# 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"

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

# 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"

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

- 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.

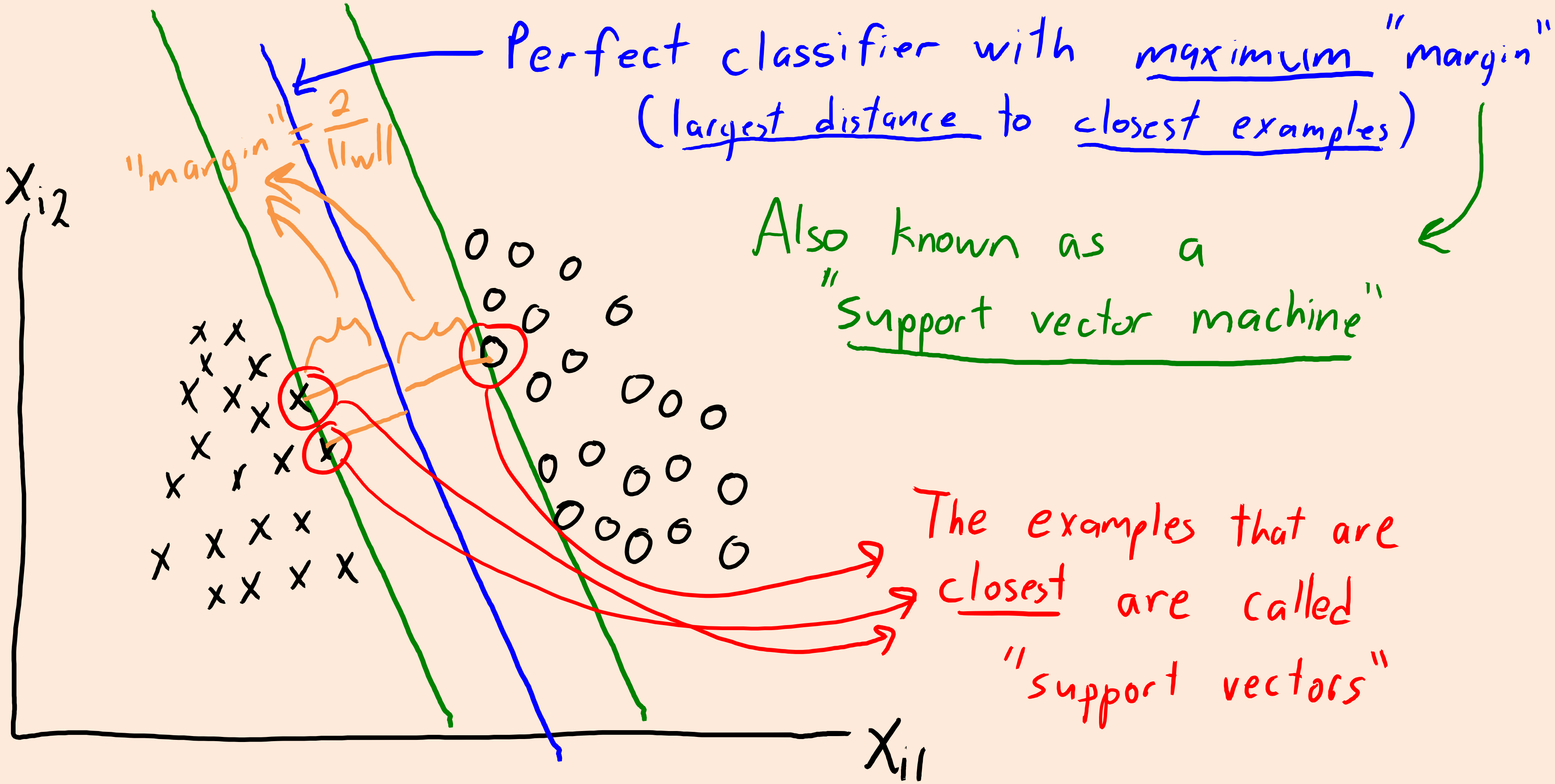
# Summary

- 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.



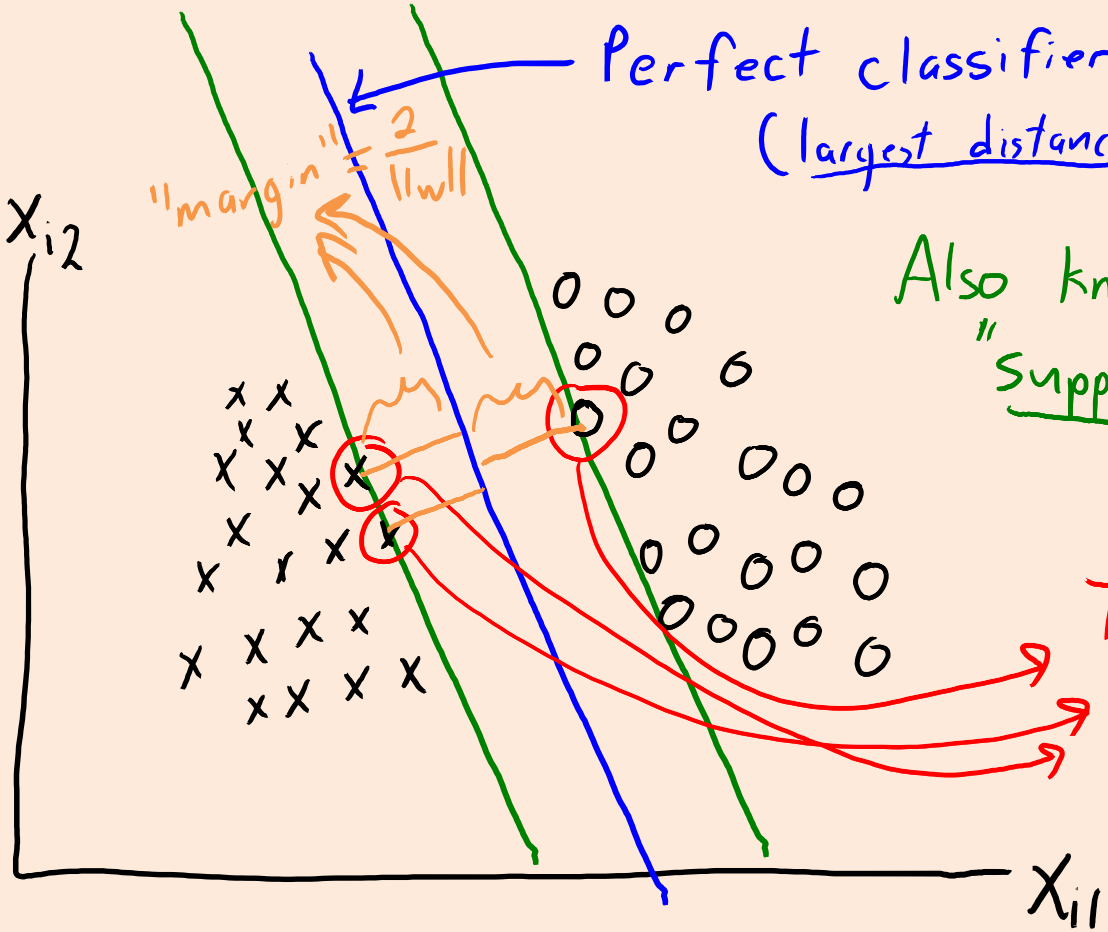
# Maximum-Margin Classifier

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

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

Also known as a "support vector machine"

Final classifier only depends on support vectors



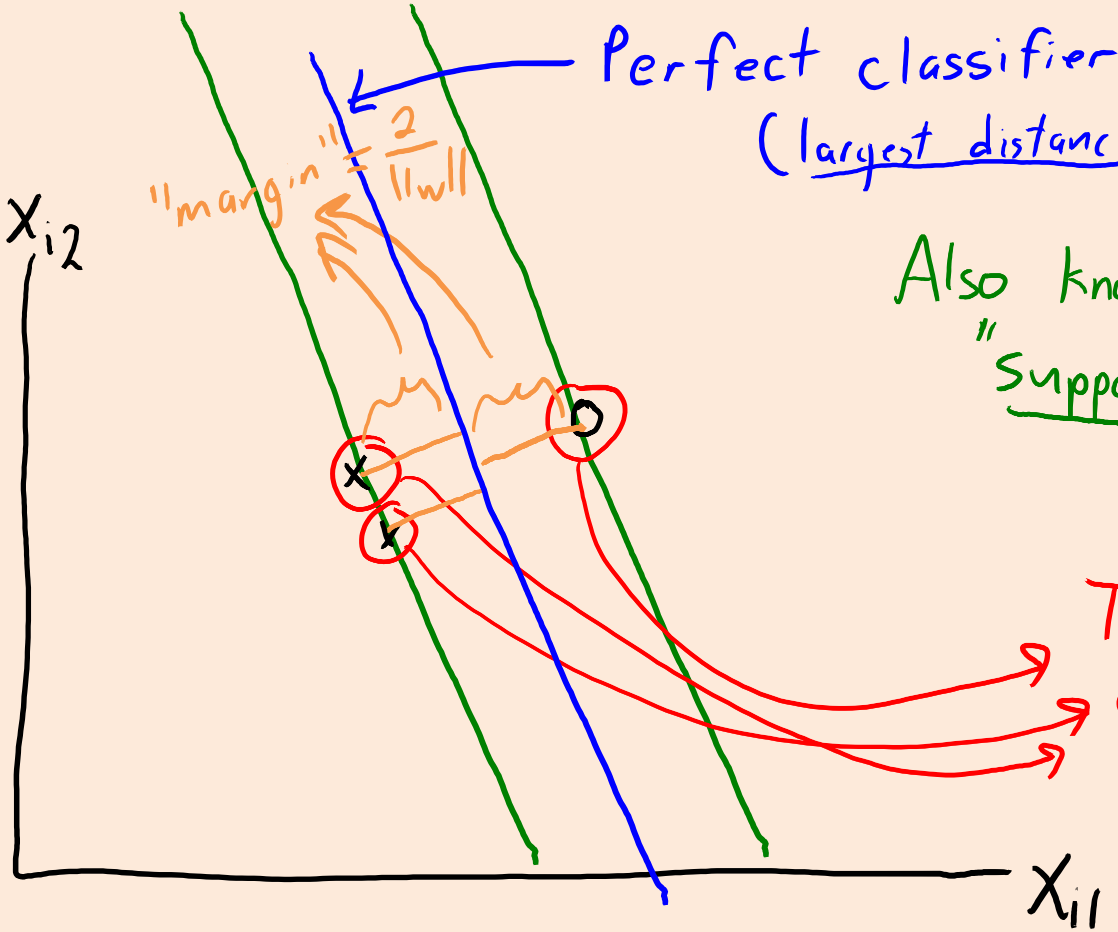

The examples that are closest are called "support vectors"

# Maximum-Margin Classifier

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

Final classifier only depends on support vectors

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




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

Also known as a "support vector machine"

The examples that are closest are called "support vectors"

# Support Vector Machines

- For linearly-separable data, SVM minimizes:

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

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

$$\begin{aligned} w^T x_i &\geq 1 \quad \text{for } y_i = 1 \\ w^T x_i &\leq -1 \quad \text{for } y_i = -1 \end{aligned} \quad \text{(classify all examples correctly)}$$

- 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:

Hinge loss for example 'i': if's the amount we violate $y_i w^T x_i \geq 1$ "slack"

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

Original SVM objective encourages large margin.

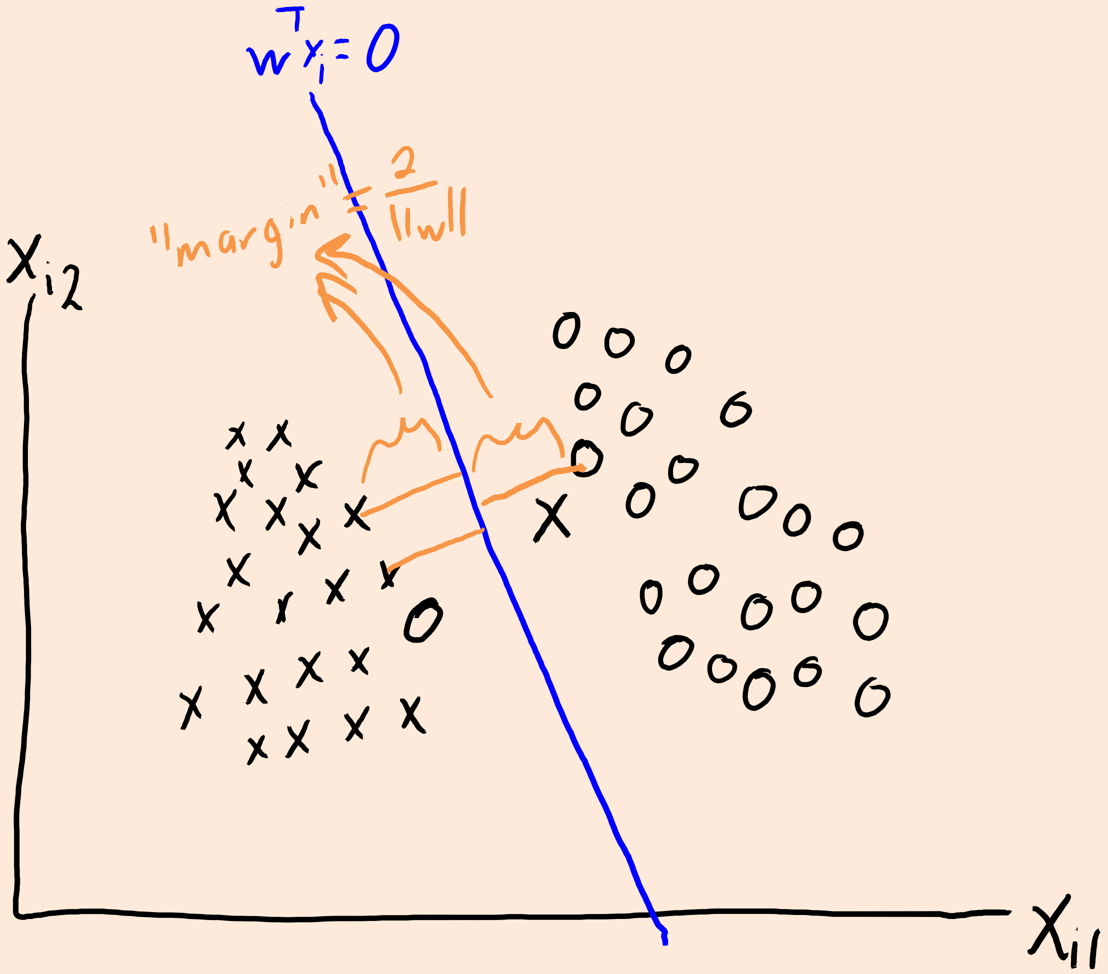
- We typically control margin/violation trade-off with parameter "λ":

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

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

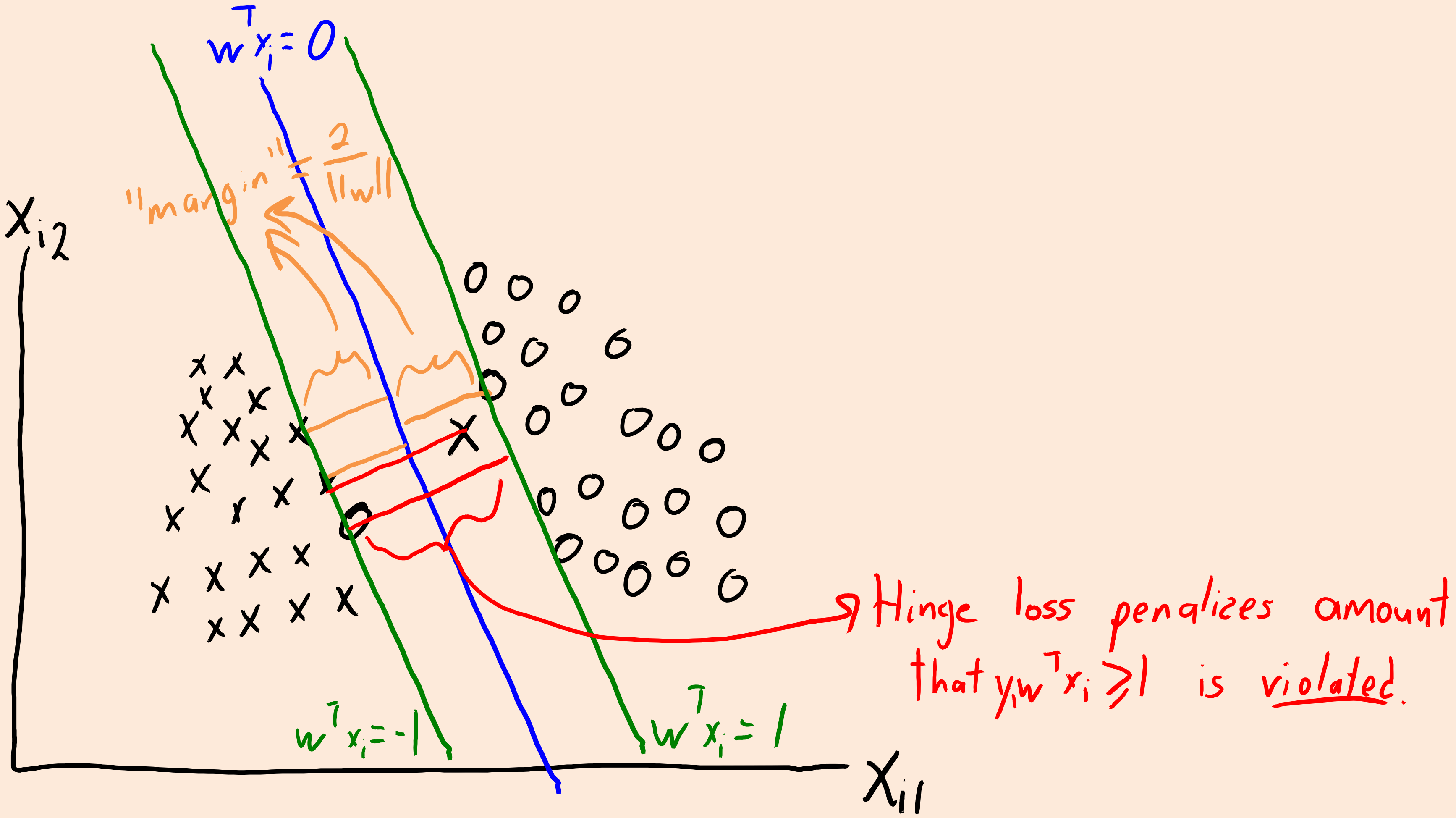
# Support Vector Machines for Non-Separable

- Non-separable case:



# Support Vector Machines for Non-Separable

- Non-separable case:



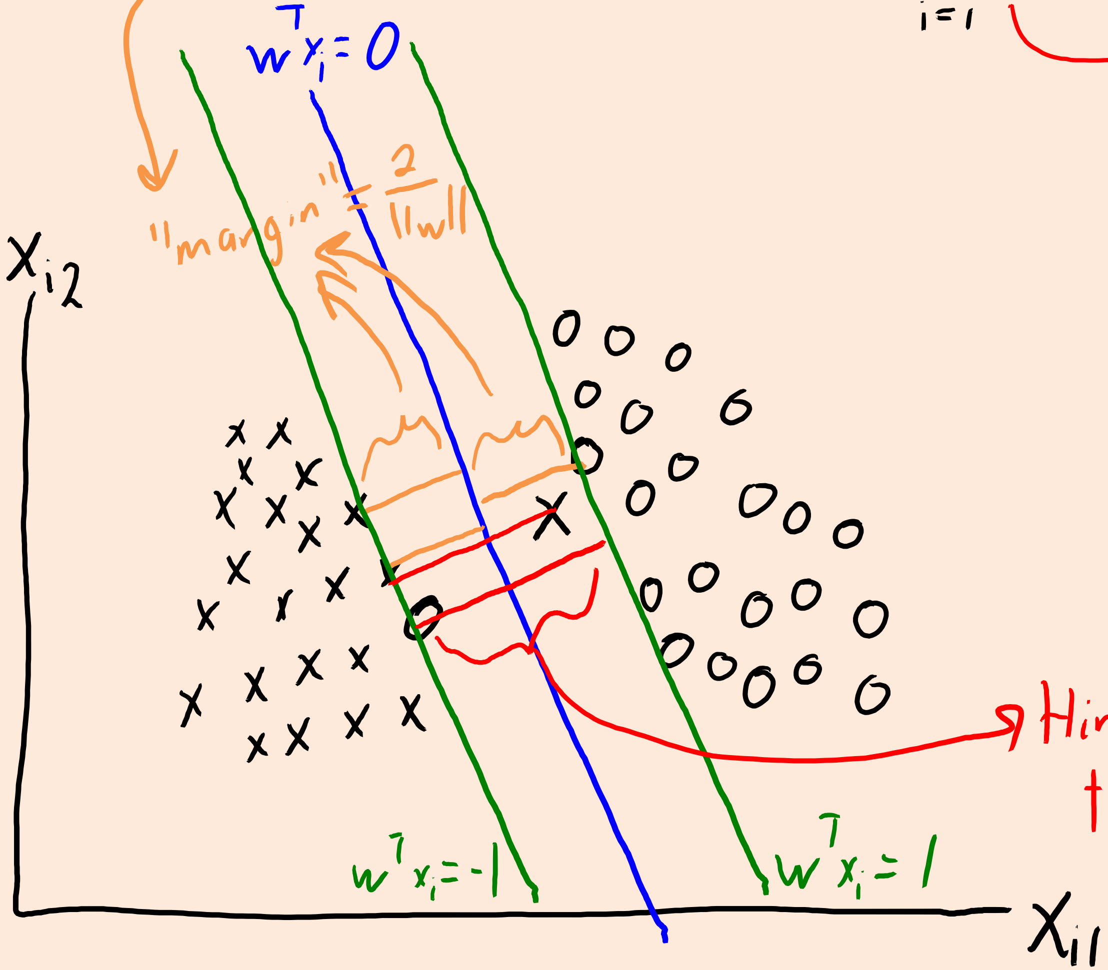
# Support Vector Machines for Non-Separable

- 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.




$\lambda$ 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:

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

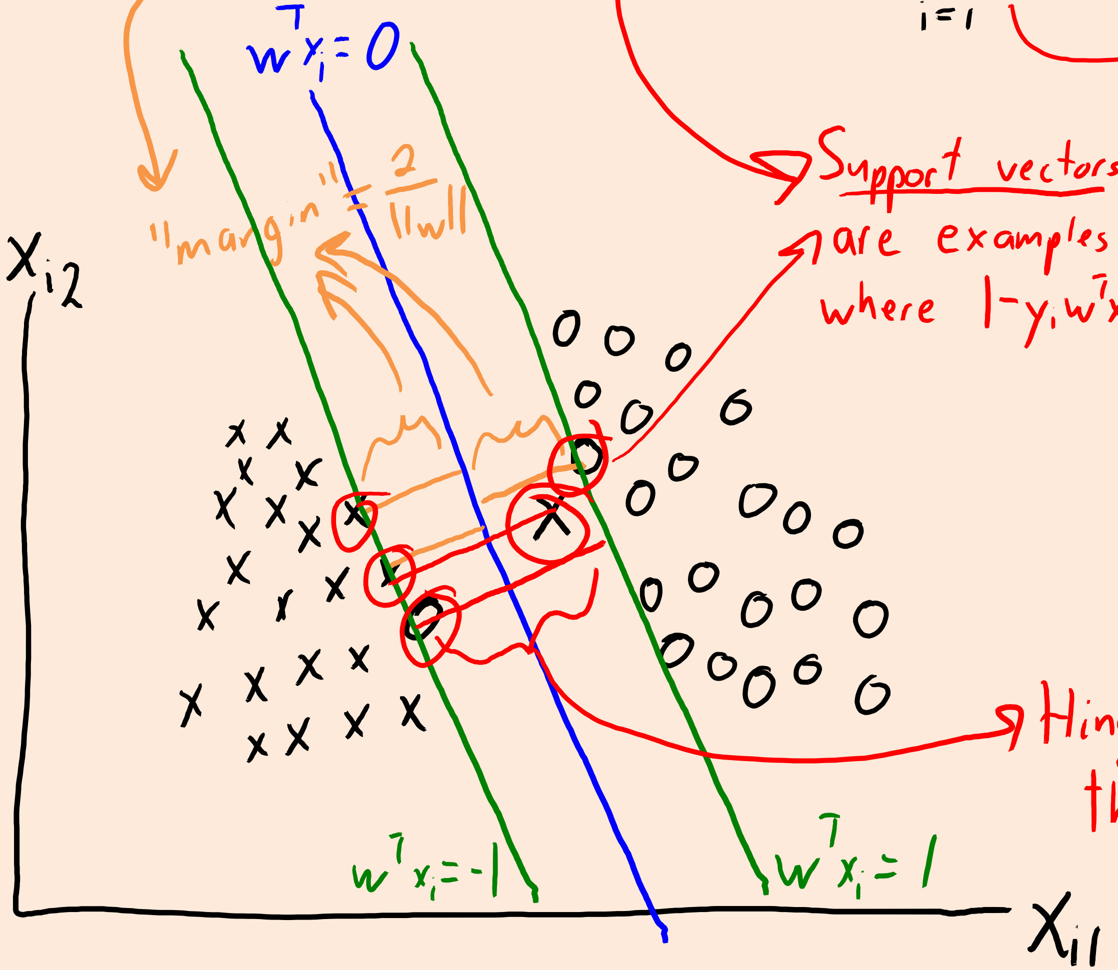
Support vectors are examples 'i' where $1 - y_i w^T x_i \geq 0$

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

$\lambda$ controls trade-off between having large margin and classifying examples correctly.

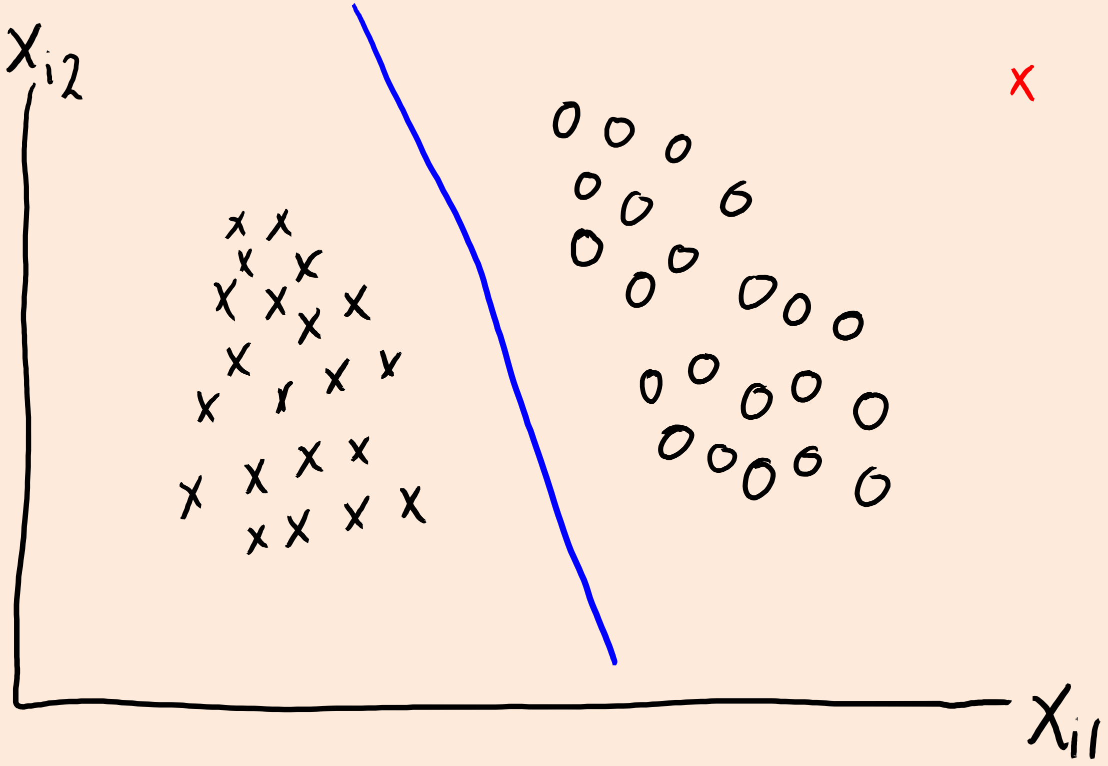
Logistic regression can be viewed as smooth approximation to SVMs.

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



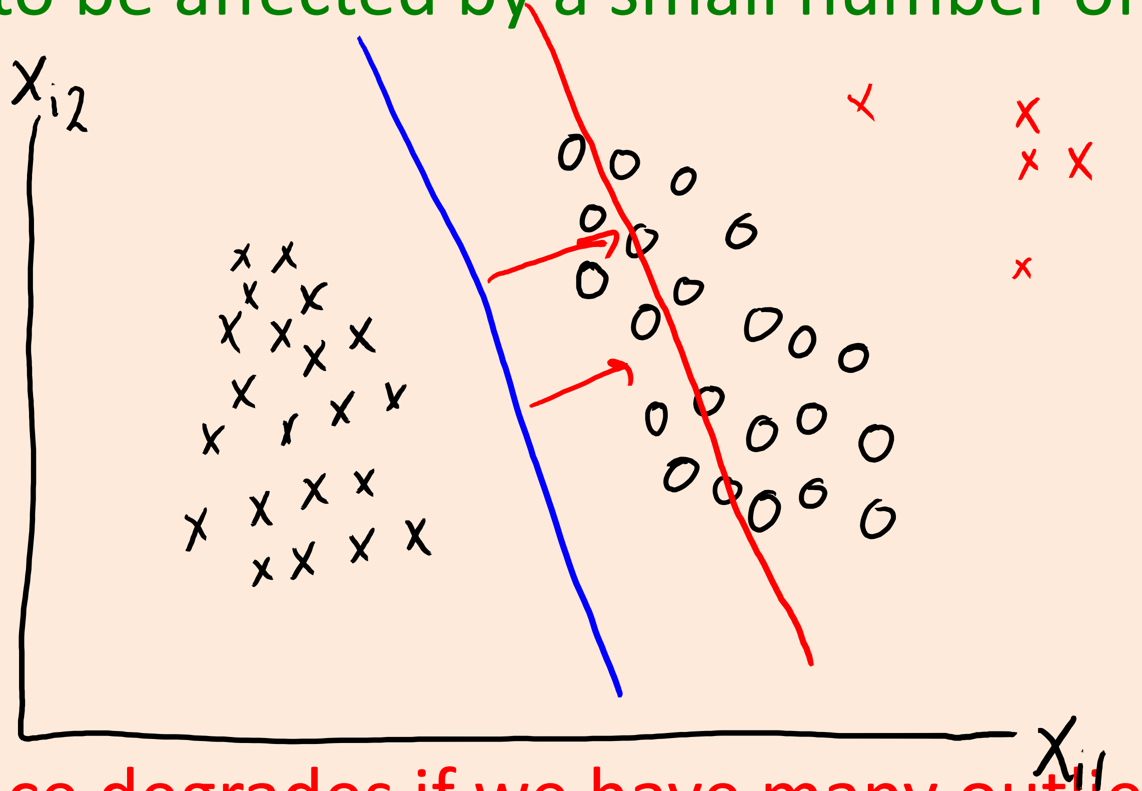
# Robustness and Convex Approximations

- 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

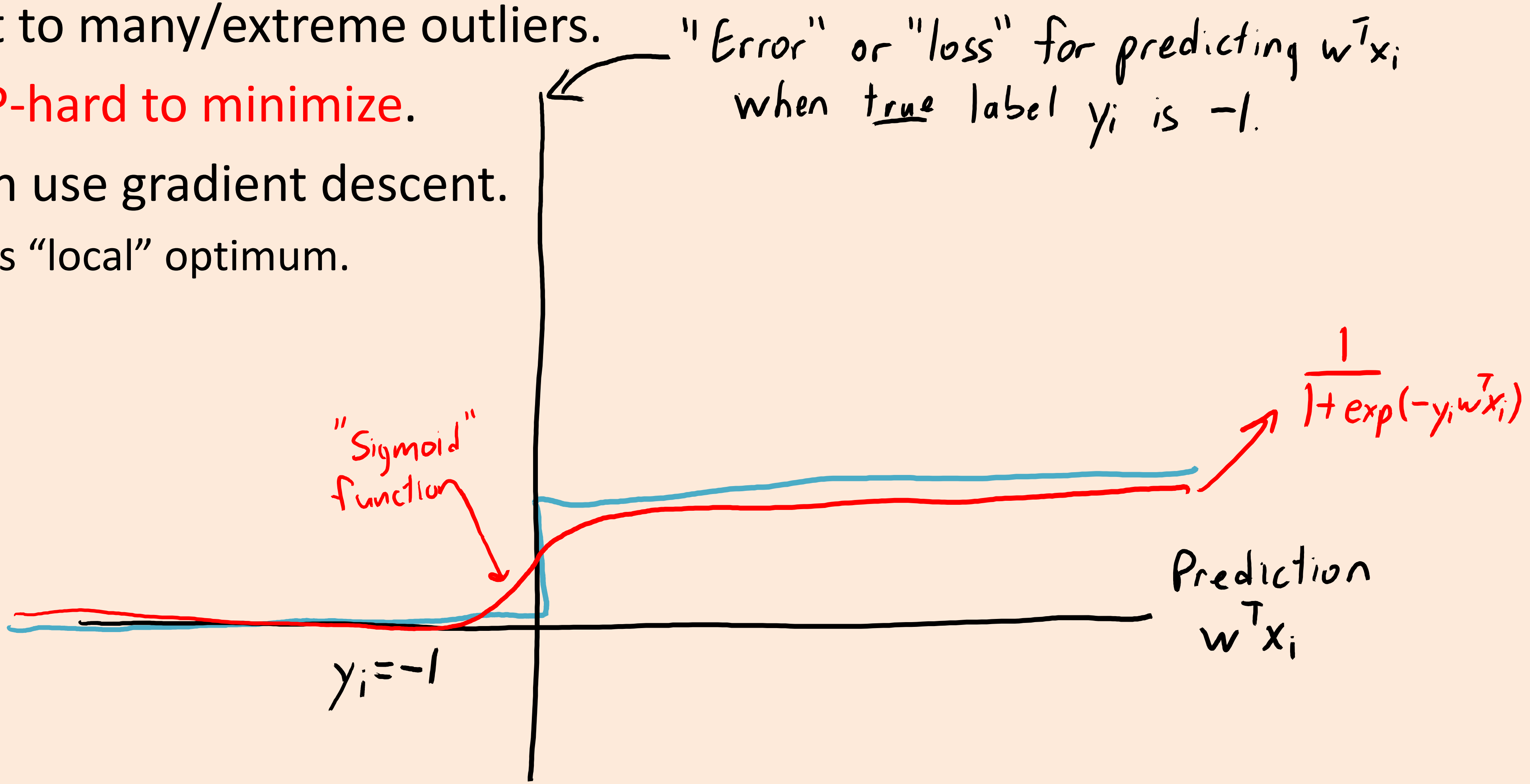
- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.




- 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 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:

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

- You would probably also want to regularize the ‘w’ with different $\lambda$.

# "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?
  - Syberian 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 Syberian husky over bulldog, Chihuahua, etc.
    - Even though we don't know if these are Syberian 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 <u>particular</u> $(c_1,c_2)$ values

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

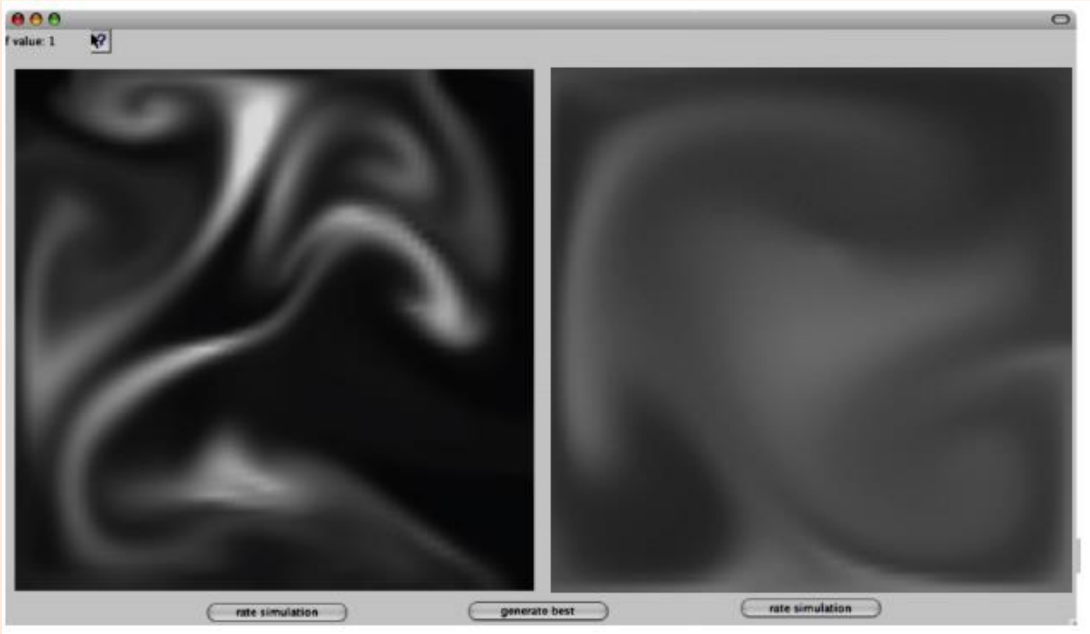
- By following the earlier steps, we can get objectives for this setting:

$$\sum_{i=1}^{n} \underbrace{\sum_{(c_1,c_2)} \max\{0, 1 - w_{c_1}^T x_i + w_{c_2}^T x_i\}}_{\text{"sum" version of multi-class SVM}} + \frac{\lambda}{2}\|W\|_F^2$$

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

# Learning with Pairwise Preferences (Ranking)

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

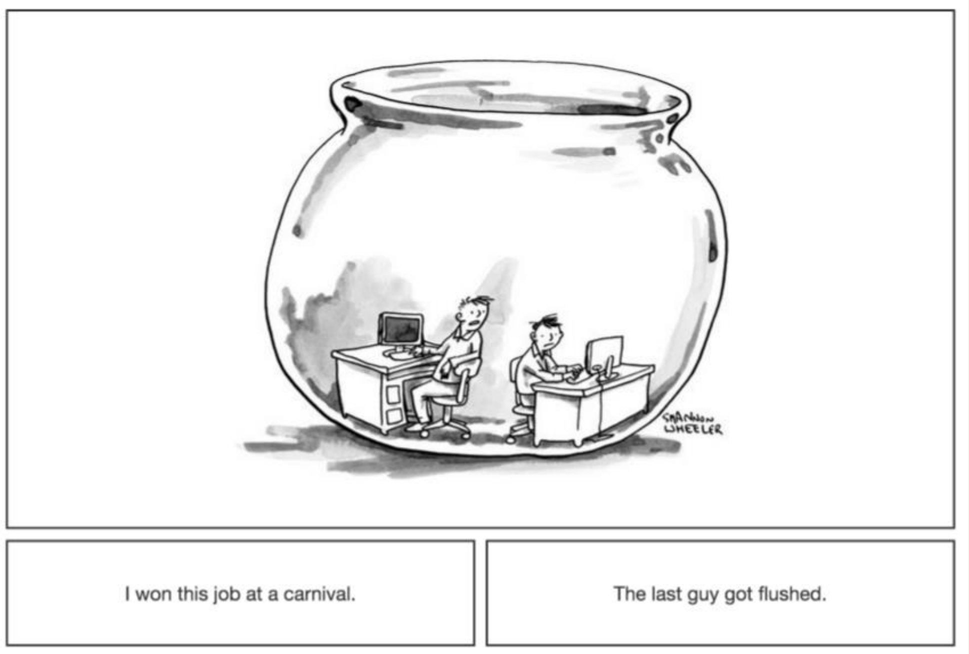



  - Don't know what the optimal parameters are, but we can ask the artist:
    - "Which one looks more like smoke"?

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