CPSC 340:
Machine Learning and Data Mining

More Linear Classifiers

Fall 2018

Admin

- Assignment 4:
  - Should be posted tonight and due Friday of next week.

- Midterm:
  - Grades posted.
  - Can view exam during Mike or my office hours this week and next week.

Last Time: Classification using Regression

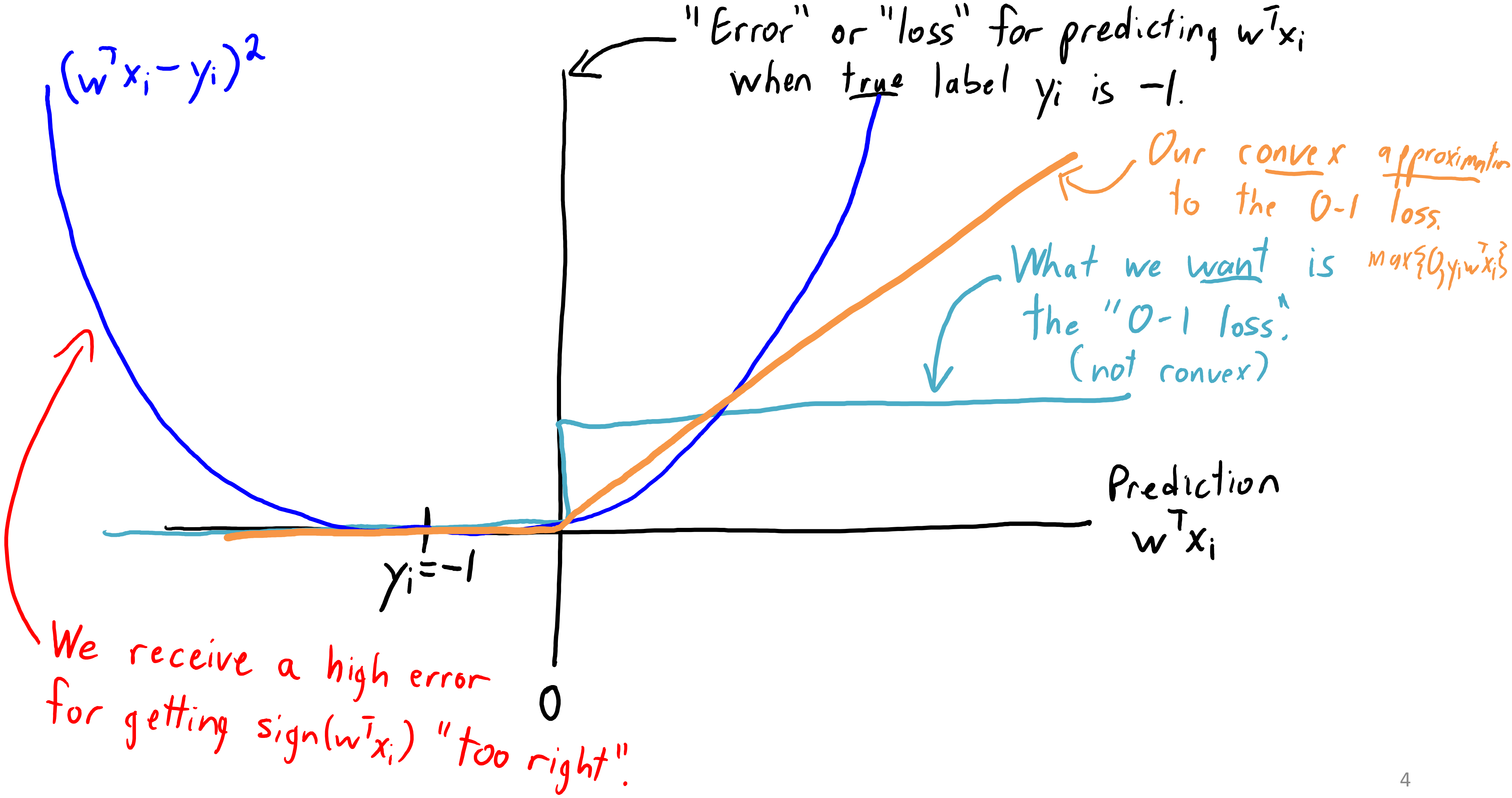
- Binary classification using sign of linear models:

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

+1 −1

- We considered three 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 error: $(\text{sign}(w^T x_i) = y_i)$?
    - Non-convex and hard to minimize in terms of 'w' (unless optimal error is 0 – perceptron).
  - Degenerate convex approximation to 0-1 error: $\max\{0, -y_i w^T x_i\}$.
    - Doesn't have the problems above, but has a degenerate solution of 0.

Hinge Loss: Convex Approximation to 0-1 Loss



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.
- 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)

- The violation of this constraint is now given by:

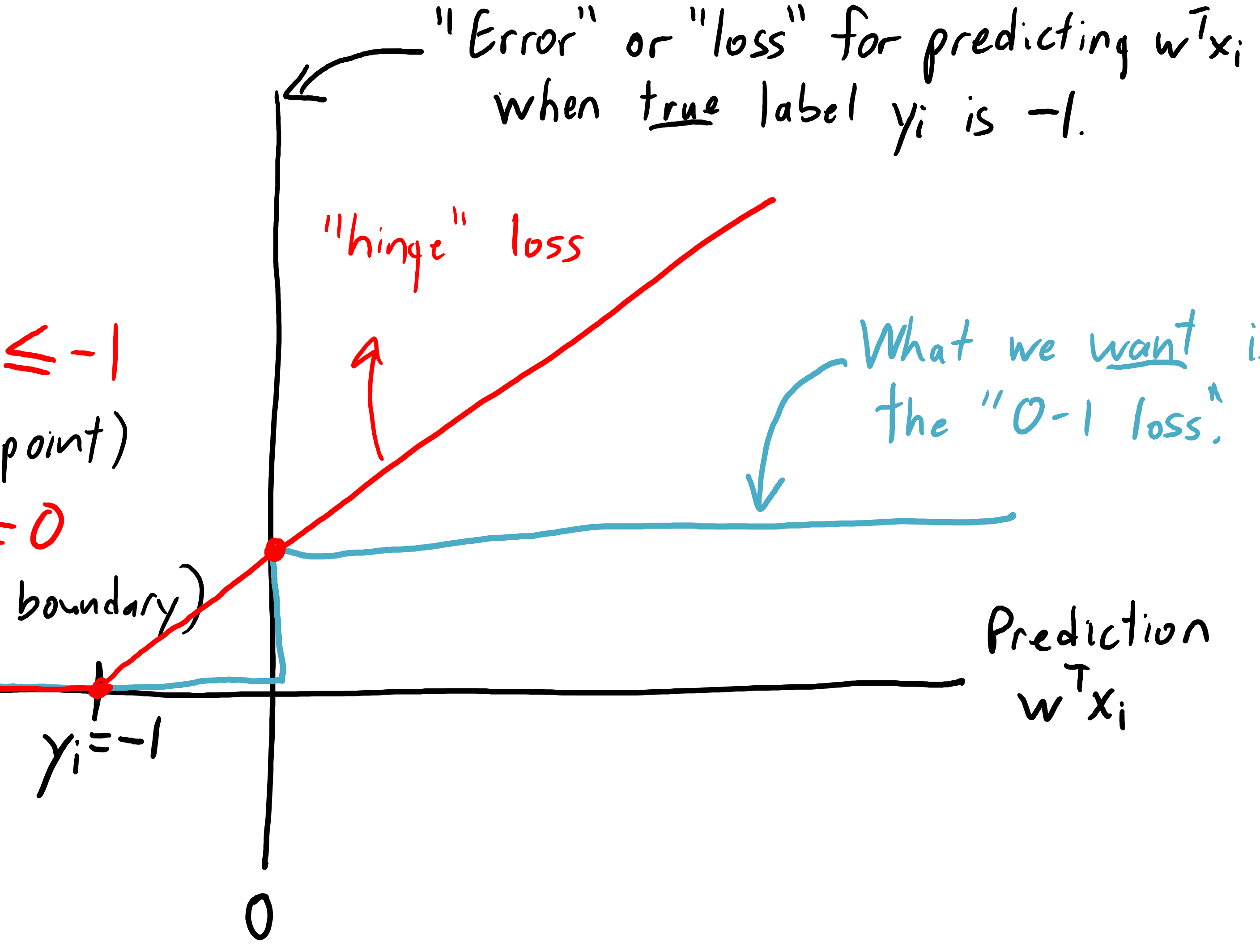
$$\max\{0, 1 - y_i w^T x_i\}$$

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

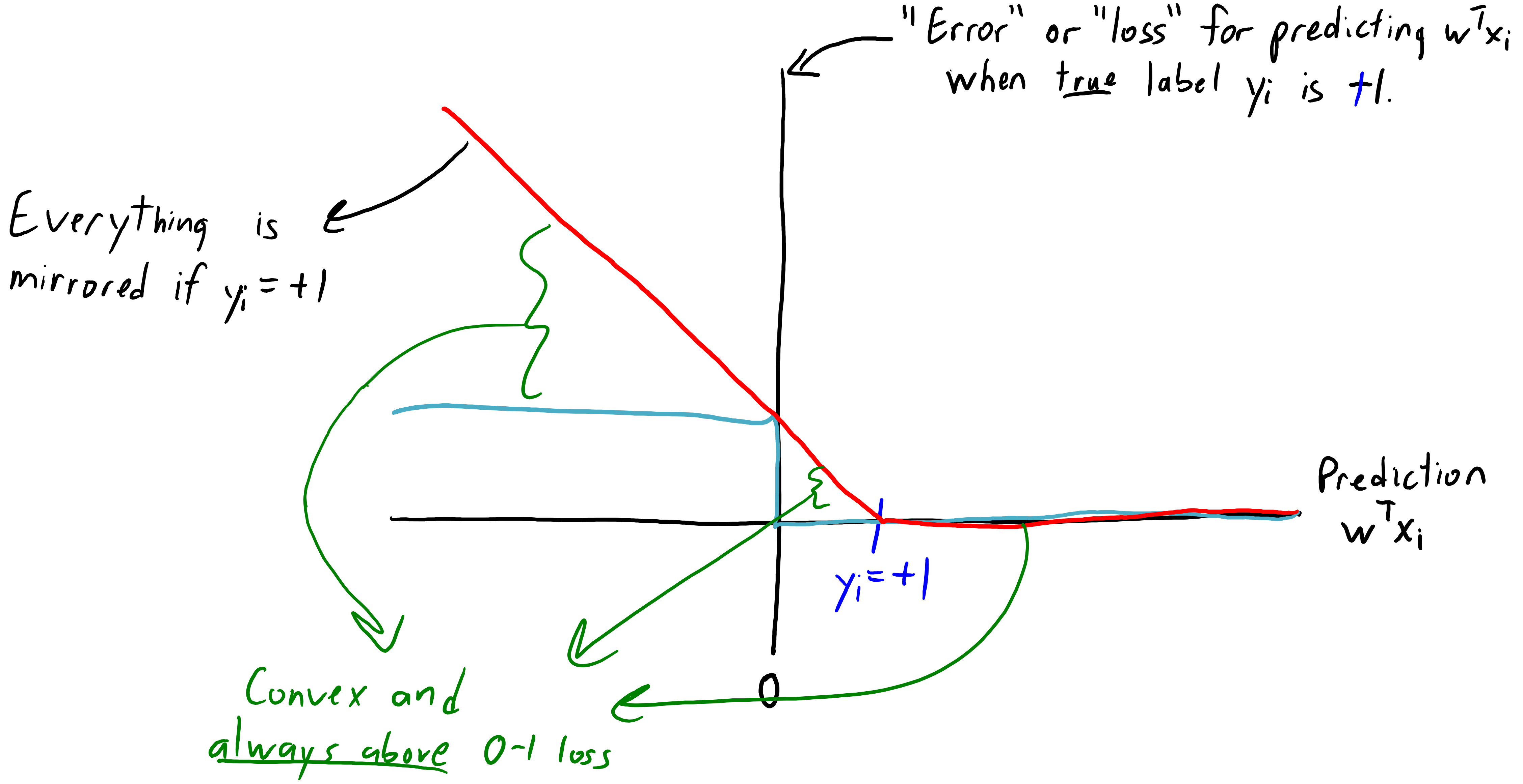
Hinge Loss: Convex Approximation to 0-1 Loss

Properties of the hinge loss:

1. Has error of 0 if $w^Tx_i \leq -1$
   (no penalty applied beyond this point)
2. Has a loss of 1 if $w^Tx_i = 0$
   (matches 0-1 loss at decision boundary)
3. Is convex and "close" to 0-1 loss.



Hinge Loss: Convex Approximation to 0-1 Loss



Hinge Loss

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

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

  – Convex upper bound on 0-1 loss.
    - If the hinge loss is 18.3, then number of training errors is at most 18.
    - So minimizing hinge loss indirectly tries to minimize training error.
    - Like perceptron, finds a perfect linear classifier if one exists.
- 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 problems.
  – SVMs can also be viewed as "maximizing the margin" (later in lecture).

'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:

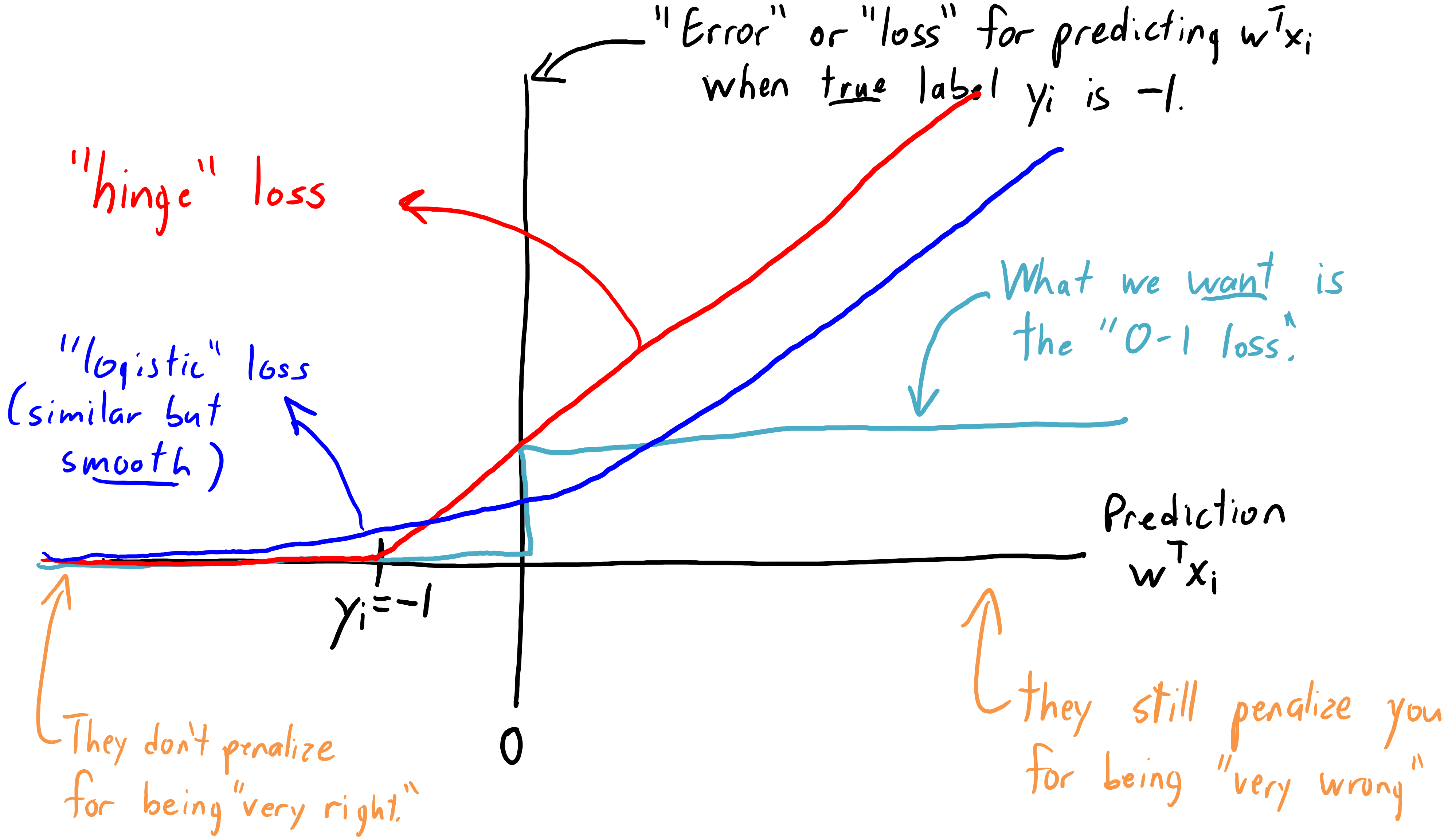
$$\max\{0, -y_i w^T x_i\} \approx \log(\underbrace{\exp(0)}_{1} + \exp(-y_i w^T x_i))$$

- 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 RBF basis and regularization.
    - With high-dimensional features and regularization.
  - Smoother predictions than random forests.

Comparison of “Black Box” Classifiers

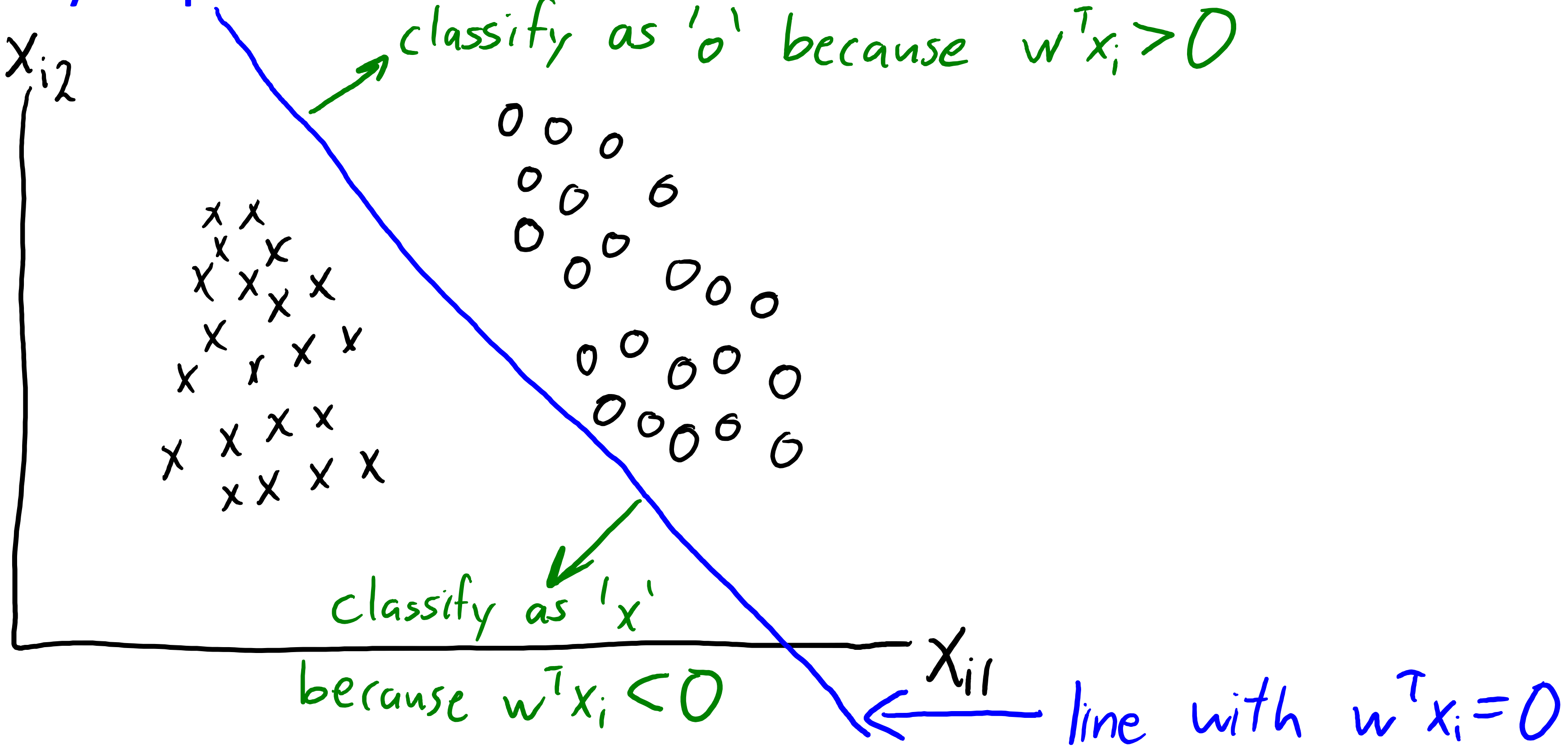
- Fernandez-Delgado et al. [2014]:
  - “Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?”

- 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?”

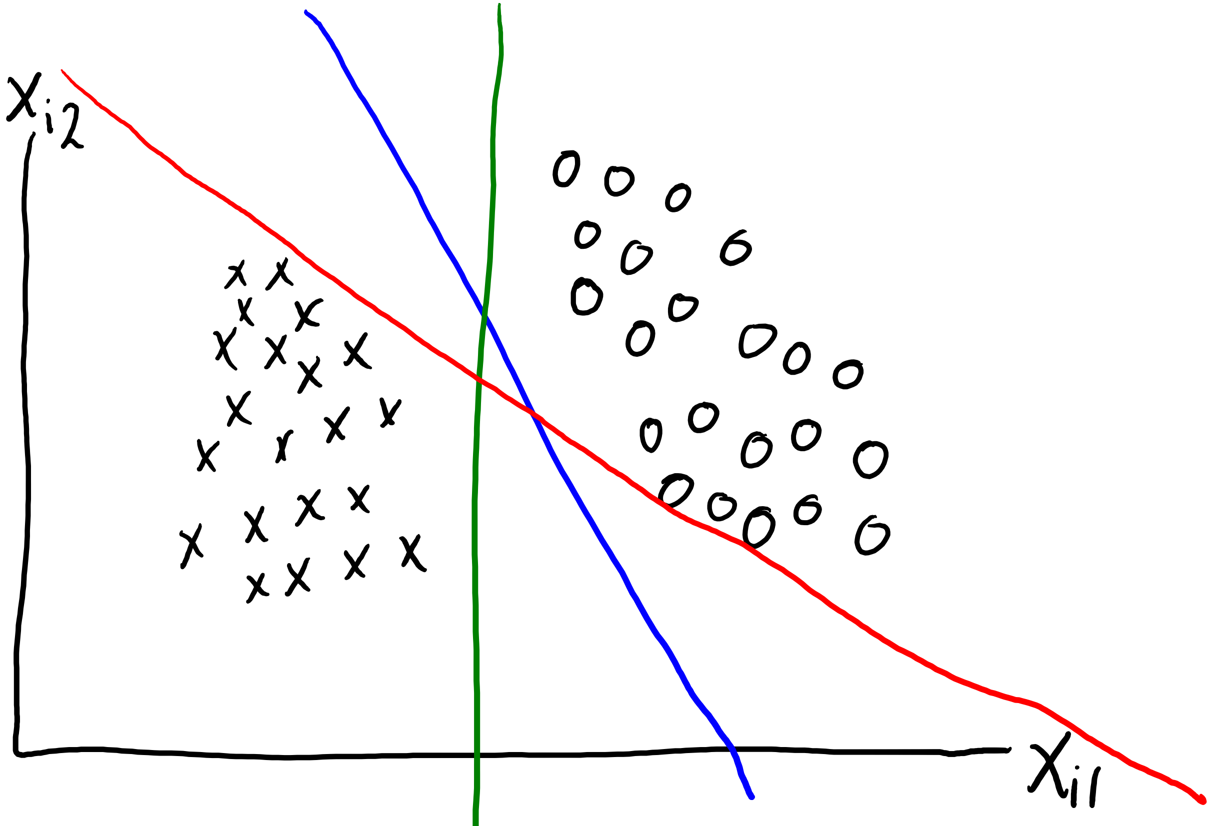
Maximum-Margin Classifier

- Consider a linearly-separable dataset.



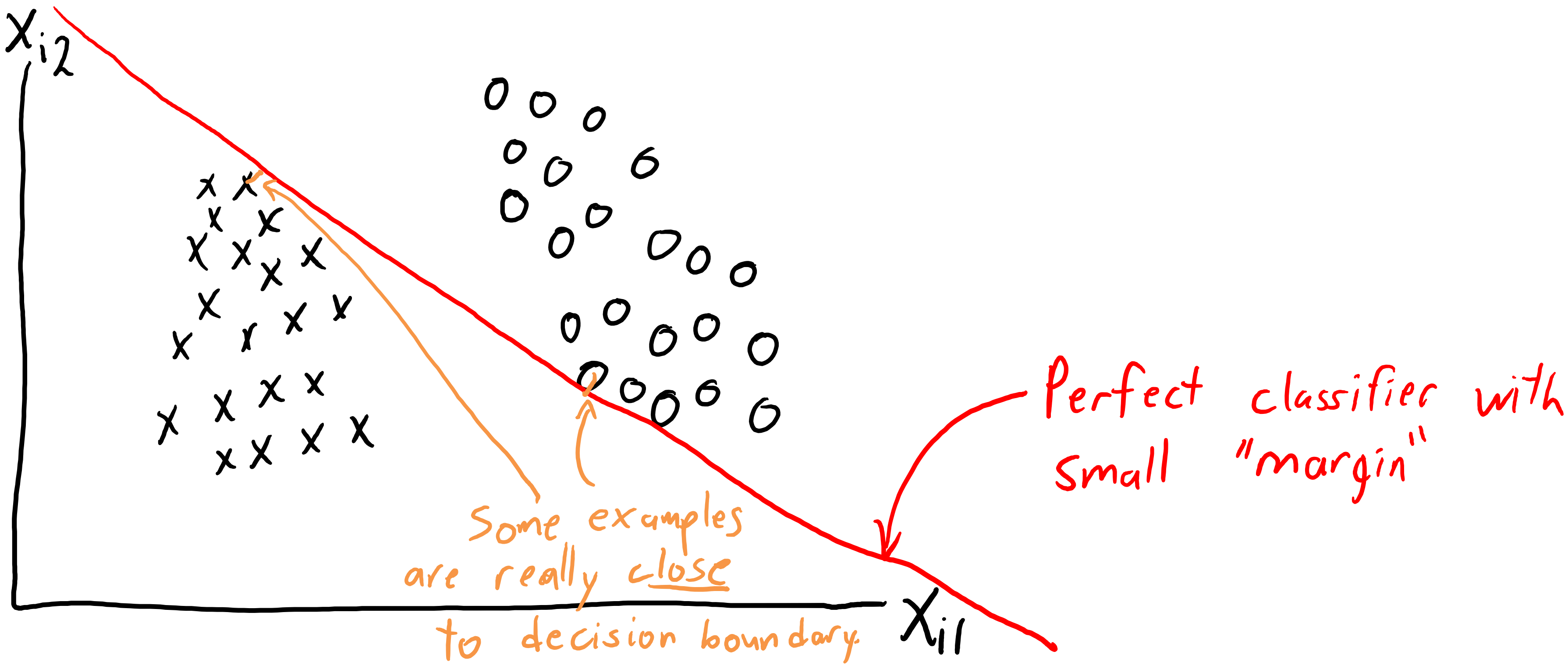
Maximum-Margin Classifier

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



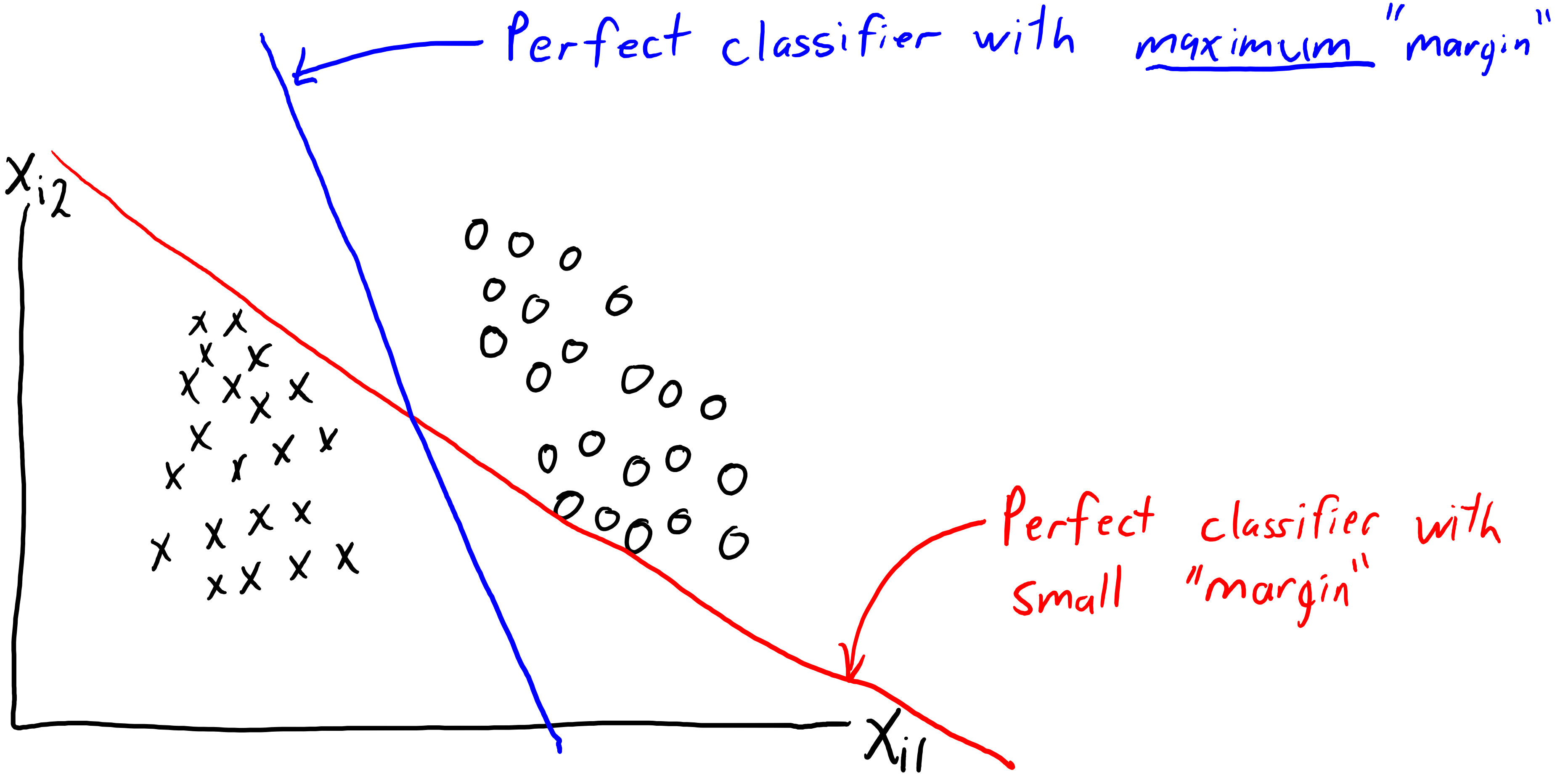
Maximum-Margin Classifier

- 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.

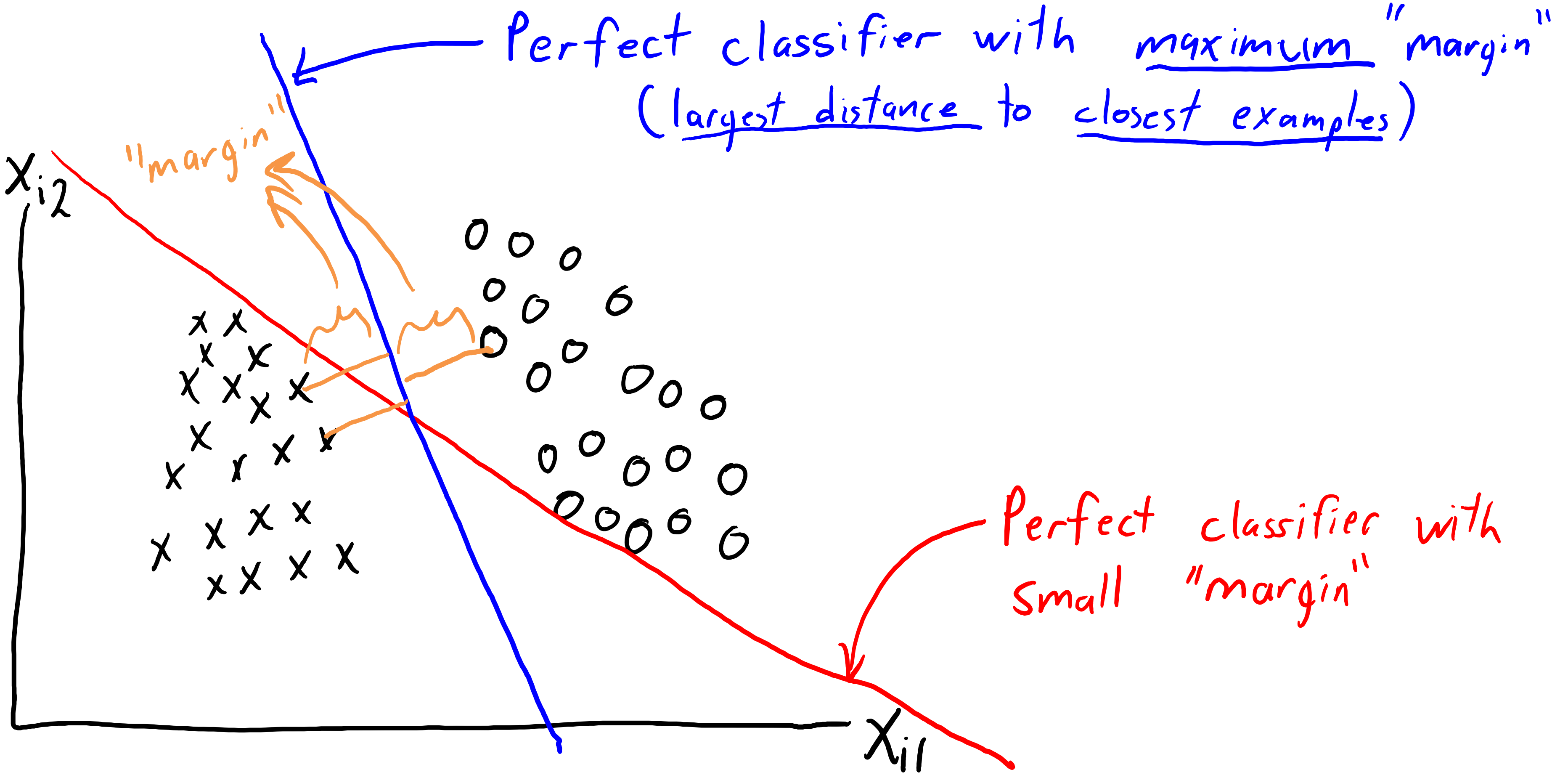


Maximum-Margin Classifier

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

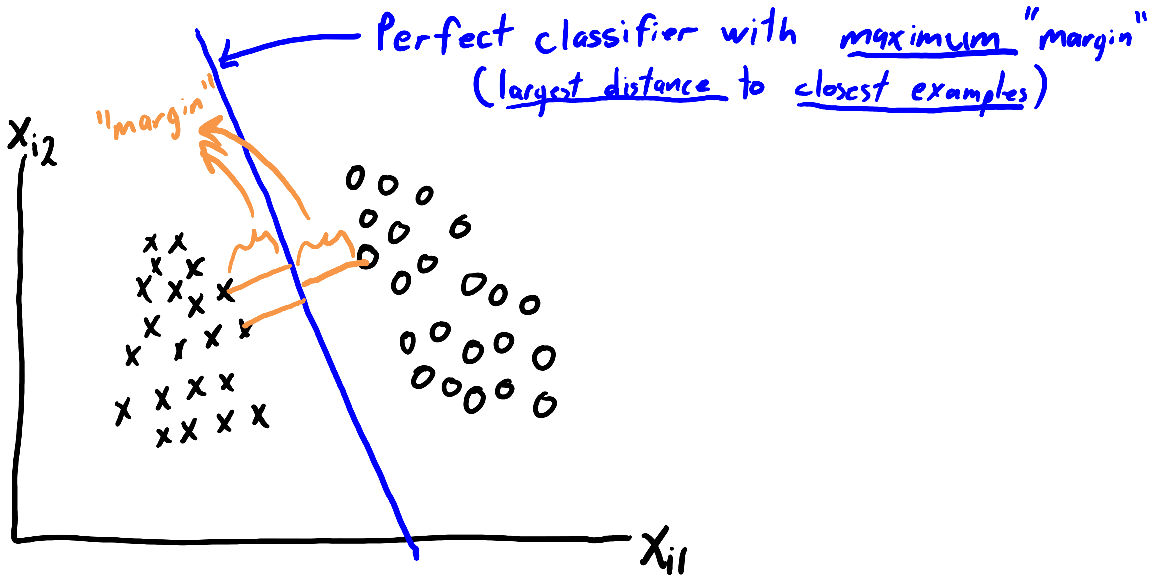
Why maximize margin?

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



Maximum-Margin Classifier

- For linearly-separable data:



- With small-enough λ > 0, SVMs find the maximum-margin classifier.
  – Origin of the name: the "support vectors" are the points closest to the line (see bonus).
  – Need λ small enough that hinge loss is 0 in solution.
- 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)

Motivation: Part of Speech (POS) Tagging

- Consider problem of finding the verb in a sentence:
  - “The 340 students jumped at the chance to hear about POS features.”
- Part of speech (POS) tagging is the problem of labeling all words.
  - >40 common syntactic POS tags.
  - Current systems have ~97% accuracy on standard test sets.
  - You can achieve this by applying “word-level” classifier to each word.
- What features of a word should we use for POS tagging?

But first…

- How do we categorical features in regression?
- Standard approach is to convert to a set of binary features:

Age	City	Income
23	Van	22,000.00
23	Bur	21,000.00
22	Van	0.00
25	Sur	57,000.00
19	Bur	13,500.00
22	Van	20,000.00

→

Age	Van	Bur	Sur	Income
23	1	0	0	22,000.00
23	0	1	0	21,000.00
22	1	0	0	0.00
25	0	0	1	57,000.00
19	0	1	0	13,500.00
22	1	0	0	20,000.00

POS Features

- Regularized multi-class logistic regression with 19 features gives ~97% accuracy:
  - Categorical features whose domain is all words ("lexical" features):
    - The word (e.g., "jumped" is usually a verb).
    - The previous word (e.g., "he" hit vs. "a" hit).
    - The previous previous word.
    - The next word.
    - The next next word.
  - Categorical features whose domain is combinations of letters ("stem" features):
    - Prefix of length 1 ("what letter does the word start with?")
    - Prefix of length 2.
    - Prefix of length 3.
    - Prefix of length 4 ("does it start with JUMP?")
    - Suffix of length 1.
    - Suffix of length 2.
    - Suffix of length 3 ("does it end in ING?")
    - Suffix of length 4.
  - Binary features ("shape" features):
    - Does word contain a number?
    - Does word contain a capital?
    - Does word contain a hyphen?

Multi-Class Linear Classification

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

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

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

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

Multi-Class Linear Classification

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

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

- In POS classification we have >40 possible labels instead of 2.
  – 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

- One vs all method for turns binary classifier into multi-class.

- Training phase:
  - For each class ‘c’, train binary classifier to predict whether example is a ‘c’.
  - So if we have ‘k’ classes, this gives ‘k’ classifiers.

- Prediction phase:
  - Apply the ‘k’ binary classifiers to get a “score” for each class ‘c’.
  - Return the ‘c’ with the highest score.

"One vs All" 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:

$$W = \underbrace{\begin{bmatrix} \text{---} \; w_1^T \; \text{---} \\ \text{---} \; w_2^T \; \text{---} \\ \vdots \\ \text{---} \; w_k^T \; \text{---} \end{bmatrix}}_{d} \Big\} k$$

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

"One vs All" Classification

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

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

  - For each class 'c', compute $w_c^T x_i$.
    - Ideally, we'll get sign($w_c^T x_i$) = +1 for one class and 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 $w_c^T x_i$ ("most positive").

Digression: Multi-Label Classification

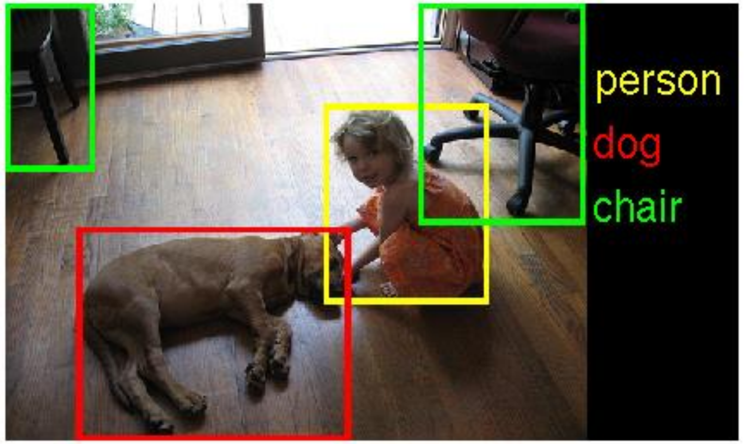
- A related problem is multi-label classification:

$$X = \left.\underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{d}\right\}n$$

$$Y = \left.\underbrace{\begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix} \end{array}}_{k}\right\}n$$

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

- 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 $\text{sign}(w_c^T x_i) = +1$ as the labels.



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

"One vs All" Multi-Class Classification

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

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

train 'k' classifiers

$$W = \begin{bmatrix} \text{---} & w_1^T & \text{---} \\ \text{---} & w_2^T & \text{---} \\ & \vdots & \\ \text{---} & w_k^T & \text{---} \end{bmatrix}$$

Predict by maximizing $w_c^T x_i$

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

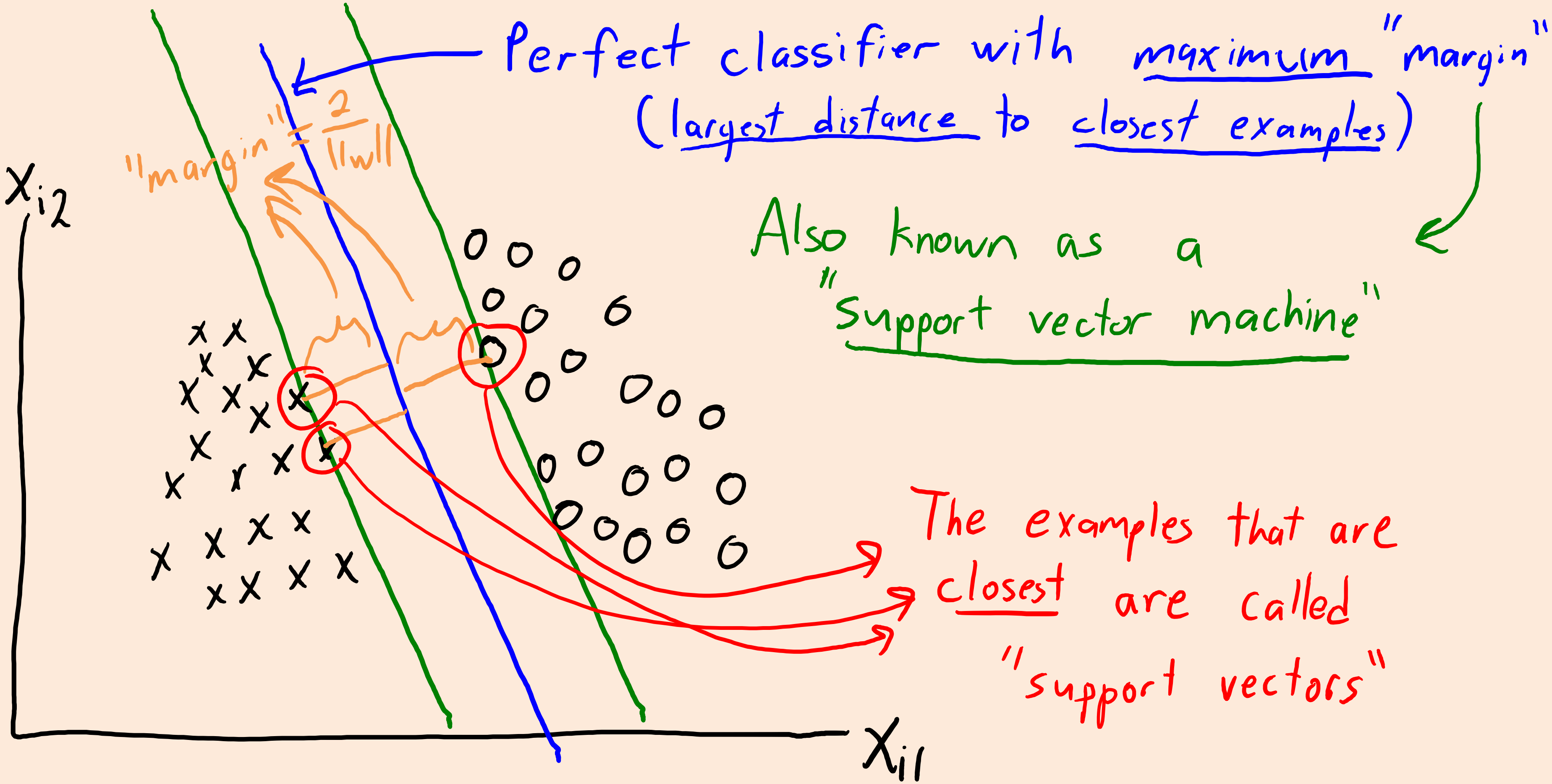
Summary

- Hinge loss is a convex upper bound on 0-1 loss.
  - SVMs add L2-regularization, can be viewed as "maximizing the margin".
- Logistic loss is a smooth convex approximation to the 0-1 loss.
  - "Logistic regression", also maximizes margin if you use gradient descent.
- SVMs and logistic regression are very widely-used.
  - A lot of ML consulting: "find good features, use L2-regularized logistic".
  - Both are just linear classifiers (a hyperplane dividing into two halfspaces).
- Word features: lexical, stem, shape.
- One vs all turns a binary classifier into a multi-class classifier.

- Next time:
  - A trick that lets you find gold and use polynomial basis with $d > 1$.

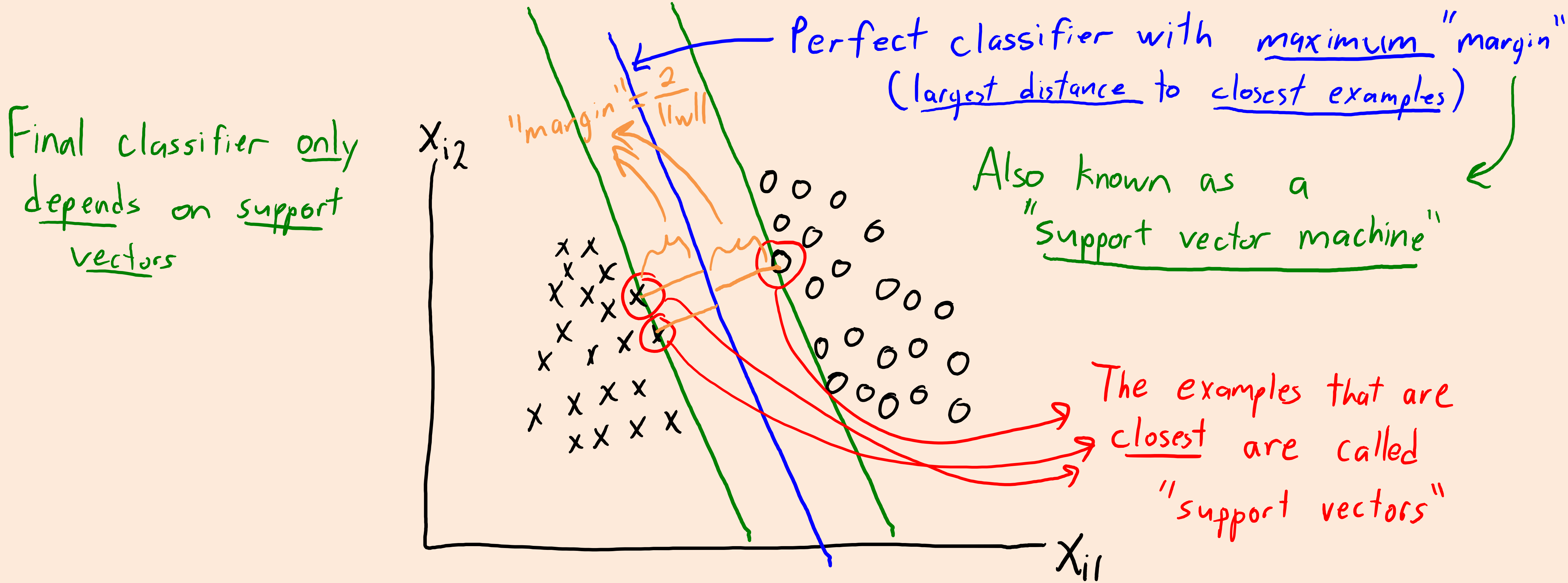
Maximum-Margin Classifier

- 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.



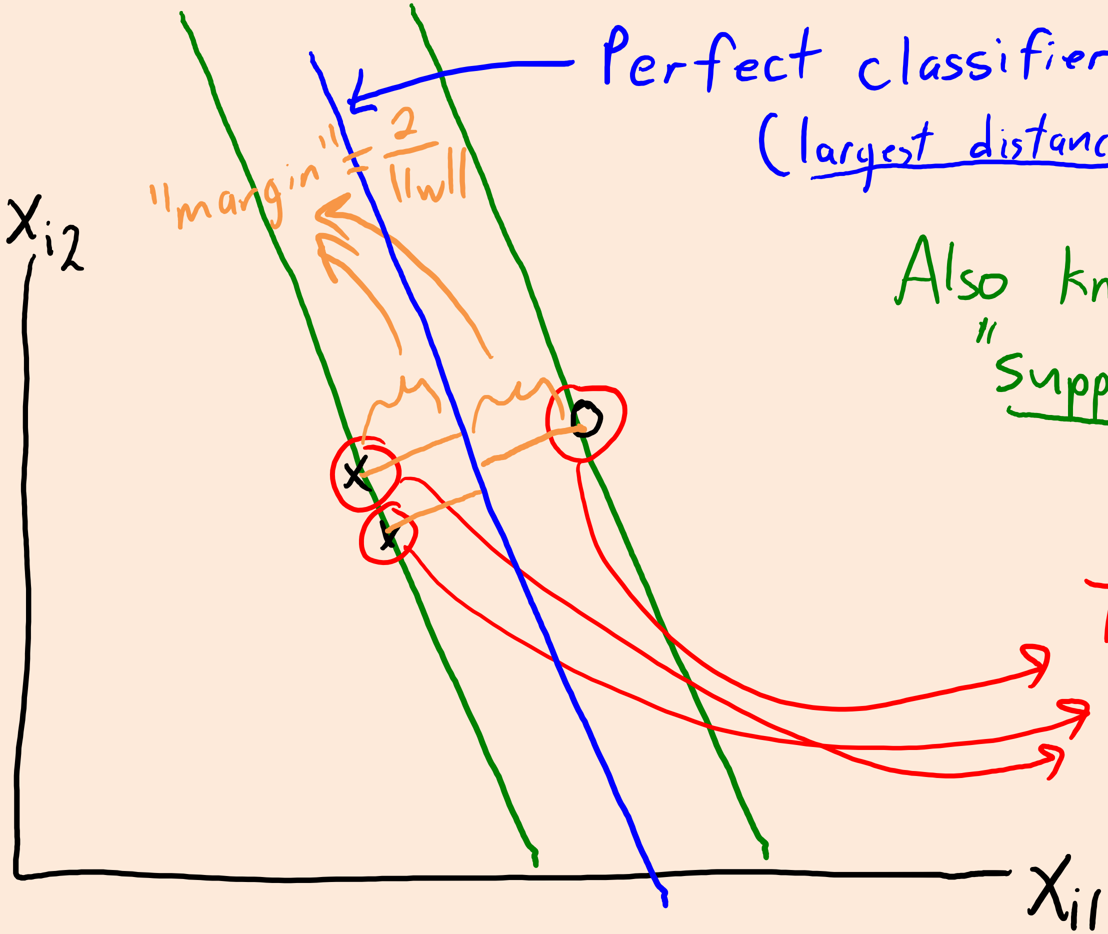
Maximum-Margin Classifier

- 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:

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

  - 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:

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

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.

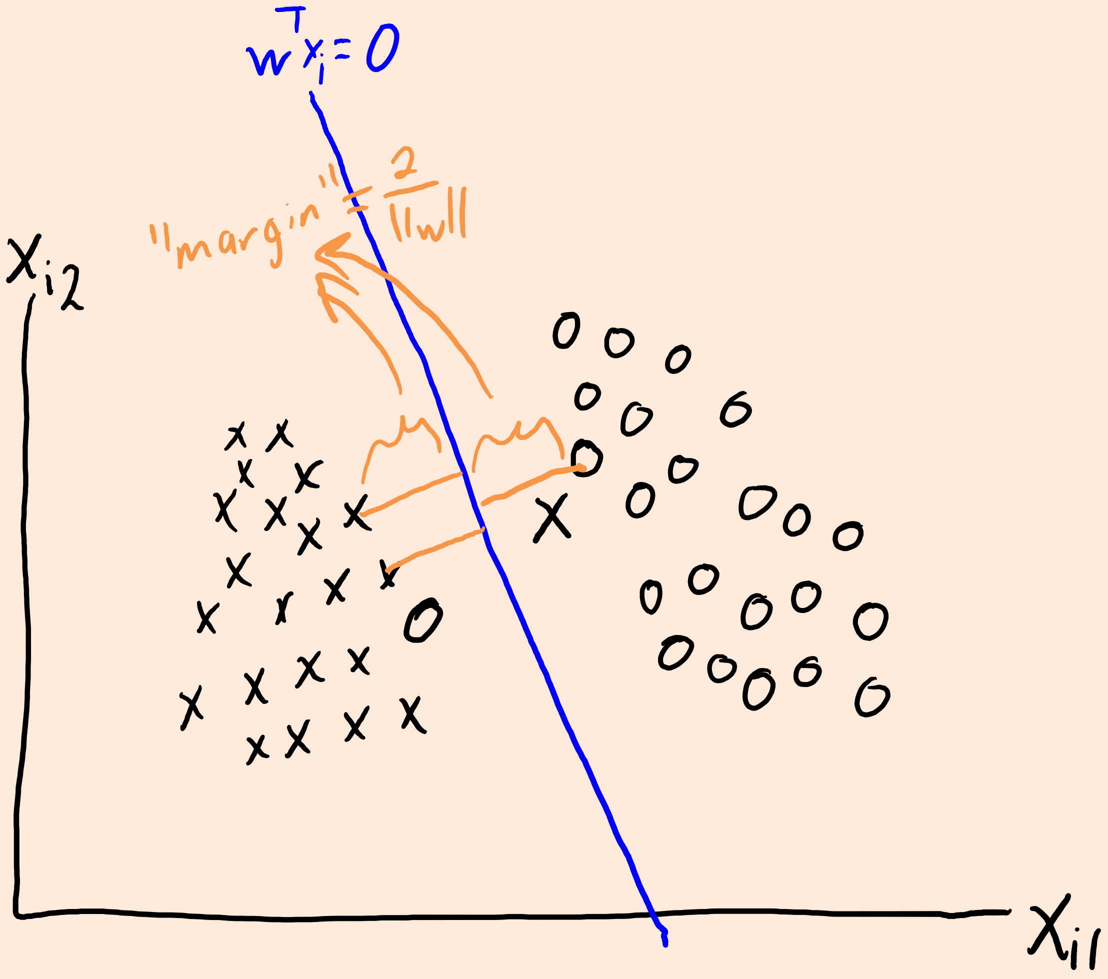
- We typically control margin/violation trade-off with parameter "λ":

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

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

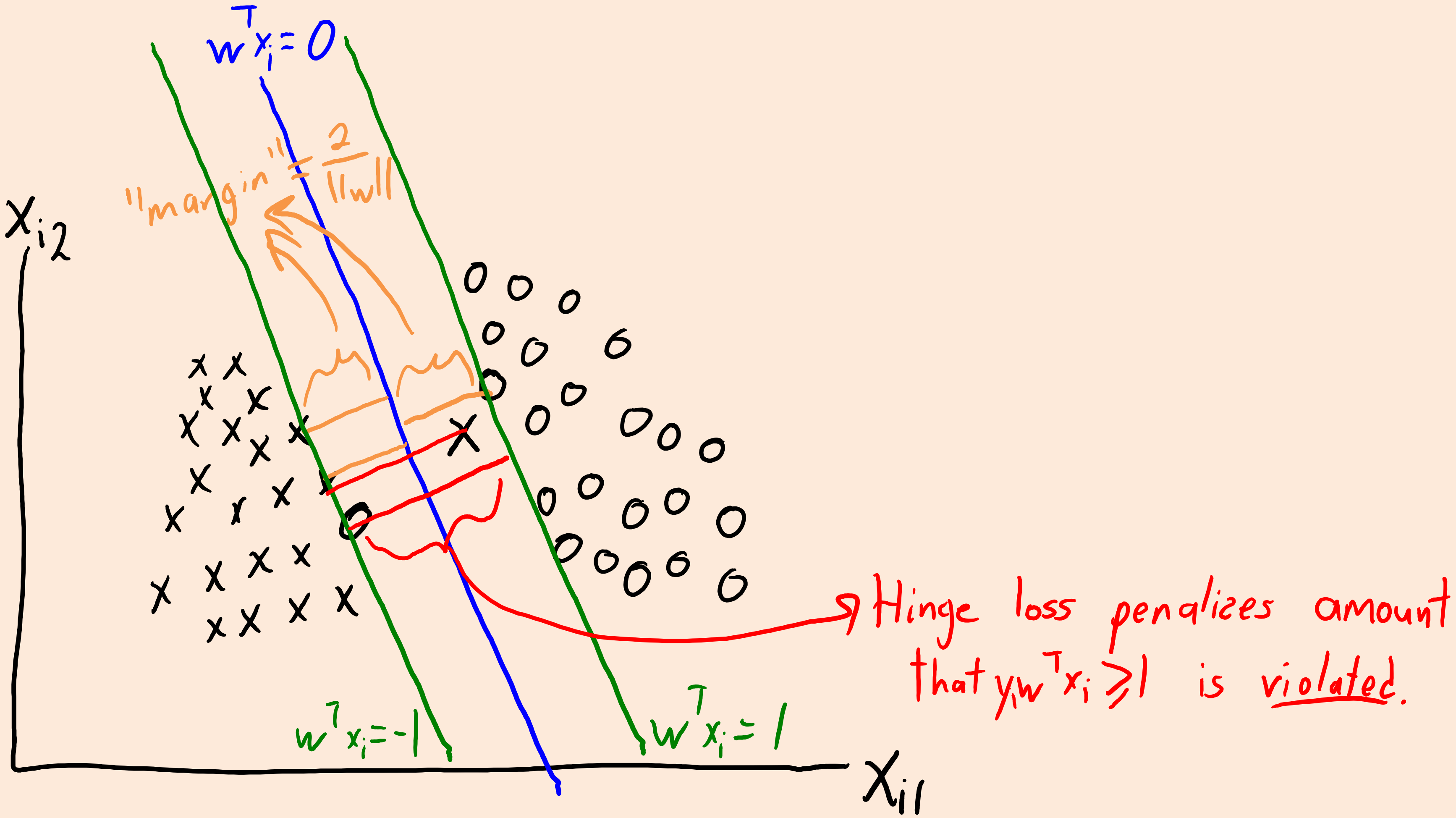
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:



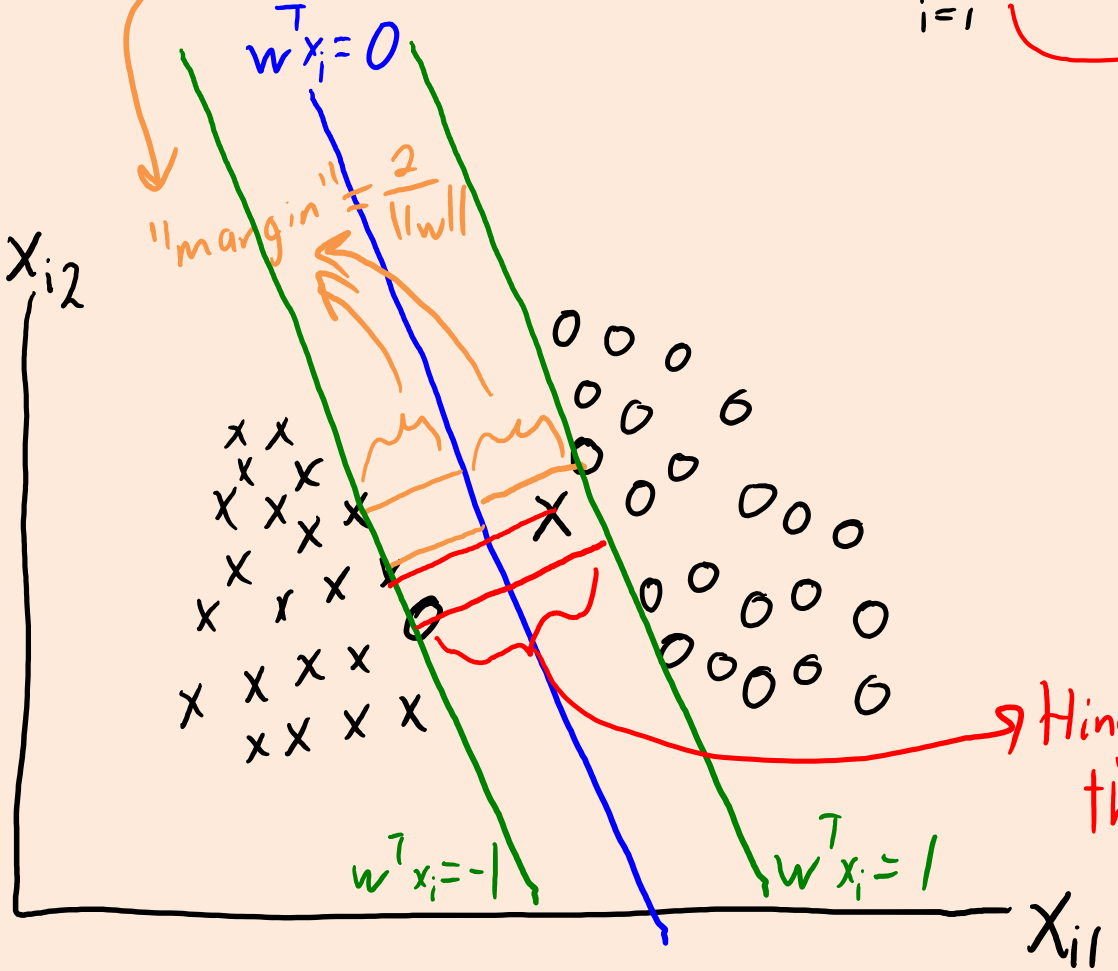
Support Vector Machines for Non-Separable

- 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.



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:

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

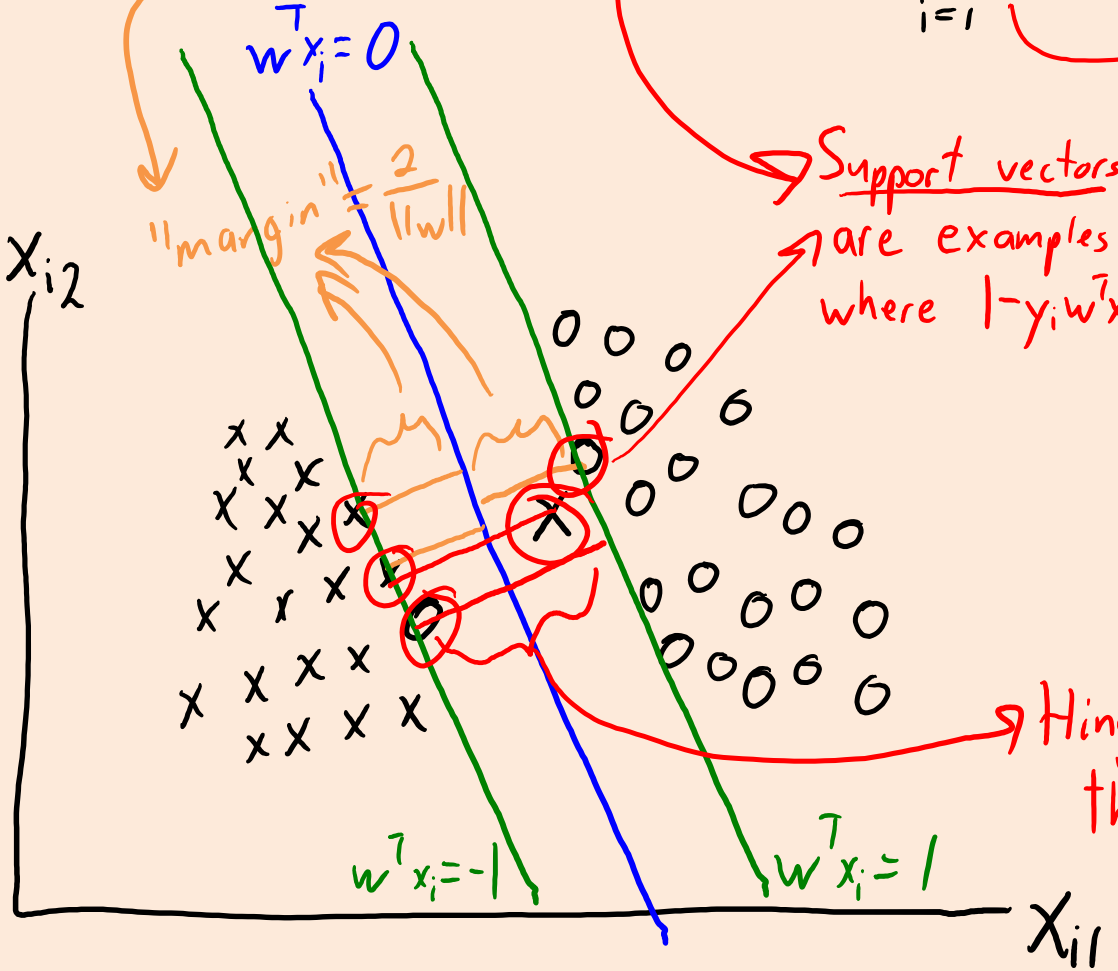
Support vectors are examples 'i' where $1 - y_i w^T x_i \geq 0$

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.

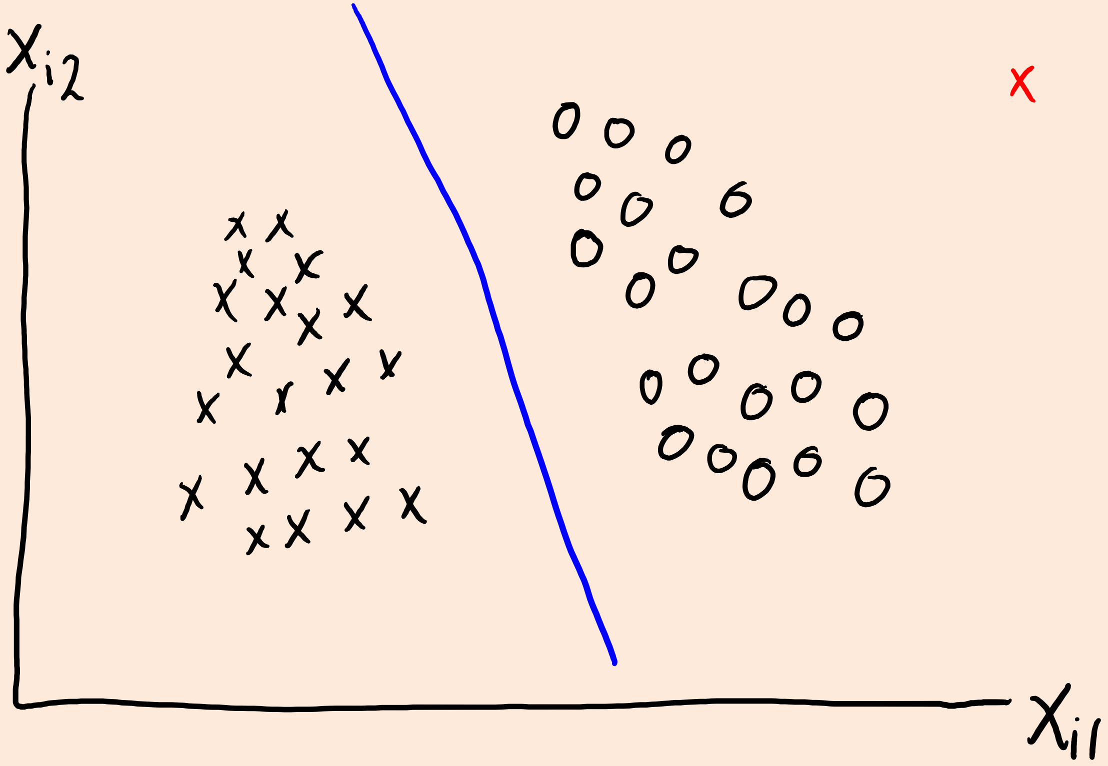
Logistic regression can be viewed as smooth approximation to SVMs.

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



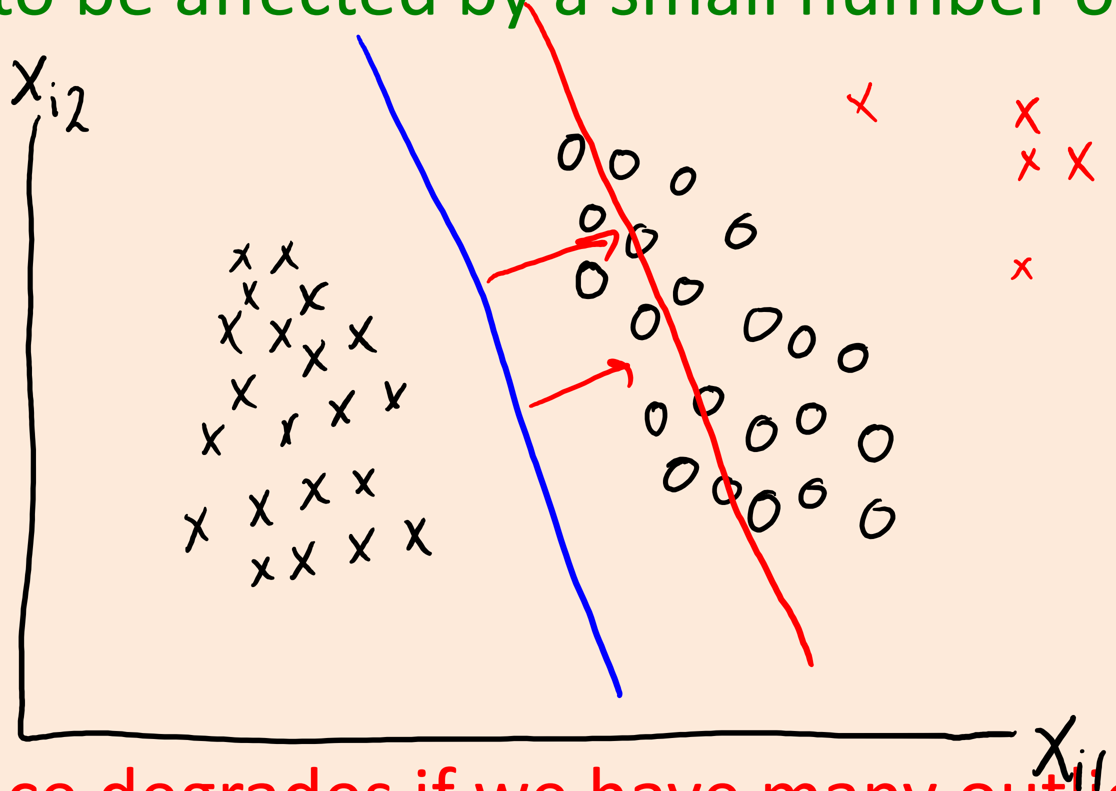
Robustness and Convex Approximations

- 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

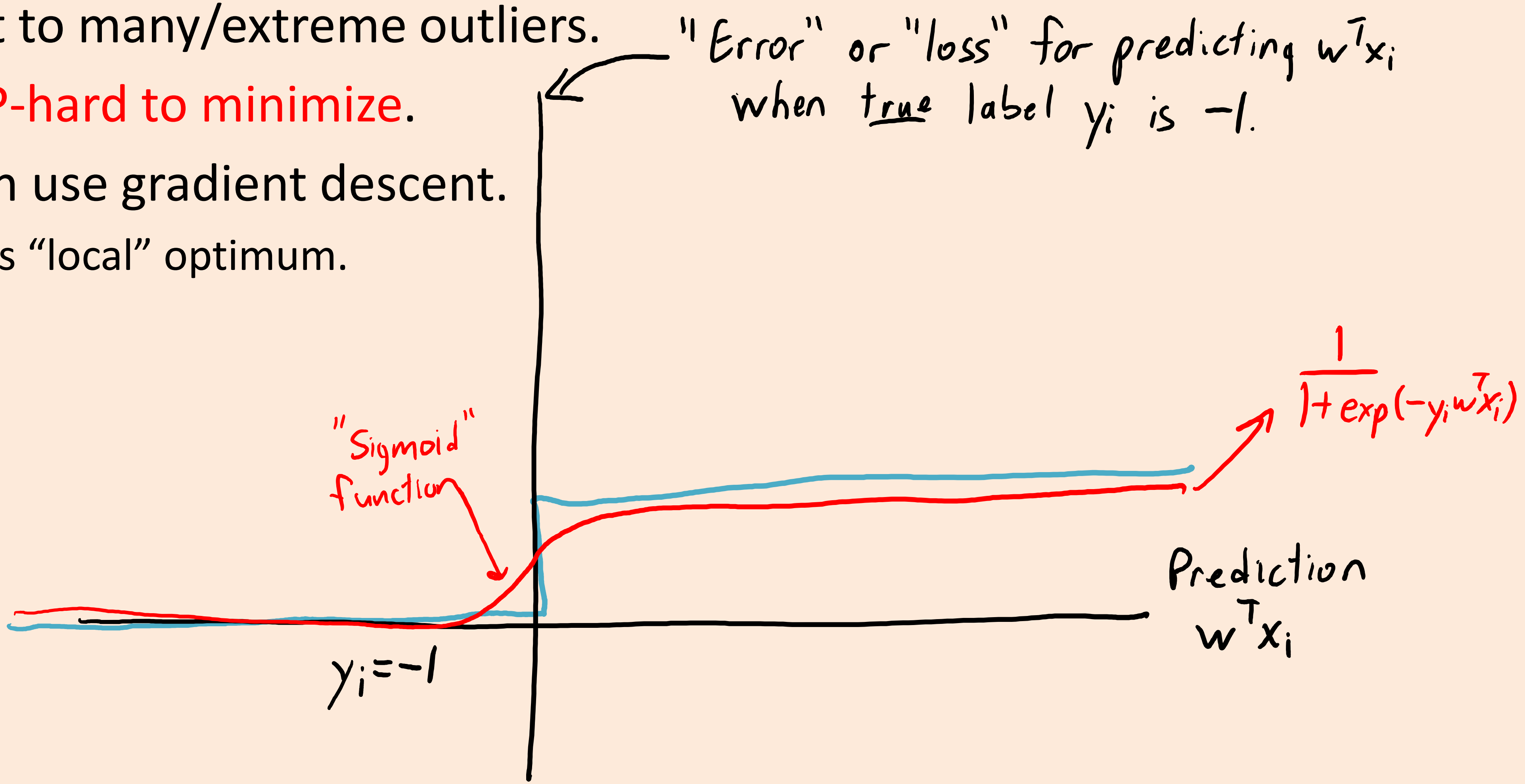
- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



- 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:

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

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

Feature Engineering

- “…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”
  – Pedro Domingos

- “Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.”
  – Andrew Ng

Feature Engineering

- Better features usually help more than a better model.

- Good features would ideally:
  - Capture most important aspects of problem.
  - Generalize to new scenarios.
  - Allow learning with few examples, be hard to overfit with many examples.

- There is a trade-off between simple and expressive features:
  - With simple features overfitting risk is low, but accuracy might be low.
  - With complicated features accuracy can be high, but so is overfitting risk.

Feature Engineering

- The best features may be dependent on the model you use.

- For counting-based methods like naïve Bayes and decision trees:
  - Need to address coupon collecting, but separate relevant “groups”.

- For distance-based methods like KNN:
  - Want different class labels to be “far”.

- For regression-based methods like linear regression:
  - Want labels to have a linear dependency on features.

Discretization for Counting-Based Methods

- For counting-based methods:
  - Discretization: turn continuous into discrete.

Age
23
23
22
25
19
22

< 20	>= 20, < 25	>= 25
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0

  - Counting age "groups" could let us learn more quickly than exact ages.
    - But we wouldn't do this for a distance-based method.

Standardization for Distance-Based Methods

- Consider features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard ‘unit’?
  – It doesn’t matter for counting-based methods.
- It matters for distance-based methods:
    - KNN will focus on large values more than small values.
    - Often we “standardize” scales of different variables (e.g., convert everything to grams).

Non-Linear Transformations for Regression-Based

- Non-linear feature/label transforms can make things more linear:
  - Polynomial, exponential/logarithm, sines/cosines, RBFs.

www.google.com/finance

Discussion of Feature Engineering

- The best feature transformations are application-dependent.
  - It's hard to give general advice.

- My advice: ask the domain experts.
  - Often have idea of right discretization/standardization/transformation.

- If no domain expert, cross-validation will help.
  - Or if you have lots of data, use deep learning methods from Part 5.

Ordinal Features

- Categorical features with an ordering are called ordinal features.

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

→

Rating
2
5
4
4
1
4
3

- If using decision trees, makes sense to replace with numbers.
  - Captures ordering between the ratings.
  - A rule like (rating ≥ 3) means (rating ≥ Good), which make sense.

Ordinal Features

- If using linear models, this would assumes ratings are equally spaced.
  – The difference between "Bad" and "Medium" is similar to the distance between "Good" and "Very Good".
- An alternative that preserves ordering with binary features:

Rating
Bad
Very Good
Good
Good
Very Bad
Good
Medium

→

≥ Bad	≥ Medium	≥ Good	Very Good
1	0	0	0
1	1	1	1
1	1	1	0
1	1	1	0
0	0	0	0
1	1	1	0
1	1	0	0

- Regression weight w_{medium} represents:
  – "How much medium changes prediction over bad".

“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.