CPSC 340: Machine Learning and Data Mining

More Linear Classifiers

Fall 2017

Admin

- Assignment 3:
  - Due Friday of next week.

- Midterm:
  - Can view your exam during instructor office hours next week, or after class this/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 $\text{sign}(w^T x_i)$

$y_i \to +1$, -1

- Problems with existing errors:
  - If $y_i = +1$ and $w^T x_i = +100$, then squared error $(w^T x_i - y_i)^2$ is huge.
  - Hard to minimize training error ("0-1 loss") in terms of 'w'.

- Motivates convex approximations to 0-1 loss…

Degenerate Convex Approximation to 0-1 Loss

- If $y_i = +1$, we get the label right if $w^T x_i > 0$.
- If $y_i = -1$, we get the label right if $w^T x_i < 0$, or equivalently $-w^T x_i > 0$.
- So "classifying 'i' correctly" is equivalent to having $y_i w^T x_i > 0$.

- One possible convex approximation to 0-1 loss:
  - Minimize how much this constraint is violated.

If $y_i w^T x_i > 0$ then you get an "error" of 0.
If $y_i w^T x_i < 0$ then you get an "error" of $-y_i w^T x_i$

$\rightarrow$ So the "error" is given by $\underbrace{\max\{0, -y_i w^T x_i\}}_{\max\{\text{constant}, \text{linear}\} \Rightarrow \text{convex}}$

Degenerate Convex Approximation to 0-1 Loss

- Our convex approximation of the error for one example is:

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

- We could train by minimizing sum over all examples:

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

- But this has a degenerate solution:
  – We have f(0) = 0, and this is the lowest possible value of ‘f’.
- There are two standard fixes: hinge loss and logistic loss.

Hinge Loss

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

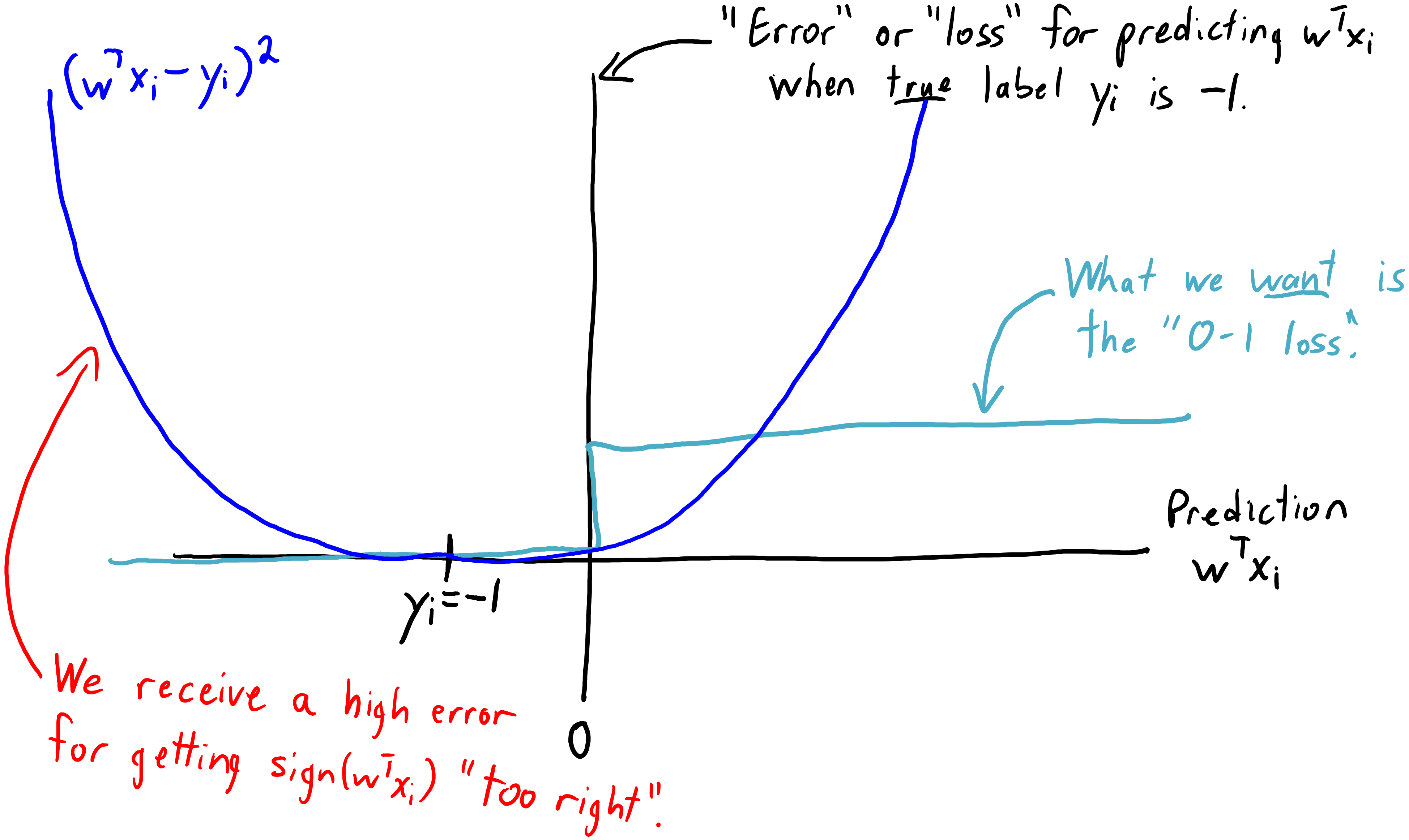
  (the "1" is arbitrary: we could make ||w|| bigger/smaller to use any positive constant)

- 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.

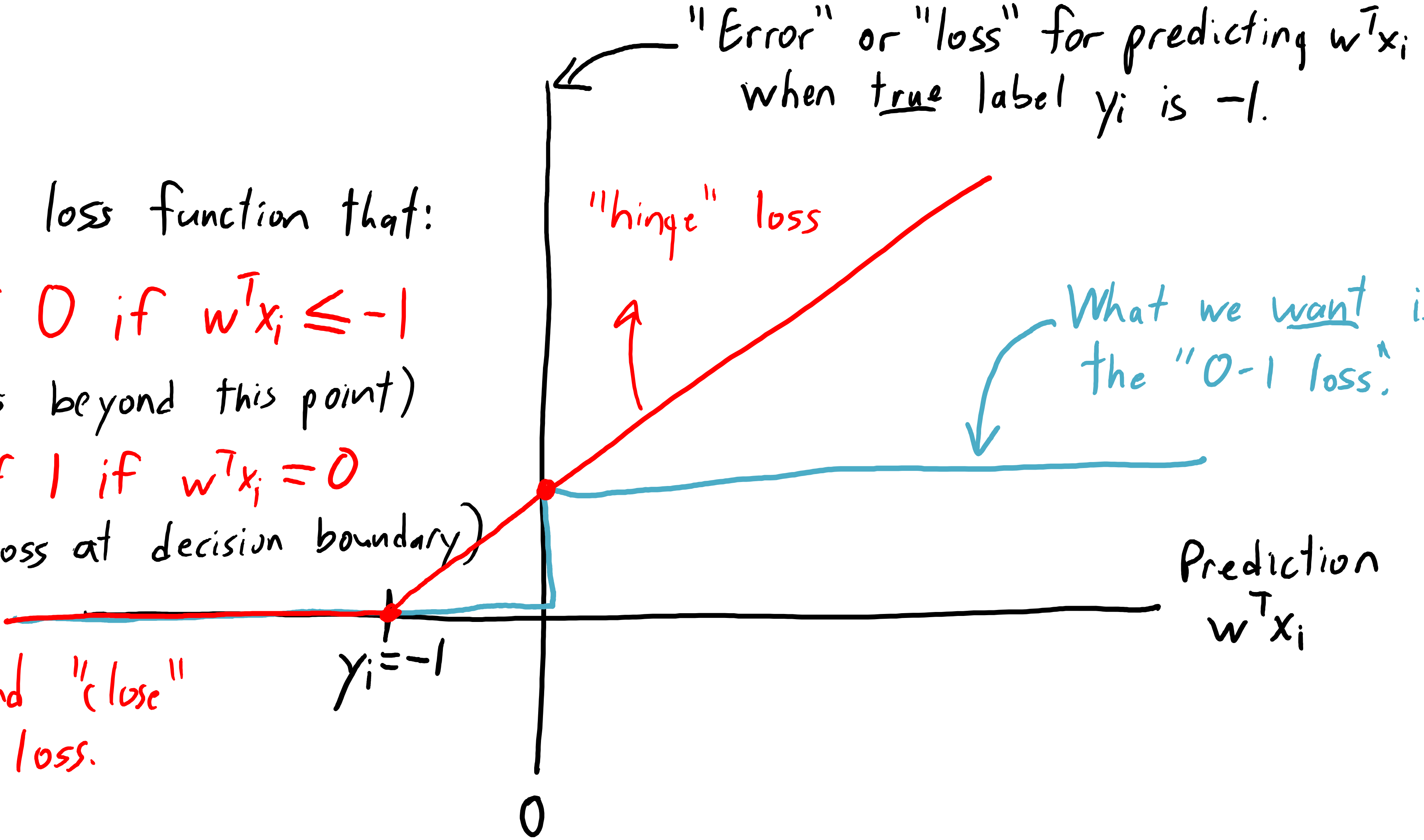
Hinge Loss: Convex Approximation to 0-1 Loss



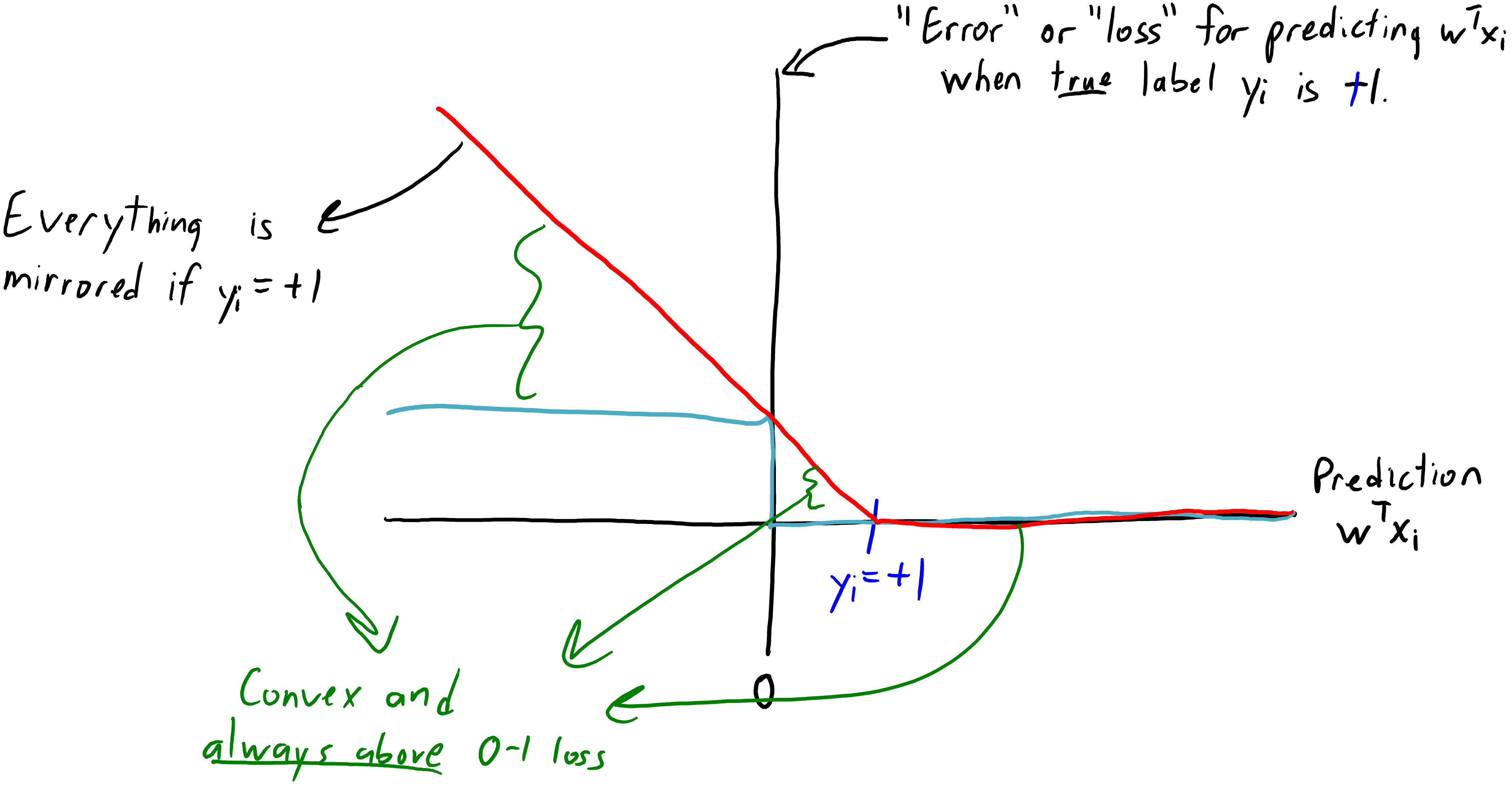
Hinge Loss: Convex Approximation to 0-1 Loss

Let's choose a loss function that:

1. Has error of 0 if $w^T x_i \leq -1$
   (no "bad" errors beyond this point)
2. Has a loss of 1 if $w^T x_i = 0$
   (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.
    - Finds a perfect linear classifier if one exists.
- Support vector machine (SVM) is hinge loss with L2-regularization.

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

- SVMs can also be viewed as "maximizing the margin" (later in lecture).

Logistic Loss

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

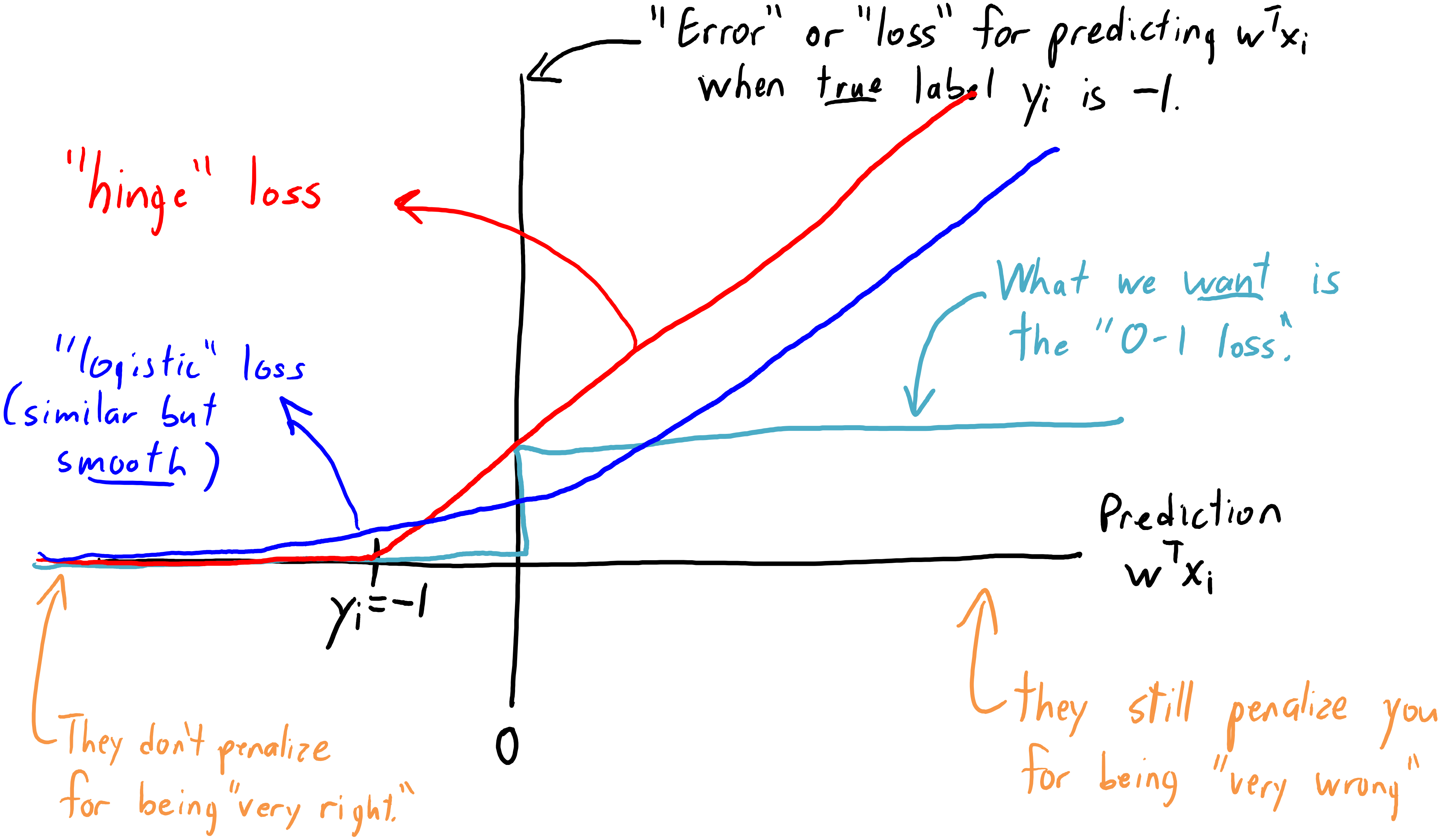
$$\max\{0, -y_i w^T x_i\} \approx \log(\underbrace{\exp(0)}_{1} + \exp(-y_i w^T x_i))$$

- 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).
    - Testing 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.

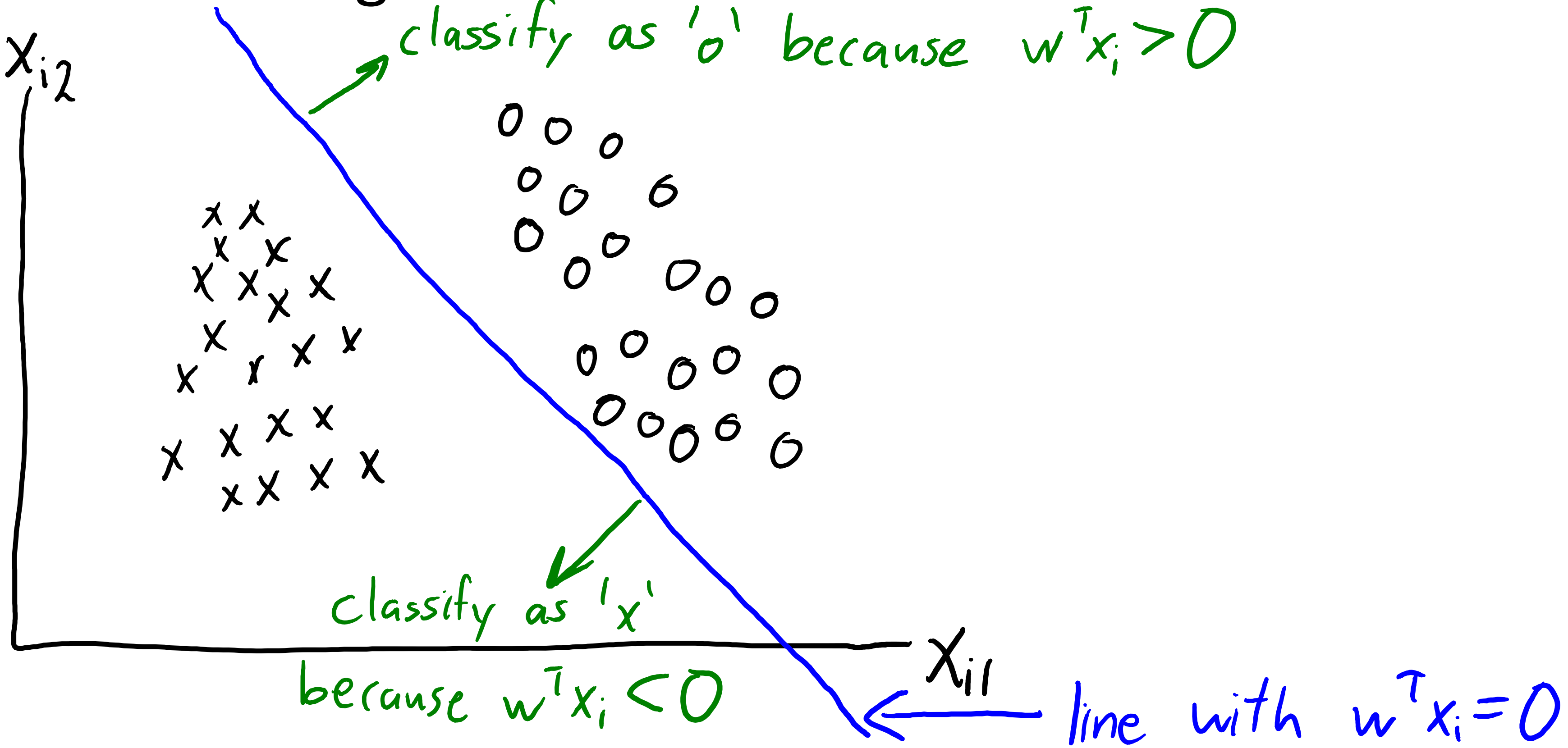
Comparison of “Black Box” Classifiers

- 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).

Last Time: Linear Classifiers

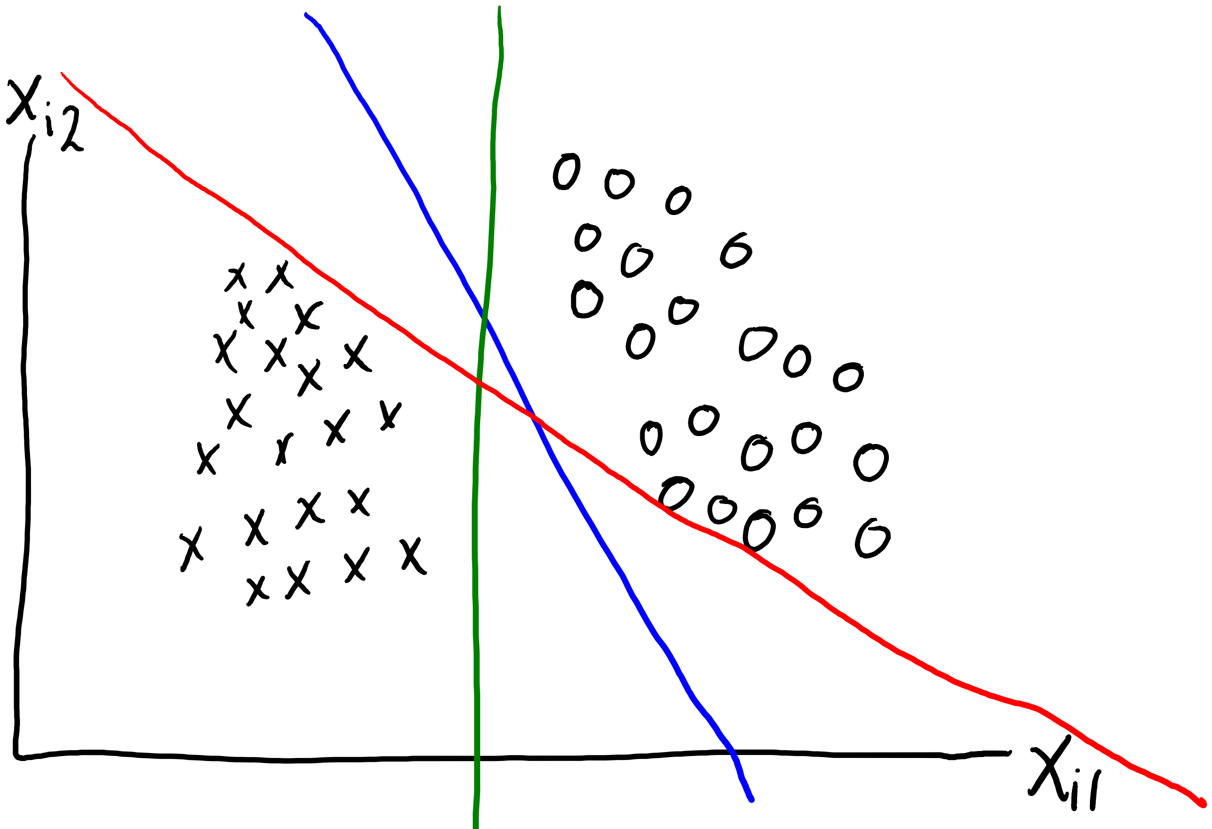
- 2D Visualization of linear regression for classification:



- "Linearly separable": a perfect linear classifier exists.

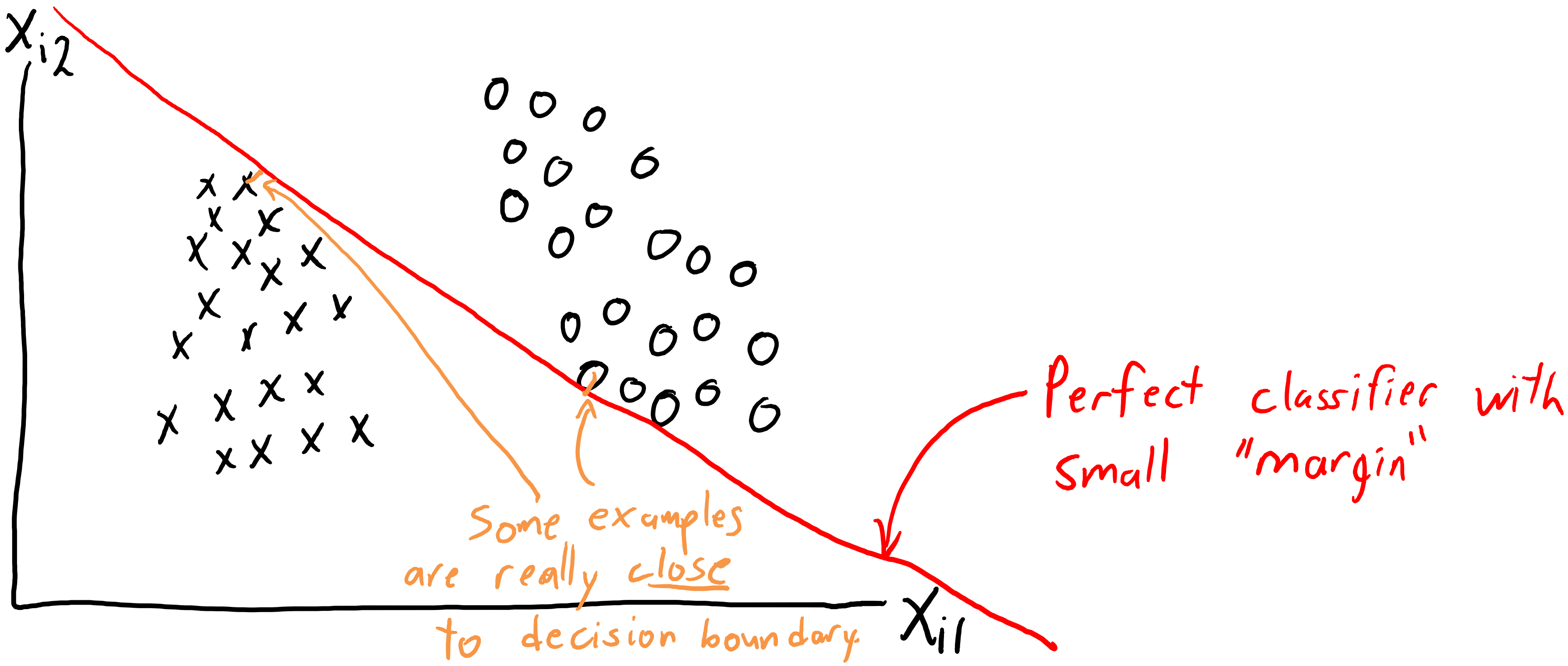
Maximum-Margin Classifier

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



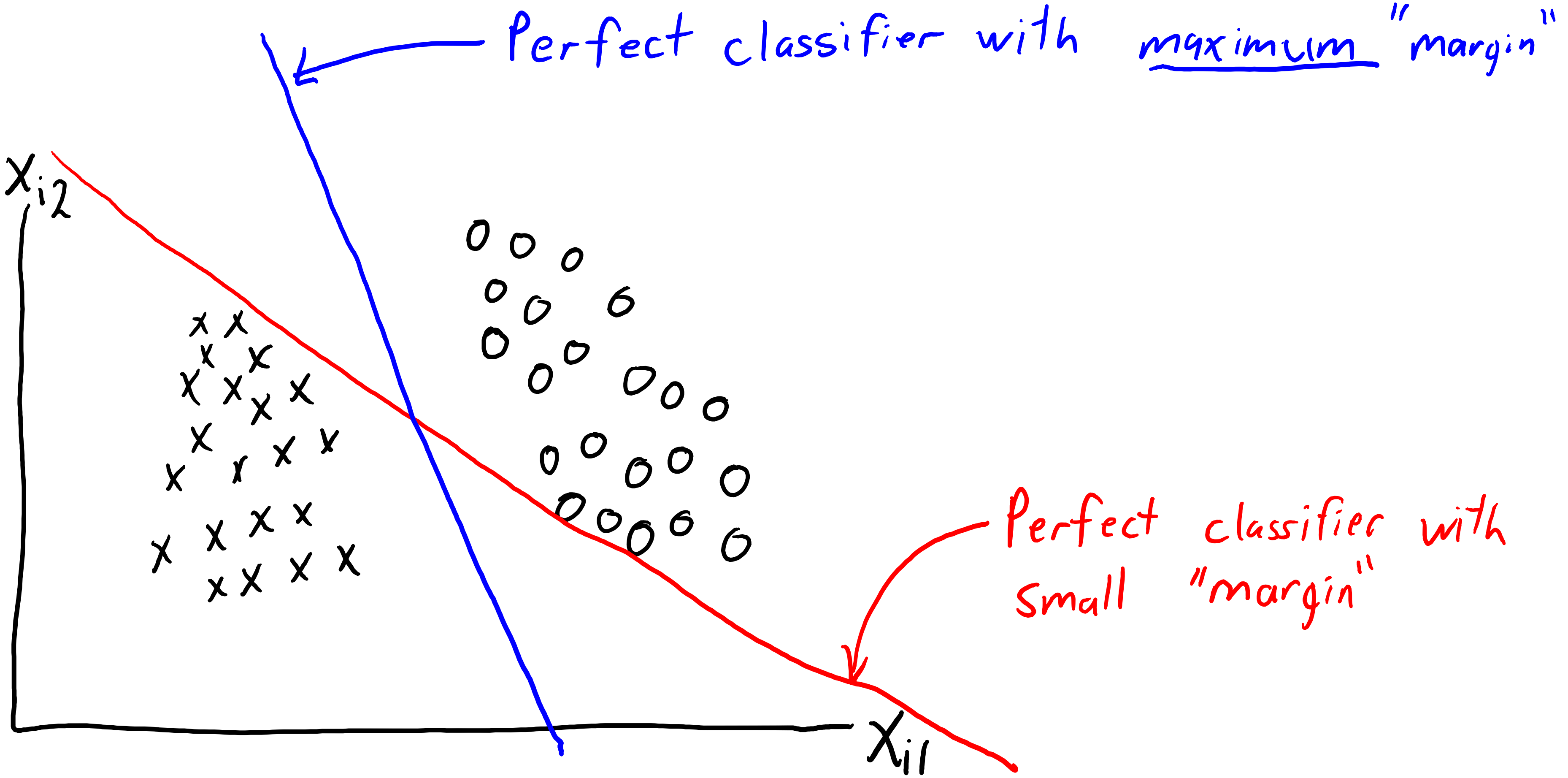
Maximum-Margin Classifier

- 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.

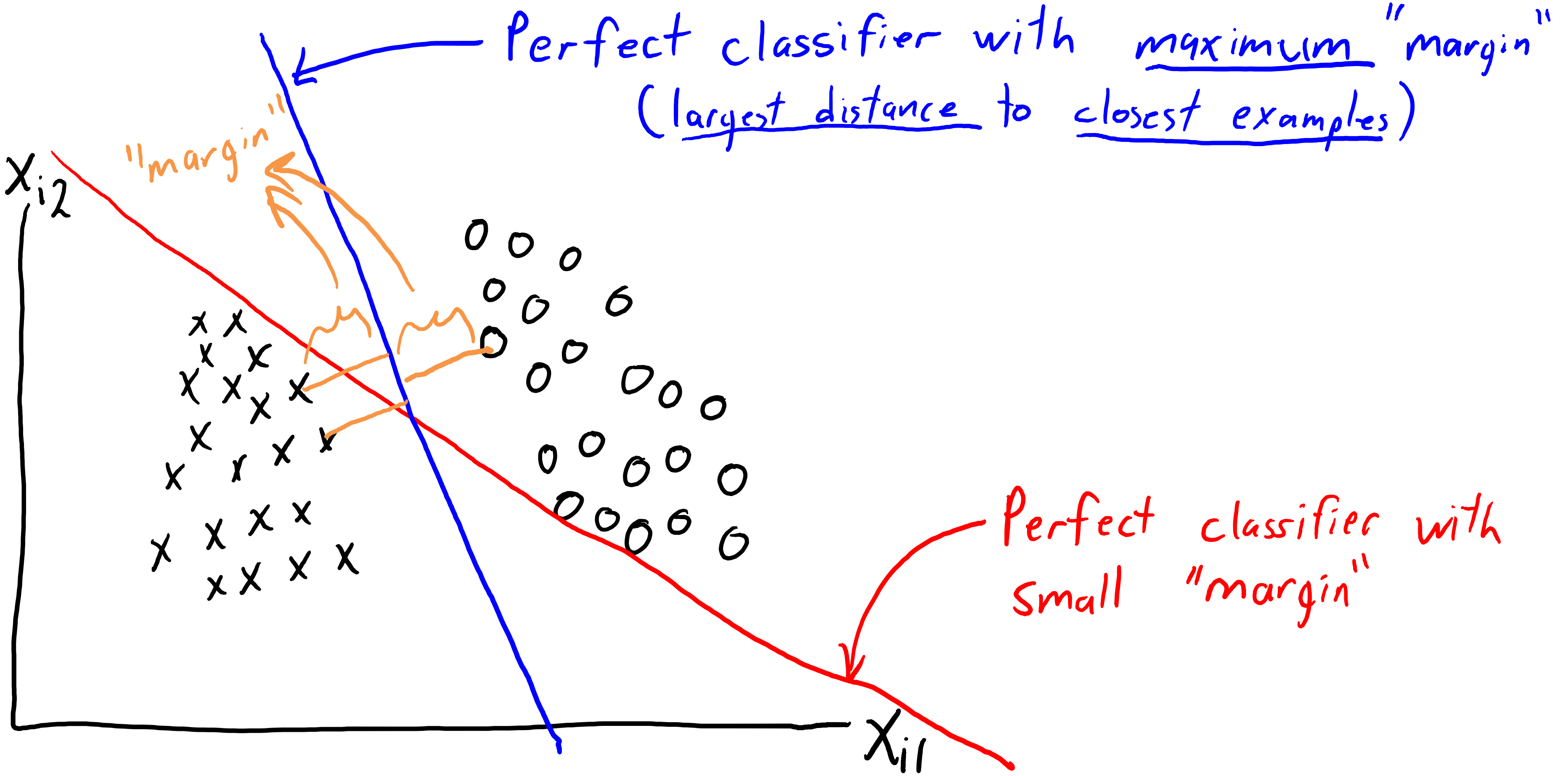


Maximum-Margin Classifier

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

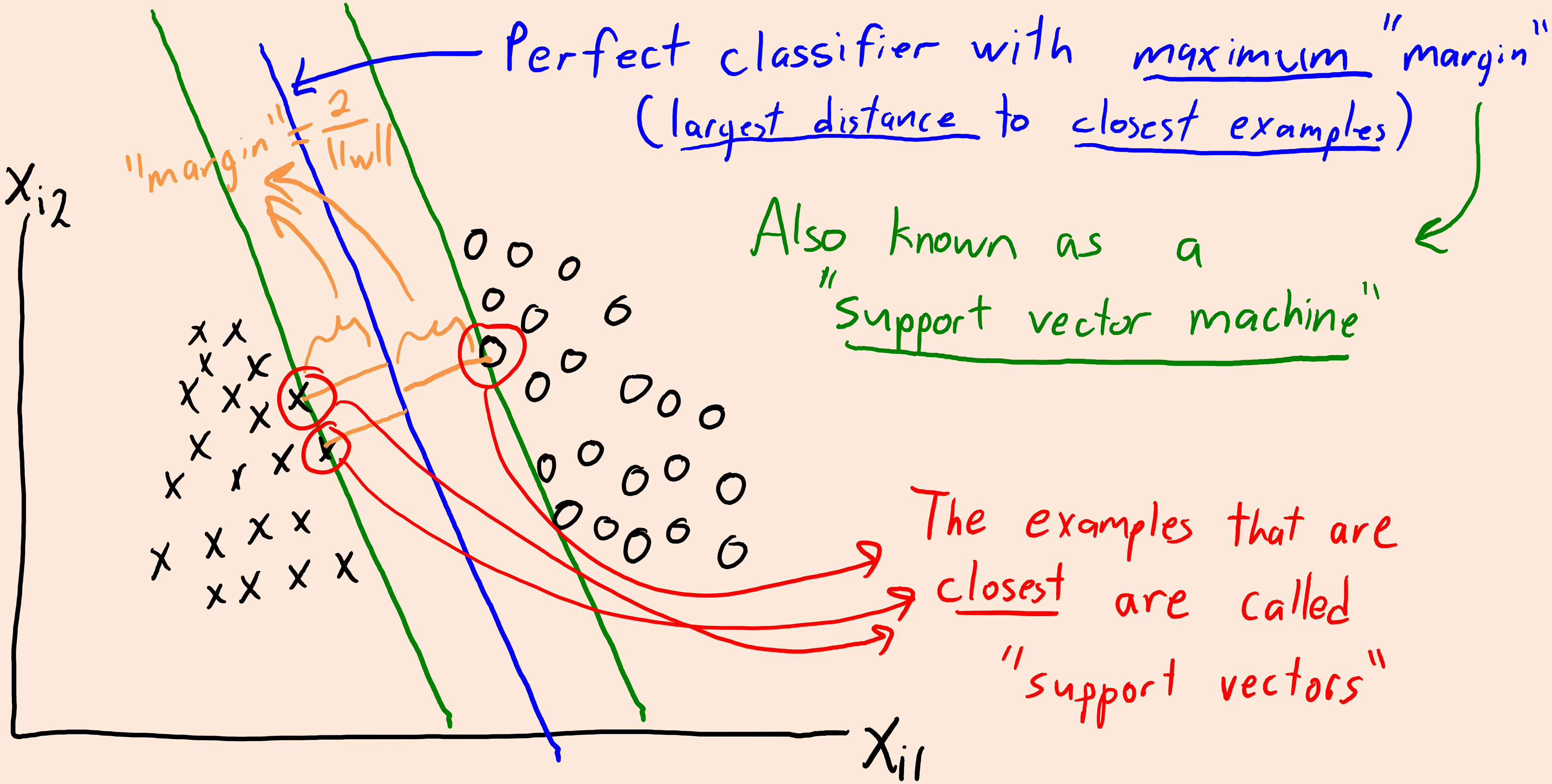
Why maximize margin?

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



Maximum-Margin Classifier

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



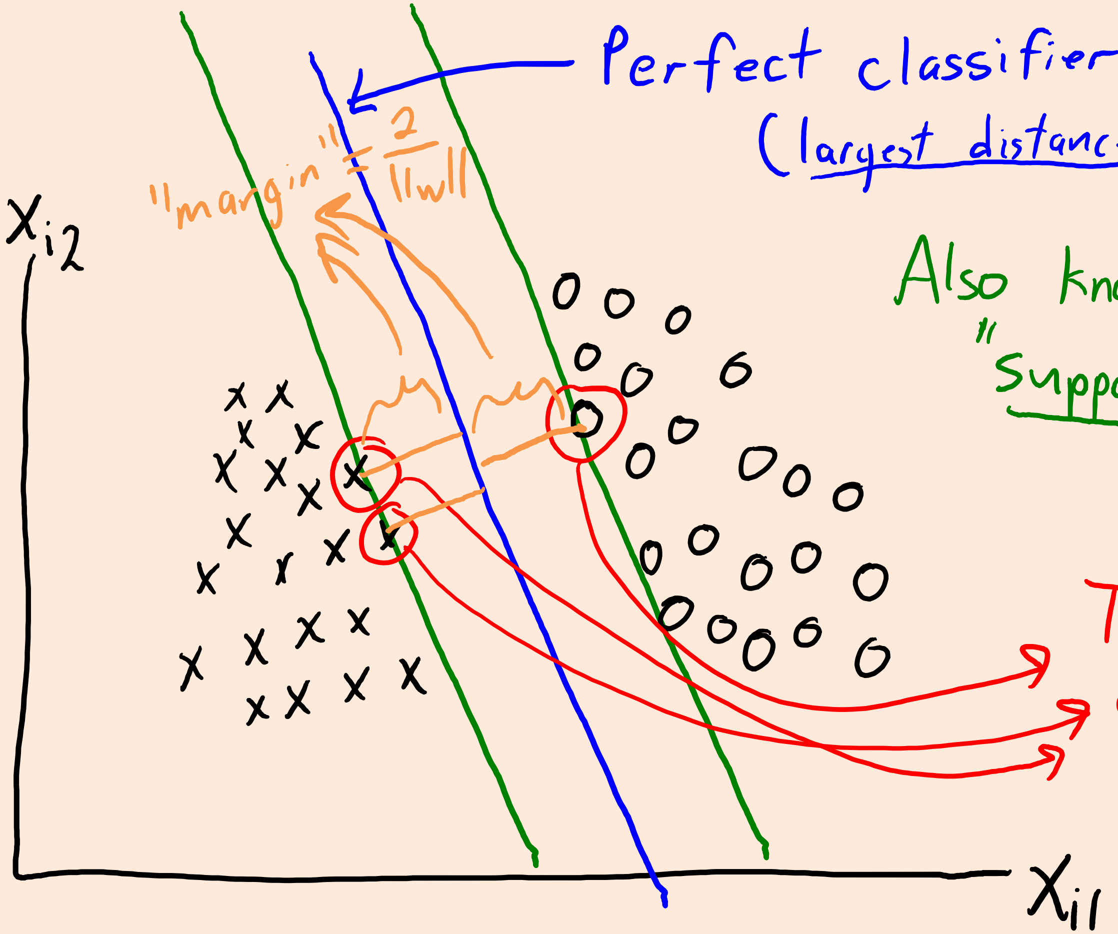
Maximum-Margin Classifier

- 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"

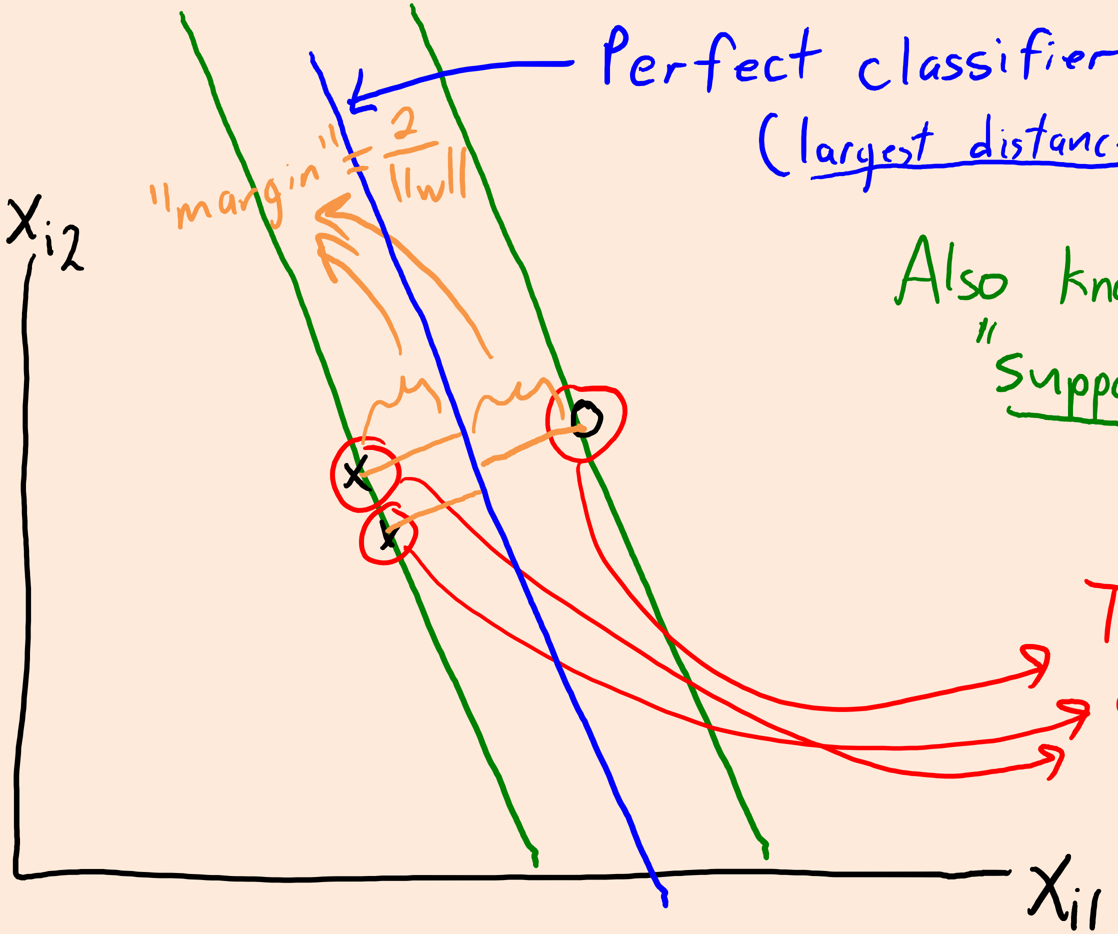
Maximum-Margin Classifier

- 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 "<u>violate constraint</u>" by $1 + w^T x_i$

⋮

Constraint violation is the <u>hinge loss</u>.

Support Vector Machines

- Try to maximizing margin and also minimizing constraint violation:

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

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.

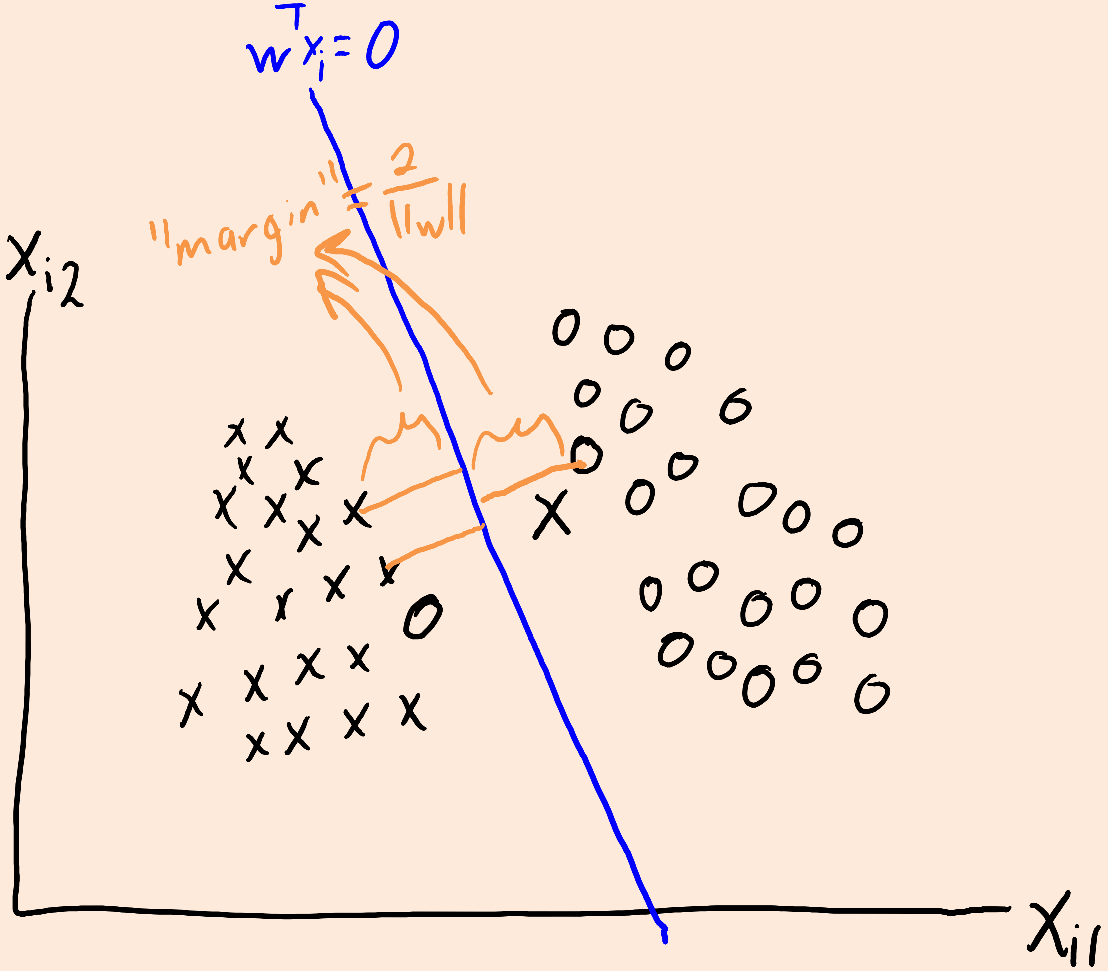
- We typically control margin/violation trade-off with parameter "λ":

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

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

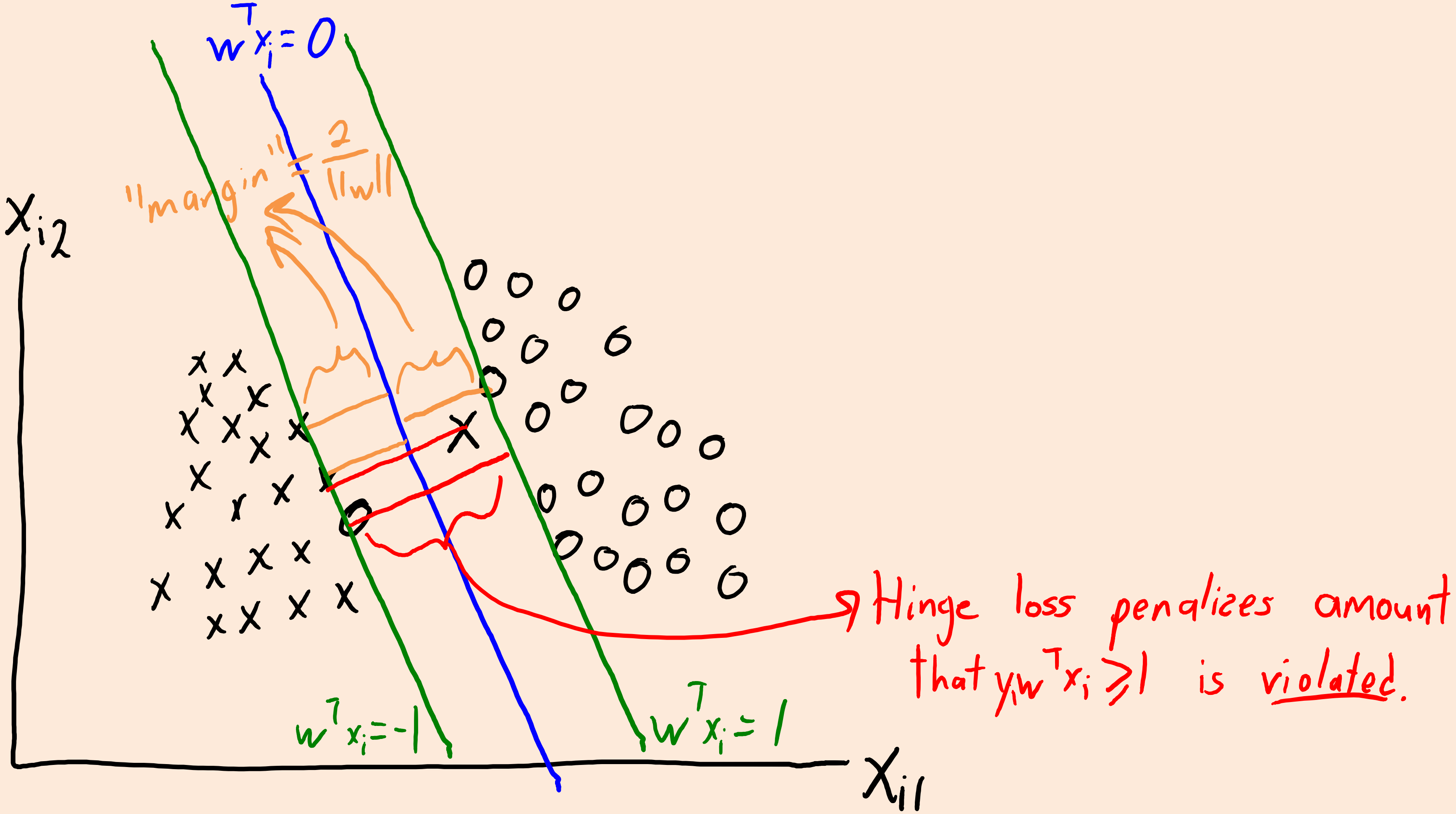
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:



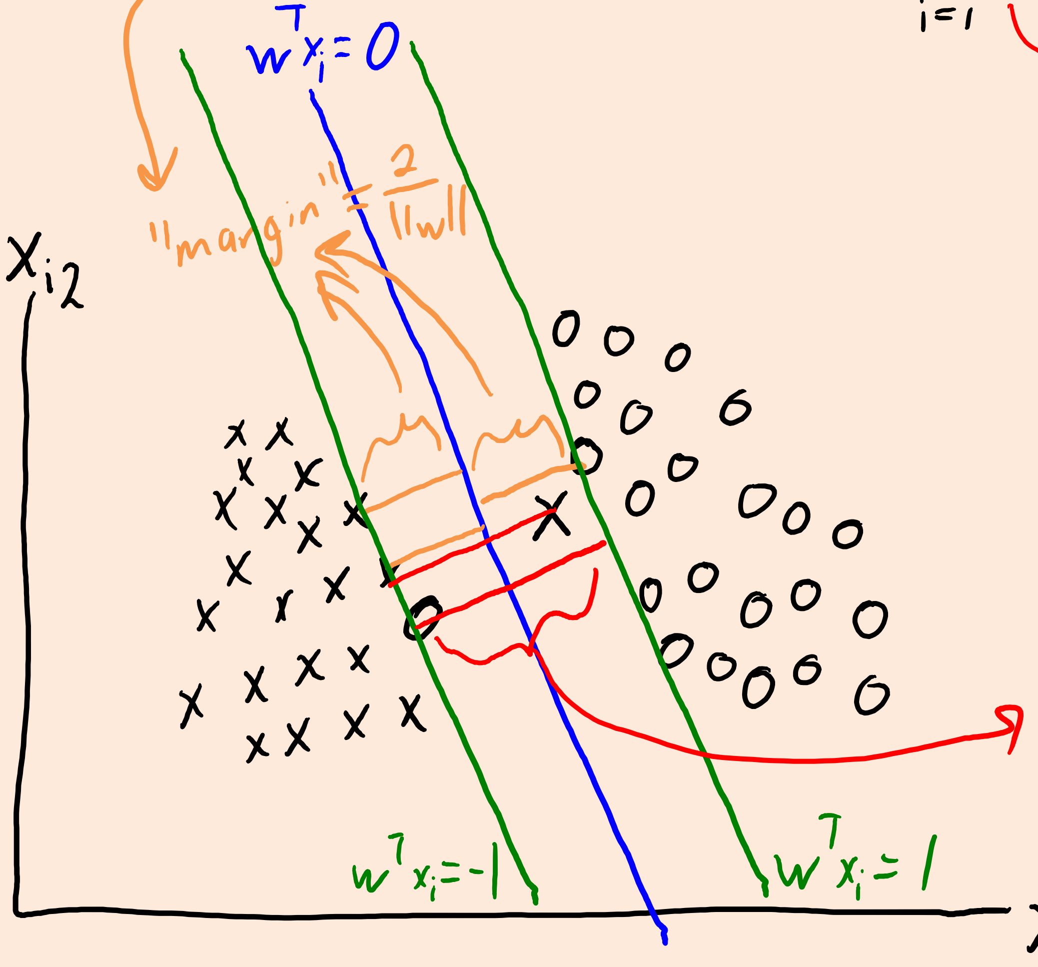
Support Vector Machines for Non-Separable

- 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.



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

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

Support Vector Machines for Non-Separable

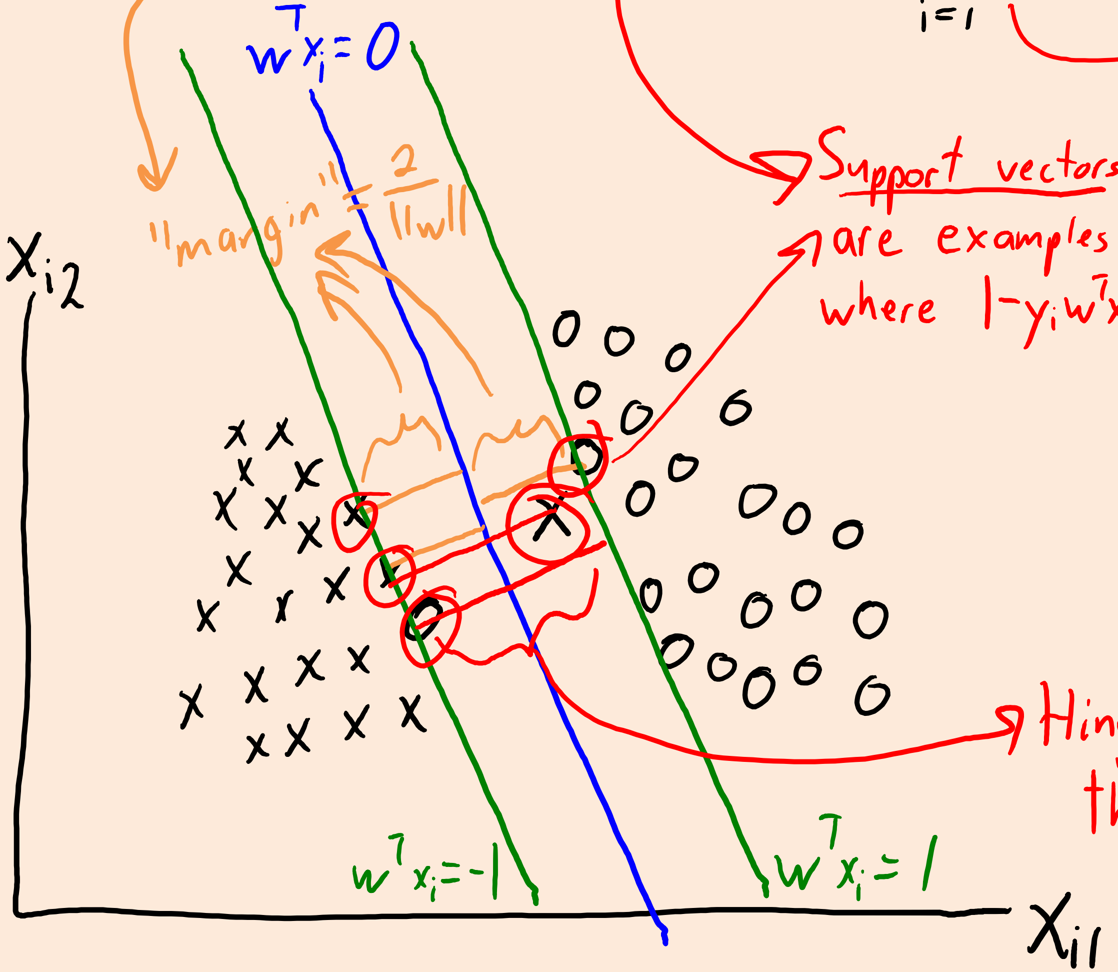
- Non-separable case:

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

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.

λ 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.

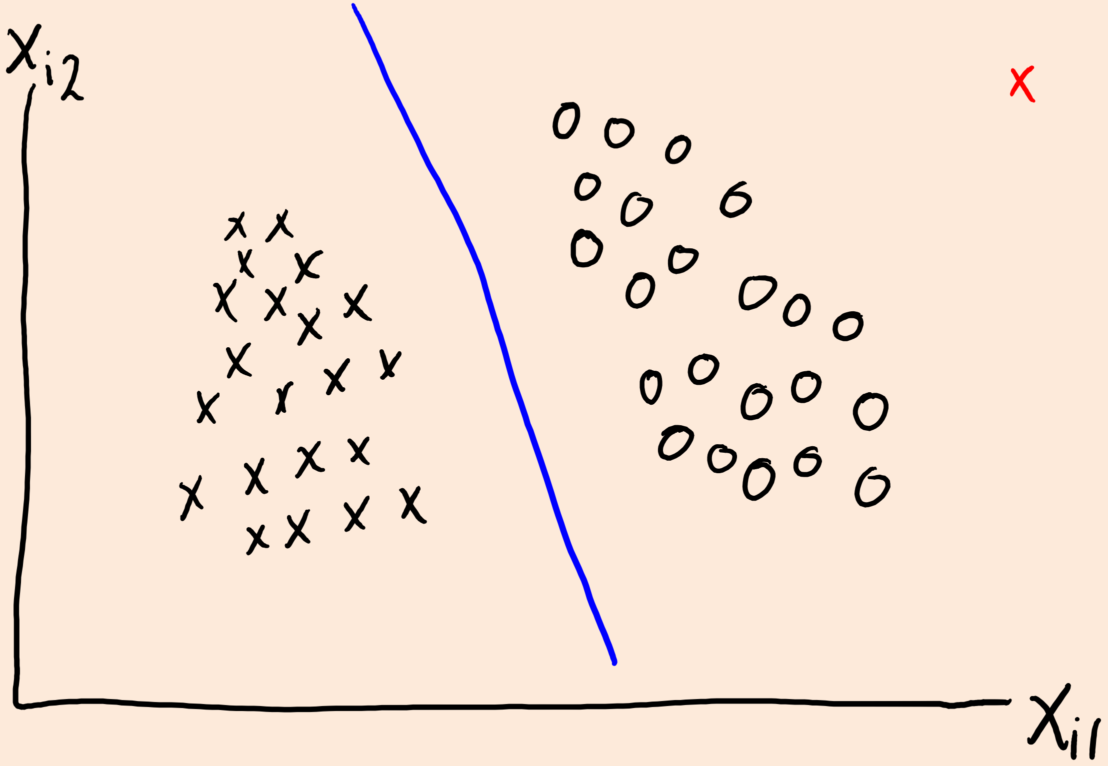
Summary

- 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".
- 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).

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

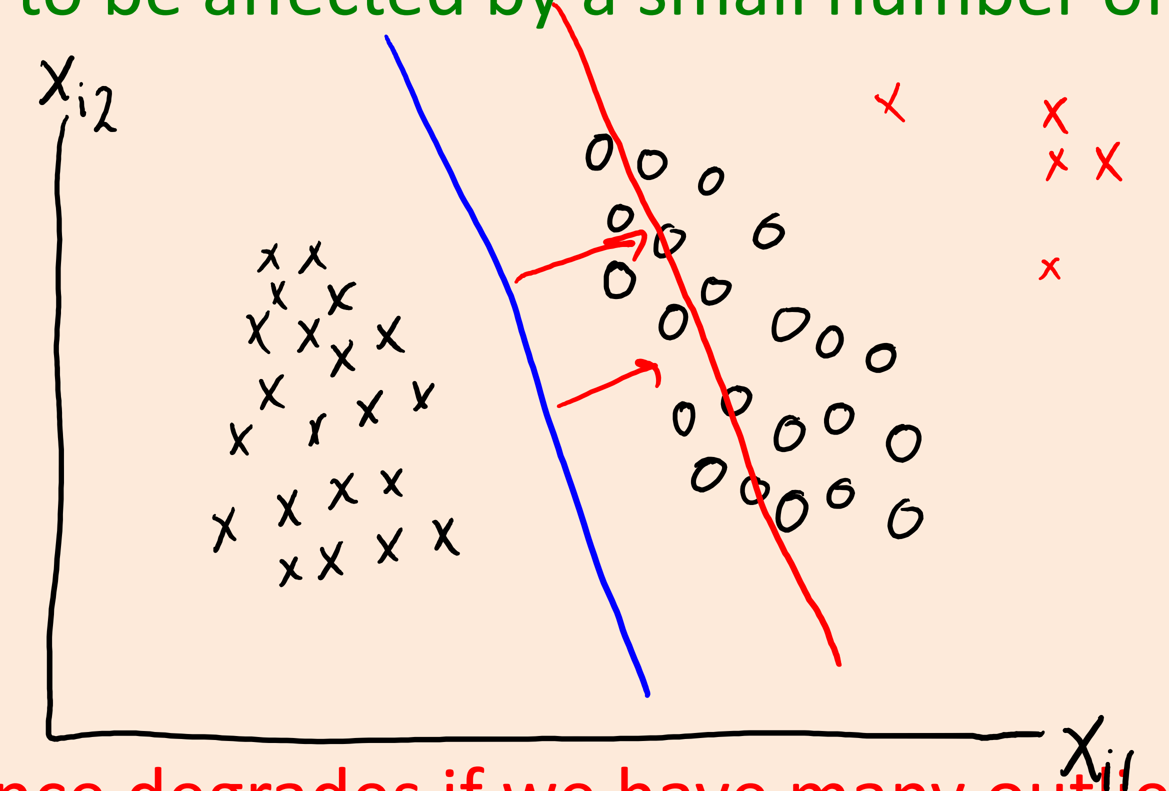
Robustness and Convex Approximations

- 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

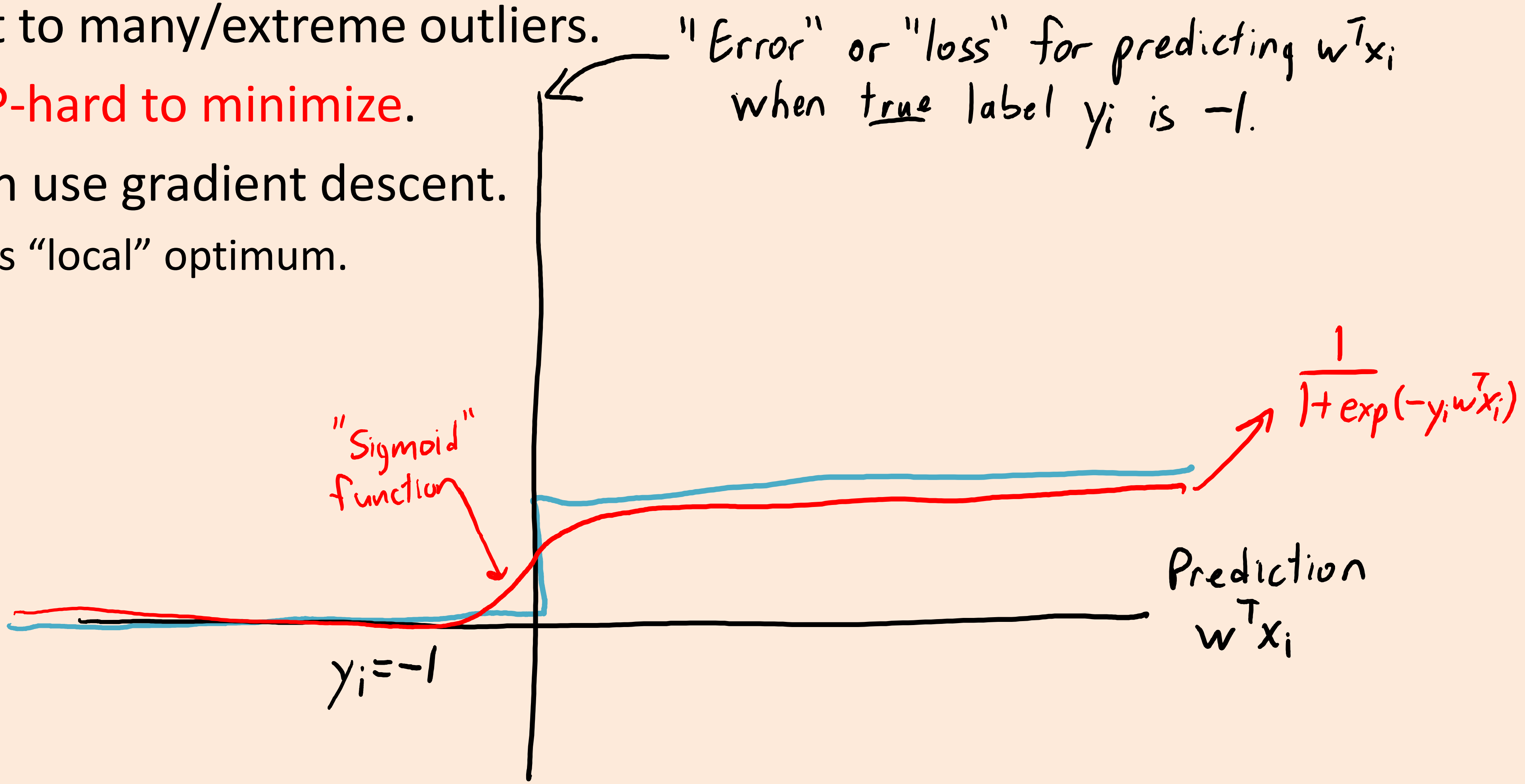
- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



- 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 λ.