CPSC 340:
Machine Learning and Data Mining

Linear Classifiers: loss functions

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Last Time: Classification using Regression

- Binary classification using sign of linear models:

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

$+1 \quad -1$

- We talked about predictions and the interpretation of ‘w’
- But what loss function do we use to learn ‘w’?

Can we just use least squares??

- Consider training by minimizing squared error with these y_i:

$$f(w) = \frac{1}{2}\|Xw - y\|^2$$

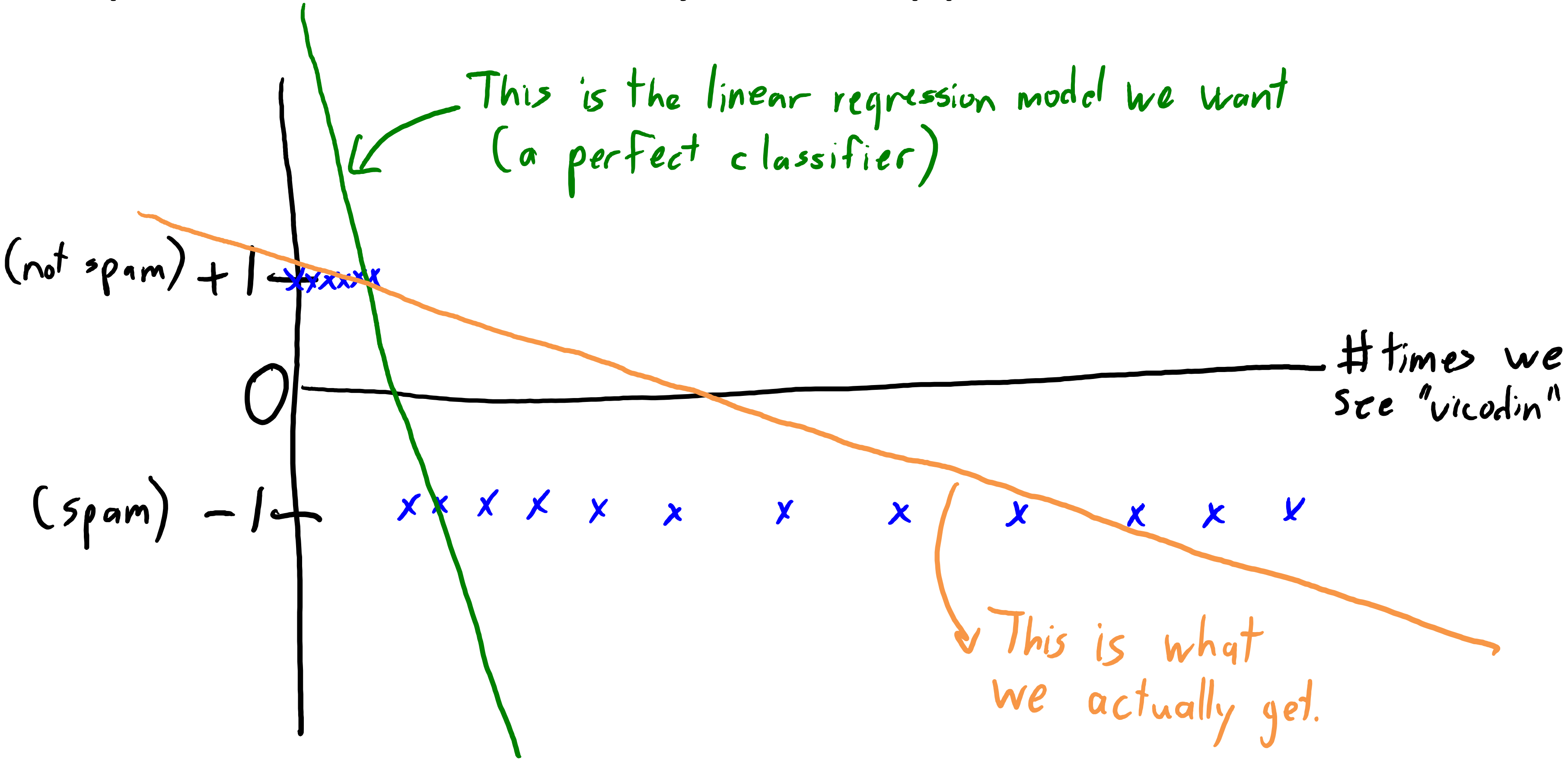
$$y = \begin{bmatrix} +1 \\ +1 \\ -1 \\ +1 \\ -1 \end{bmatrix}$$

- If we predict $w^Tx_i = +0.9$ and $y_i = +1$, error is small: $(0.9 - 1)^2 = 0.01$.
- If we predict $w^Tx_i = -0.8$ and $y_i = +1$, error is big: $(-0.8 - 1)^2 = 3.24$.
- If we predict $w^Tx_i = +100$ and $y_i = +1$, error is huge: $(100 - 1)^2 = 9801$.

- Least squares penalized for being "too right".
  - +100 has the right sign, so the error should be zero.

Can we just use least squares??

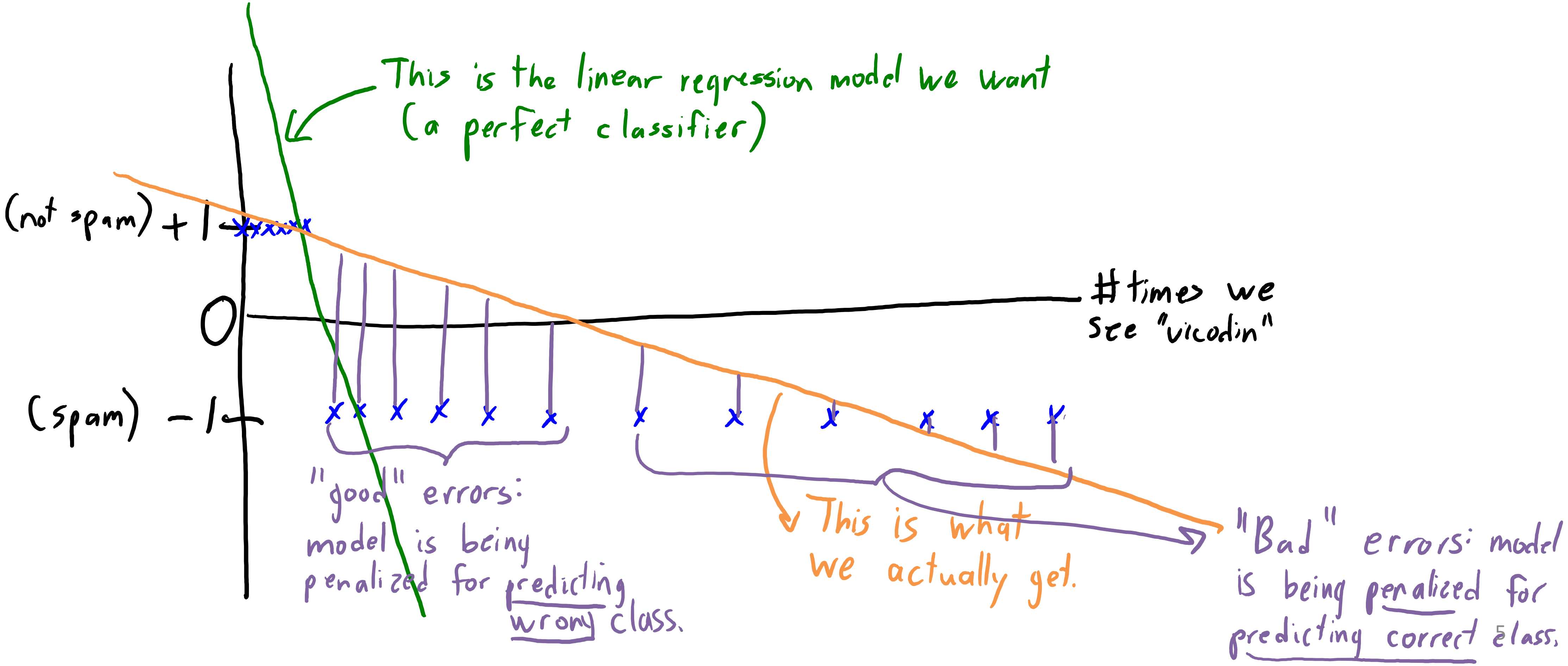
- Least squares behaves weirdly when applied to classification:



- Make sure you understand why the green line achieves 0 training error.

Can we just use least squares??

- What went wrong?
  - “Good” errors vs. “bad” errors.

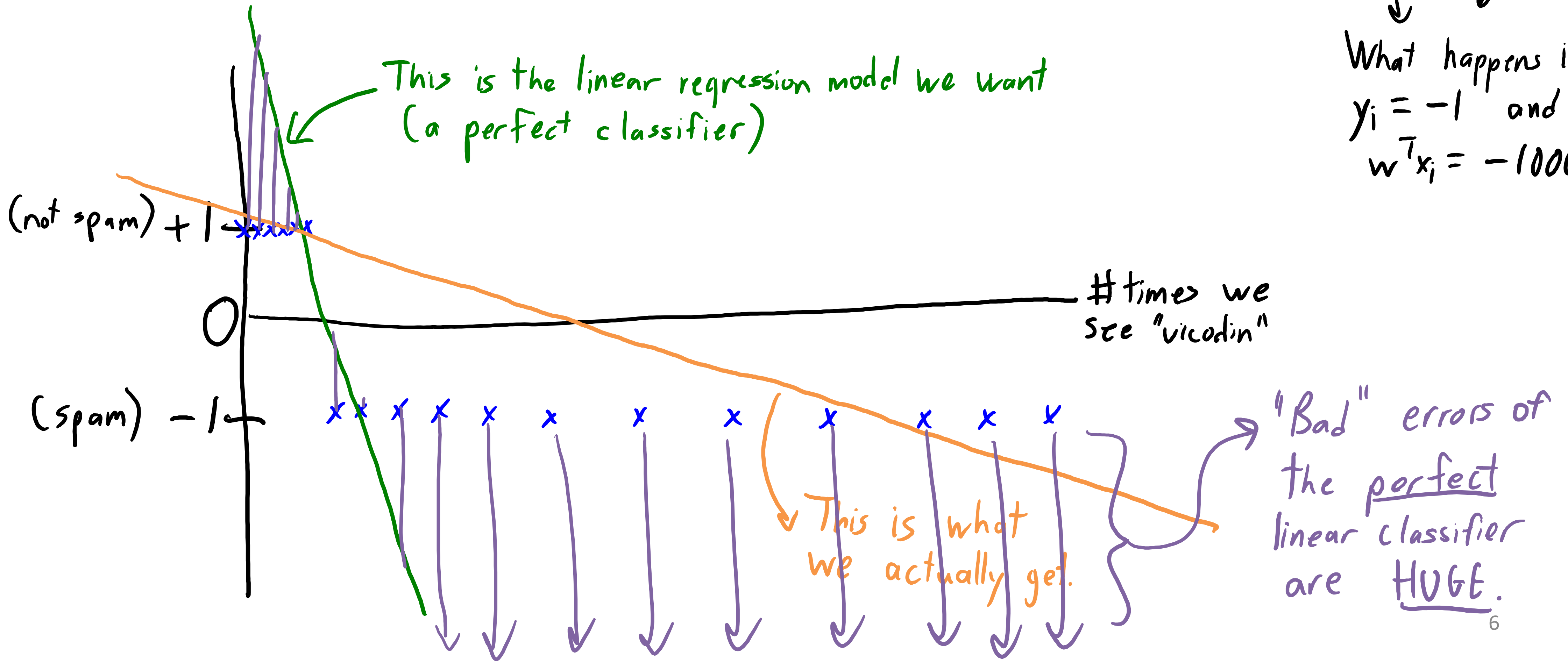


Can we just use least squares??

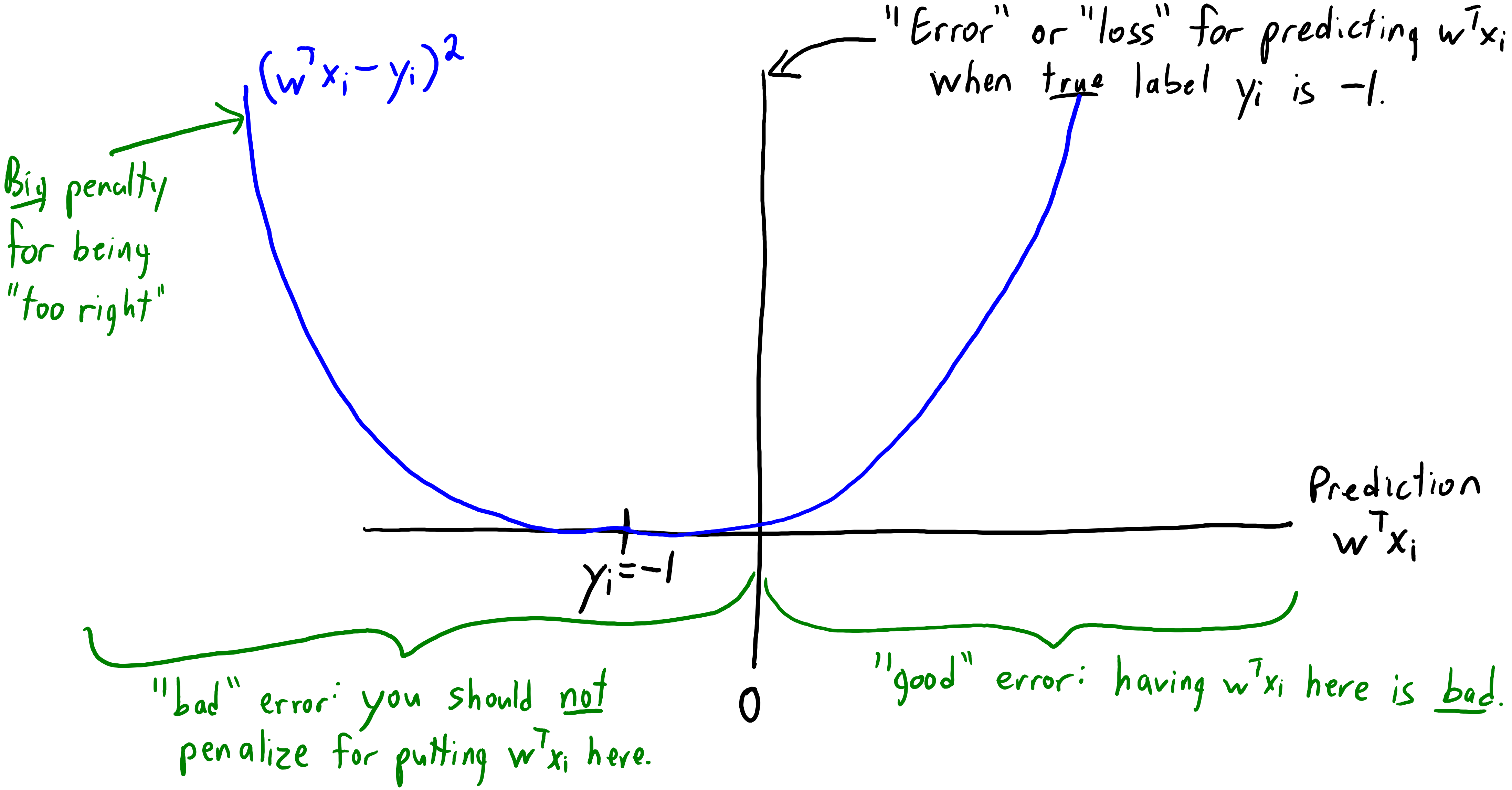
- What went wrong?
  - "Good" errors vs. "bad" errors.

$$f(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

What happens if $y_i = -1$ and $w^T x_i = -1000$?



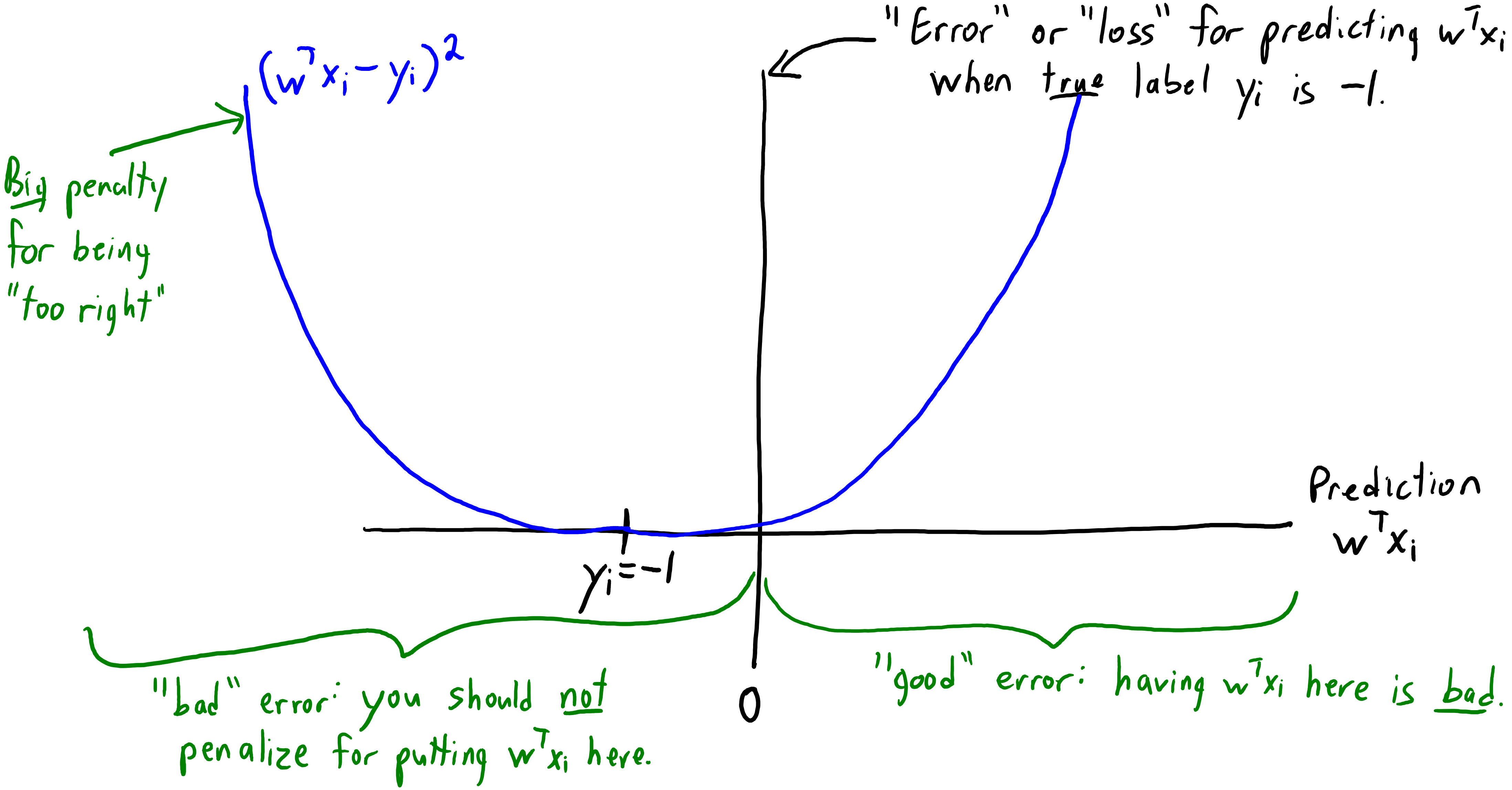
Comparing Loss Functions



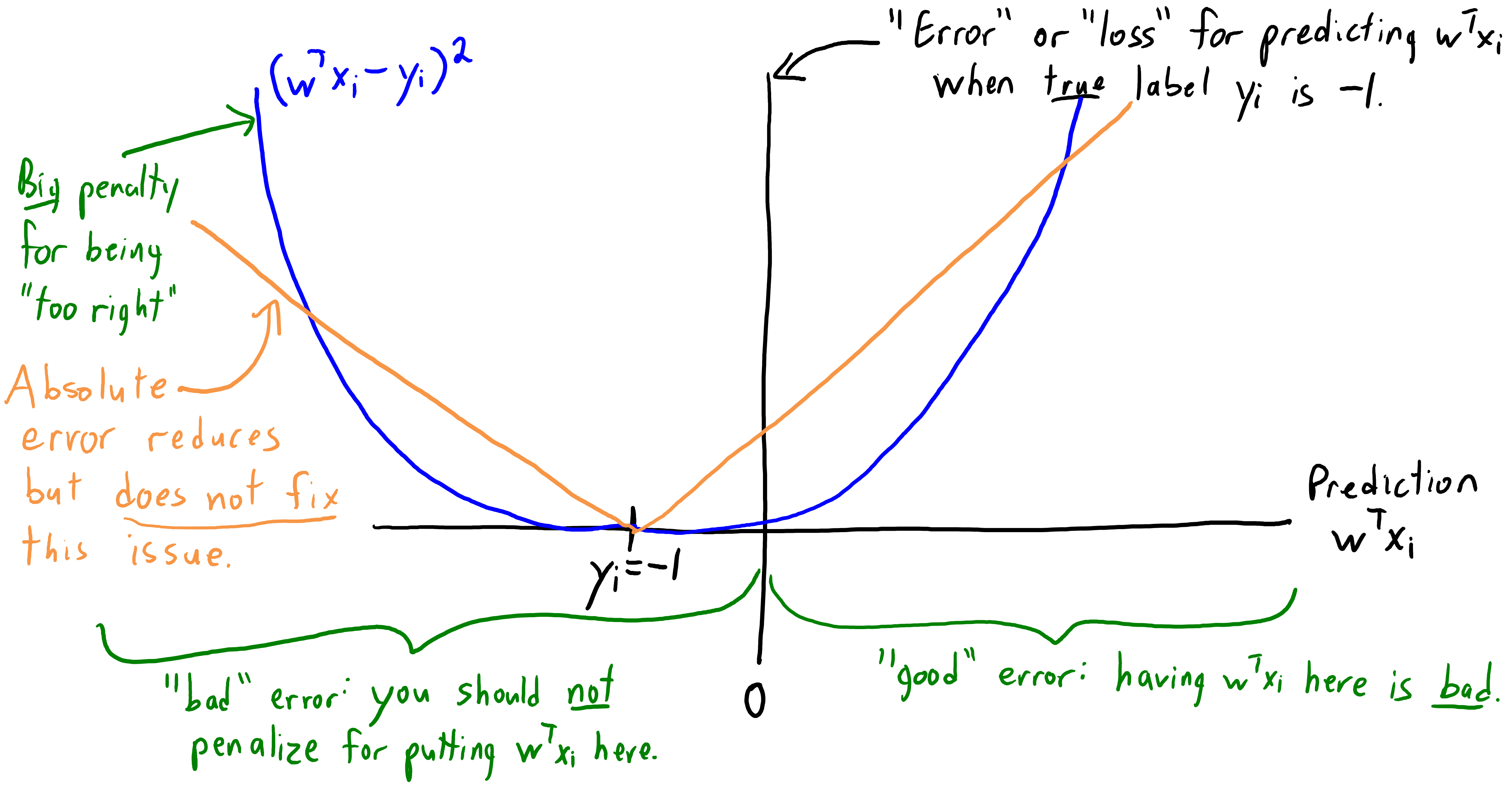
Thoughts on the previous (and next) slide

- We are now plotting the loss vs. the predicted $w^\top x_i$.
  - This is totally different from plotting in the data space (y_i vs. x_i).
- The loss is a sum over training examples.
  - We're plotting the individual loss for a particular training example.
  - In the figure, this example has label $y_i = -1$ so the loss is centered at -1. (The plot would be mirrored in the case of $y_i = +1$.)
    - We only need to show one case or the other to get our point across.
  - Note that with regular linear regression the output y_i could be any number and thus the parabola could be centred anywhere. But here we've restricted ourselves to y_i={-1,+1}.
- (The next slide is the same as the previous one)

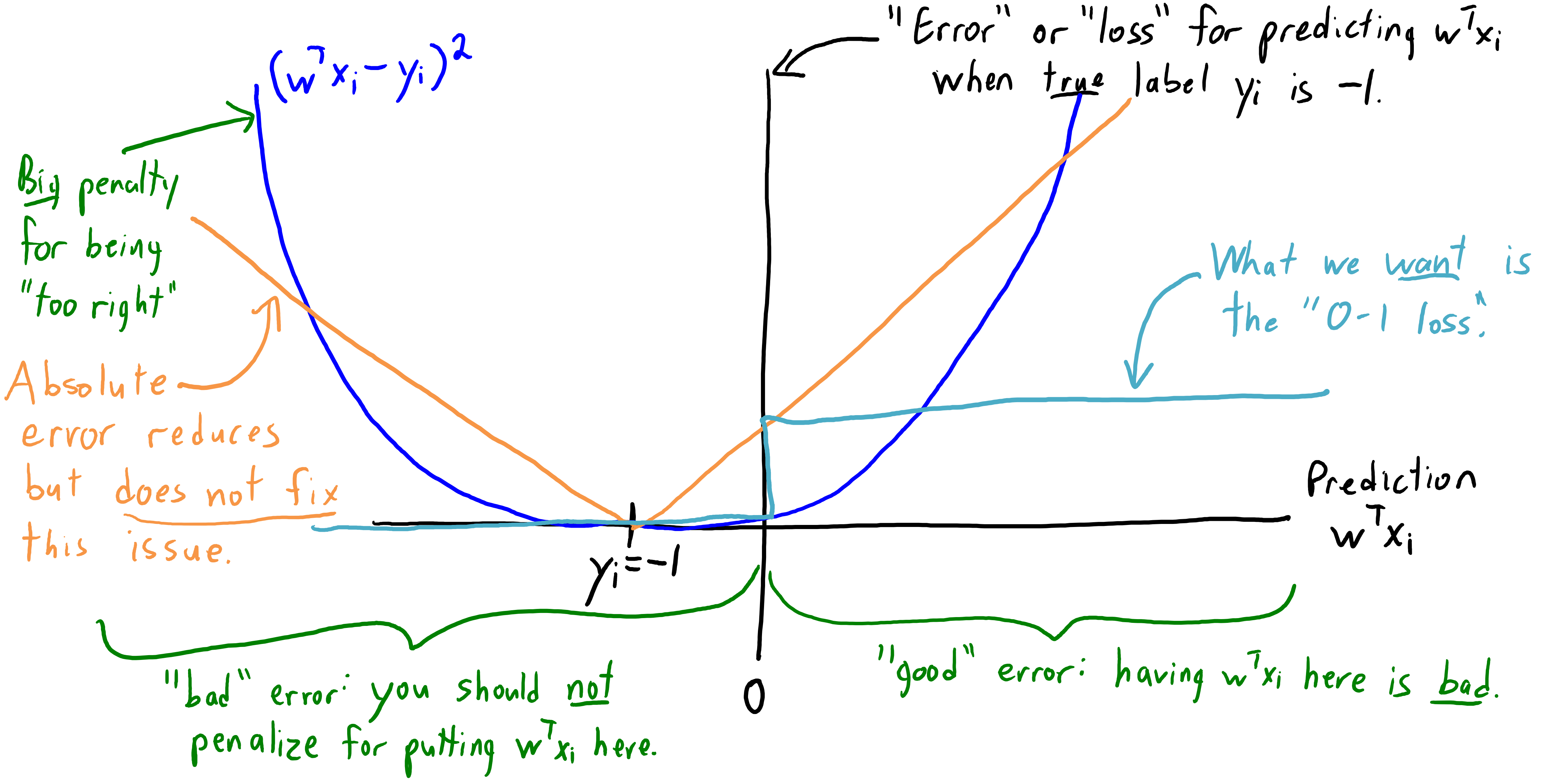
Comparing Loss Functions



Comparing Loss Functions



Comparing Loss Functions



0-1 Loss Function

- The 0-1 loss function is the number of classification errors:
  - We can write using the L0-norm as $||\text{sign}(Xw) - y||_0$.
  - Unlike regression, in classification it's reasonable that $\text{sign}(w^Tx_i) = y_i$.

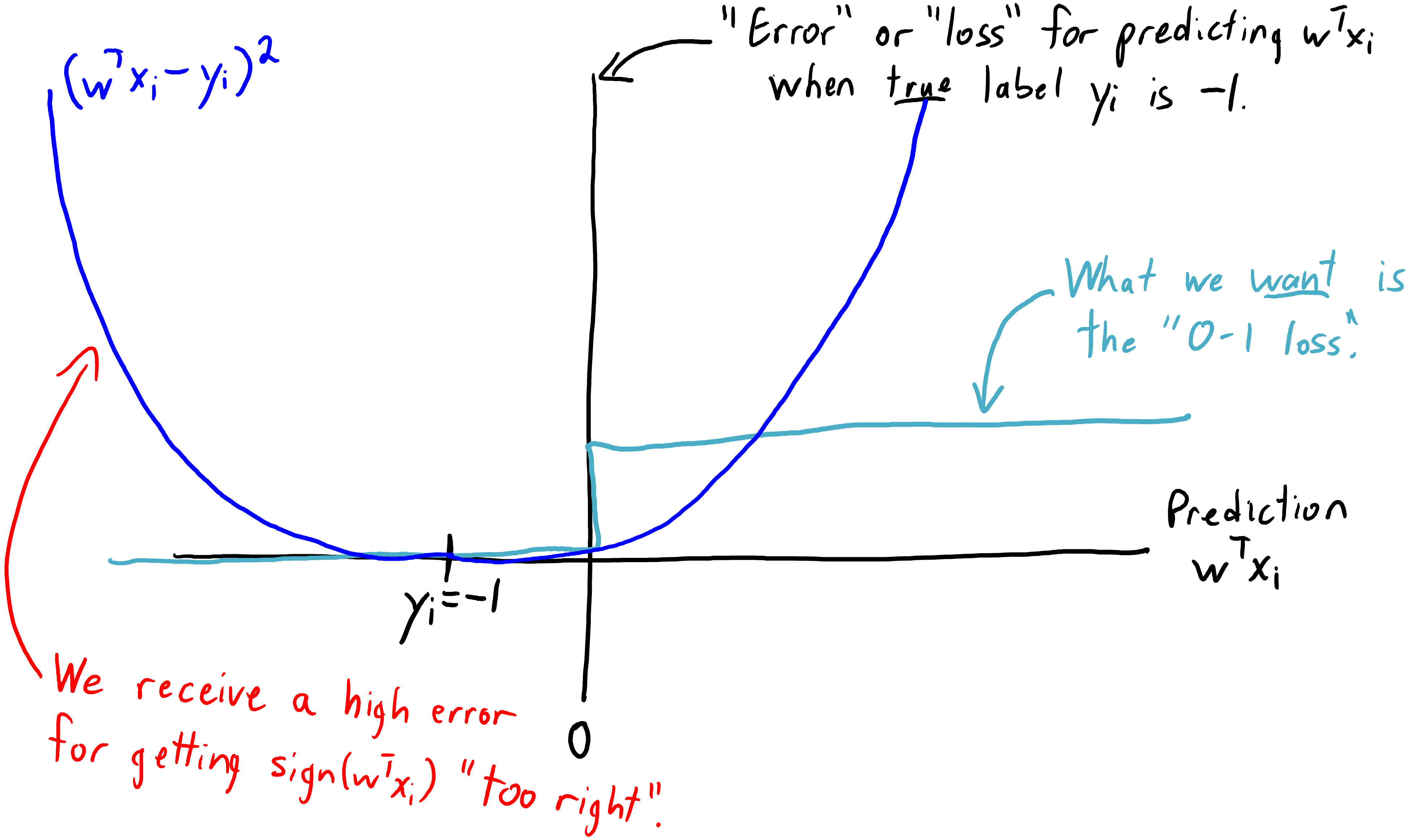
- Unfortunately the 0-1 loss is non-convex in 'w'.
  - It's easy to minimize if a perfect classifier exists (perceptron).
  - Otherwise, finding the 'w' minimizing 0-1 loss is a hard problem.

  - Gradient is zero everywhere so you don't know "which way to go" in w-space.
  - Note this is NOT the same type of problem we had with using the squared loss.
    - We can minimize the squared error, but it might giver a bad model for classification.

(Jupyter notebook demo / notes)

- NOTE: the next 4 slides are being replaced with the Jupyter notebook. I do not want to delete them in case they are usual for you to refer to, and I do not want to move them to Bonus since they aren’t bonus material. But I won’t cover them in lecture.

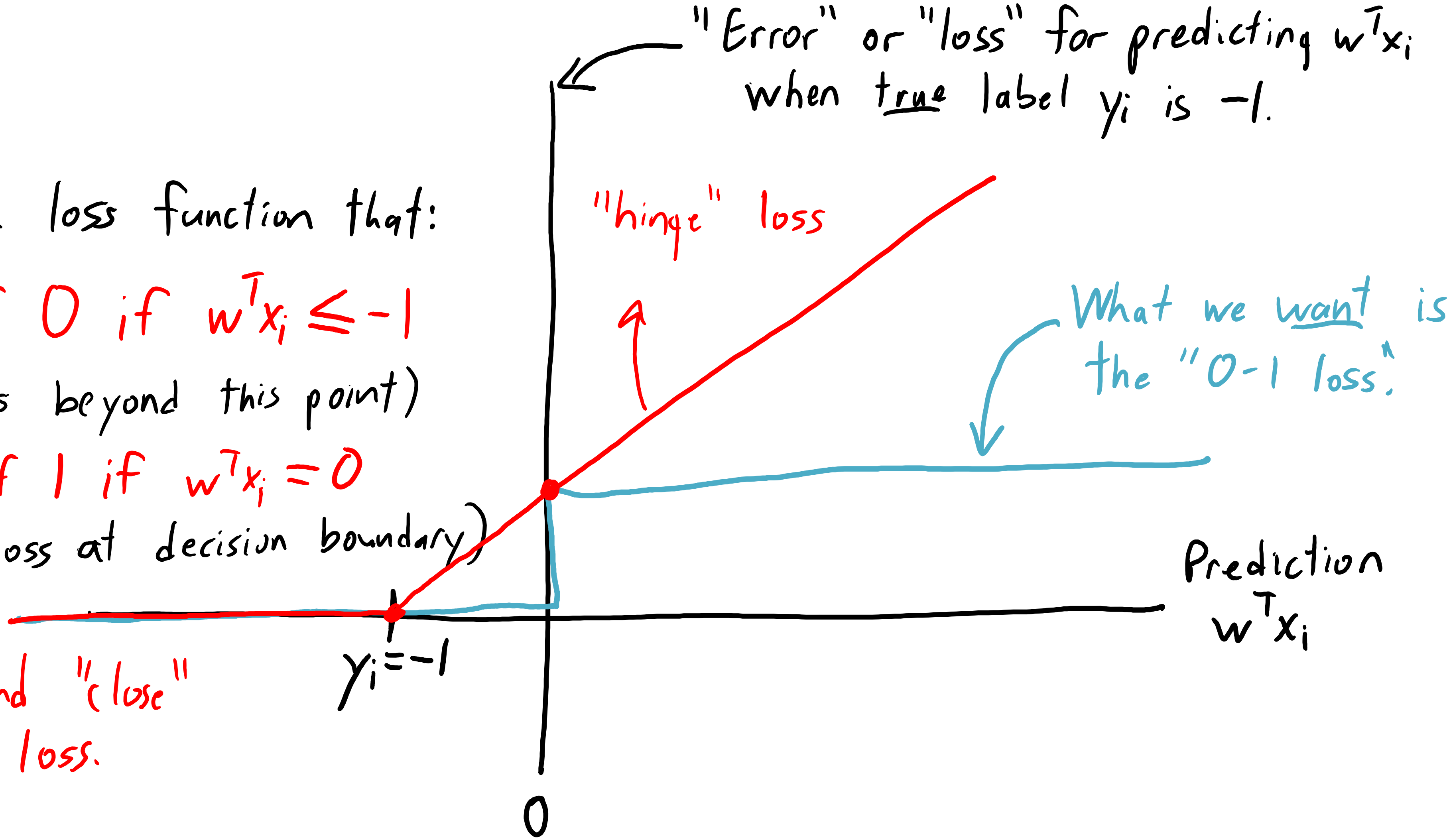
Hinge Loss: Convex Approximation to 0-1 Loss



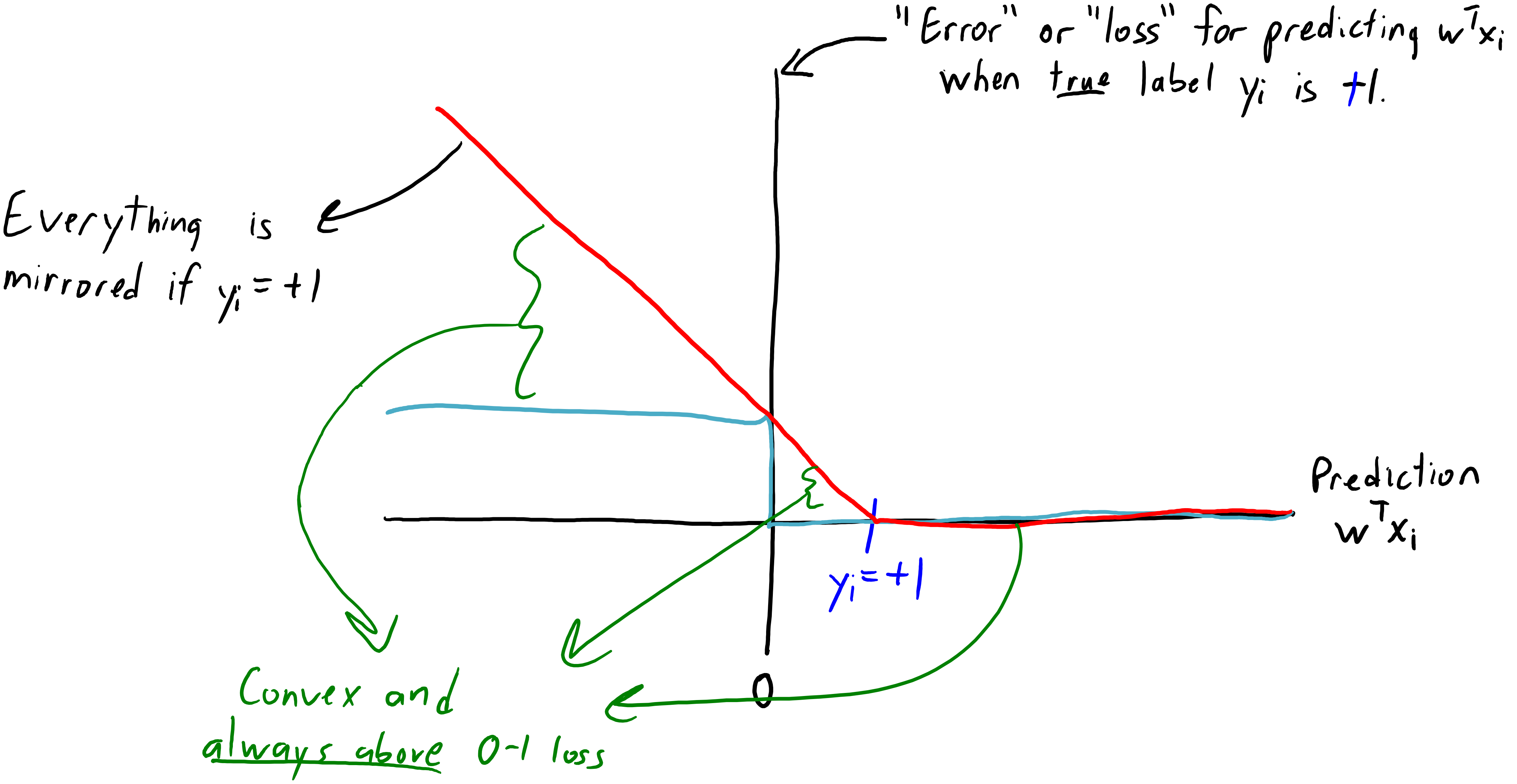
Hinge Loss: Convex Approximation to 0-1 Loss

Let's choose a loss function that:

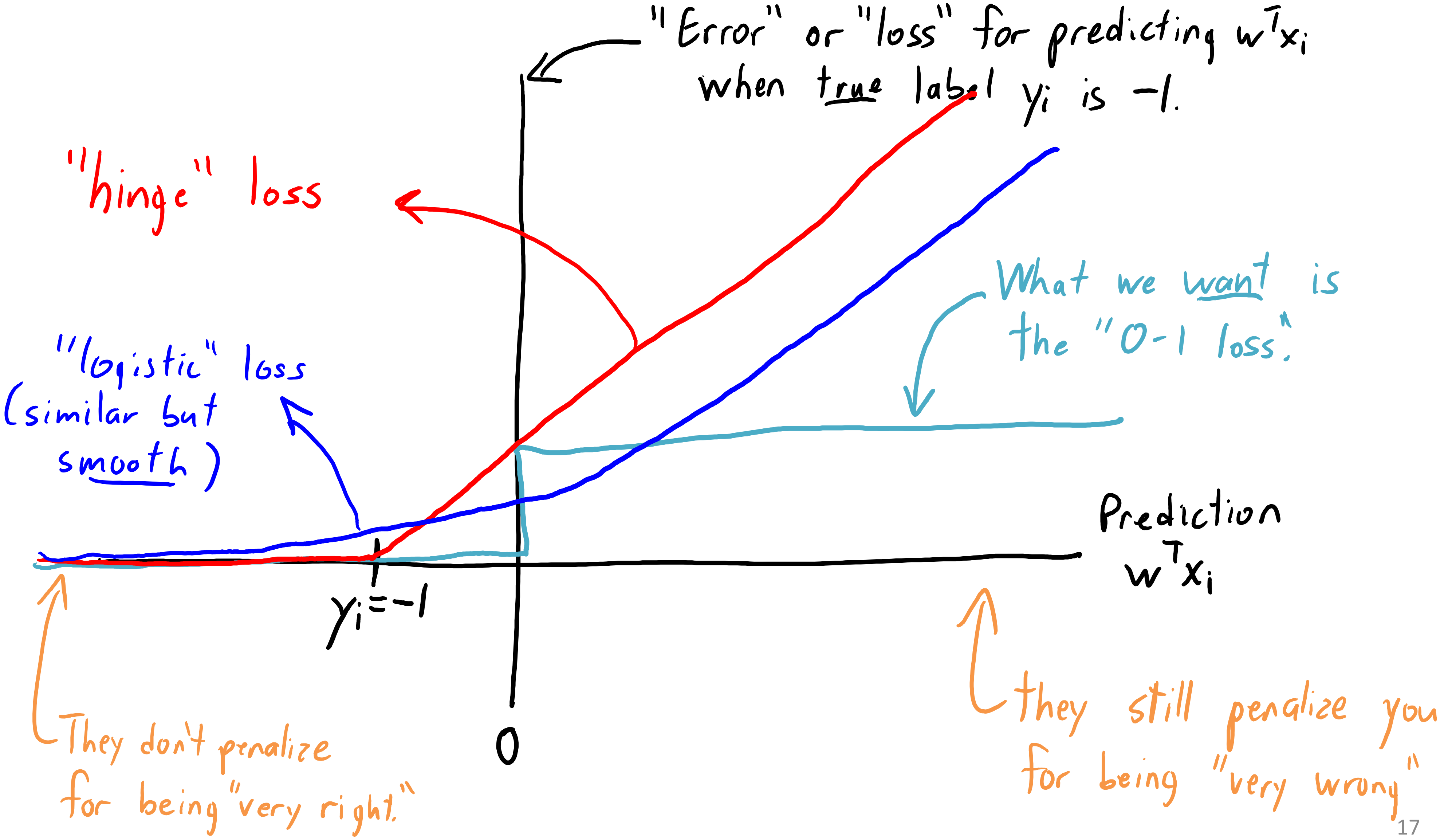
1. Has error of 0 if $w^T x_i \leq -1$
   (no "bad" errors beyond this point)
2. Has a loss of 1 if $w^T x_i = 0$
   (matches 0-1 loss at decision boundary)
3. Is convex and "close" to 0-1 loss.



Hinge Loss: Convex Approximation to 0-1 Loss



Convex Approximations 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 because each error incurs a loss of at least 1.
    - 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).

Location of the "hinge"

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

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

- Why not have the hinge at 0 instead of 1?
  - In that case, we'd have a trivial solution at w=0
    - f(0)=0 and f(w)≥0 so w=0 minimizes f.
  - Putting the hinge at some positive value avoids this problem.
  - The "1" is arbitrary and is just an overall scaling factor for w.
  - See bonus slides for more info

Logistic Loss

- Logistic loss:

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

- This is the "logistic loss" and model is called "logistic regression".
  – Convex and differentiable: minimize this with gradient descent.
  – You should also add regularization.
  – We'll see later that the probabilities it outputs have a meaningful interpretation.

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^Tx_i.
    - (For now we haven’t said how to minimize the SVM loss since it’s not smooth)
  - Weights w_j are easy to understand.
    - It’s how much x_j changes the prediction and in what direction.
  - We can often get a 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?”

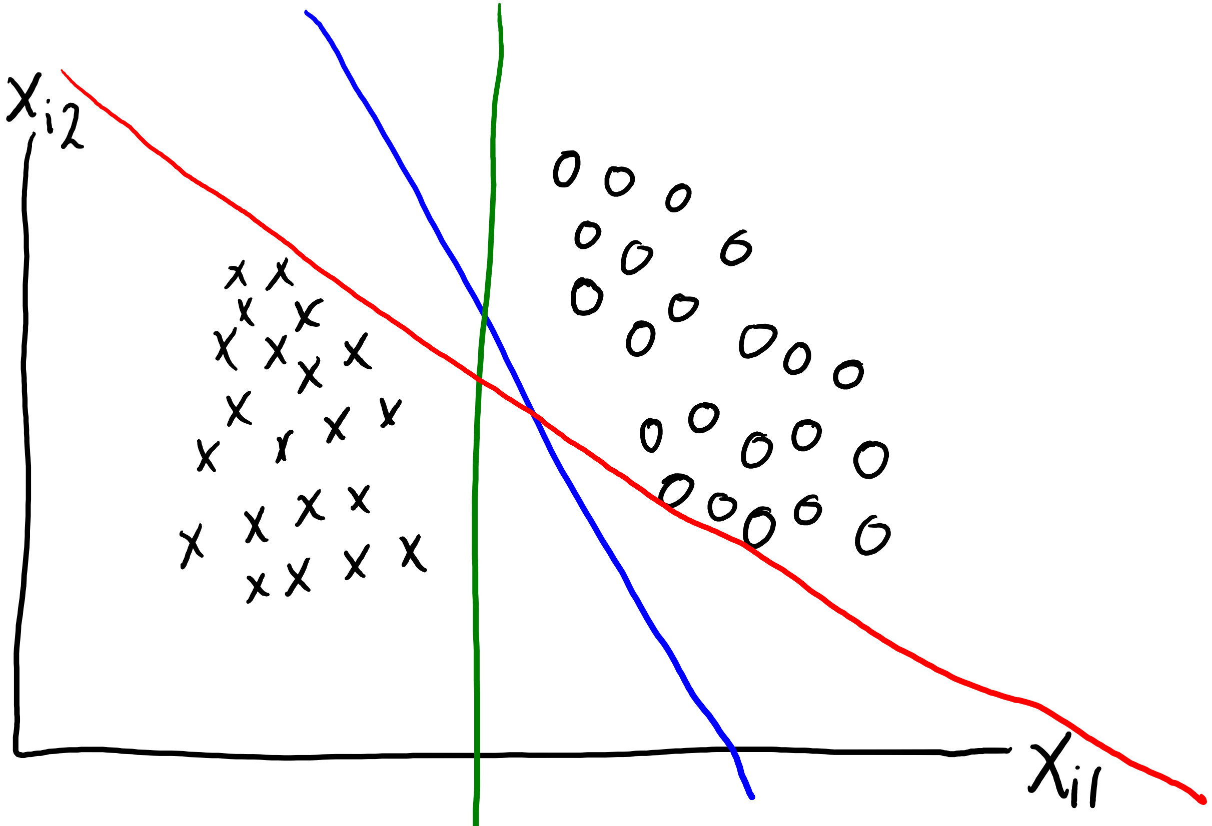
- Compared 179 classifiers on 121 datasets.
- Random forests are most likely to be the best classifier.
- Next best class of methods was SVMs (L2-regularization, RBFs).

Maximum-Margin Classifier

- You should know the word “margin” because you might hear it
- Personally I believe this is not the best way to understand SVM
- Thus the following slides are mainly for completeness
- More on max-margin in the bonus slides

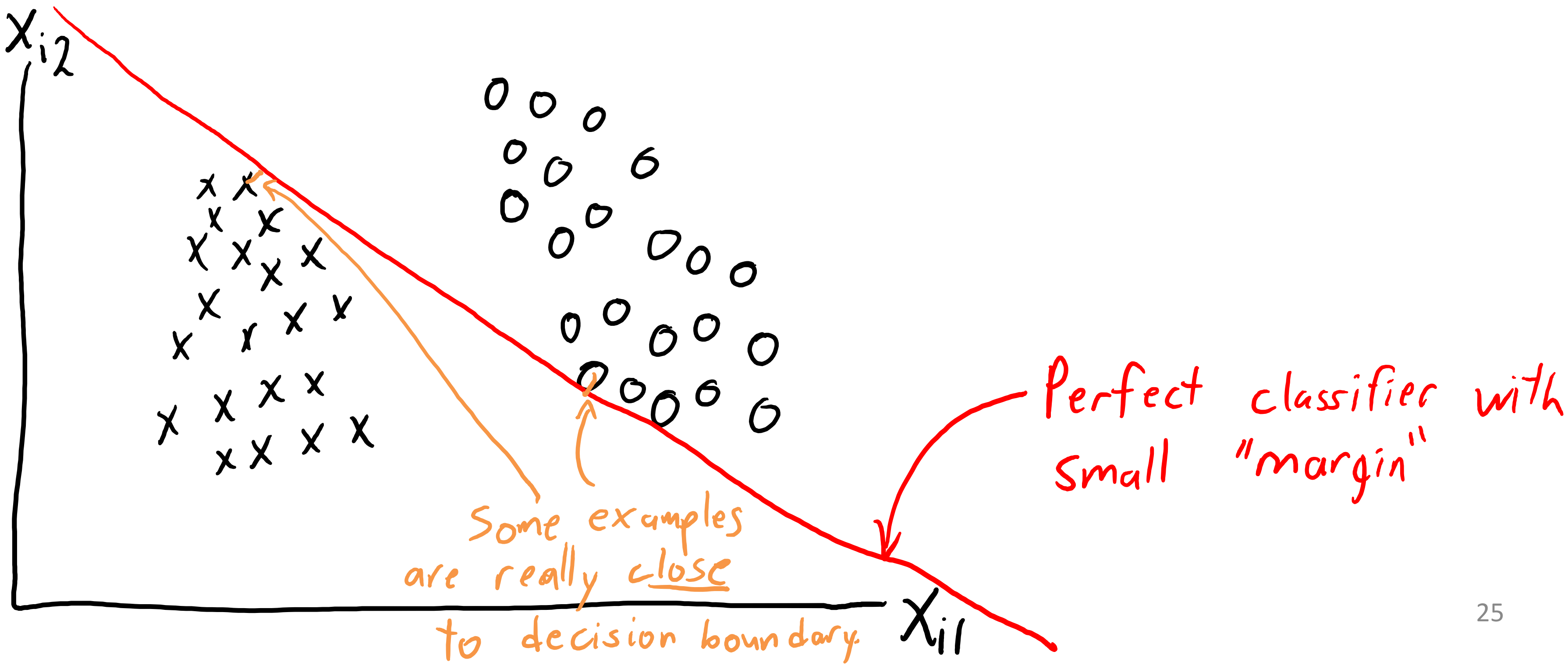
Maximum-Margin Classifier

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



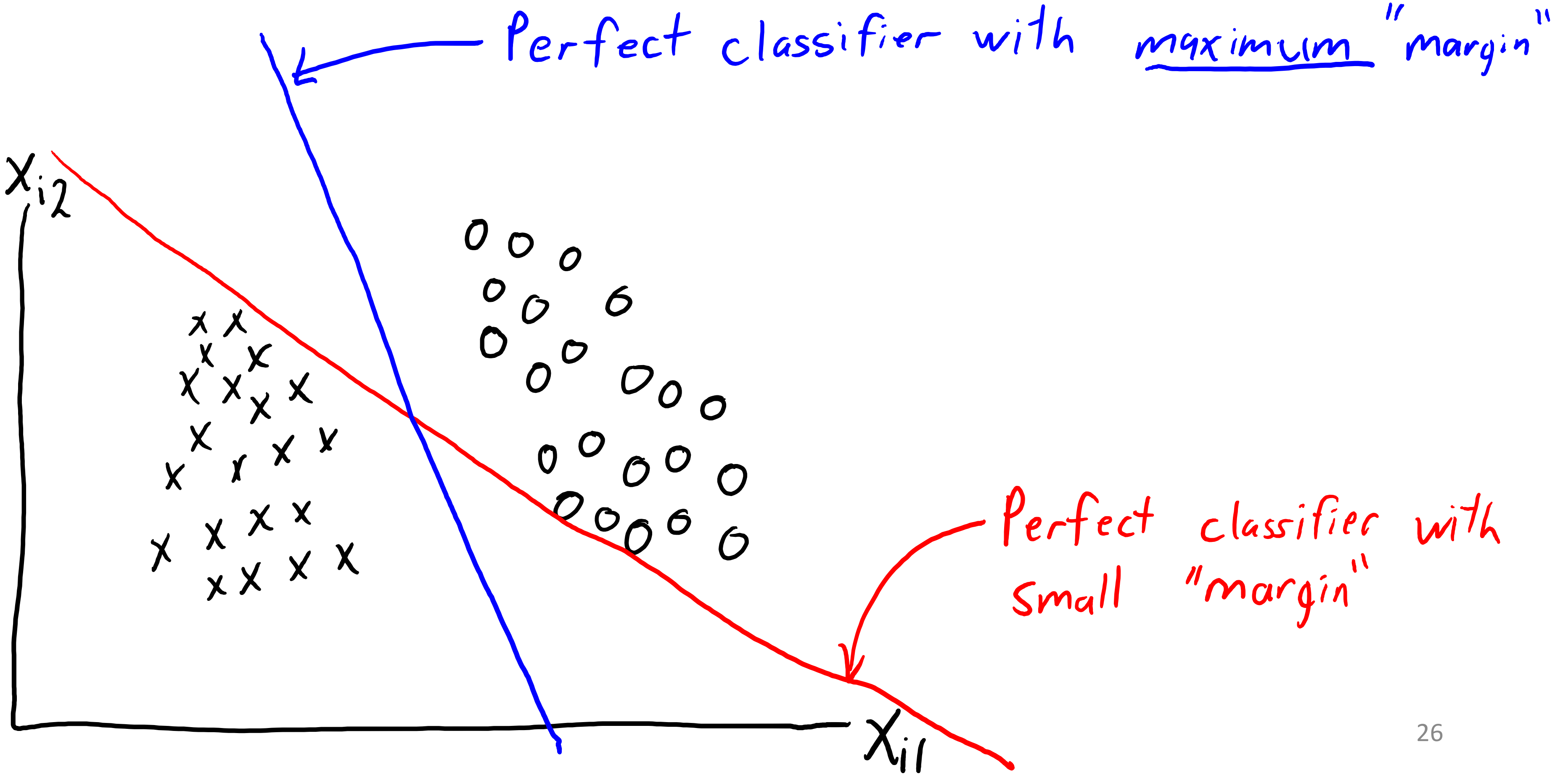
Maximum-Margin Classifier

- 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.



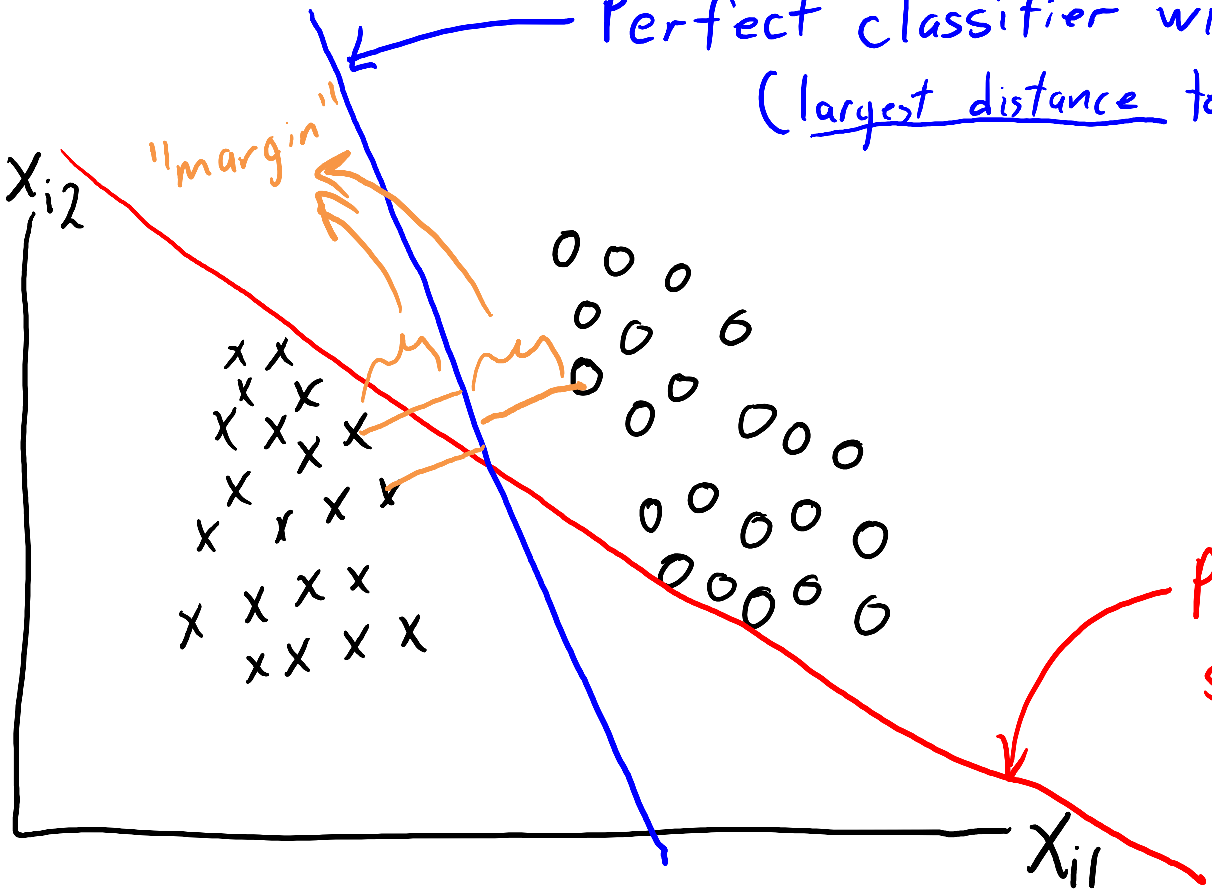
Maximum-Margin Classifier

- 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.

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



Perfect classifier with small "margin"

Maximum-Margin Classifier

- We want to "maximize the minimum distance"
  - We saw this sort of "minimax" problem with brittle regression:
    - Minimize the maximum distance from data to line (maximum residual)
- But we also don't like errors, so we penalize them
  - The objective becomes an error penalty term plus a max-margin term
  - One can massage these into the hinge loss + L2-regularization (bonus)
- SVM solving ties to constrained optimization (outside scope of 340)

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 regression”.
  - Both are just linear classifiers (a hyperplane dividing into two halfspaces)

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)

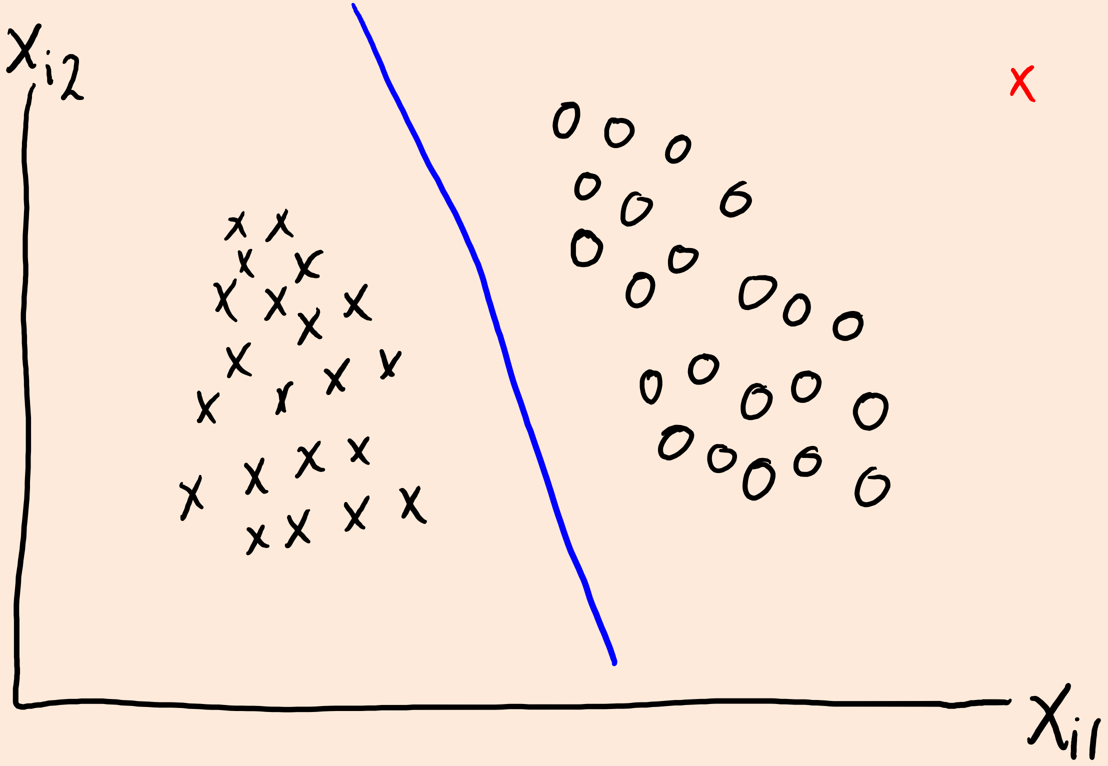
- The violation of this constraint is now given by:

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

- 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.

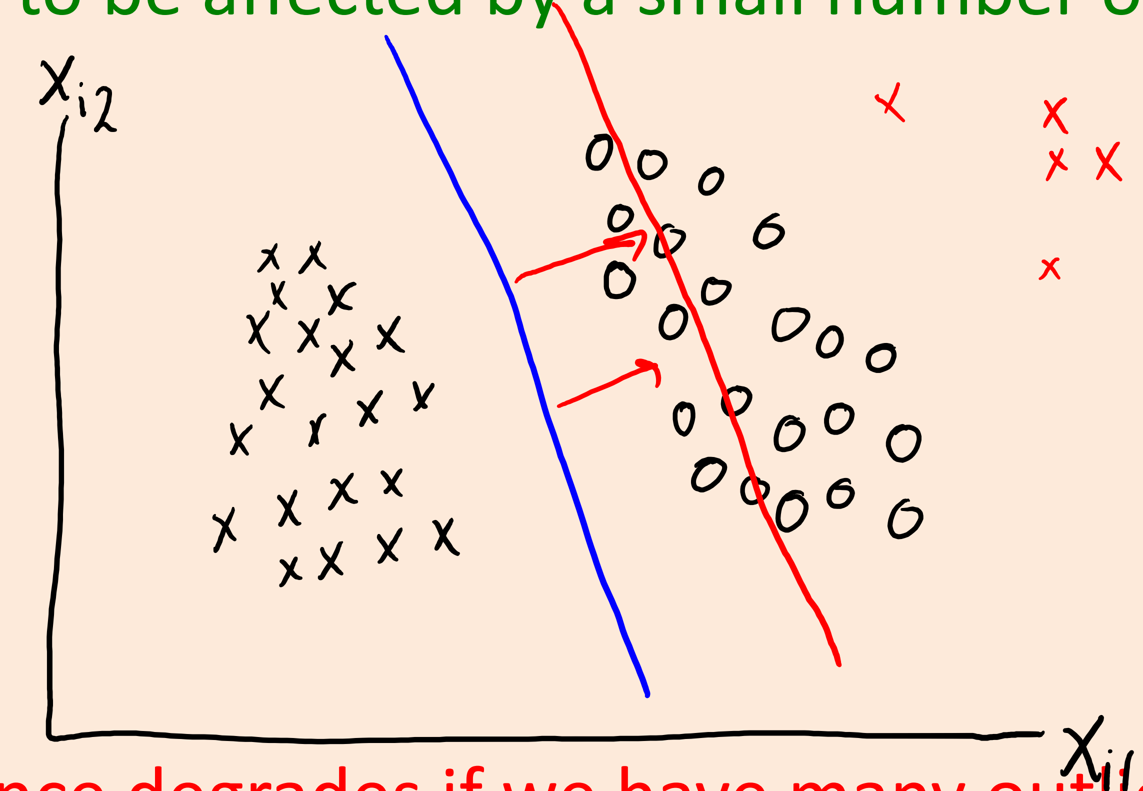
Robustness and Convex Approximations

- 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

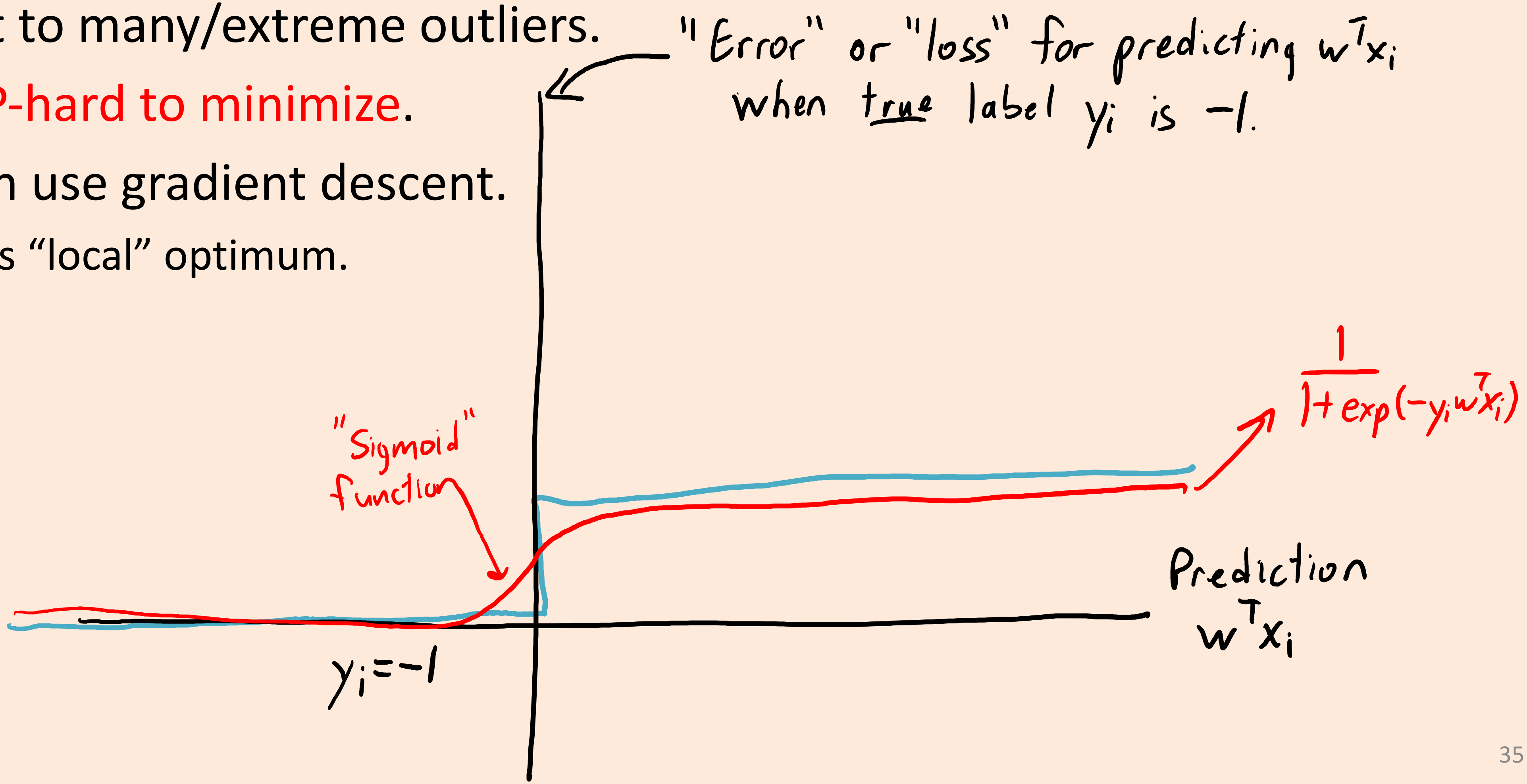
- Because the hinge/logistic grow like absolute value for mistakes, they tend not to be affected by a small number of outliers.



- 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.

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

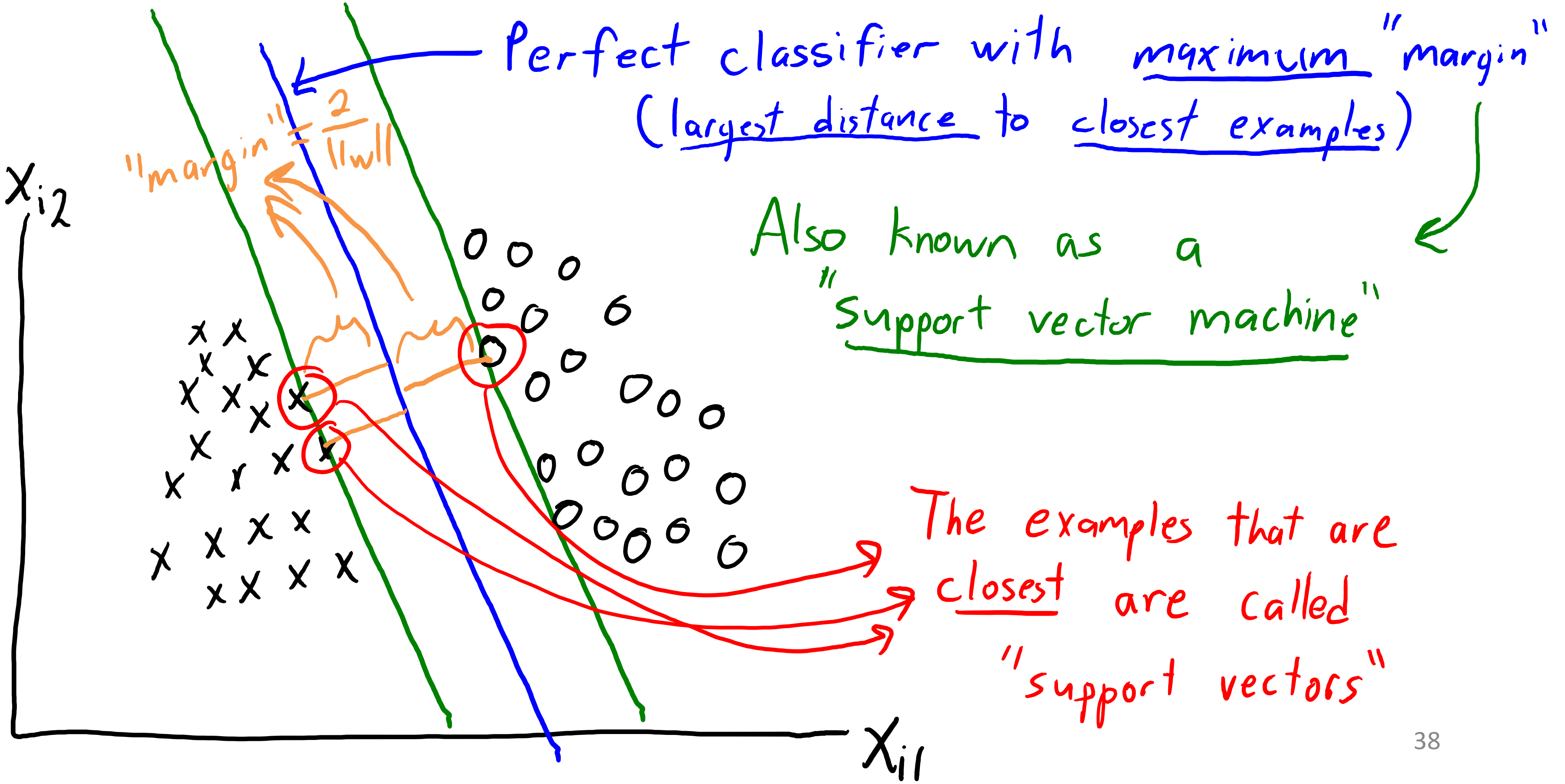
- If $w^T x_i$ gets the sign wrong, we can “correct” the mis-classification by modifying v_i.
- A problem is that we can ignore the ‘w’ and get a tiny training error by just updating the v_i variables.
- But we want most v_i to be zero, so “robust logistic regression” puts an L1-regularizer on the v_i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

Maximum-Margin Classifier

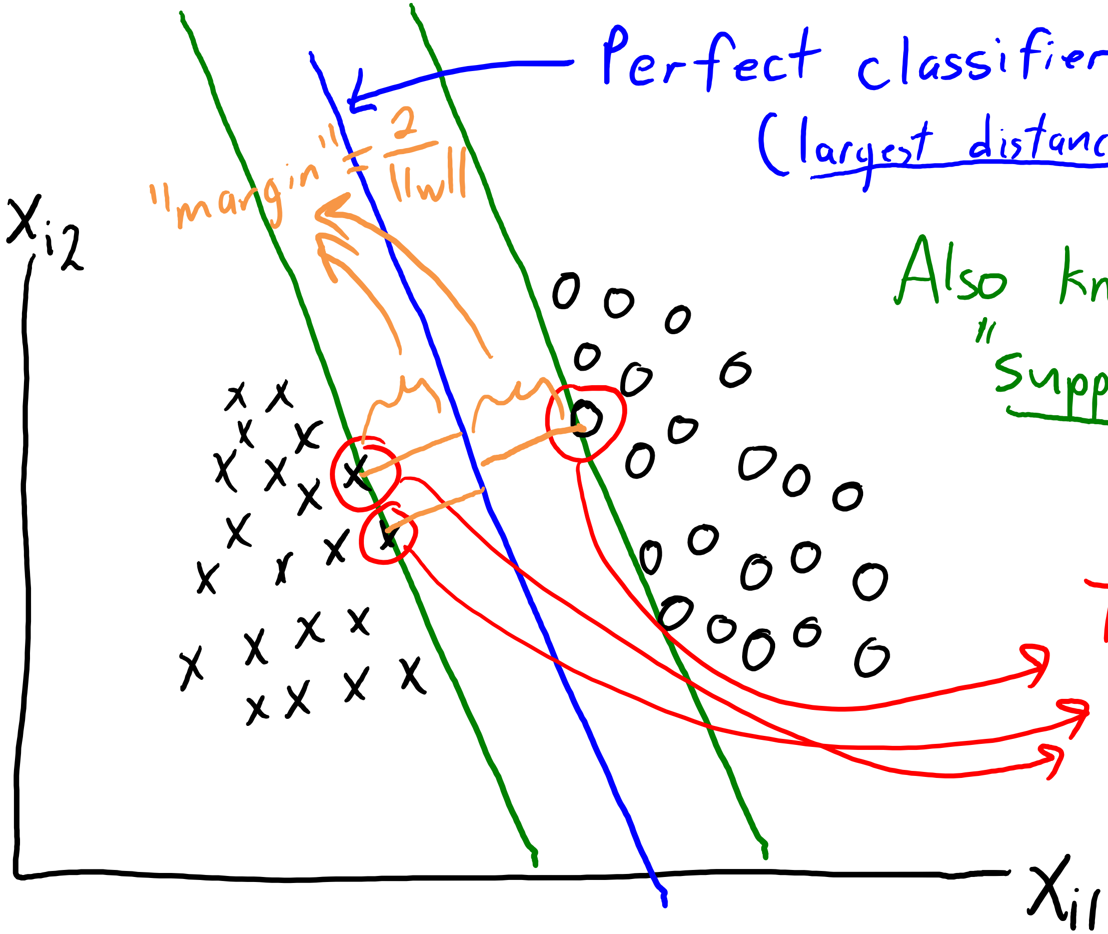
- Consider a linearly-separable dataset.
  - Maximum-margin classifier: choose the farthest from both classes.



Maximum-Margin Classifier

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

Final classifier only depends on support vectors



Also known as a "support vector machine"

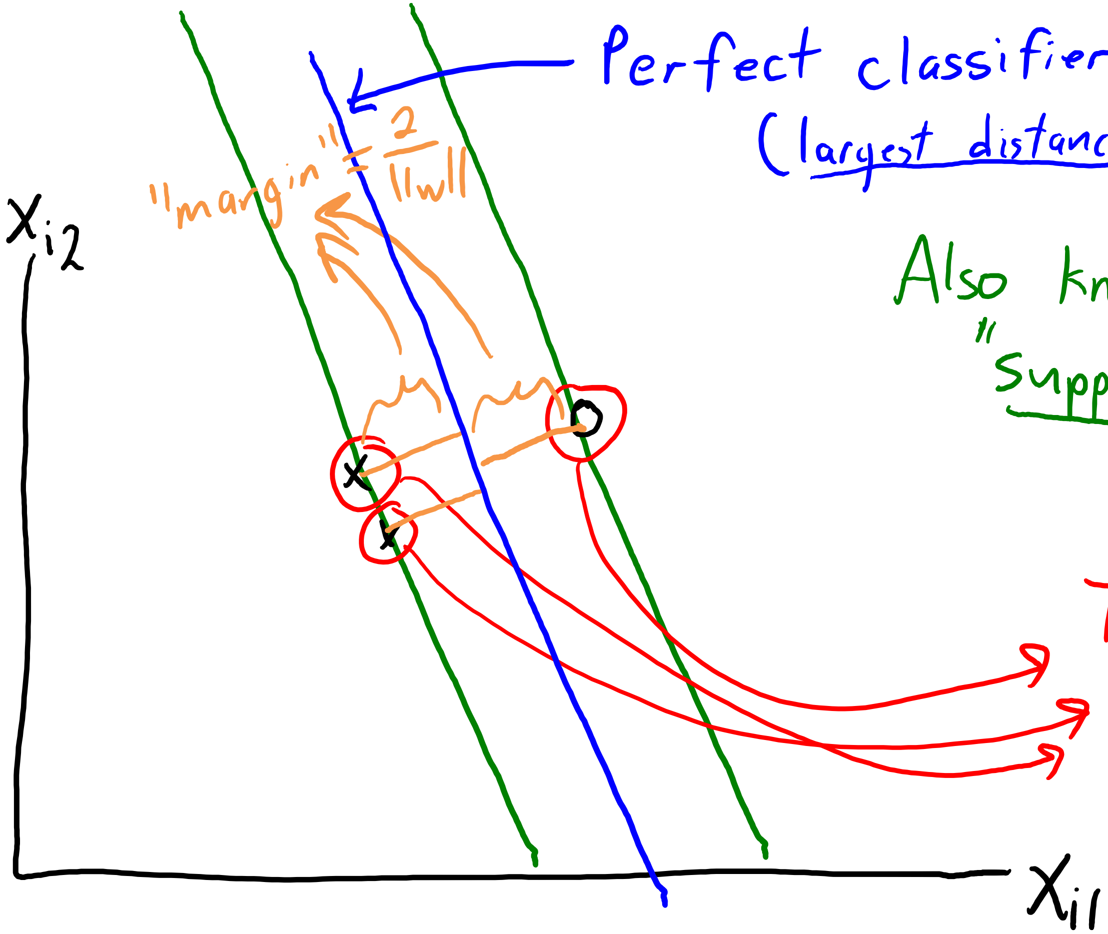
Maximum-Margin Classifier

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

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

Also known as a "support vector machine"

Final classifier only depends on support vectors



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

The examples that are closest are called "support vectors"

Support Vector Machines

- For linearly-separable data, SVM minimizes:

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

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

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

(classify all examples correctly)

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

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

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

⋮

Constraint violation is the hinge loss.

Support Vector Machines

- Try to maximizing margin and also minimizing constraint violation:

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

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.

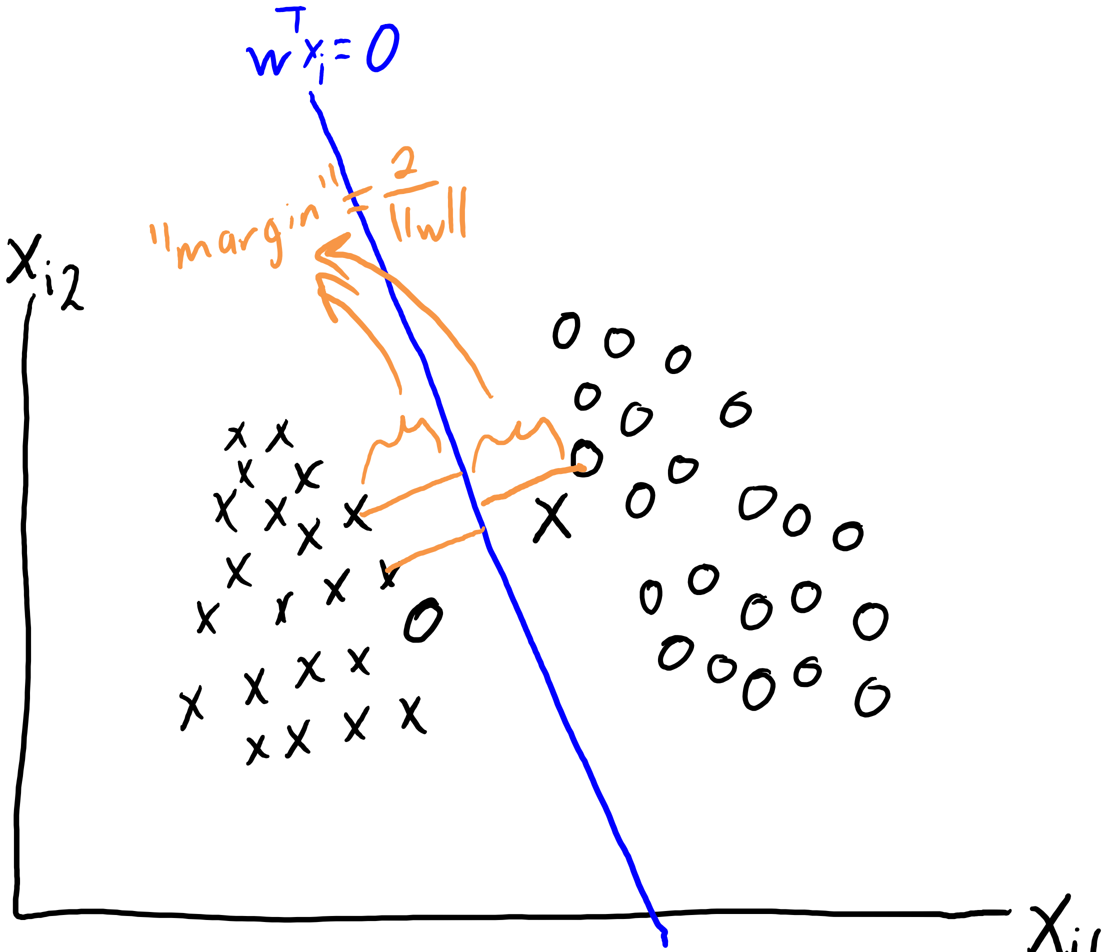
- We typically control margin/violation trade-off with parameter "λ":

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

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

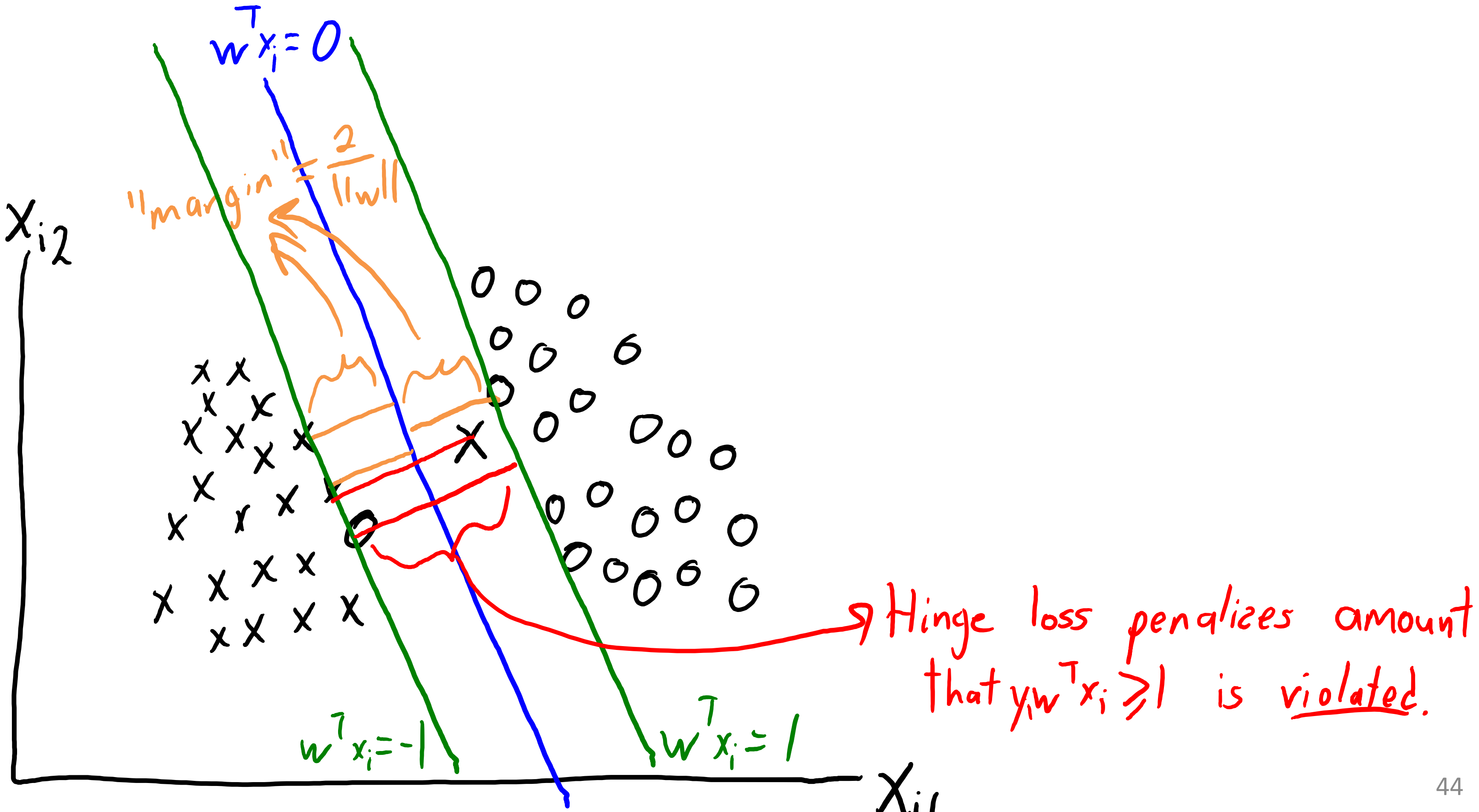
Support Vector Machines for Non-Separable

- Non-separable case:



Support Vector Machines for Non-Separable

- Non-separable case:



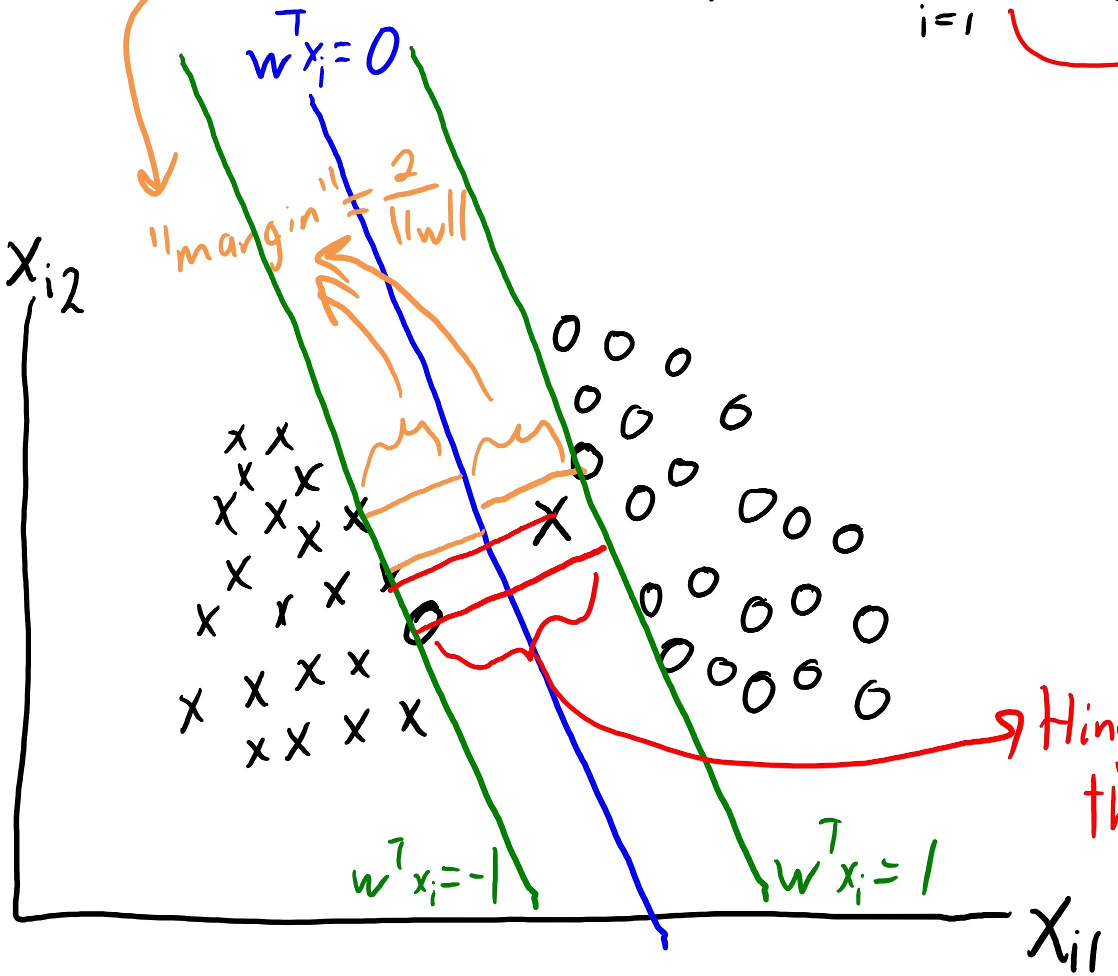
Support Vector Machines for Non-Separable

- 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.



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

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

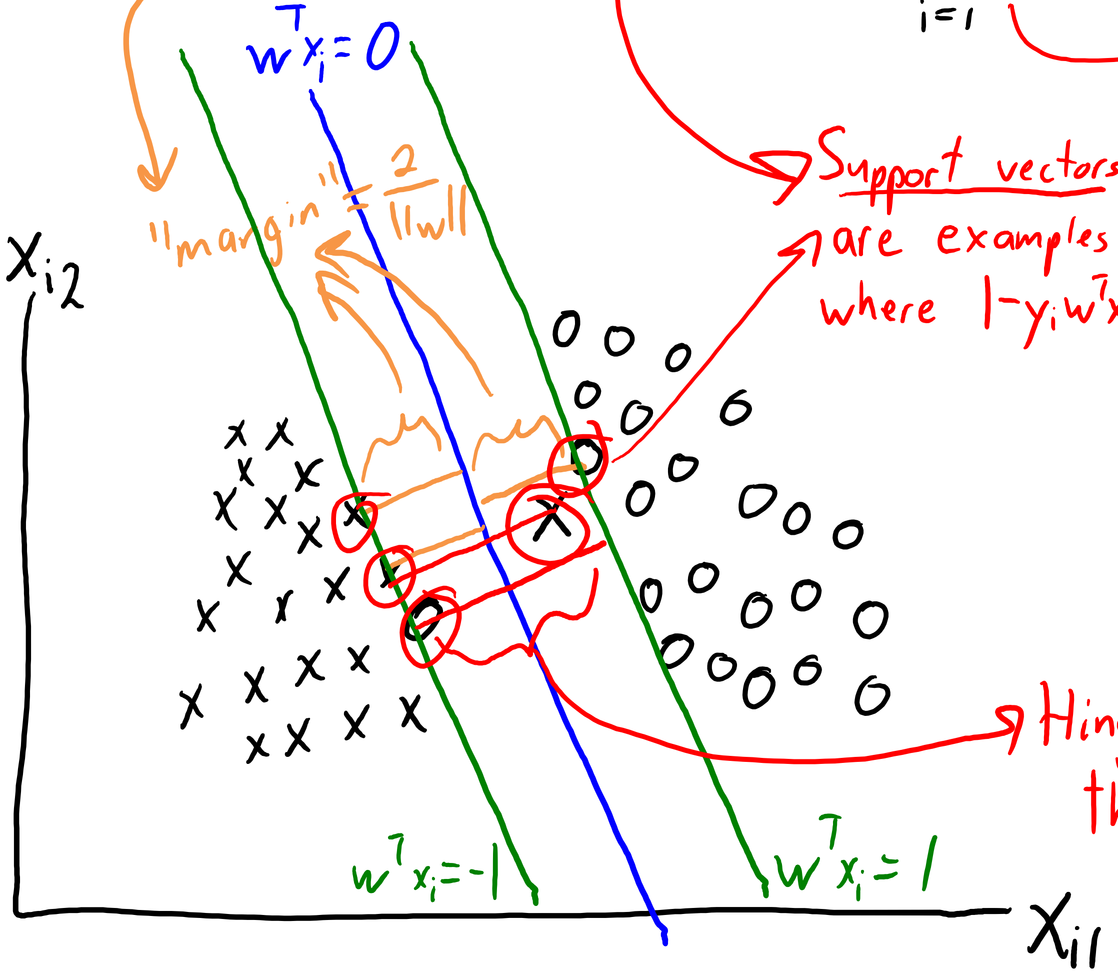
Support Vector Machines for Non-Separable

- Non-separable case:

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

Logistic regression can be viewed as smooth approximation to SVMs.

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



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

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.