

# CPSC 540: Machine Learning

Robust Regression, Logistic Regression, MLE and MAP

Winter 2016

# Admin

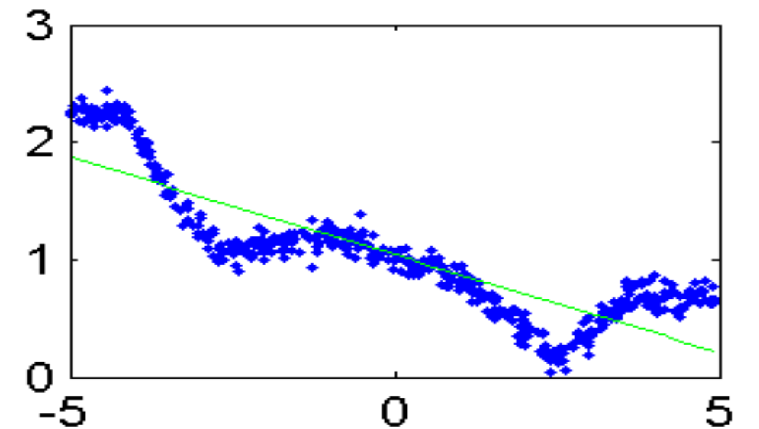
- **Room:** Search for new room is in progress.
  - Waiting for final numbers.
- **Auditing/enrollment forms:**
  - Drop-off/pickup your forms at the end of class.
  - For enrollment, I need your prerequisite forms.
- **CPSC and EECE graduate students:**
  - **submit your prereq form in class** by Thursday.
- **Assignment 1:**
  - Due Tuesday, **start early!**

# Last Time: Nonlinear Basis

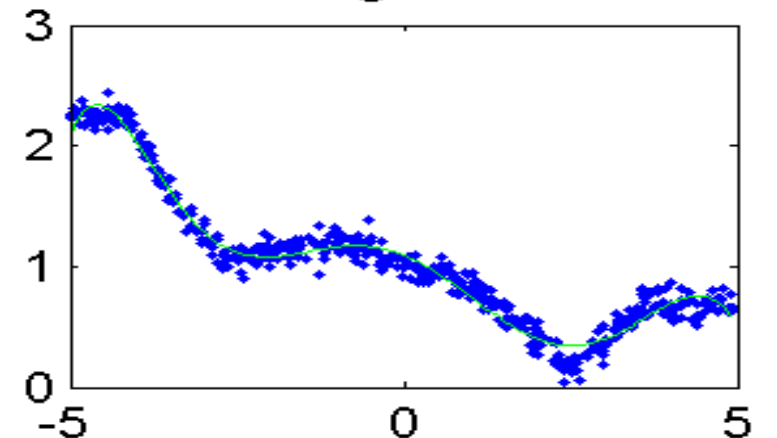
- **Change of basis** allows nonlinear functions with linear regression:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$X_{\text{poly}} = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \dots & (x_1)^p \\ \vdots & x_2 & (x_2)^2 & \dots & (x_2)^p \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & x_n & (x_n)^2 & \dots & (x_n)^p \end{bmatrix}$$



Degree 7



# Last Time: Training vs. Testing

- In supervised learning we are given a **training set**  $X$  and  $y$ .
  - But what we care about is **test error**: are prediction accurate on **new data**?
- In order to say anything about new data, **need assumptions**:
  - **IID assumption**: training and test data drawn from same distribution.
- Often, we have an explicit **test set** to approximate test error.

Data:  $X, y, X_{test}, y_{test}$

1. Train:  $model = fit(X, y)$

2. Predict test set labels  $\hat{y} = predict(model, X_{test})$

3. Evaluate  $error = diff(\hat{y}, y_{test})$

- Golden rule: **this test set cannot influence training in any way.**
  - Otherwise, not valid approximation of test error.

# What if we don't have an explicit test set?

- Possible training procedures if you **only have a training set**:

1. Randomly split training set into “train” and “validate” set.
2. Train **model** based on **train set**.
3. Report **validate set accuracy** with this model.

$$X = \begin{bmatrix} \text{train} \\ \text{validate} \end{bmatrix} \quad y = \begin{bmatrix} \text{train} \\ \text{validate} \end{bmatrix}$$

- We can **trust this accuracy** is reasonable.
  - Validation set gives unbiased approximation of test error.

# What if we don't have an explicit test set?

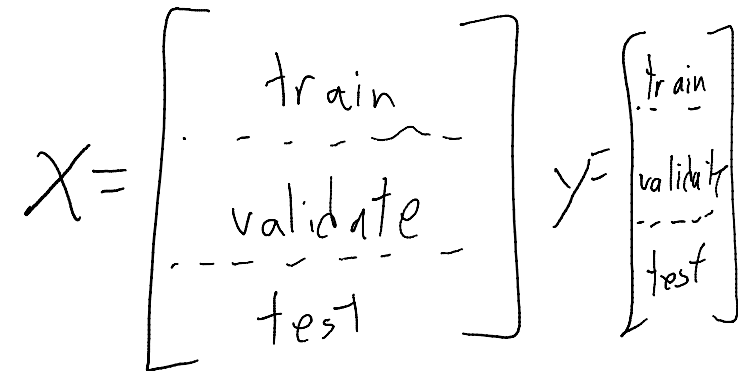
- Possible training procedures if you **only have a training set**:
  1. Randomly split training set into “train” and “validate” set.
  2. Train **10 models** based on **train set** (e.g., 10 different bases)
  3. Choose one with **highest accuracy** on **validate set**.
  4. Report **validate set accuracy** with this model.
- We should be a **little skeptical** of this accuracy:
  - We **violated golden rule** on validation set:
    - Approximation of test error was used to choose model.
  - But we probably not overfitting much: only 10 models considered.

# What if we don't have an explicit test set?

- Possible training procedures if you **only have a training set**:
  1. Randomly split training set into “train” and “validate” set.
  2. Train **1 billion models** based on **train set**.
  3. Choose one with **highest accuracy** on **validate set**.
  4. Report **validate set accuracy** with this model.
- We should be a **very skeptical** of this accuracy:
  - We **badly violated golden rule** on validation set:
    - High chance of overfitting to validation set.

# What if we don't have an explicit test set?

- Possible training procedures if you **only have a training set**:
  1. Randomly split training set into “train”, “validate”, and “**test**” set.
  2. Train **1 billion models** based on **train set**.
  3. Choose one with **highest accuracy** on **validate set**.
  4. Report **test set accuracy** with this model.



- We can **trust this accuracy** is reasonable.
  - We might still overfit to validate set, but test set not used during training.



# What if we don't have an explicit test set?

- Similar reasoning applies to **cross-validation**:
  - Selecting between **10 models** using cross-validation on full data set:
    - Cross-validation error of best model will be **a bit optimistic**.
  - Selecting between **1B models** using cross-validation on full data set:
    - Cross-validation error of best model could be **meaningless**.
  - **Proper cross-validation** procedure:
    - Randomly split data into “**train/crossValidate**” and “**test**” set.
    - Choose model with lowest cross-validation error on “**train/crossValidate**” set.
    - Report error on “**test**” set which did not influence final model.

$$X = \begin{bmatrix} \text{train/crossVal} \\ \dots \\ \text{test} \end{bmatrix} \quad y = \begin{bmatrix} \text{train/crossVal} \\ \dots \\ \text{test} \end{bmatrix}$$

# Fundamental Trade-Off and Regularization

- Bias-variance and other learning theory results to trade-off:

1. How small you can make the training error.

vs.

2. How well training error approximates the test error.

- **Simple models**: high training error but don't overfit:
- **Complex models**: low training error but overfit.
- **Regularization**: reduces overfitting in complex models.
  - Common approach is L2-regularization:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \frac{1}{2} \|Xw - y\|^2 + \frac{\lambda}{2} \|w\|^2$$

- Increases training error, but **typically decreases test error**.
- Increasing number of training examples 'n' has a similar effect on trade-off.

# Parametric vs. Non-Parametric

- Polynomials are not the only **possible bases**:
  - Common to use exponentials, logarithms, trigonometric functions, etc.
  - The **right basis will vastly improve performance**.
  - But when you have a lot of features, the **right basis may not be obvious**.
- The above bases are **parametric** model:
  - The size of the model *does not depend* on the number of training examples 'n'.
  - As 'n' increases, you can estimate the model more accurately.
  - But at some point, more data doesn't help because model is too simple.
- Alternative is **non-parametric** models:
  - **Size of the model grows** with the number of training examples.
  - Model gets more complicated as you get more data.
  - You can model very complicated functions where you don't know the right basis.

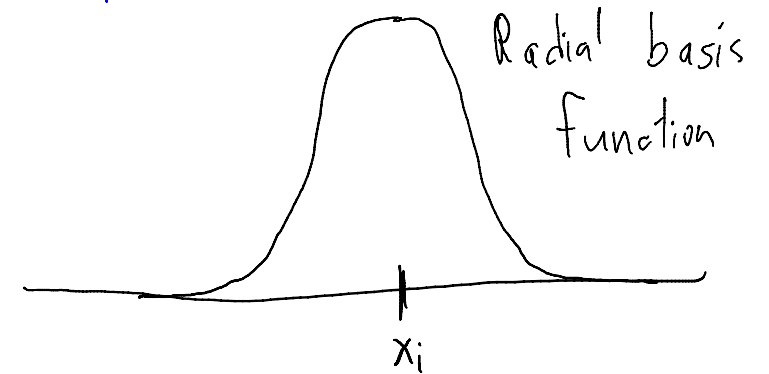
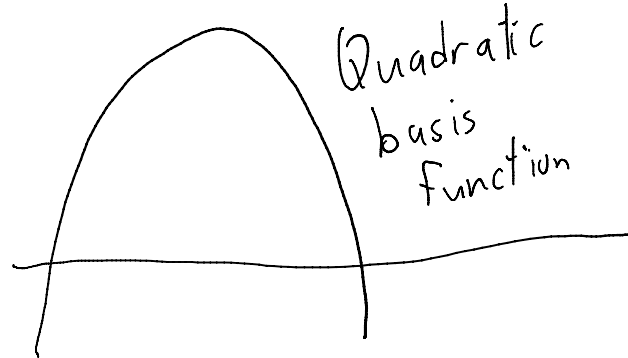
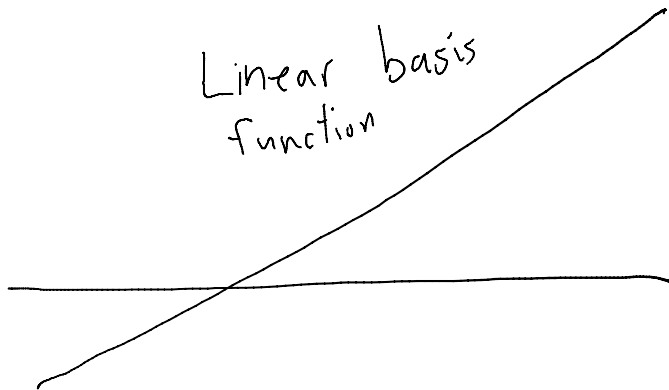
# Non-Parametric Basis: RBFs

- Radial basis functions (RBFs):
  - Non-parametric bases that depend on distances to training points.
- Most common example is Gaussian or squared exponential:

$$f(x) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$$

training example

parameter (speed that it goes to zero)



Where did constant  $\sqrt{2\pi}$  go?

- not needed:




$$w^T x_i = \left(\frac{1}{c} w\right)^T (c x_i)$$

# Non-Parametric Basis: RBFs

Note each basis function depends on all original features:  
 $\|x - x_i\|^2 = \sum_{j=1}^d (x_j - x_{ij})^2$

- Radial basis functions (RBFs):
  - Non-parametric bases that depend on distances to training points.
- Most common example is Gaussian or squared exponential:

$$X_{\text{rbf}} = \begin{bmatrix} \exp\left(-\frac{\|x_1 - x_1\|^2}{2\sigma^2}\right) & \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right) & \dots & \exp\left(-\frac{\|x_1 - x_n\|^2}{2\sigma^2}\right) \\ \exp\left(-\frac{\|x_2 - x_1\|^2}{2\sigma^2}\right) & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots \\ \exp\left(-\frac{\|x_n - x_1\|^2}{2\sigma^2}\right) & \dots & \dots & \exp\left(-\frac{\|x_n - x_n\|^2}{2\sigma^2}\right) \end{bmatrix}$$

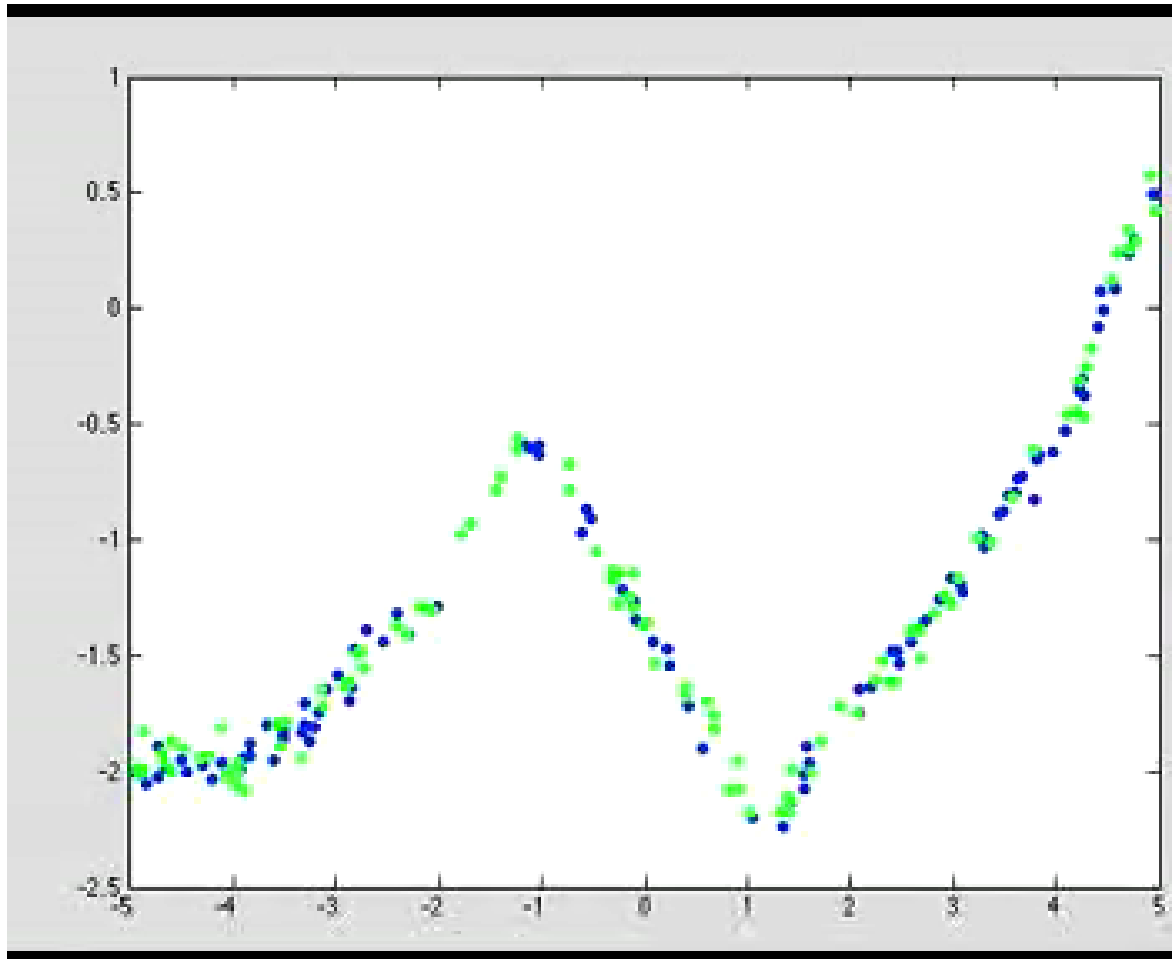
$\hat{y}_i = w_1$    $+ w_2$    $+ \dots + w_n$  

$\rightarrow$   $X_{\text{rbf}}$  is  $n$  by  $n$ .

- Gaussian RBFs are universal approximators (compact subsets of  $\mathbb{R}^d$ )
  - Can approximate any continuous function to arbitrary precision.

# Non-Parametric Basis: RBFs

- RBF basis for different values of  $\sigma$ :



Could add bias and linear basis:

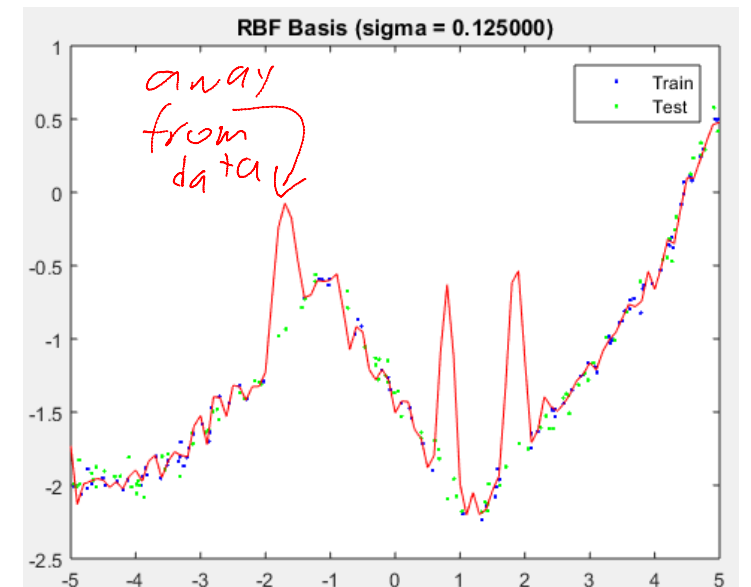
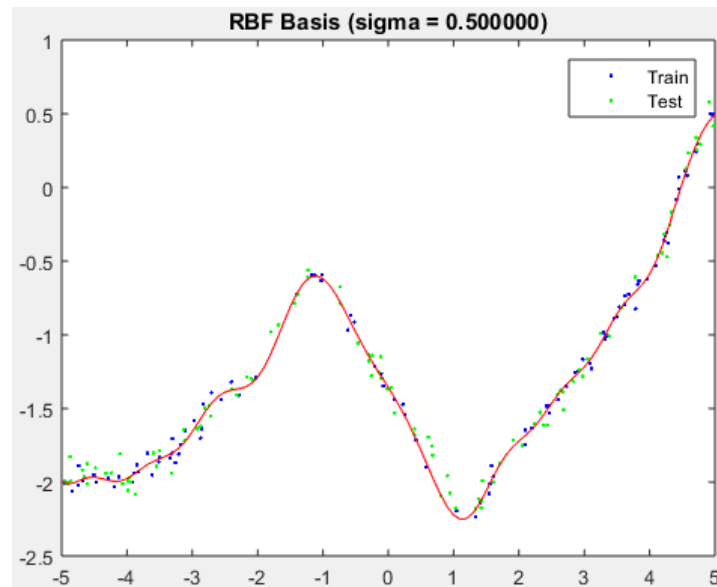
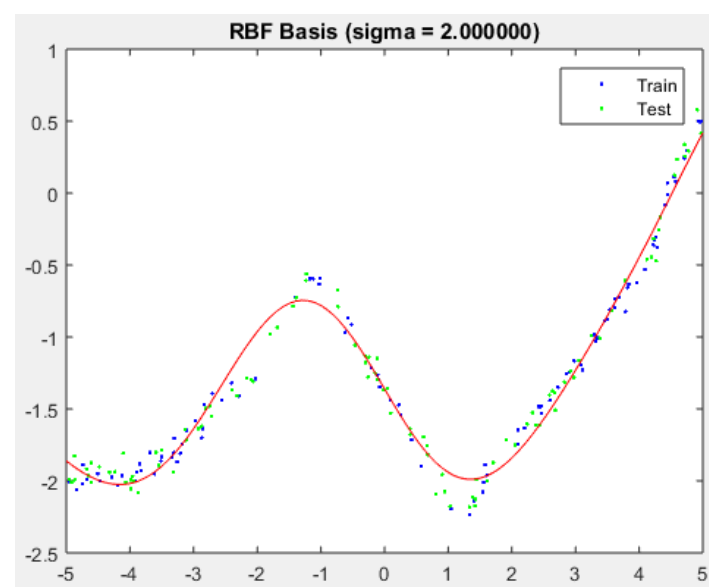
$$\bar{X} = \begin{bmatrix} 1 & -x_1 & \dots \\ \vdots & -x_2 & \vdots \\ \vdots & -x_3 & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & -x_n & \vdots \end{bmatrix} \quad X_{\text{rbf}}$$

1      d      n

Now it defaults to linear regression instead of 0 away from data.

# RBFs, Regularization, and Validation

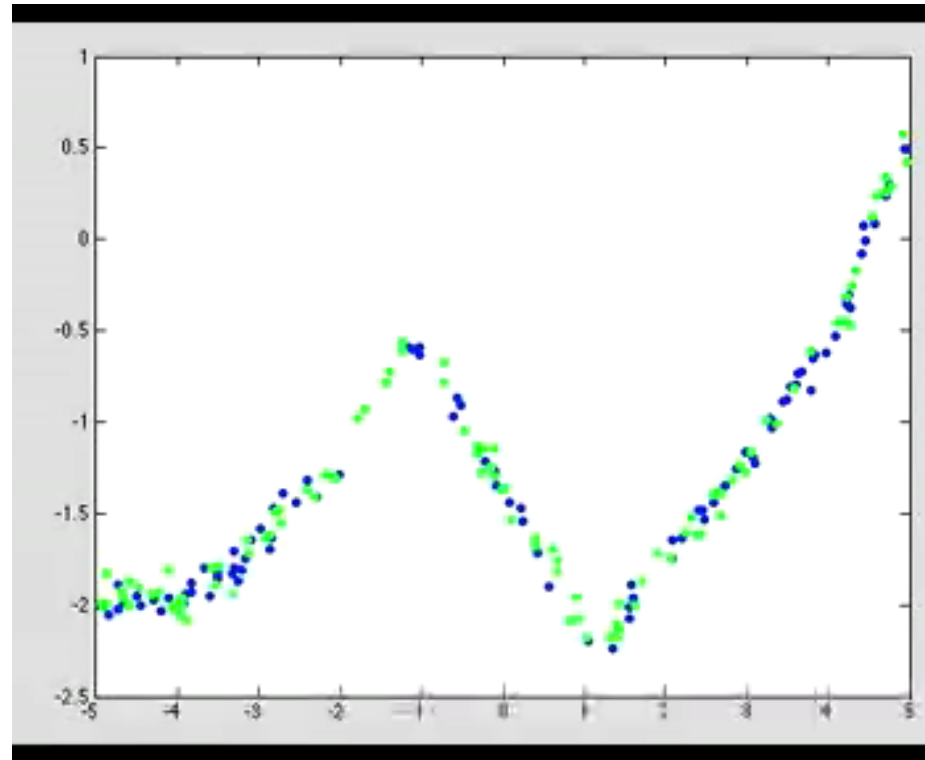
- Very effective model:
  - RBF basis with L2-regularization and cross-validation to choose  $\sigma$  and  $\lambda$ .



- **Expensive at test time:** need distance to all training examples.

# RBFs, Regularization, and Validation

- RBF basis with L2-regularization for different values of  $\sigma$  and  $\lambda$ .



- At least one of these models is often a good fit.



# Today: Alternatives to Squared Error

- Squared error is **computationally convenient** choice:

- Solution involves solving a linear system.

$$w = (X^T X + \lambda I)^{-1} X^T y$$

- But it's usually **not the right choice**:

- Corresponds to assuming error are normally distributed (later in lecture).

- Makes it **sensitive to outliers** or large errors.

- Makes it **inappropriate with restrictions** on  $y$  (like binary or censored).

- Today:

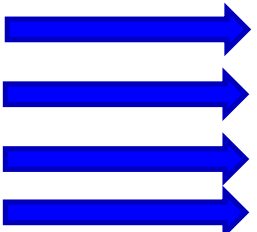
- **Alternatives to squared error**, and deriving other alternatives.

- Computational implications of these alternatives.

# Least Squares with Outliers

- Consider fitting least squares with an **outlier** in the labels:
  - Observation that is unusually different from the others.

	Egg	Milk	Fish	Wheat	Shellfish	Peanuts	...	IgE
Day 1	0	0.7	0	0.3	0	0		700
Day 2	0.3	0.7	0	0.6	0	0.01		740
Day 3	0	0	0	0.8	0	0		50
Day 4	0.3	0.7	1.2	0	0.10	0.01		40000

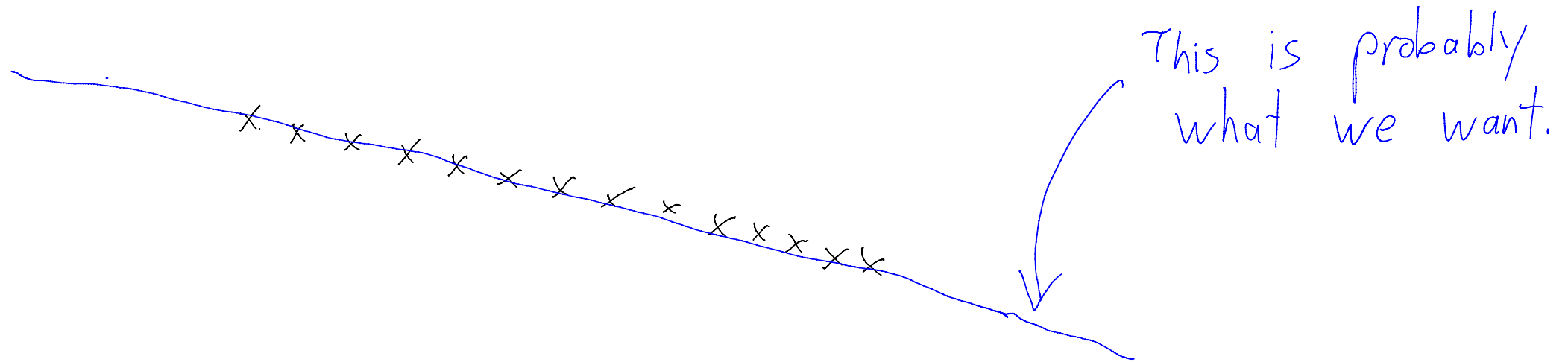


- Some sources of outliers:
  - Errors, contamination of data from different distribution, rare events.

# Least Squares with Outliers

- Consider fitting least squares with an **outlier** in the labels:

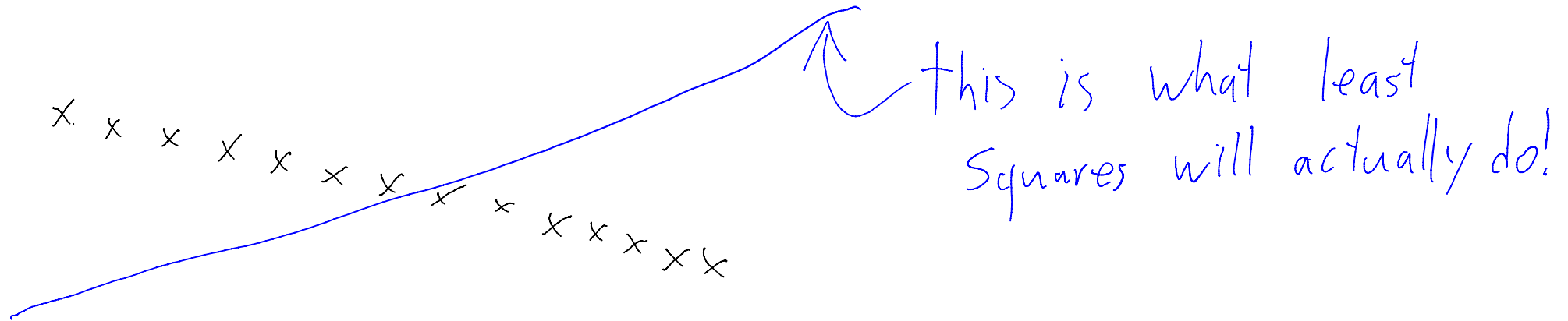
$x$  ← "outlier": it's not like the others.



# Least Squares with Outliers

- Consider fitting least squares with an **outlier** in the labels:

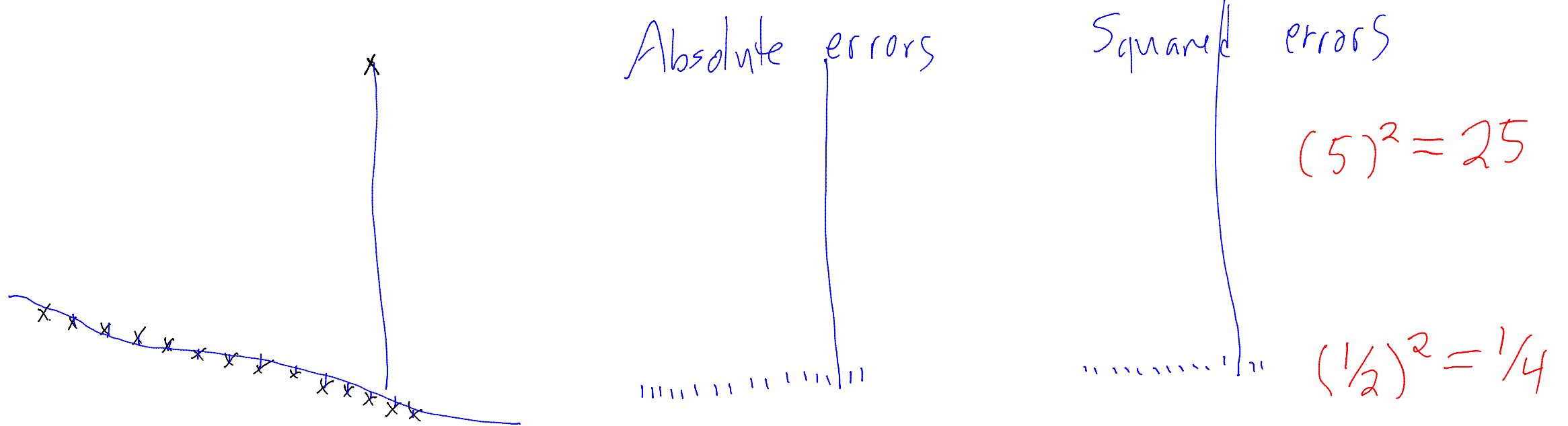
$x$  ← "outlier": it's not like the others.



- **Least squares is very sensitive to outliers.**

# Least Squares with Outliers

- Squaring error shrinks small errors, and **magnifies large errors**:



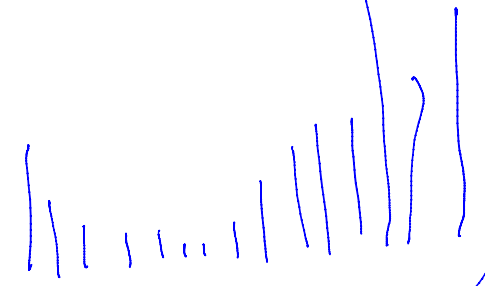
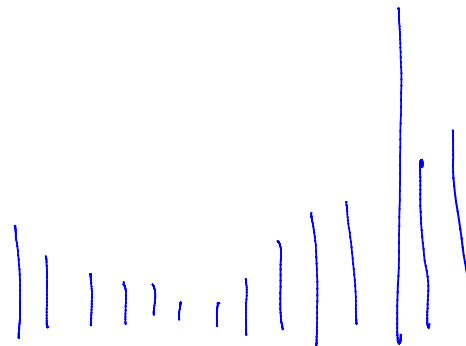
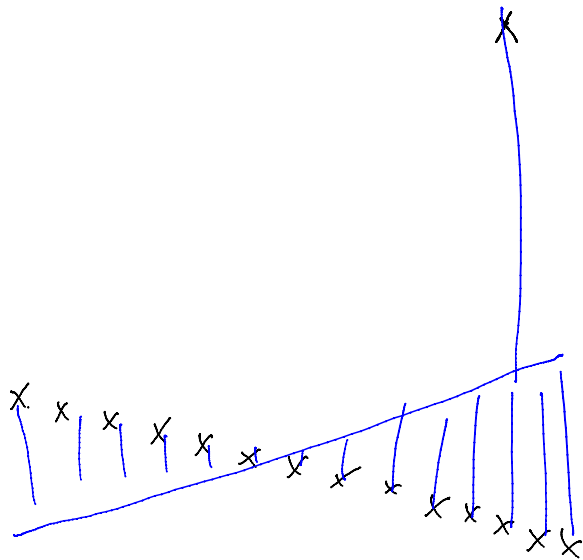
- Outliers (large error) influence 'w' much more than other points.

# Least Squares with Outliers

- Squaring error shrinks small errors, and **magnifies large errors:**

Absolute Errors:

Squared Errors:



sum of these is smaller.

- Outliers (large error) influence 'w' much more than other points.
  - Good if outlier means 'plane crashes', bad if it means 'data entry error'.

# Robust Regression

- **Robust regression** objectives put less focus on far-away points.
- For example, just use **absolute error**:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n |w^T x_i - y_i|$$

- Now decreasing **'small' and 'large' errors is equally important.**
- In matrix notation, we can write this as minimizing **L1-norm**:

$$\|r\|_1 = \sum_{i=1}^n |r_i|$$

Let "residual" vector

'r' have elements

$$r_i = w^T x_i - y_i$$

$$\text{so } r = Xw - y$$

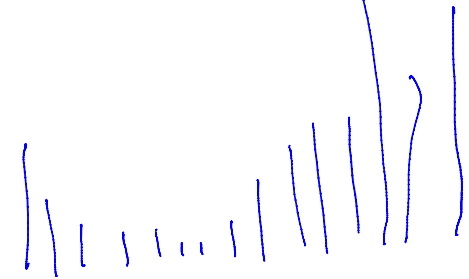
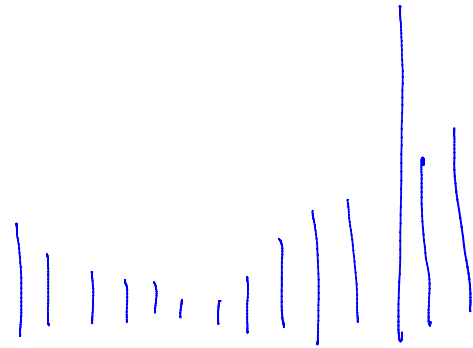
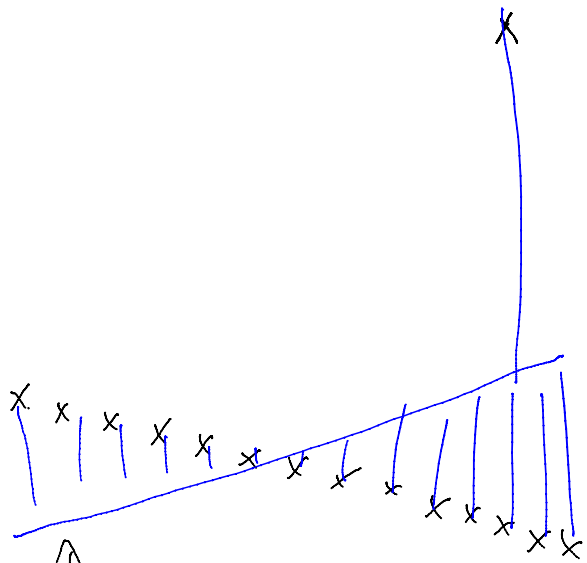
$$\operatorname{argmin}_{w \in \mathbb{R}^d} \|Xw - y\|_1$$

# Squared Error vs. Absolute Error

- Comparing squared error absolute error:

Absolute Errors:

Squared Errors:

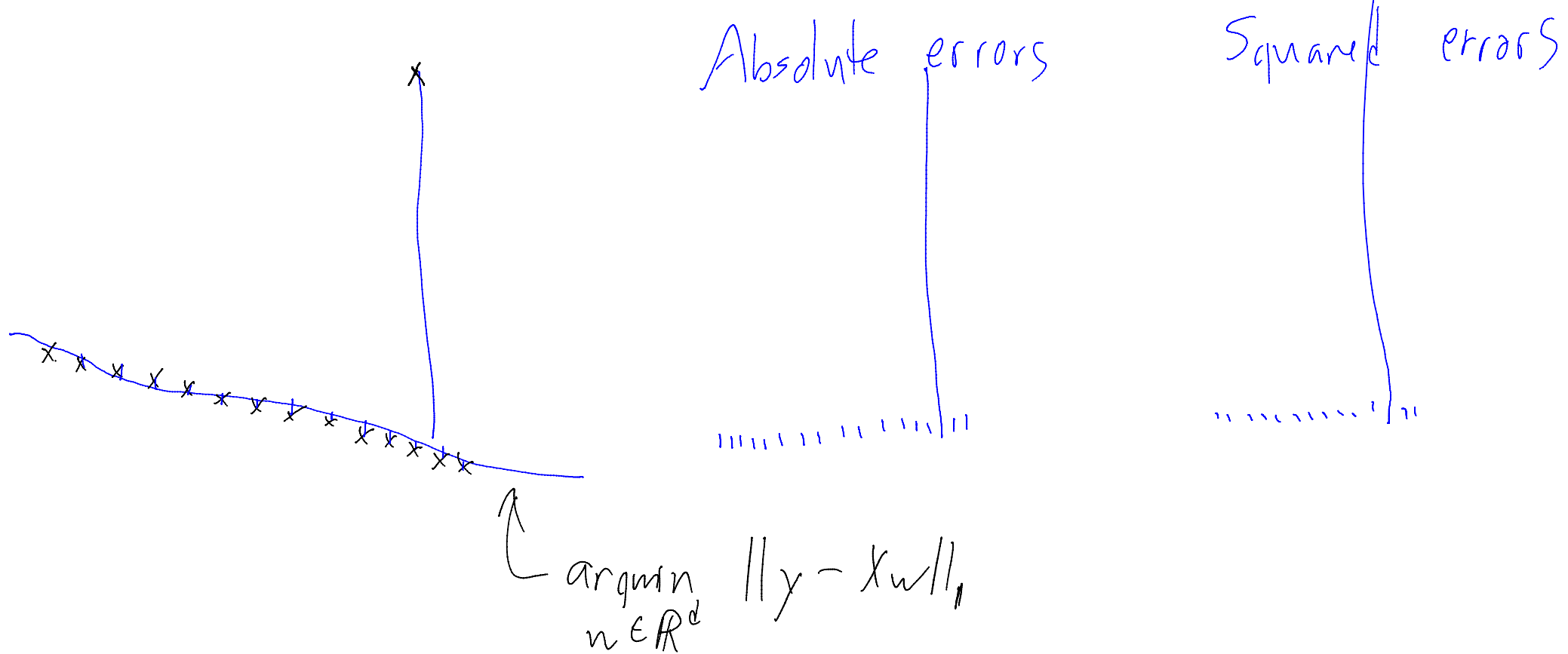


$\text{argmin}_{w \in \mathbb{R}^d} \frac{1}{2} \|y - Xw\|^2$



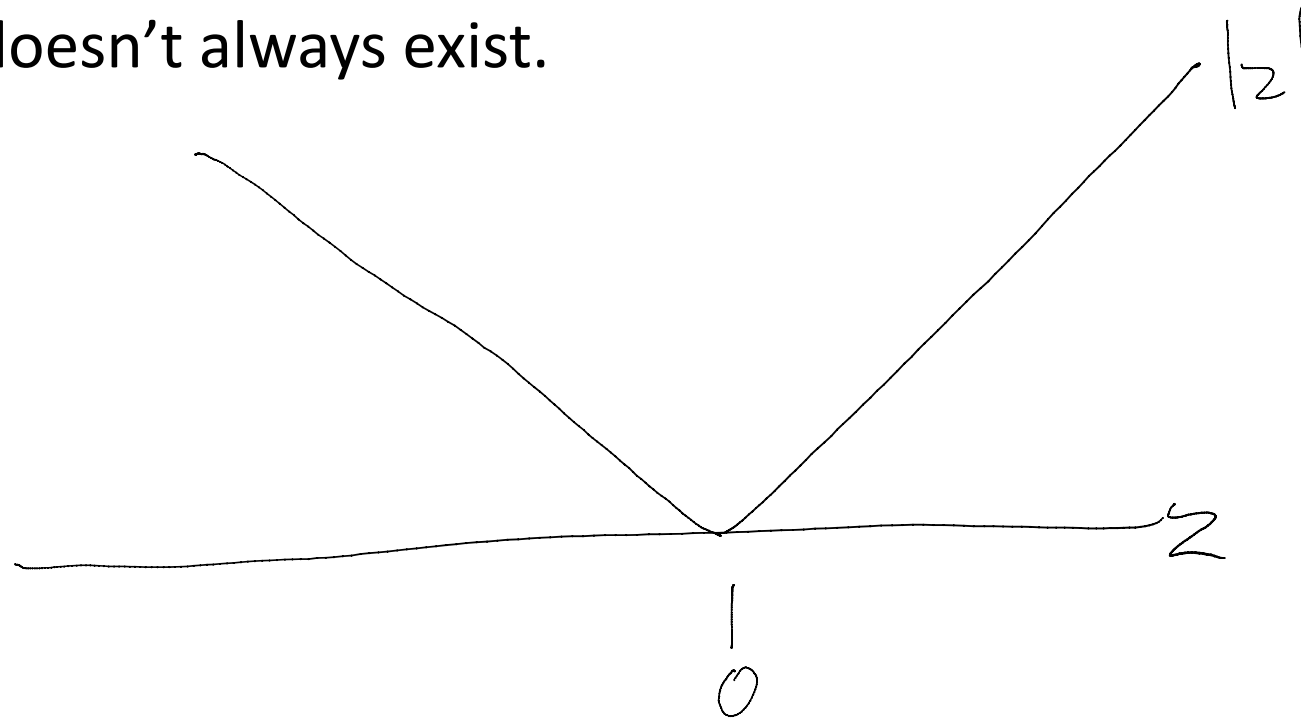
# Squared Error vs. Absolute Error

- Comparing squared error absolute error:



# Regression with the L1-Norm

- Unfortunately, **minimizing the absolute error is harder:**
  - Gradient doesn't always exist.



Why not smooth?

- E.g., Huber loss.
- Introduces parameter.
- Smoothing might make the problem harder.

- Generally, **harder to minimize non-smooth** than smooth functions.
- But we can formulate minimize absolute error as a **linear program**.

# Converting into Constrained Problems

- Key observation:
  - Absolute value is **maximum of smooth functions**:  $|w| = \max\{w, -w\}$
- We can convert to minimizing **smooth function with constraints**:
  1. Replace **maximum with new variable**, constrained to upper-bound max.
  2. Replace individual constraint with **constraint for each element** of max.

$$\operatorname{argmin}_{w \in \mathbb{R}} |w| \iff \operatorname{argmin}_{w \in \mathbb{R}} \max\{w, -w\} \iff \operatorname{argmin}_{w \in \mathbb{R}, r \in \mathbb{R}} r, \text{ s.t. } r \geq \max\{w, -w\}$$

We must have  $r = |w|$  at solution, otherwise constraints are violated or we could decrease 'r'.

$$\operatorname{argmin}_{w \in \mathbb{R}, r \in \mathbb{R}} r, \text{ s.t. } r \geq w \text{ and } r \geq -w$$

# Minimizing Absolute Error as Linear Program

- We can apply the same steps to a **sum of max functions**:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n |w^T x_i - y_i| \Leftrightarrow \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \max\{w^T x_i - y_i, y_i - w^T x_i\}$$

$$\Leftrightarrow \operatorname{argmin}_{w \in \mathbb{R}^d, r \in \mathbb{R}^n} \sum_{i=1}^n r_i \quad \text{subject to } r_i \geq \max\{w^T x_i - y_i, y_i - w^T x_i\} \text{ for all 'i'}$$

$$\Leftrightarrow \operatorname{argmin}_{w \in \mathbb{R}^d, r \in \mathbb{R}^n} \sum_{i=1}^n r_i \quad \text{subject to } r_i \geq w^T x_i - y_i \text{ and } r_i \geq y_i - w^T x_i \text{ for all 'i'}$$

- This is a **linear program**:
  - Minimizing a **linear function subject to linear constraints**.
  - We can efficiently solve 'medium-sized' linear programs: Matlab's 'linprog'.
  - There are other linear program formulations of this problems.

(pause)

# Motivation: Identifying Important E-mails

- We have a big collection of e-mails:
  - Marked as ‘important’ if user took some action based on them.



- We want to write a program that identifies ‘important’ e-mails?
- Can we formulate as supervised learning?

# Supervised Learning Representation for E-mails

- For e-mail 'i', the target label  $y_i$  is binary:
  - +1: “e-mail is important”.
  - -1: “e-mail is not important”.
  - **Classification**: supervised learning with discrete labels.
- What are the right **features**  $x_i$  (basis) for e-mails?
  - Use **bag of words**:
    - “CPSC”, “Expedia”, “vicodin”.
    - Binary “Expedia” feature is 1 if phrase “Expedia” is in the message, and 0 otherwise.
  - Could add phrases:
    - “you’re a winner”, “CPSC 540”.
  - Could add regular expressions:
    - <recipient name>, <sender domain == “mail.com”>

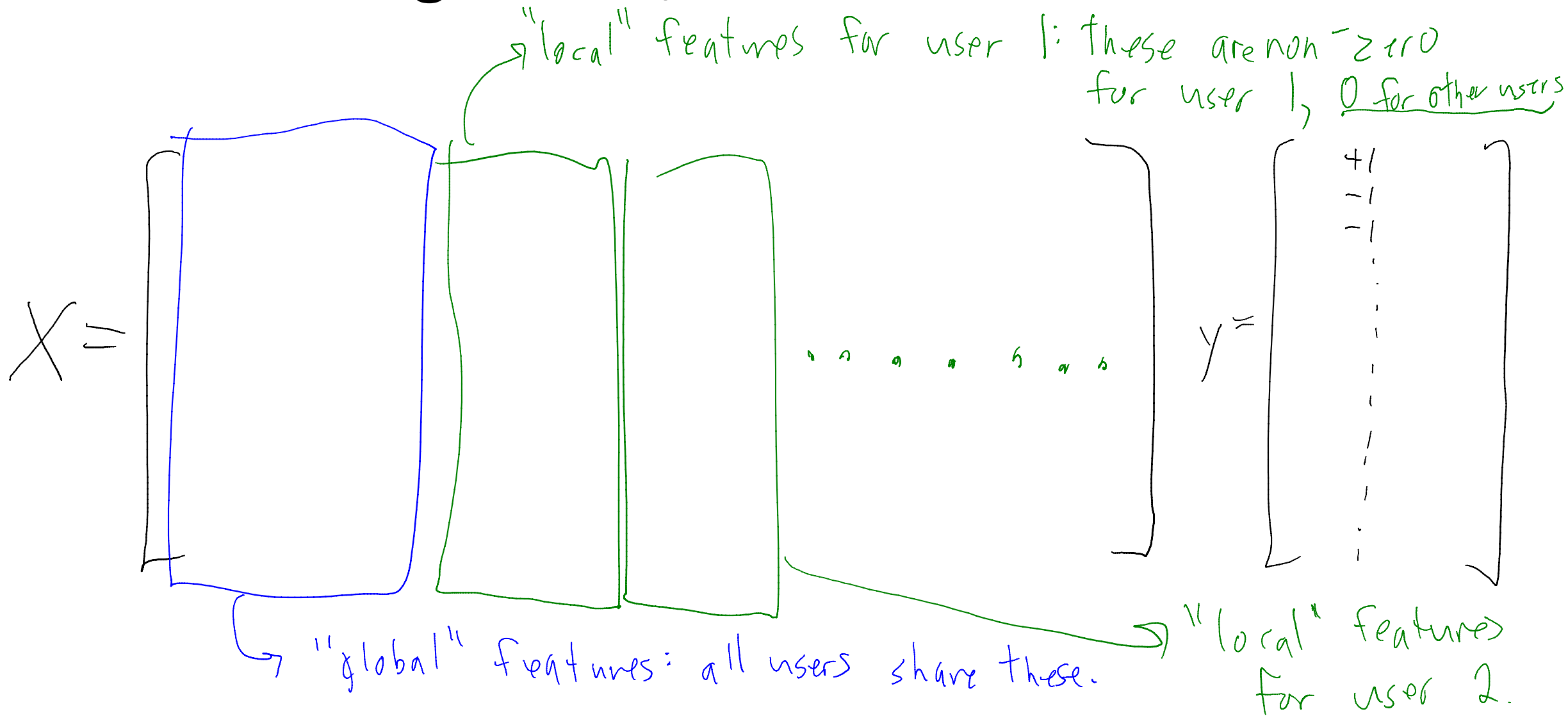
# Supervised Learning Representation for E-mails

$$X = \begin{matrix} & \text{"CPSC"} & \text{"Expedia"} & \text{"Vicodin"} & \langle \text{recipient name} \rangle \\ \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{matrix} & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{matrix} \quad Y = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

- Can we make **personalized** predictions?
  - Some messages ‘universally’ important:
    - “This is your mother, something terrible happened, give me a call ASAP.”
  - Some messages may be important to one user but not others.



# The Big Global/Local Feature Table



# Predicting Importance of E-mail For New User

- Consider a new user:
  - Start out with no information about them.
  - Use **global** features to predict what is important to generic user.

$$\hat{y}_i = \text{sign}(w_g^T x_g) \rightarrow \text{features/parameters shared across all users.}$$

- With more data, update **global** features and **user's local** features:
  - **Local** features make prediction *personalized*.

$$\hat{y}_i = \text{sign}(w_g^T x_g + w_u^T x_u) \rightarrow \text{features/parameters specific to user "u"}$$

- G-mails system: classification with **logistic regression**.

# Classification Using Regression?

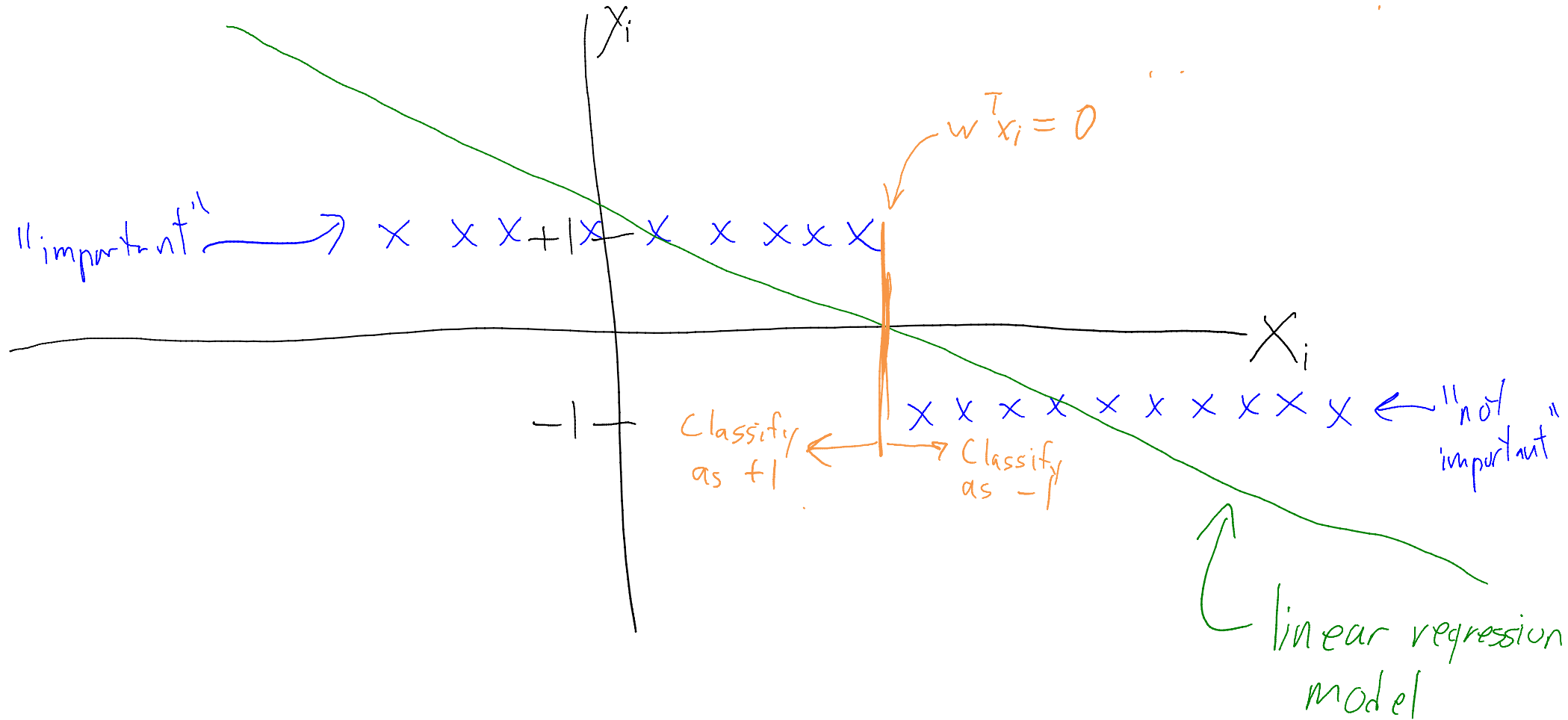
- Usual approach to do **binary classification with regression**:
  - Code  $y_i$  as '+1' for one class and '-1' for the other class.
- Fit a linear regression model:

$$\begin{aligned}\hat{f}_i &= w_1 x_{i1} + w_2 x_{i2} + \dots + w_d x_{id} \\ &= w^T x_i\end{aligned}$$

- Classify by **take the sign** (i.e., closer '-1' or '+1'?):

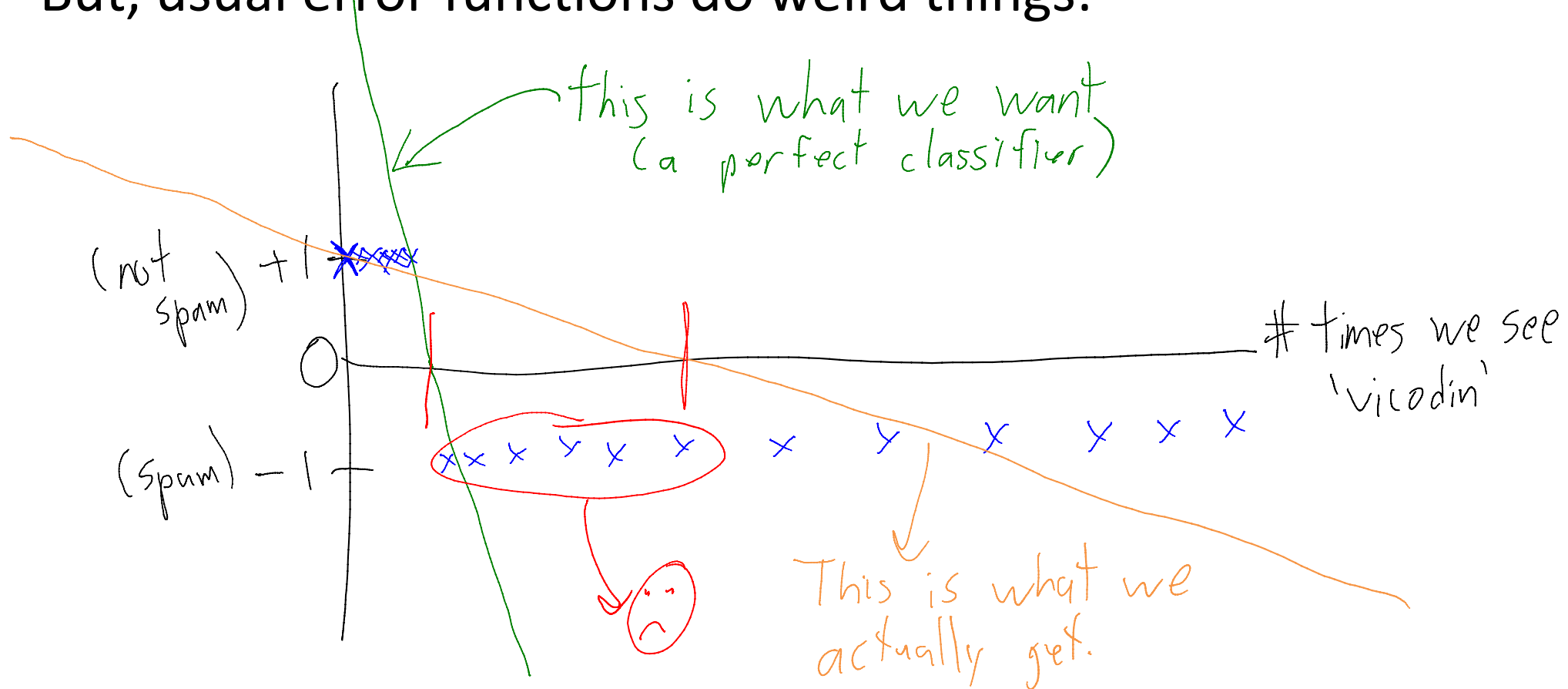
$$\hat{y}_i = \text{sign}(w^T x_i).$$

# Classification using Regression



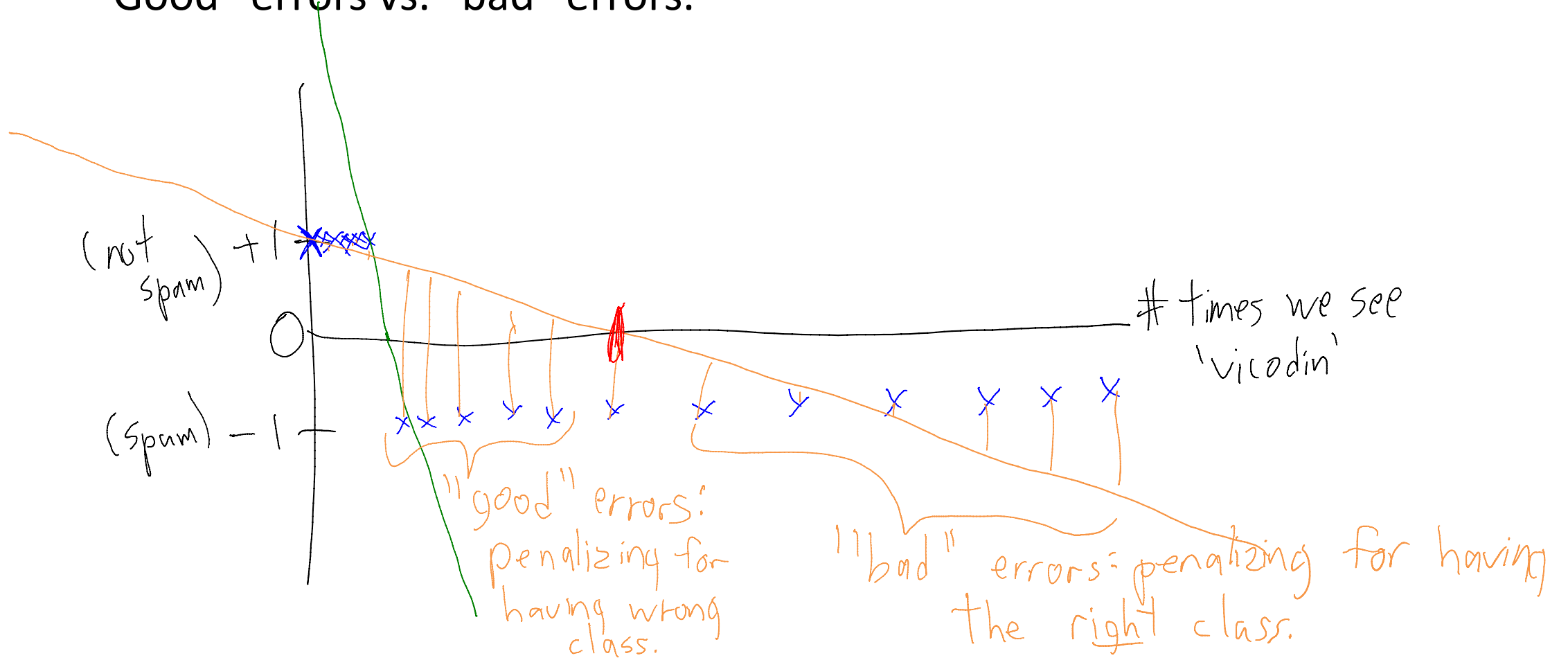
# Classification using Regression

- Can use our tricks (e.g., RBF basis, regularization) for classification.
- But, usual error functions do weird things:



# Classification Using Regression

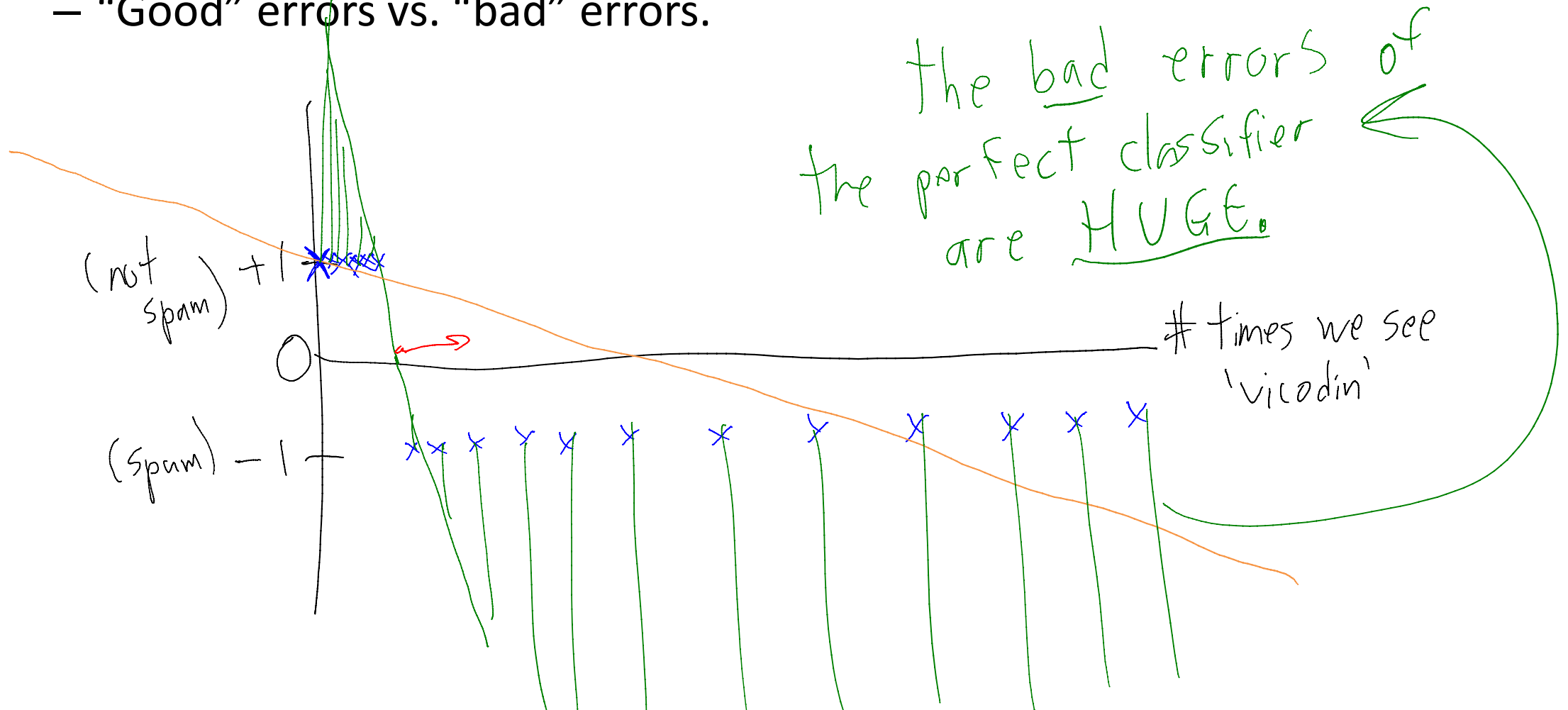
- What went wrong?
  - “Good” errors vs. “bad” errors.



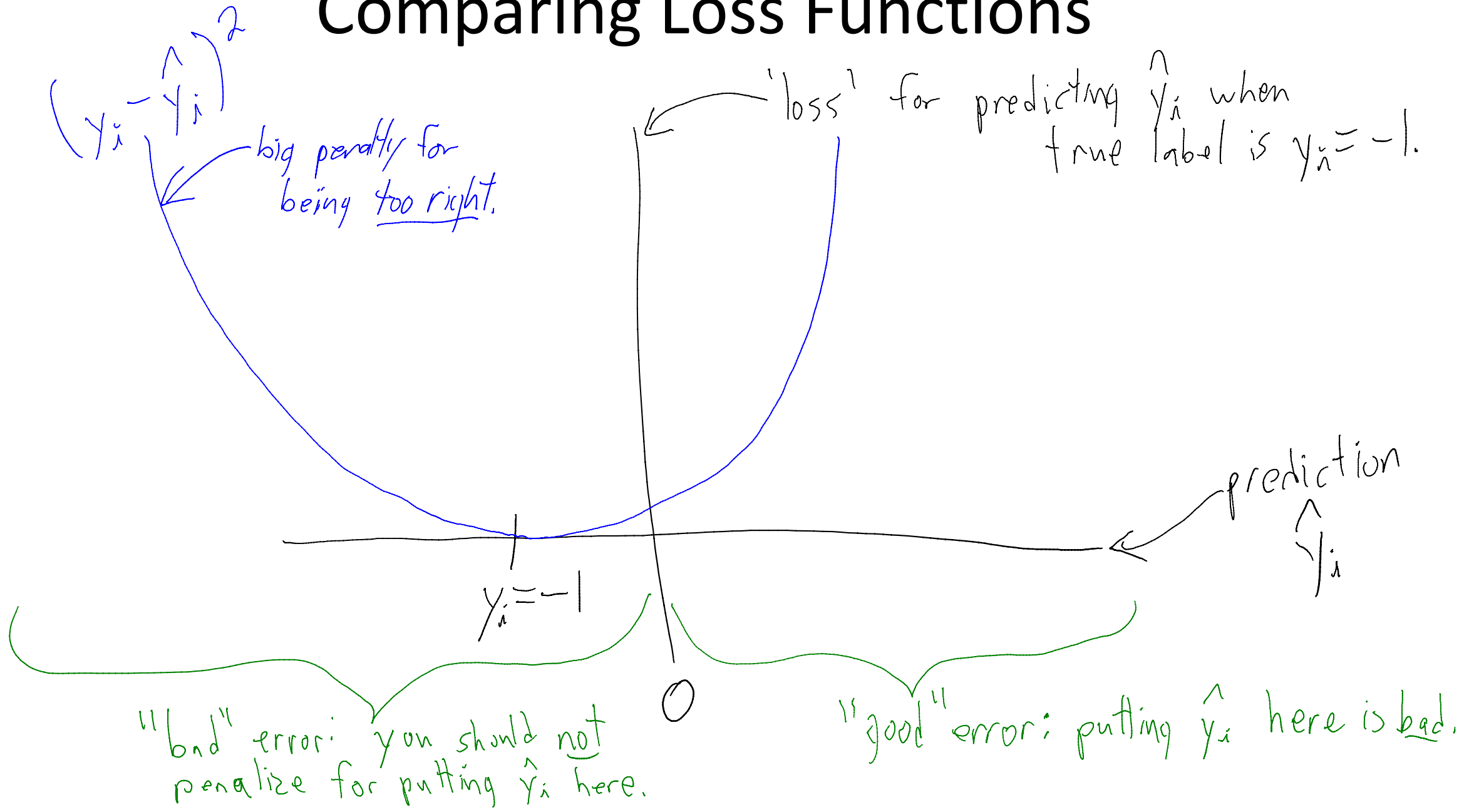
# Classification Using Regression

Squared error  
vs. sign

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

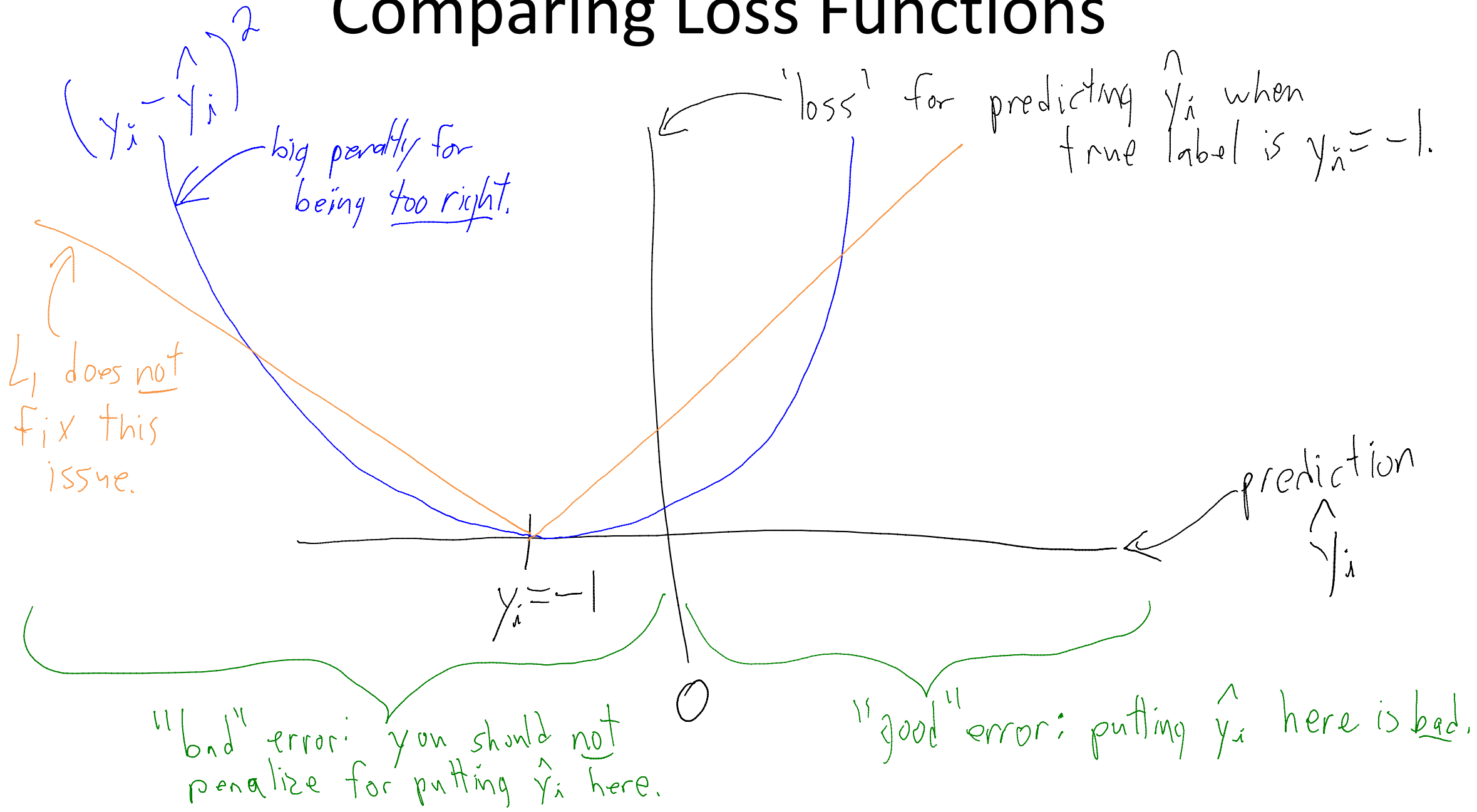


# Comparing Loss Functions

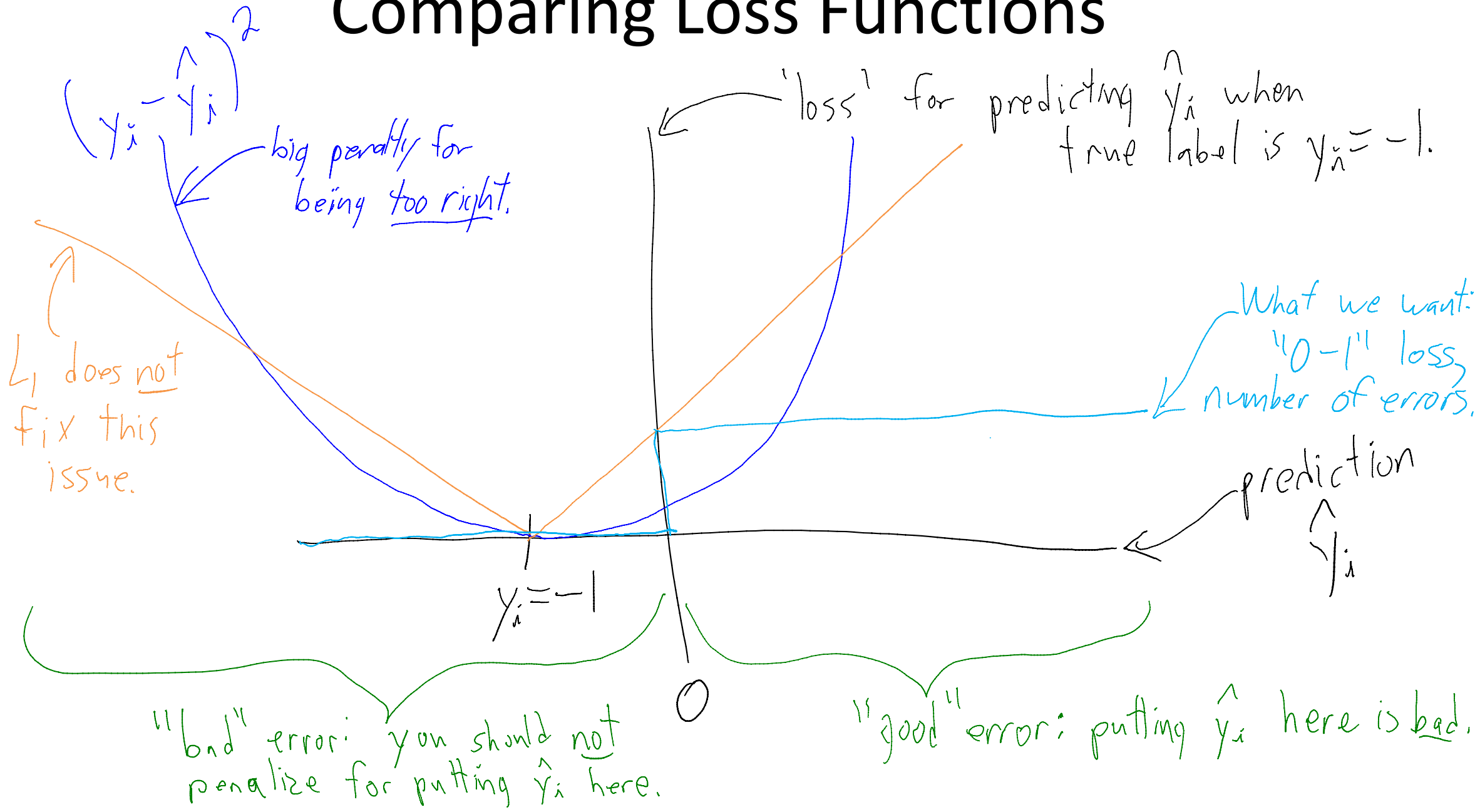




# Comparing Loss Functions



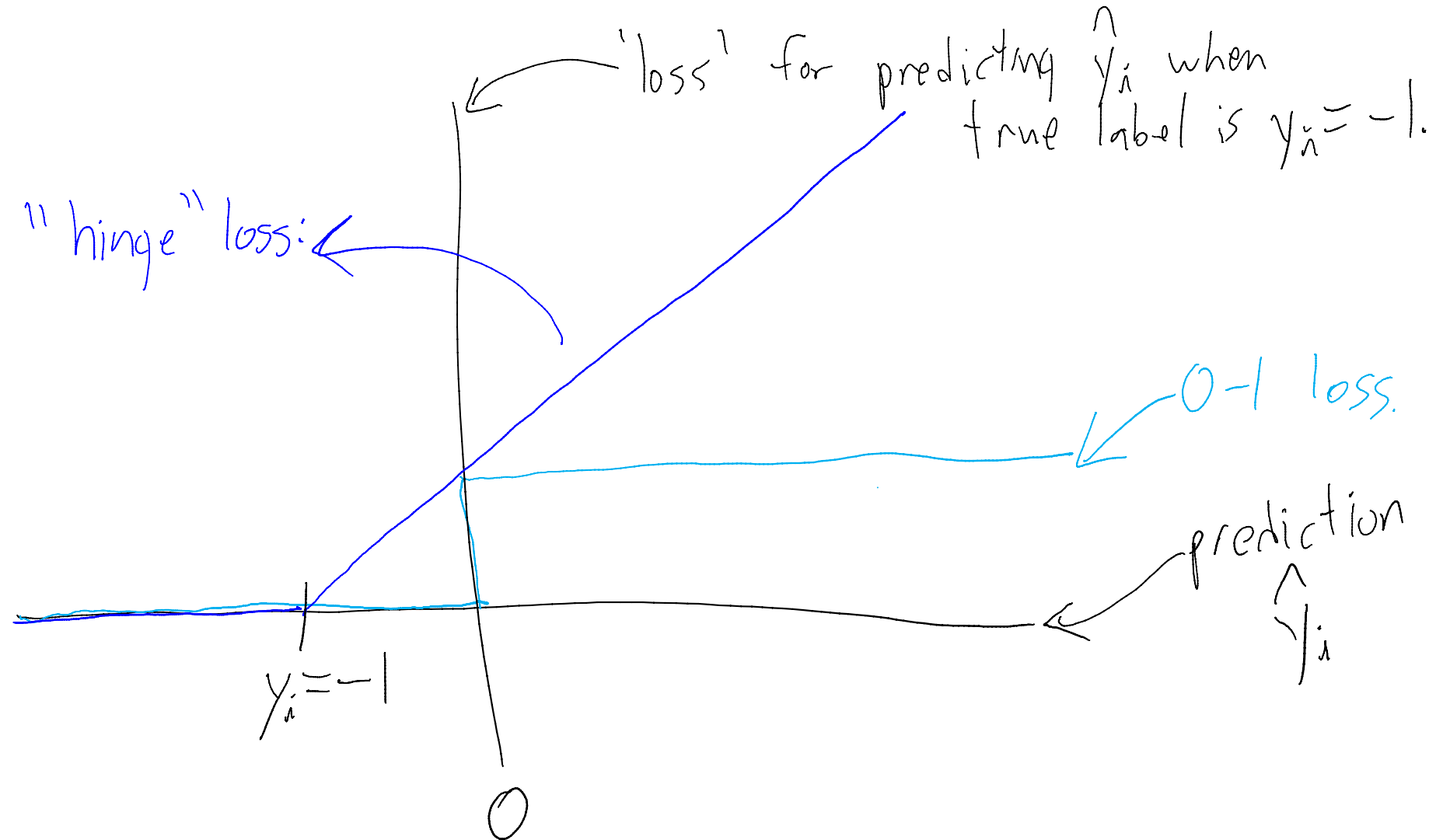
# Comparing Loss Functions



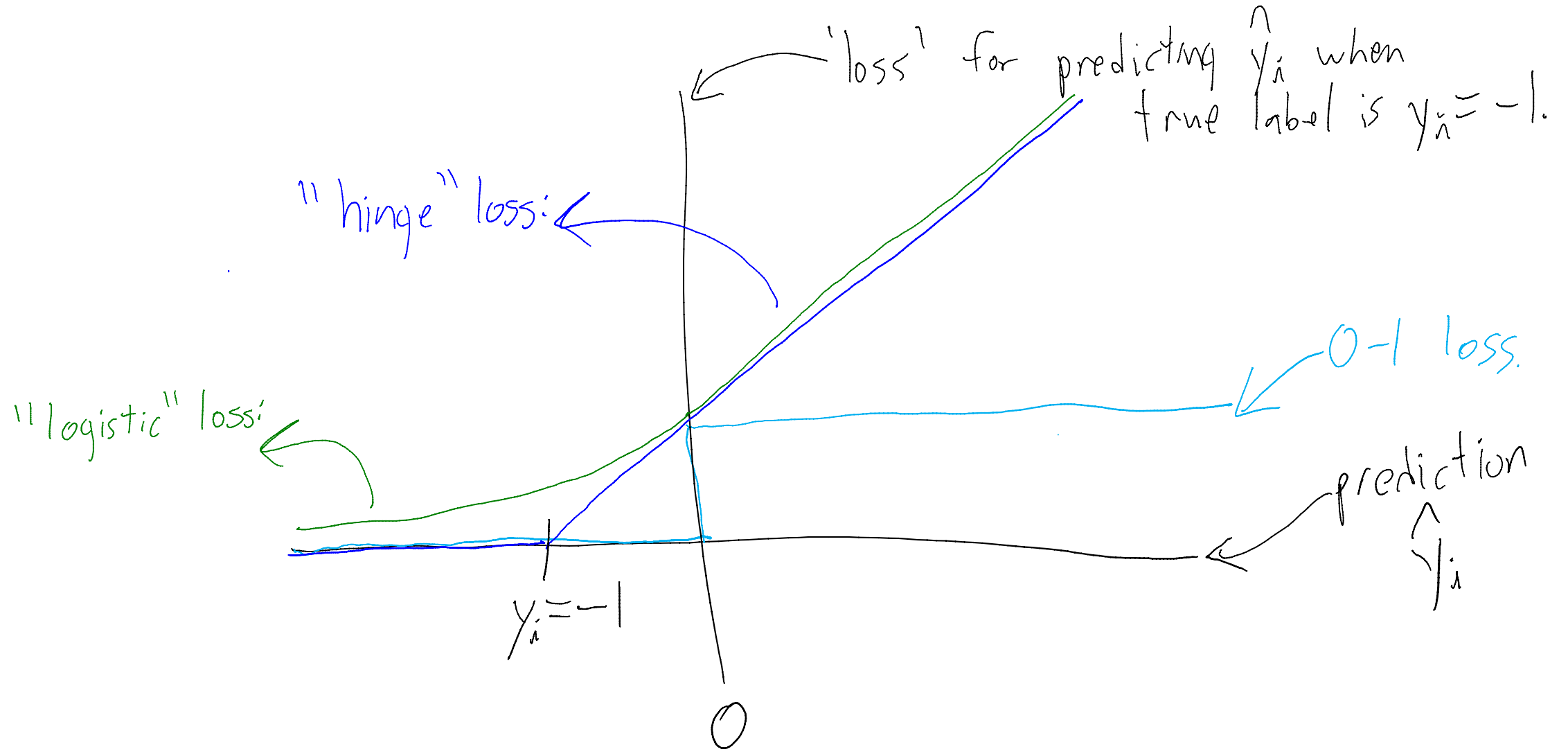
# 0-1 Loss Function and Tractable Approximations

- The **0-1 loss function** is the number of errors after taking the sign.
  - If a perfect classifier exists, you can find one as a linear program.
  - Otherwise, it's **NP-hard** to minimize 0-1 loss:
    - We do not expect that efficient algorithms exist.
- Tractable alternatives to 0-1 loss:
  - **Hinge loss**: upper-bound on 0-1 loss that can be written as linear program.
  - **Logistic loss**: differentiable function similar to hinge loss.

# 0-1 Loss Function and Tractable Approximations



# 0-1 Loss Function and Tractable Approximations



# Hinge Loss and Support Vector Machines

- Hinge loss is given by:

$$\operatorname{Argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \max \{0, 1 - \underline{y_i w^T x_i}\}$$

- Can be written as a **linear program** using our max trick.
- Solution will be a perfect classifier, if one exists.
- **Support vector machine (SVM)** is hinge loss with L2-regularization.

$$\operatorname{Argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \max \{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2} \|w\|^2$$

- Can be written as a **quadratic program** using our max trick
  - Quadratic objective with linear constraints.
- Solution will be perfect classifier, if one exists and  $\lambda$  is small enough.
- **Maximizes margin**: maximizes distance of data to decision boundary.

# Logistic Regression

- **Logistic regression** minimizes logistic loss:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

- You can/should also add regularization:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i)) + \frac{\lambda}{2} \|w\|^2$$

- These can't be written as linear/quadratic programs:
  - But they're **differentiable**: we'll discuss how to solve them next time.

# Logistic Regression and SVMs

- SVMs and logistic regression are used EVERYWHERE!
- Why?
  - Training and testing are both fast, even for “large-scale” problems.
  - It is easy to understand what the weights ‘ $w_j$ ’ mean.
  - With high-dimensional features and regularization, often good test error.
  - Otherwise, often good test error with RBF basis and regularization.
  - For logistic regression, predictions have probabilistic interpretation.

If  $p(y_i = +1 | w, x_i) = \text{sigm}(w^T x_i)$  then minimizing logistic loss corresponds to maximum likelihood estimate.

$\text{sigm}(z) = \frac{1}{1 + \exp(-z)}$



# Maximum Likelihood Estimation

- Maximum likelihood estimate (MLE) in an abstract setting:
  - We have a dataset 'D'.
  - We want to pick a model 'h' from among set of models H.
  - We define the likelihood as a probability density  $p(D | h)$ .
  - We choose the model 'h' that maximizes the likelihood:

$$\hat{h} = \operatorname{argmax}_{h \in H} p(D | h)$$

- If the data consists of 'n' IID samples 'D<sub>i</sub>', then we equivalently have:

$$\hat{h} = \operatorname{argmax}_{h \in H} \prod_{i=1}^n p(D_i | h) \quad \text{since independence implies } p(D | h) = \prod_{i=1}^n p(D_i | h)$$

- MLE has appealing properties as  $n \rightarrow \infty$  (take STAT 560/561)

# Negative Log-Likelihood

- In linear regression we predict  $y_i$  conditioned on  $x_i$ :

Conditional likelihood  $p(y_i | w, x_i)$

- MLE estimate of 'w' with IID data is:

$$\operatorname{argmax}_{w \in \mathbb{R}^d} p(y | w, X) \iff \operatorname{argmax}_{w \in \mathbb{R}^d} \prod_{i=1}^n p(y_i | w, x_i)$$

- We can equivalently **minimize negative log-likelihood (NLL)**:

$$\operatorname{argmax}_{w \in \mathbb{R}^d} \log \left( \prod_{i=1}^n p(y_i | w, x_i) \right) \iff \operatorname{argmax}_{w \in \mathbb{R}^d} \sum_{i=1}^n \log(p(y_i | w, x_i))$$

*(if  $a > b$  then  $\log(a) > \log(b)$ )*  
*So argmax is the same*

*( $\log(ab) = \log(a) + \log(b)$ )*

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log(p(y_i | w, x_i)) \iff \operatorname{argmax} f$$

*↕*  
 $\operatorname{argmin} -f$

# MLE Interpretation of Logistic Regression

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

then we have  $p(y_i = -1 | w, x_i) = 1 - p(y_i = +1 | w, x_i)$   
 $= \frac{1}{1 + \exp(w^T x_i)}$

We can write both cases as

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

Compute MLE by minimizing negative log-likelihood:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log(p(y_i | w, x_i)) = \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log\left(\frac{1}{1 + \exp(-y_i w^T x_i)}\right)$$

$$= \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i))$$

Logistic loss

$$\left( -\log\left(\frac{1}{2}\right) = -\log(1) + \log(2) \right. \\ \left. = \log(2) \right)$$

# MLE Interpretation of Least Squares

Consider continuous  $y_i$  with Gaussian likelihood:  $p(y_i | w, x_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)$

We also sometimes write this as  $y_i \sim N(w^T x_i, \sigma^2)$   $\propto \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)$   
↓  
equal up to constant not depending on  $y_i$

MLE with Gaussian likelihood is least squares:

$$\begin{aligned} \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log(p(y_i | w, x_i)) &\iff \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log\left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)\right) \\ &\iff \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \underbrace{-\log\left(\frac{1}{\sigma\sqrt{2\pi}}\right)}_{\text{constant}} + \frac{(w^T x_i - y_i)^2}{2\sigma^2} \quad (\log(\exp(z)) = z) \\ &\iff \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n \underbrace{\frac{1}{2\sigma^2}}_{\text{constant}} (w^T x_i - y_i)^2 \iff \operatorname{argmin}_{w \in \mathbb{R}^d} \|Xw - y\|^2 \end{aligned}$$

# Discussion: Probabilistic Interpretation

- Why is **probabilistic interpretation** important?

- We can return a **probabilistic prediction**:

Instead of  $\hat{y}_i = 1$ , say that  $p(\hat{y}_i = 1 | w, x_i) = 99\%$   
or  $p(\hat{y}_i = 1 | w, x_i) = 51\%$

- For complicated  $y_i$ , it may be **easier to define probability** than loss.

- We can talk about **maximizing utility**:

Predict / True	True 'spam'	True 'not spam'
Predict 'spam'	TP: 0	FP: 100
Predict 'not spam'	FN: 10	TN: 0

Predict "not spam"  
even if  $\hat{y}_i = \text{"spam"}$   
if expected cost of  
"not spam" is lower.

$$E [ C(\hat{y}_i = \text{spam}) ] = p(y_i = \text{spam} | x_i) C(\hat{y}_i = \text{spam}, y_i = \text{spam}) \\ + p(y_i = \text{not spam} | x_i) C(\hat{y}_i = \text{spam}, y_i = \text{not spam})$$

# Problem with Maximum Likelihood

- Maximum likelihood estimate:

$$\operatorname{argmax}_{h \in H} p(D|h)$$

- Data viewed as random variable, model comes from fixed family.
- A problem with MLE:
  - data could be very likely in some **very unlikely model** from family.
  - E.g., complex model overfits by memorizing the data.

# Maximum a Posteriori (MAP) Estimation

- Maximum a posteriori (MAP) estimate maximizes reverse:

$$\operatorname{argmax}_{h \in H} p(h|D)$$

- Model is a random variable, and we need to **find most likely model**.

- Using Bayes' rule, we have  $p(h|D) = \frac{p(D|h)p(h)}{p(D)} \propto p(D|h)p(h)$ .

$$\operatorname{argmax}_{h \in H} \underbrace{p(h|D)}_{\text{posterior}} \iff \operatorname{argmax}_{h \in H} \underbrace{p(D|h)}_{\text{likelihood}} \underbrace{p(h)}_{\text{prior}}$$

- Prior  $p(h)$  is 'belief' that 'h' is the correct model before seeing data:
  - Can take into account that complex models are likely to overfit.

# MAP Estimation and Regularization

- MAP is equivalent to minimizing NLL plus negative log-prior.

$$\begin{aligned} \operatorname{argmax}_{w \in \mathbb{R}^d} p(w | y, X) &\iff \operatorname{argmax}_{w \in \mathbb{R}^d} p(y | w, X) p(w | X) && \text{(Bayes' rule)} \\ &\iff \operatorname{argmax}_{w \in \mathbb{R}^d} p(y | w, X) p(w) && \text{(Assume } w \text{ and } X \text{ are independent)} \\ &\iff \operatorname{argmax}_{w \in \mathbb{R}^d} \prod_{i=1}^n [p(y_i | w, x_i)] p(w) && \text{(IID assumption)} \\ &\iff \operatorname{argmax}_{w \in \mathbb{R}^d} \log \left( \prod_{i=1}^n [p(y_i | w, x_i)] p(w) \right) && \text{(log is monotonic)} \\ &\iff \operatorname{argmax}_{w \in \mathbb{R}^d} \sum_{i=1}^n \log(p(y_i | w, x_i)) + \log(p(w)) && \text{(log}(abc) = \log(a) + \log(b) + \log(c)) \\ &\iff \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log(p(y_i | w, x_i)) - \log(p(w)) && \text{(argmax } f \equiv \operatorname{argmin} -f) \end{aligned}$$



# MAP Estimation and Regularization

- So MAP estimation looks like fitting regularized loss function:

$$\operatorname{argmax}_{w \in \mathbb{R}^d} p(w | y, X) \iff \operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log(p(y_i | w, x_i)) - \log(p(w))$$

- L2-regularization corresponds to independent Gaussian prior:

$$w_j \sim \mathcal{N}(0, \lambda^{-1}) \iff p(w_j | \lambda) \propto \exp\left(-\frac{\lambda}{2} w_j^2\right)$$

$$p(w | \lambda) \propto \prod_{j=1}^d \exp\left(-\frac{\lambda}{2} w_j^2\right) = \exp\left(-\frac{\lambda}{2} \sum_{j=1}^d w_j^2\right) = \exp\left(-\frac{\lambda}{2} \|w\|^2\right)$$

So MAP estimate is:

$$\operatorname{argmin}_{w \in \mathbb{R}^d} \sum_{i=1}^n -\log(p(y_i | w, x_i)) + \frac{\lambda}{2} \|w\|^2$$

# Summary

- **Radial basis functions**: non-parametric universal basis.
- **Robust regression models**: more suitable when we have outliers.
- **Converting non-smooth** problems to constrained smooth problems.
- **SVMs and logistic regression**: more suitable losses for classification.
- **MLE and MAP**: probabilistic interpretation to losses/regularizers.
  
- Next time:
  - Why is 0-1 hard but logistic regression easy?
  - How do we solve “large-scale” problems?