CPSC 340:
Machine Learning and Data Mining

Non-Parametric Feature Transforms

Summer 2021

Admin

- Midterm is tomorrow.
  - Manually-graded portion on Gradescope
    - 55 minutes
    - Handwritten or typeset
  - Auto-graded potion on Canvas
    - 45 minutes
    - Multiple choice
  - The two portions are equally weighted
- Please don’t ask broad questions on Piazza tomorrow
  - If you have issues with exams, etc., make a private post
- Assignment 4 is due Monday, June 7, 2021

In This Lecture

1. Standardization (5 minutes)
2. Gaussian RBF (20 minutes)
3. Linear Classifiers Intro (20 minutes)

Last Time: Regularization

- L0-regularization (AIC, BIC, Mallow's Cp, Adjusted R^2, ANOVA):
  - Adds penalty on the number of non-zeros to select features.

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_0$$

- L2-regularization (ridge regression):
  - Adding penalty on the L2-norm of 'w' to decrease overfitting:

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

- L1-regularization (LASSO):
  - Adding penalty on the L1-norm decreases overfitting and selects features:

$$f(w) = \|Xw - y\|^2 + \lambda \|w\|_1$$

Coming Up Next

STANDARDIZATION

Features with Different Scales

- Consider continuous 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 decision trees or naïve Bayes.
    - They only look at one feature at a time.
  - It doesn't matter for least squares:
    - w_j*(100 mL) gives the same model as w_j*(0.1 L) with a different w_j.

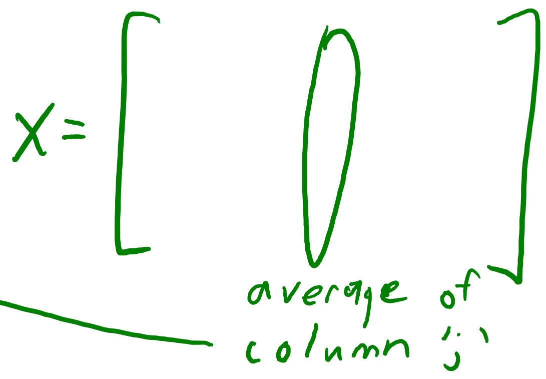
Features with Different Scales

- Consider continuous 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 matters for k-nearest neighbours:
    - "Distance" will be affected more by large features than small features.
  - It matters for regularized least squares:
    - Penalizing $(w_j)^2$ means different things if features 'j' are on different scales.

Standardizing Features



- It is common to standardize continuous features:
  - For each feature:
    1. Compute mean and standard deviation:

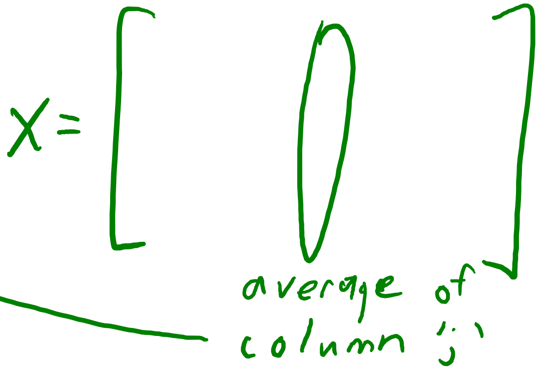
$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$$

    2. Subtract mean and divide by standard deviation ("z-score")

Replace x_{ij} with $\dfrac{x_{ij}-\mu_j}{\sigma_j}$

  - Now changes in 'w_j' have similar effect for any feature 'j'.
- How should we standardize test data?
  - Wrong approach: use mean and standard deviation of test data.
  - Training and test mean and standard deviation might be very different.
  - Right approach: use mean and standard deviation of training data.

Standardizing Features



- It is common to standardize continuous features:
  - For each feature:
    1. Compute mean and standard deviation:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$$

    2. Subtract mean and divide by standard deviation ("z-score")

Replace x_{ij} with $\frac{x_{ij}-\mu_j}{\sigma_j}$

  - Now changes in 'w_j' have similar effect for any feature 'j'.
- If we're doing 10-fold cross-validation:
  - Compute μ_j and σ_j based on the 9 training folds (e.g., average over 9/10s of data).
  - Standardize the remaining ("validation") fold with this "training" μ_j and σ_j.
  - Re-standardize for different folds.

Standardizing Target

- In regression, we sometimes standardize the targets y_i.
  - Puts targets on the same standard scale as standardized features:

$$\text{Replace } y_i \text{ with } \frac{y_i - \mu_y}{\sigma_y}$$

- With standardized target, setting w = 0 predicts ____________:
  - High regularization makes us predict closer to the average value.
- Again, make sure you standardize test data with the training stats.
- Other common transformations of y_i are logarithm/exponent:

$$\text{Use } \log(y_i) \text{ or } \exp(\gamma y_i)$$

  - Makes sense for geometric/exponential processes.

Coming Up Next

GAUSSIAN RADIAL BASIS FUNCTION

Weighted Sum of "Basis Functions"

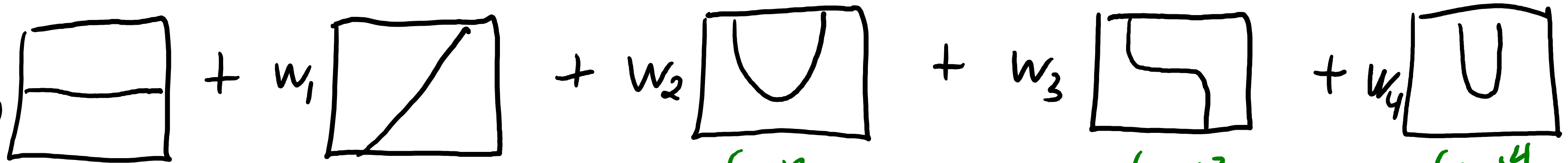
- Features for linear models with "change of basis" are functions

"basis function" $f_j : \mathbb{R} \rightarrow \mathbb{R}$

$$y_i = w_0 f_0(x_i) + w_1 f_1(x_i) + w_2 f_2(x_i) + \cdots + w_p f_p(x_i)$$ "on-the-fly" transformation

$$y_i = v_1 z^1 + v_2 z^2 + v_3 z^3 + \cdots + v_{p+1} z^{p+1}$$ "offline" transformation

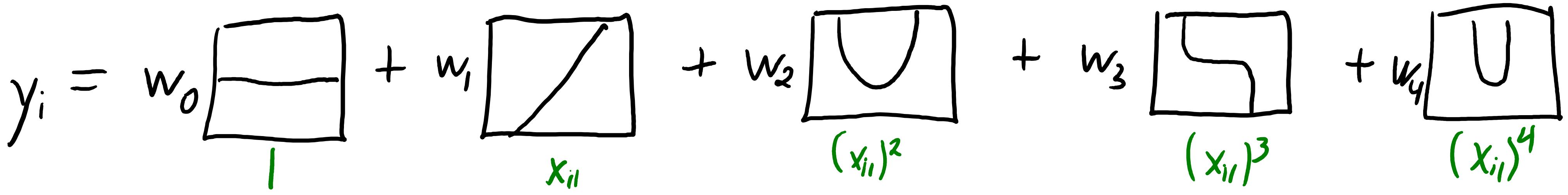
- We've been using linear models with polynomial bases:



$$y_i = \underbrace{w_0\, 1 + w_1\, x_{i1} + w_2\, (x_{i1})^2 + w_3\, (x_{i1})^3 + w_4\, (x_{i1})^4}_{\text{Weighted sum of basis functions}}$$

Parametric vs. Non-Parametric Transforms

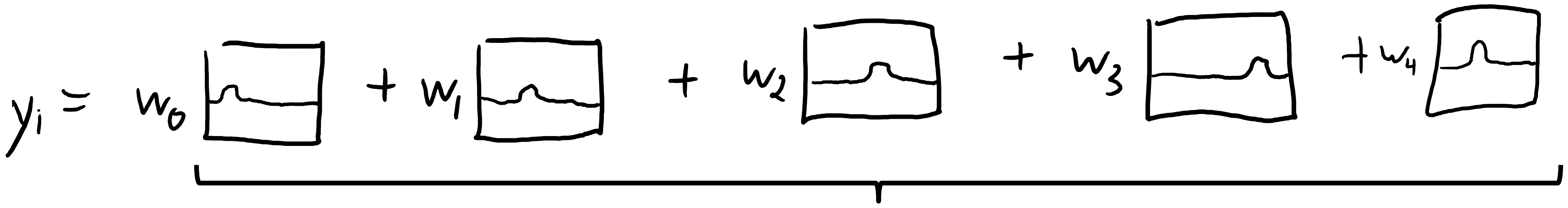
- We've been using linear models with polynomial bases:



- But polynomials are not the only possible bases:
  - Exponentials, logarithms, trigonometric functions, etc.
  - The right basis will vastly improve performance.
  - If we use the wrong basis, our accuracy is limited even with lots of data.
  - But the right basis may not be obvious.

Parametric vs. Non-Parametric Transforms

- Alternative: non-parametric bases:
  - Size of basis (number of features) grows with ‘n’.
  - Model gets more complicated as you get more data.
  - Can model complicated functions where you don’t know the right basis.
    - With enough data.
  - Classic example is “Gaussian RBFs” (“Gaussian” == “normal distribution”).

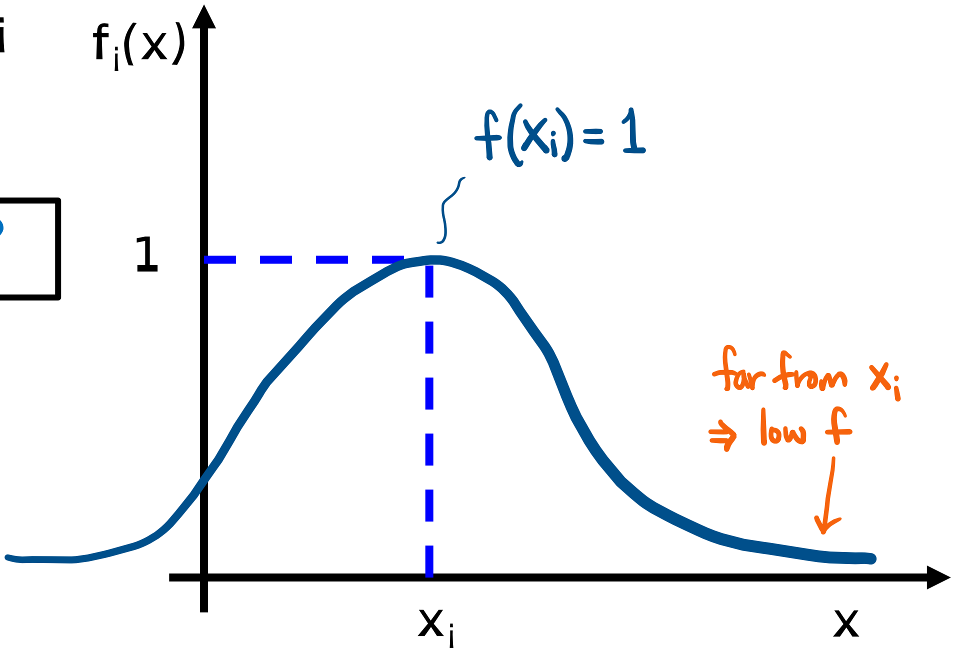


Weighted sum of basis functions

"Local Bumps"

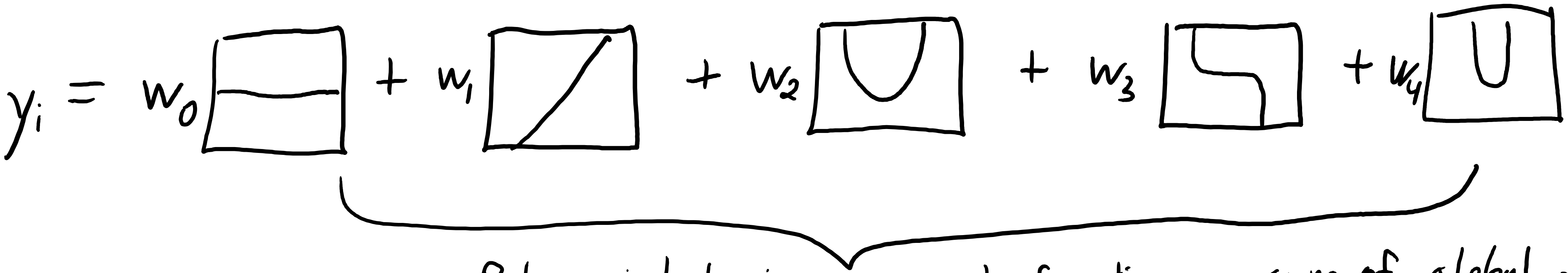
"basis function" $f_i : \mathbb{R} \to \mathbb{R}, \quad i = 1, 2, \cdots, n$

- Gaussian RBF's basis functions are "local bumps"
  - Each training example xi defines its own local bump
  - d=1: bell-curve centered at x_i

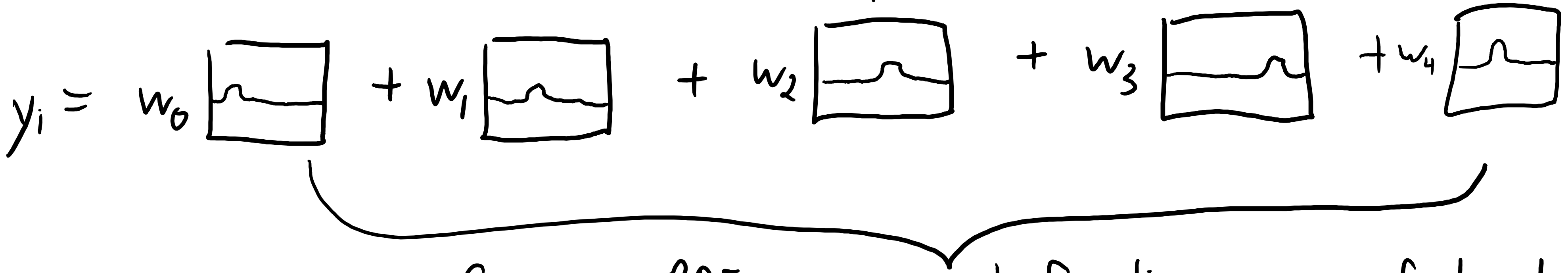


Q: How many local bumps are there?

Gaussian RBFs: A Sum of “Bumps”



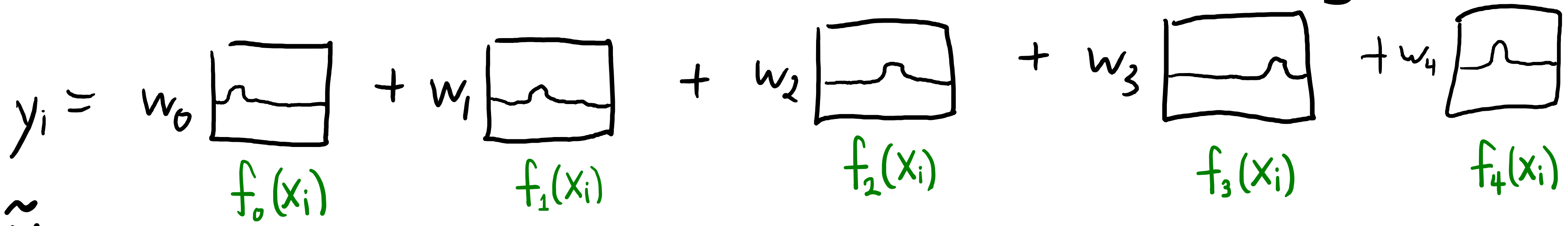
Polynomial basis represents function as sum of <u>global</u> polynomials.



Gaussian RBFs represent function as sum of <u>local</u> “bumps”

Q: How do we predict $\hat{y}_i$ for a test example x_i?

Prediction with Gaussian RBF Regression



$$\tilde{x}_i = 1.4$$

$$\hat{y}_i = w_0 \underbrace{f_0(1.4)}_{\text{bump 0 "score"}} + w_1 \underbrace{f_1(1.4)}_{\text{bump 1 "score"}} + w_2 \underbrace{f_2(1.4)}_{\text{bump 2 "score"}} + w_3 \underbrace{f_3(1.4)}_{\text{bump 3 "score"}} + w_4 \underbrace{f_4(1.4)}_{\text{bump 4 "score"}}$$

- Prediction is weighted combination of “bump scores”
- These “bump scores” are defined by training examples
  - an instance of “learned features”

"Change of Basis" for Gaussian RBFs

bump score based on x_1

bump score based on x_j (example j)

$$X = {}_{n}\underbrace{\begin{bmatrix} \\ x_j \\ \\ \end{bmatrix}}_{1} \leftarrow j \longrightarrow \underset{\text{RBF features}}{Z} = {}_{n}\underbrace{\begin{bmatrix} 1 & & & & \\ & 1 & & x_{ij} & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}}_{n} \leftarrow i$$

x_i's bump score based on x_j

Q: Why are there 1s on the diagonal?

Q: Does polynomial basis give us "learned features" too?

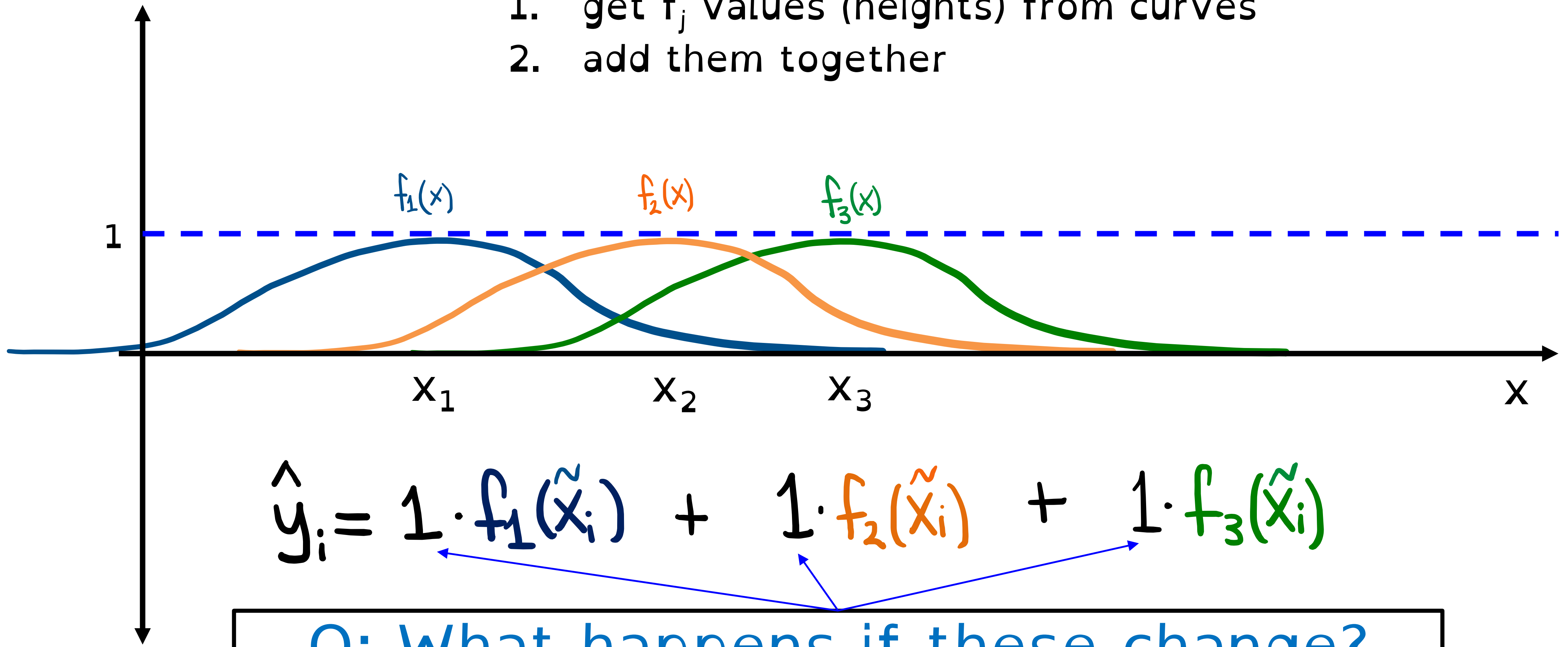
$$X = {}_{n}\underbrace{\begin{bmatrix} \\ \\ \end{bmatrix}}_{1} \longrightarrow \underset{\text{Poly Basis features}}{Z} = {}_{n}\underbrace{\begin{bmatrix} & & & & \\ & & & & \end{bmatrix}}_{p+1}$$

Gaussian RBFs: Universal Approximator

- Gaussian RBFs are universal approximators (compact subets of $\mathbb{R}^d$)
  - Enough bumps can approximate any continuous function to arbitrary precision.
  - Achieve optimal test error as ‘n’ goes to infinity.

Visualizing RBF Regression

- To predict $\hat{y}_i$ from $\tilde{x}_i$,
  1. get f_j values (heights) from curves
  2. add them together

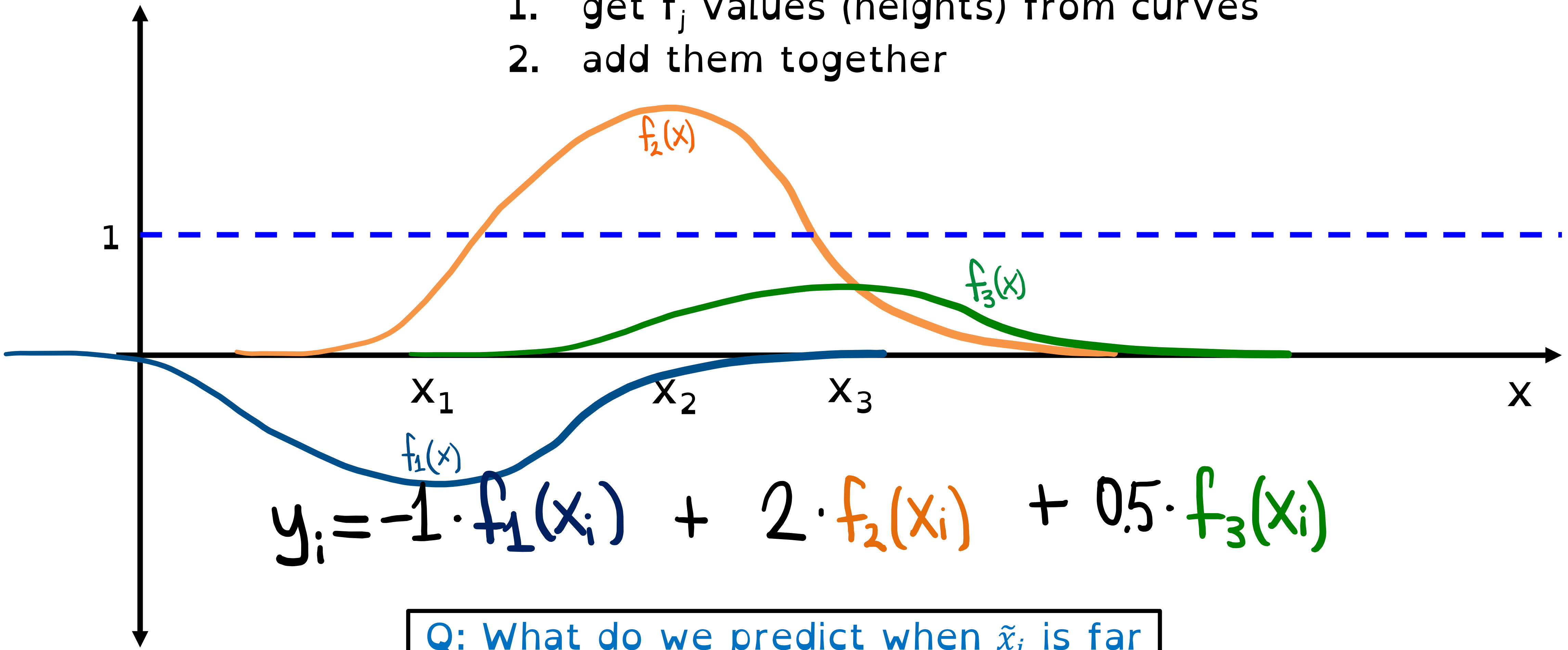


$$\hat{y}_i = 1 \cdot f_1(\tilde{x}_i) + 1 \cdot f_2(\tilde{x}_i) + 1 \cdot f_3(\tilde{x}_i)$$

Q: What happens if these change?

Visualizing RBF Regression

- To predict $\hat{y}_i$ from $\tilde{x}_i$,
  1. get f_j values (heights) from curves
  2. add them together

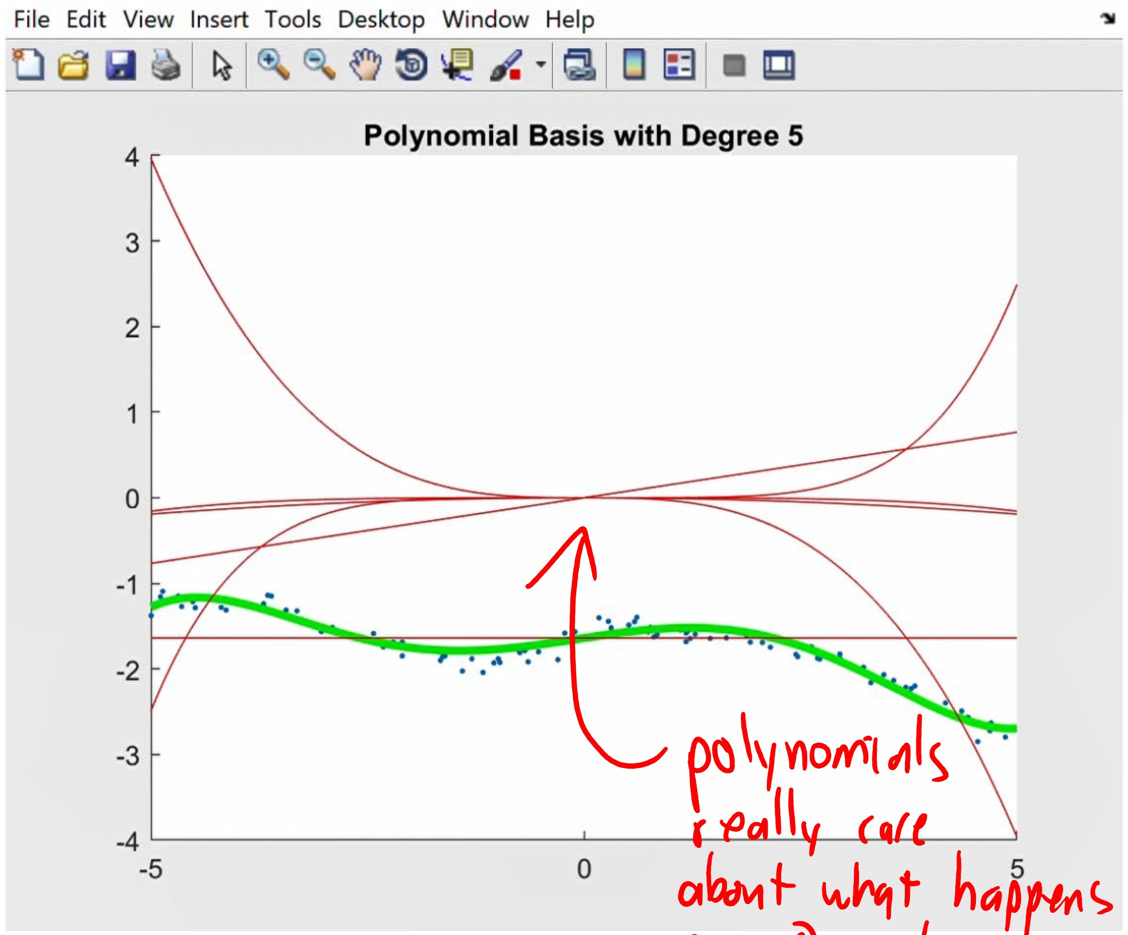


$$y_i = -1 \cdot f_1(x_i) + 2 \cdot f_2(x_i) + 0.5 \cdot f_3(x_i)$$

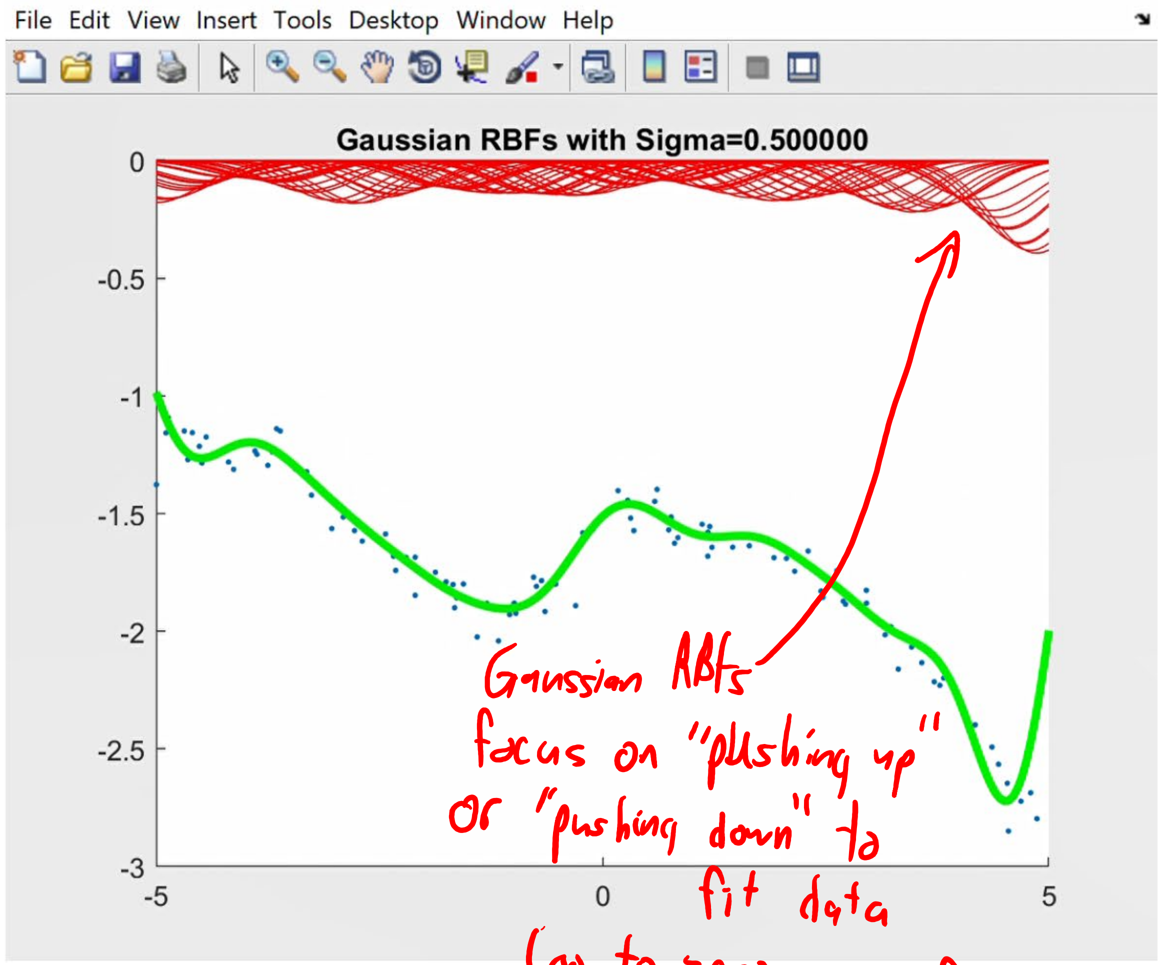
Q: What do we predict when $\tilde{x}_i$ is far away from other examples?

Gaussian RBFs: A Sum of "Bumps"

- More-realistic version (green is regression line, red is each basis):



polynomials really care about what happens near 0, and are wonky at edges of data

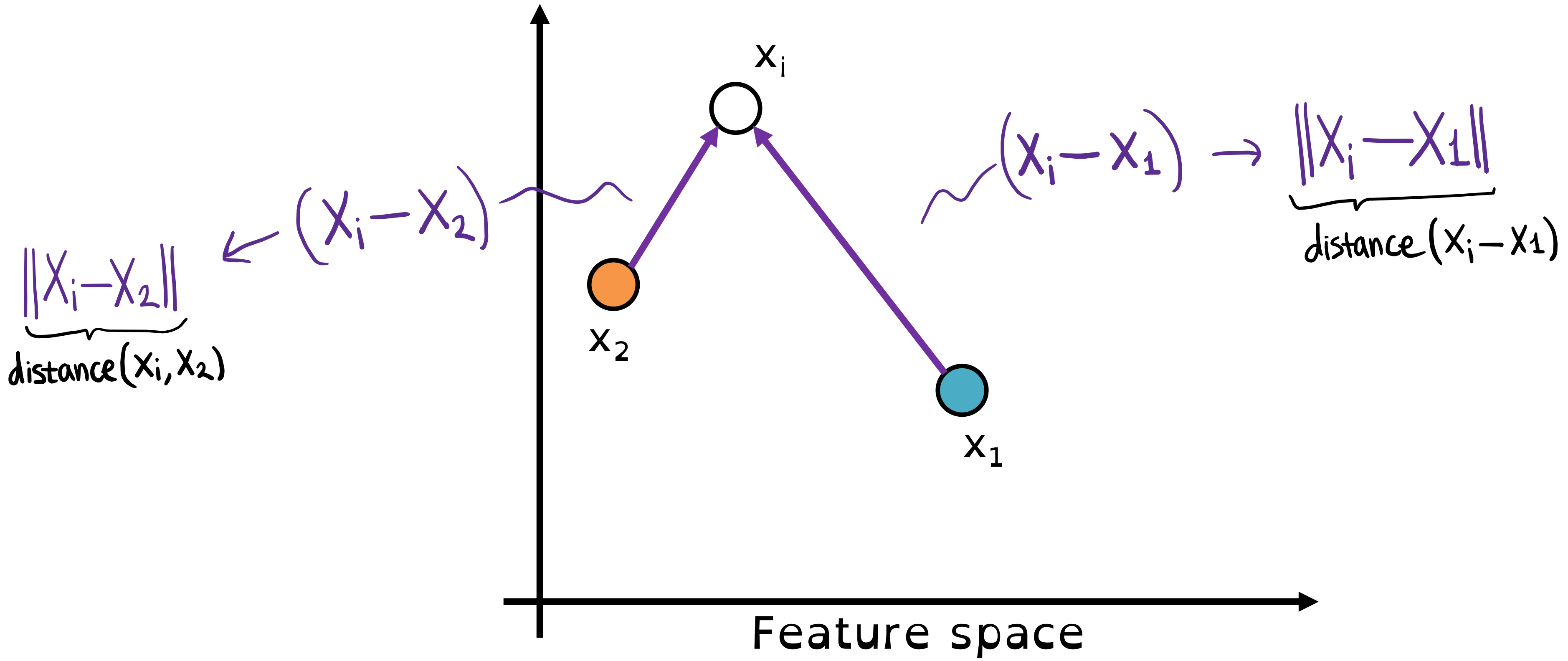


Gaussian RBFs focus on "pushing up" or "pushing down" to fit data (go to zero away from data)

Coming Up Next

GAUSSIAN RBF
IN HIGHER DIMENSIONS

Recall: Distance

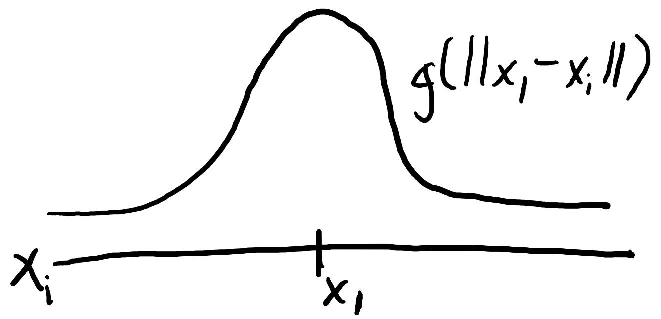

x_i
x_2
x_1
Feature space
$(x_i - x_2)$
$\underbrace{\|x_i - x_2\|}_{\text{distance}(x_i, x_2)}$
$(x_i - x_1) \rightarrow \underbrace{\|x_i - x_1\|}_{\text{distance}(x_i - x_1)}$

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - A set of non-parametric bases that depend on distances to training points.

Replace $x_i = (\underbrace{x_{i1}, x_{i2}, \dots, x_{in}}_{\text{'d' features}})$ with $z_i = (\underbrace{g(\|x_i - x_1\|), g(\|x_i - x_2\|), \dots, g(\|x_i - x_n\|)}_{\text{'n' features}})$

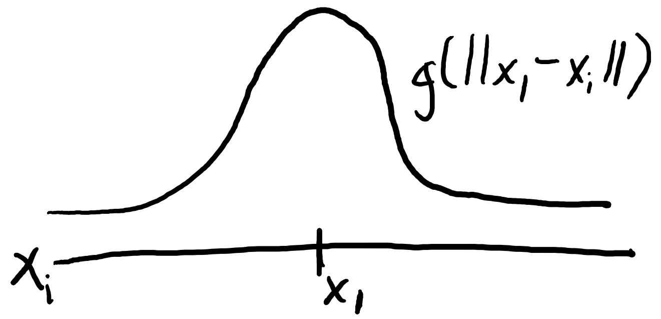
  - Have 'n' features, with feature 'j' depending on distance to example 'i'.
    - Typically the feature will decrease as the distance increases:



Gaussian RBFs: Formal Details

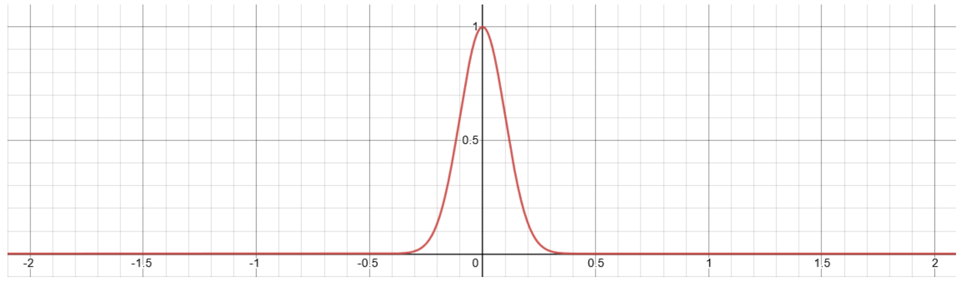
- What is a radial basis functions (RBFs)?
  - Most common choice of ‘g’ is Gaussian RBF:

$$g : \mathbb{R} \rightarrow \mathbb{R} \qquad g(\varepsilon) = \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$$

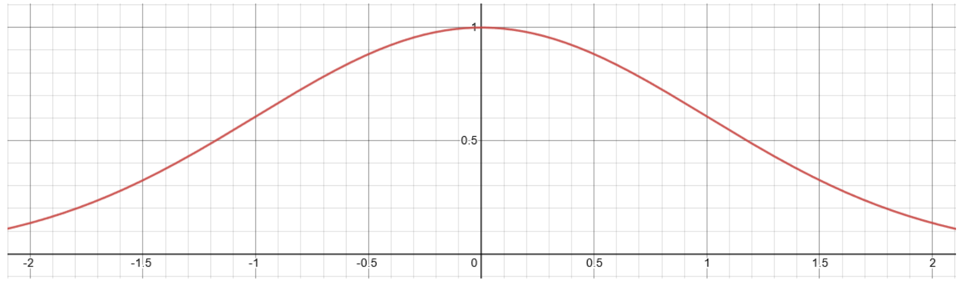


Q: What does $g(\epsilon)$ look like if σ is small?
What does it look like if σ is large?

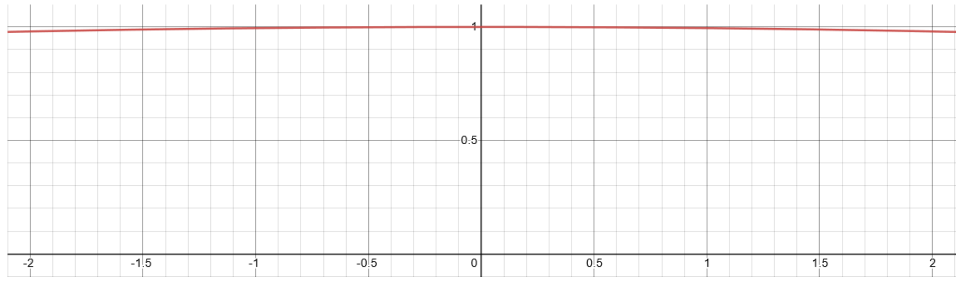
σ and Curve Width



$\sigma = 0.1$



$\sigma = 1.0$



$\sigma = 10.0$

- How does σ affect the model complexity?
  - As σ increases, the model complexity (increases/decreases)
- Low sensitivity to change in feature values => low complexity of model

Gaussian RBFs: Formal Details

- What is a radial basis functions (RBFs)?
  - The training and testing matrices when using RBFs:

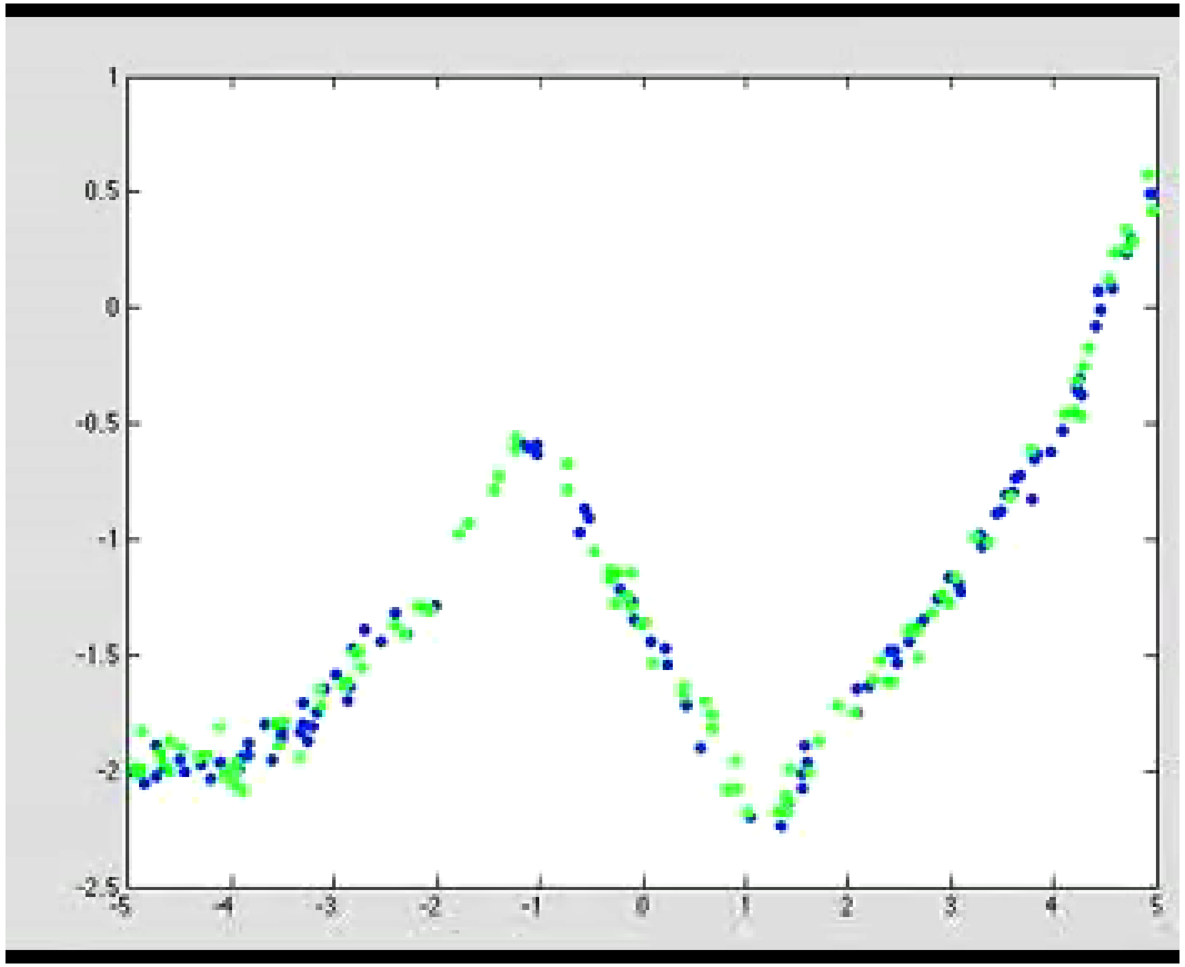
$$\text{Replace } X = \underbrace{\left.\begin{bmatrix} \quad & \quad \\ & \\ & \end{bmatrix}\right\}n}_{d} \quad \text{by} \quad Z = \underbrace{\left.\begin{bmatrix} g(\|x_1-x_1\|) & g(\|x_1-x_2\|) & \cdots & g(\|x_1-x_n\|) \\ g(\|x_2-x_1\|) & g(\|x_2-x_2\|) & \cdots & g(\|x_2-x_n\|) \\ \vdots & & \ddots & \vdots \\ g(\|x_n-x_1\|) & g(\|x_n-x_2\|) & \cdots & g(\|x_n-x_n\|) \end{bmatrix}\right\}n}_{n}$$

$$\text{To make } \underline{\text{predictions}} \text{ on } \tilde{X} = \underbrace{\left.\begin{bmatrix} \quad & \quad \\ & \\ & \end{bmatrix}\right\}t}_{d} \quad \text{use} \quad \tilde{Z} = \underbrace{\left.\begin{bmatrix} & & \\ & g(\|\tilde{x}_i - x_j\|) & \\ & & \end{bmatrix}\right\}t}_{n}$$

Number of "features" is number of training examples.

Non-Parametric Basis: RBFs

- Least squares with Gaussian RBFs for different σ values:



Could add bias and linear basis:

$$Z = \begin{bmatrix} 1 & — x_1 — & g(\|x_1 - x_1\|) & \cdots & g(\|x_1 - x_n\|) \\ 1 & — x_2 — & \vdots & \ddots & \vdots \\ 1 & — x_3 — & & & \\ \vdots & \vdots & & & \\ 1 & — x_n — & g(\|x_1 - x_n\|) & \cdots & g(\|x_n - x_n\|) \end{bmatrix}$$

$$\underbrace{\quad}_{1} \quad \underbrace{\qquad}_{d} \quad \underbrace{\qquad\qquad\qquad}_{n}$$

Q: What do we predict when $\tilde{x}_i$ is far away from other examples?

RBFs and Regularization

- Gaussian Radial basis functions (RBFs) predictions:

$$\hat{y}_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$

$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

  - Flexible bases that can model any continuous function.
  - But with ‘n’ data points RBFs have ‘n’ basis functions.

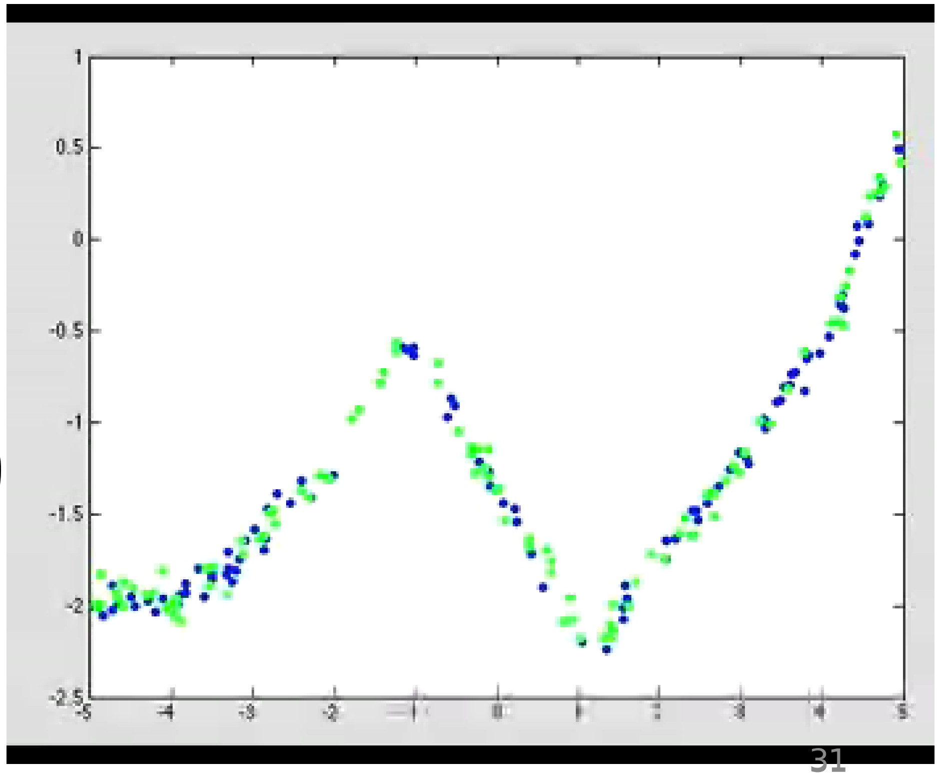
- How do we avoid overfitting with this huge number of features?
  - We regularize ‘w’ and use validation error to choose σ and λ.

RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error.

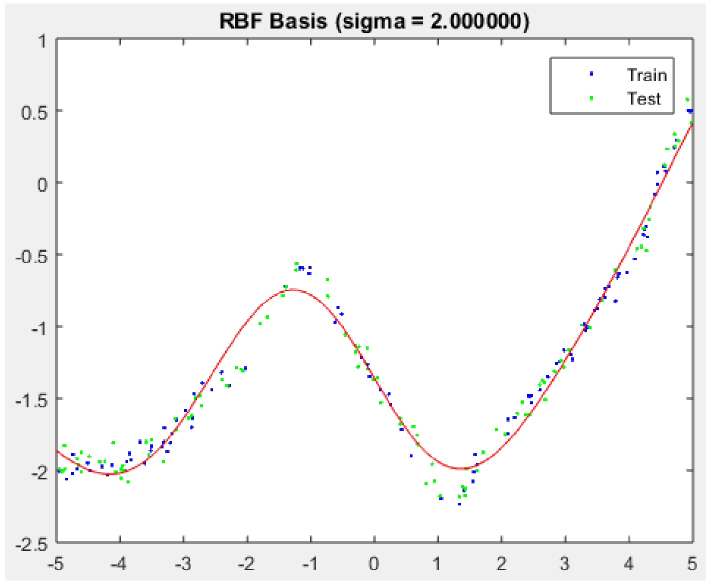
for each value of λ and σ:

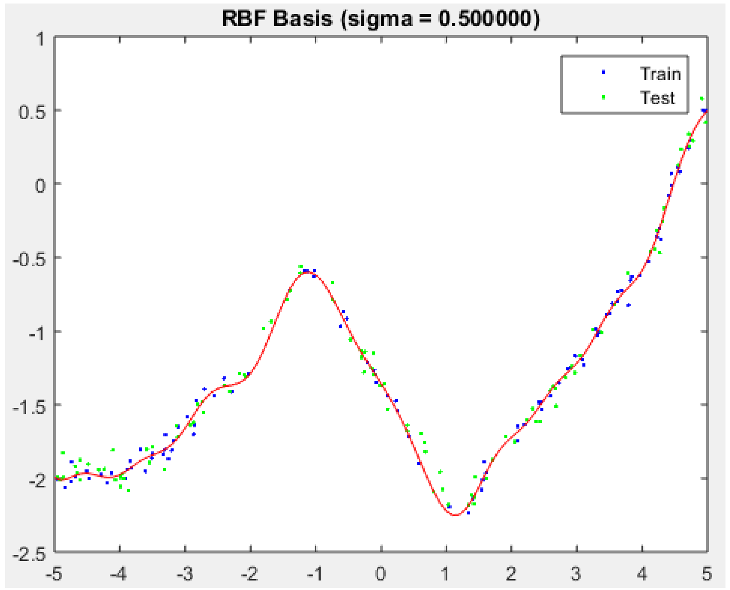
- Compute Z on training data (and σ)
- Compute best v: $v = (Z^T Z + \lambda I)^{-1} Z^T y$
- Compute $\tilde{Z}$ on validation data (using train data distances)
- Make predictions $\hat{y} = \underset{t\times n}{\tilde{Z}} \underset{n\times 1}{v}$
- Compute validation error $\|\hat{y} - \tilde{y}\|^2$

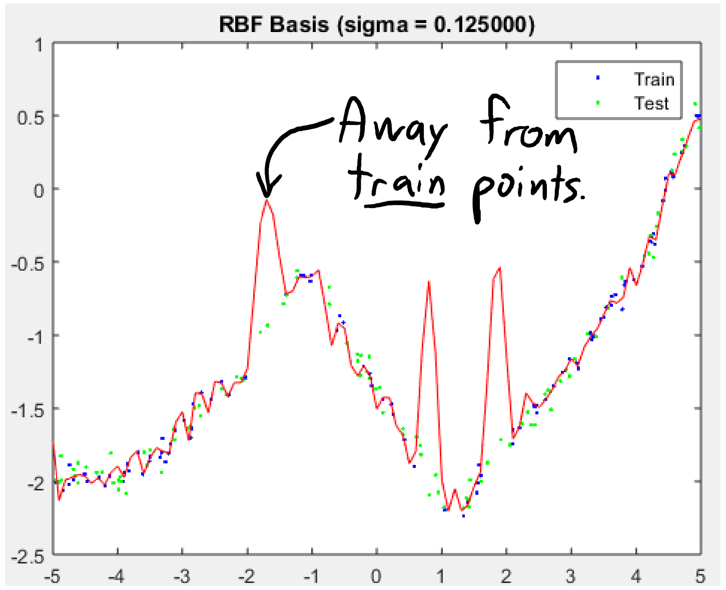


RBFs, Regularization, and Validation

- A model that is hard to beat:
  - RBF basis with L2-regularization and cross-validation to choose σ and λ.
  - Flexible non-parametric basis, magic of regularization, and tuning for test error!







  - Expensive at test time: needs ______________________________.

Hyper-Parameters of Gaussian RBFs

- In this setting we have 2 hyper-parameters (σ and λ).
- More complicated models have even more hyper-parameters.
  - Searching all values is unviable (increases ___________ risk).
- Simplest approaches:
  - Exhaustive search: discretize and try all combinations
  - Random search: try random values.

Hyper-Parameter Optimization

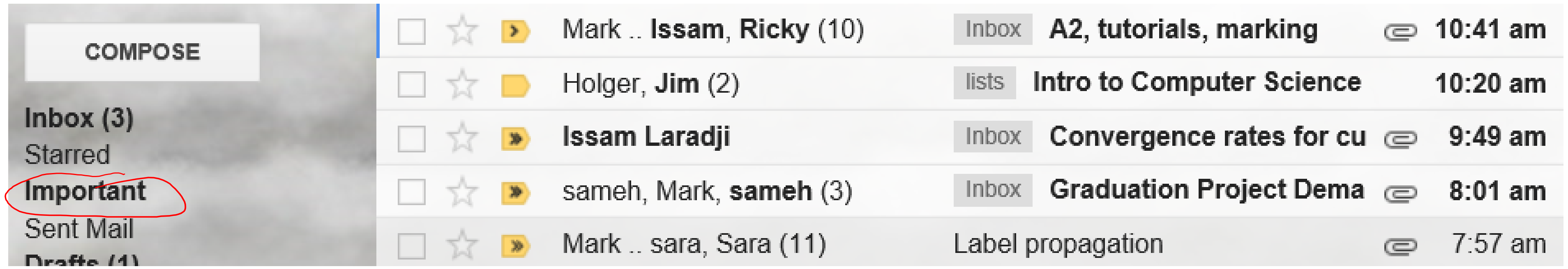
- Other common hyper-parameter optimization methods:
  - Exhaustive search with pruning:
    - If it "looks" like test error is getting worse as you decrease λ, stop decreasing it.
  - Coordinate search:
    - Optimize one hyper-parameter at a time, keeping the others fixed.
    - Repeatedly go through the hyper-parameters
  - Stochastic local search:
    - Generic global optimization methods (simulated annealing, genetic algorithms, etc.).
  - Bayesian optimization (Mike's PhD research topic):
    - Use RBF regression to build model of how hyper-parameters affect validation error.
    - Try the best guess based on the model.

Coming Up Next

LINEAR CLASSIFIERS INTRO

Motivation: Identifying Important Emails

- How can we automatically identify 'important' emails?



COMPOSE

Inbox (3)
Starred
Important
Sent Mail
Drafts (1)

Mark .. Issam, Ricky (10)	Inbox	A2, tutorials, marking	10:41 am
Holger, Jim (2)	lists	Intro to Computer Science	10:20 am
Issam Laradji	Inbox	Convergence rates for cu	9:49 am
sameh, Mark, sameh (3)	Inbox	Graduation Project Dema	8:01 am
Mark .. sara, Sara (11)		Label propagation	7:57 am

- A binary classification problem ("important" vs. "not important").
  - Labels are approximated by whether you took an "action" based on mail.
  - High-dimensional feature set (that we'll discuss later).

- Gmail uses regression for this binary classification problem.

Binary Classification Using Regression?

- Recall: we had classification problems in Part 1:
  - Food allergies, spam filtering, character recognition, Netflix recommendation, etc.

- Binary classification: 2 classes in label y

- Usually, we encode $y_i = \{0, 1\}$
- For linear classifiers, we encode $y_i = \{-1, +1\}$
  - e.g. +1 means “important”, -1 means otherwise.

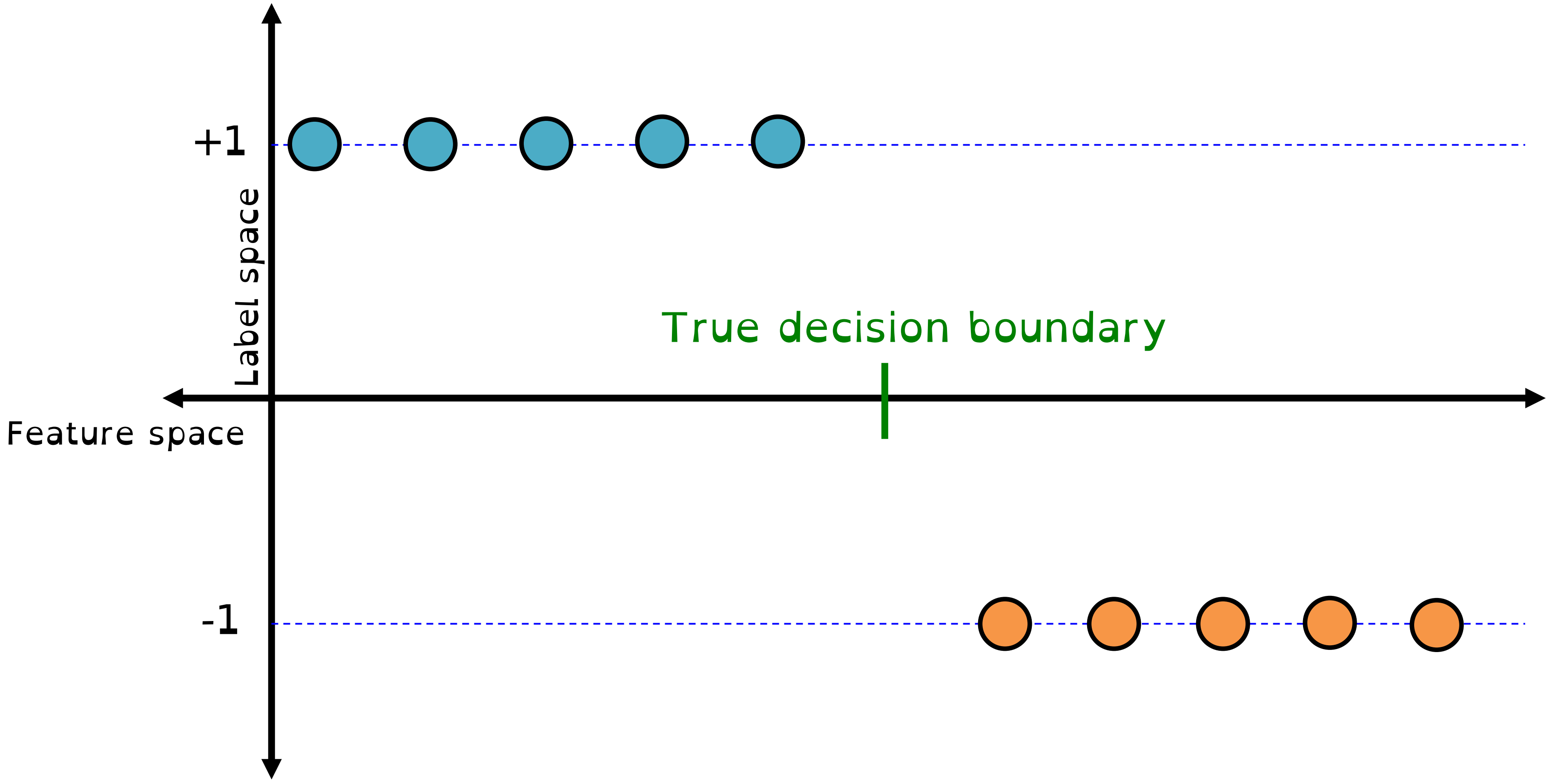
Visualizing Binary Classification

- Assumption: somewhere along the feature space, there's a boundary that (roughly) splits +1s and -1s.

- If a perfect boundary exists, the data is called "linearly separable"

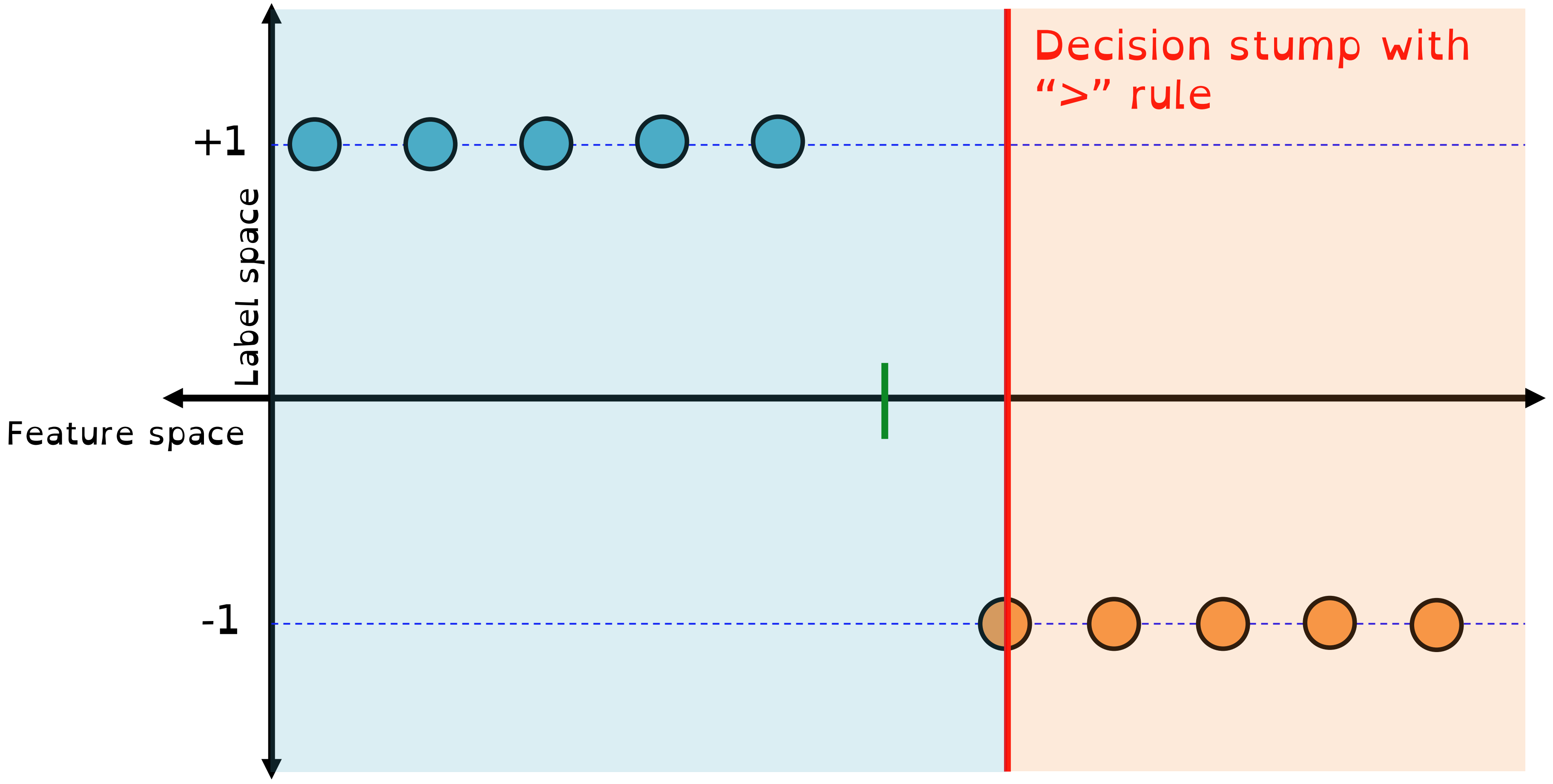
Visualizing Binary Classification

- Assumption: somewhere along the feature space, there's a boundary that (roughly) splits +1s and -1s.



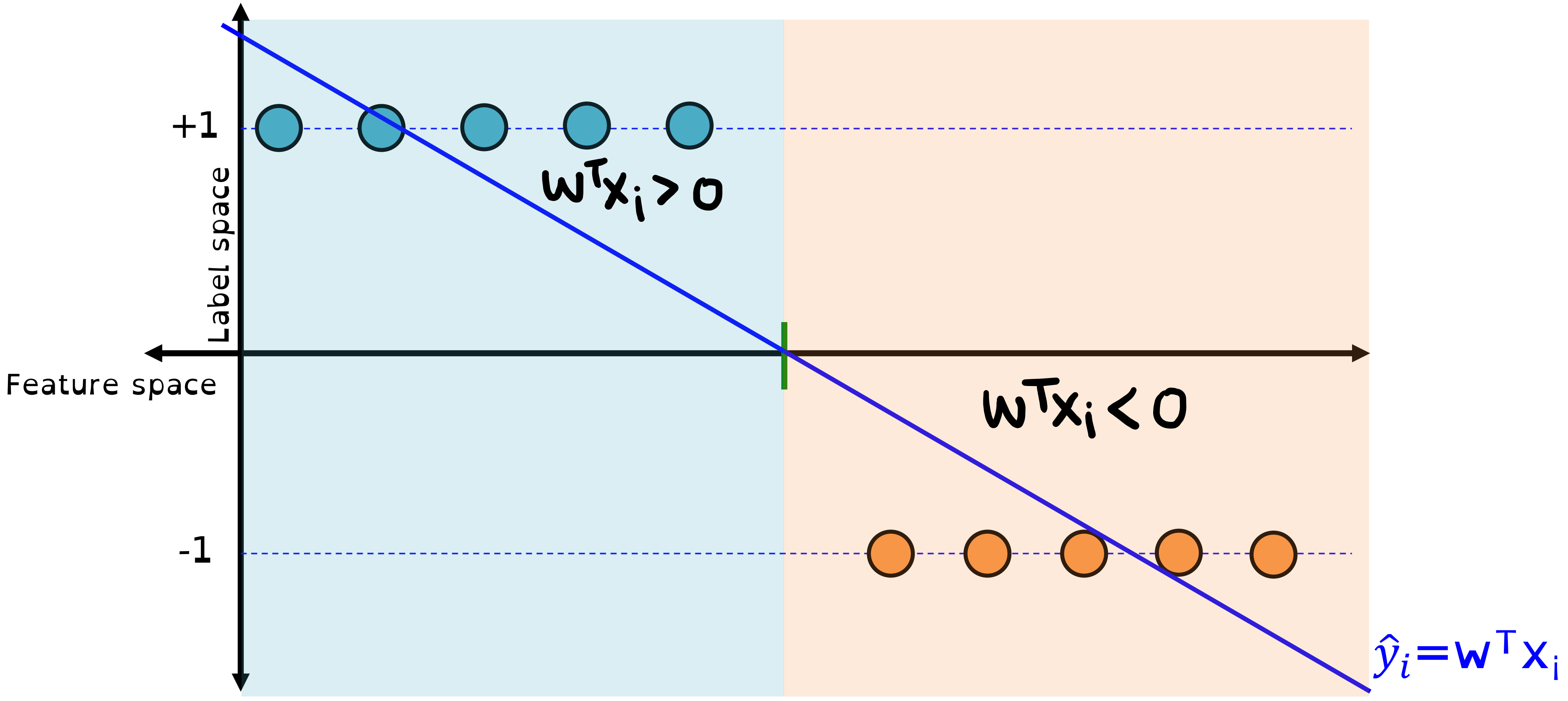
Visualizing Binary Classification

- Assumption: somewhere along the feature space, there's a boundary that (roughly) splits +1s and -1s.



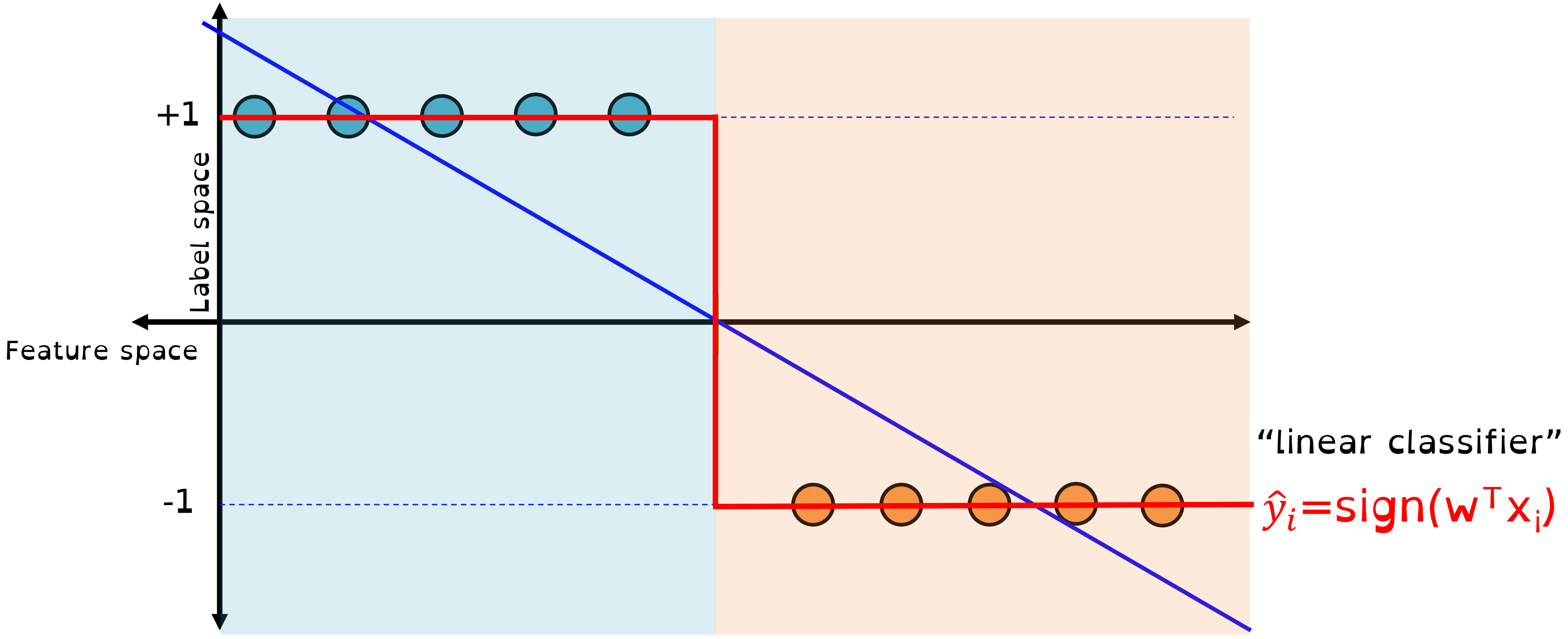
Visualizing Binary Classification

- Assumption: somewhere along the feature space, there's a boundary that (roughly) splits +1s and -1s.



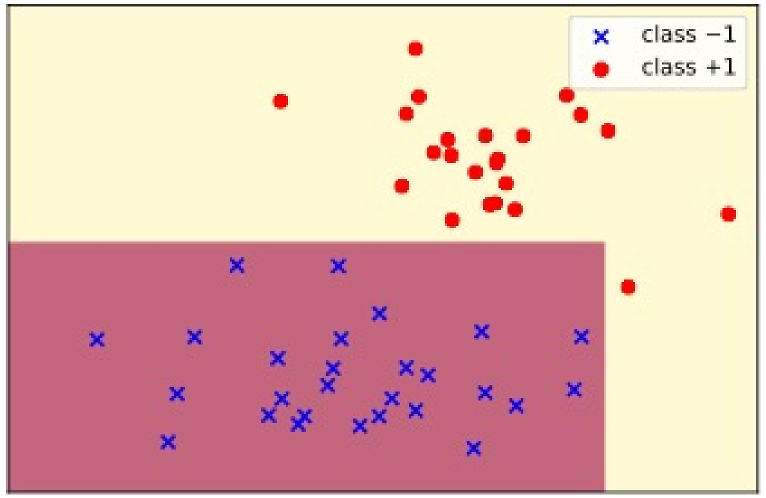
Visualizing Binary Classification

- Assumption: somewhere along the feature space, there's a boundary that (roughly) splits +1s and -1s.



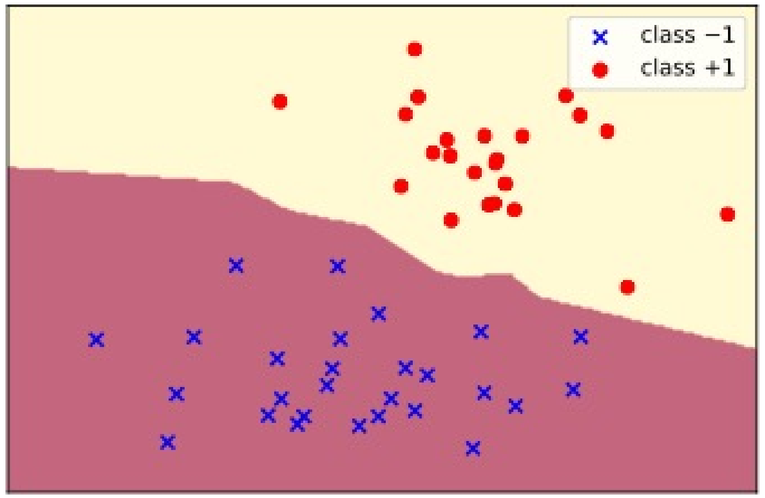
Decision Boundaries in 2D

decision tree



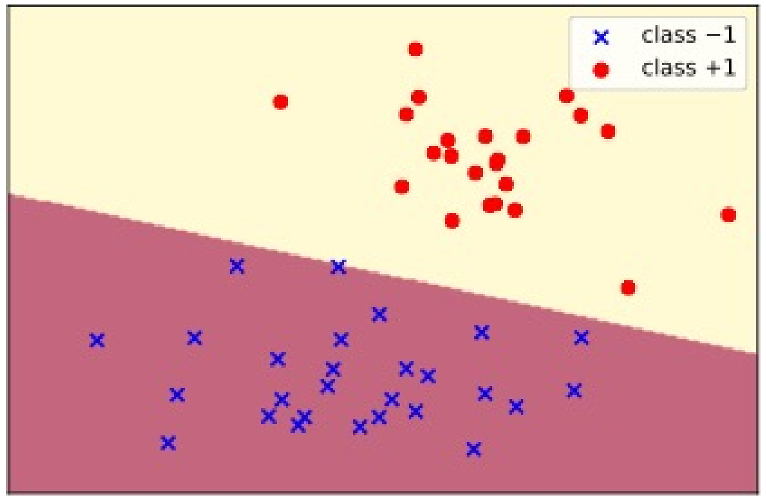
Feature space

KNN



Feature space

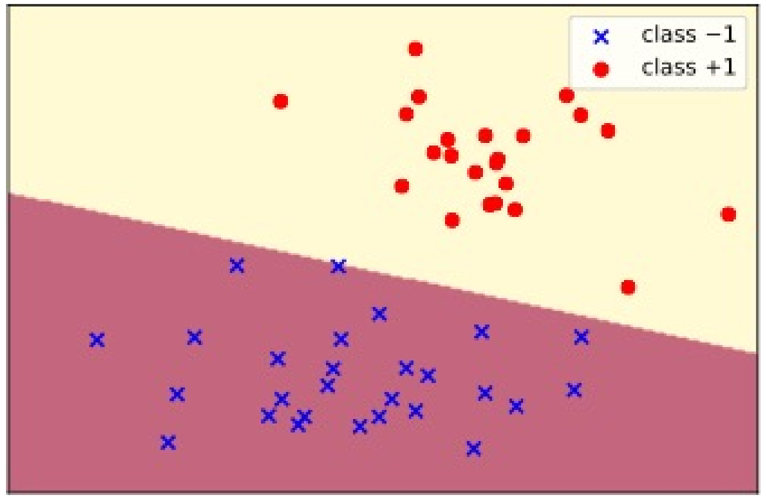
linear classifier



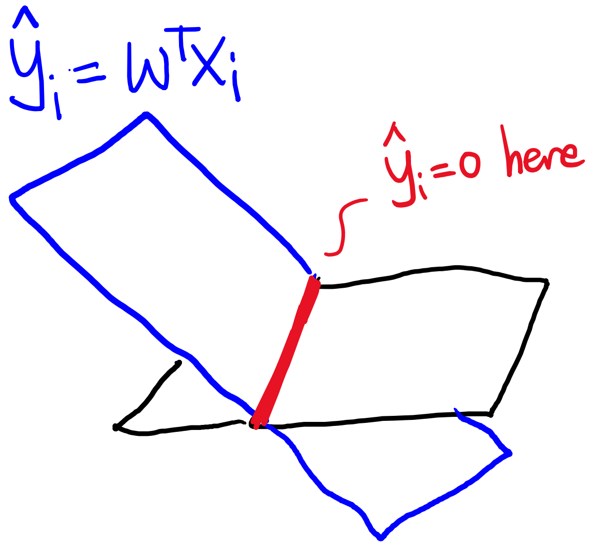
Feature space

Decision Boundaries in 2D

linear classifier



Feature space



- Linear classifier would be a $\hat{y}_i = w^T x_i$ function coming out of screen:
  - The boundary is at $\hat{y}_i = 0$.

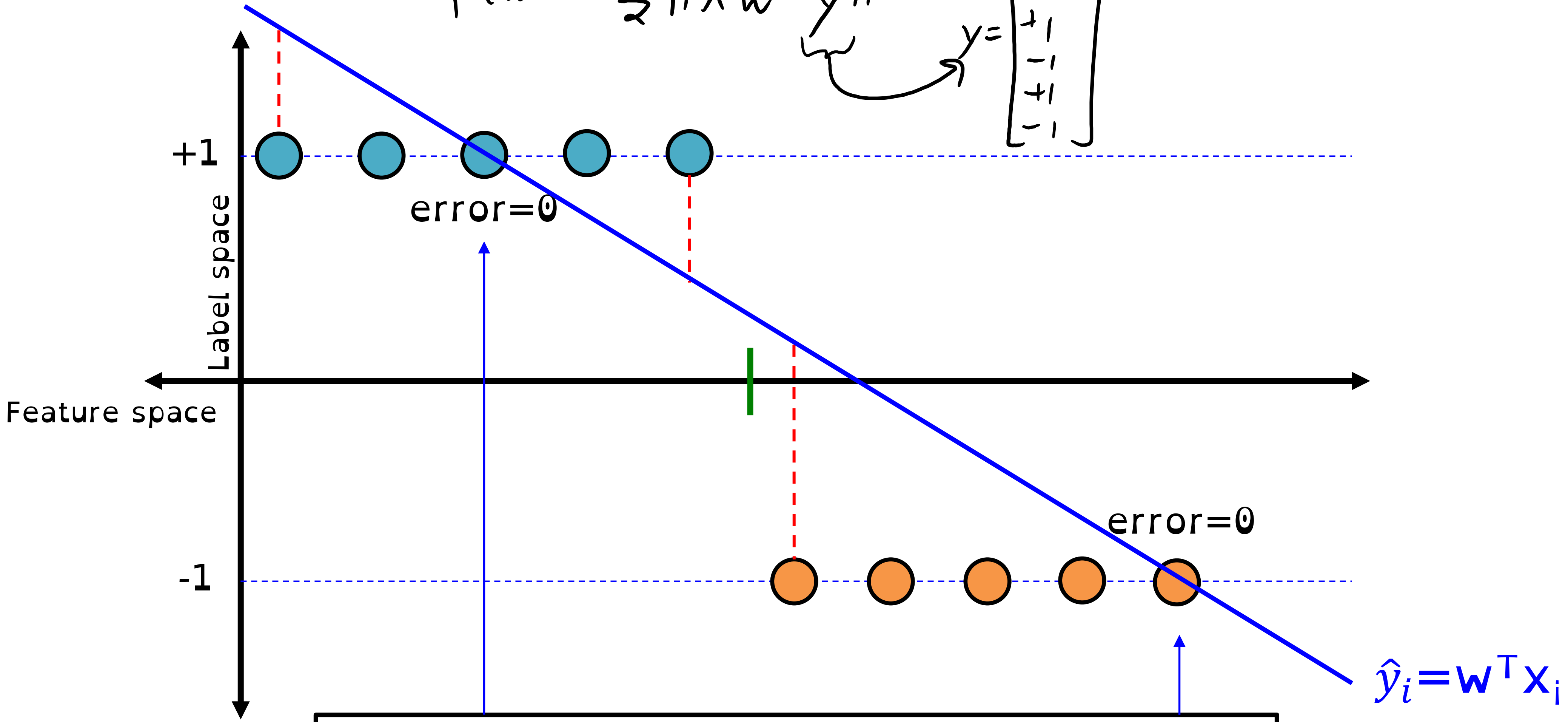
Coming Up Next

LOSSES FOR BINARY CLASSIFIERS

Should we use least squares for classification?

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

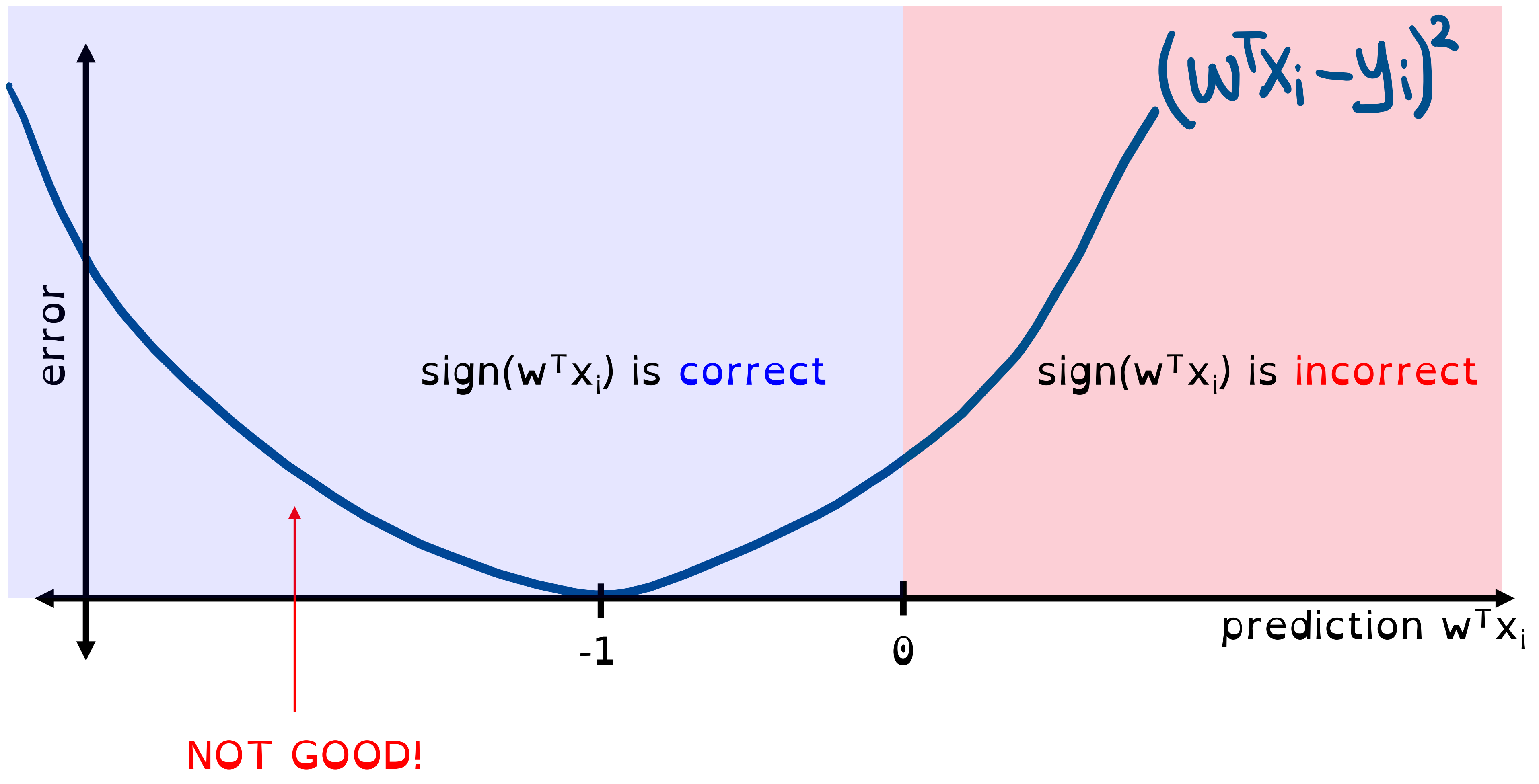
$$y = \begin{bmatrix} +1 \\ +1 \\ -1 \\ +1 \\ -1 \end{bmatrix}$$



Q: Do these points deserve to have error=0 and others don't?

Should we use least squares for classification?

Given example (x_i, -1)



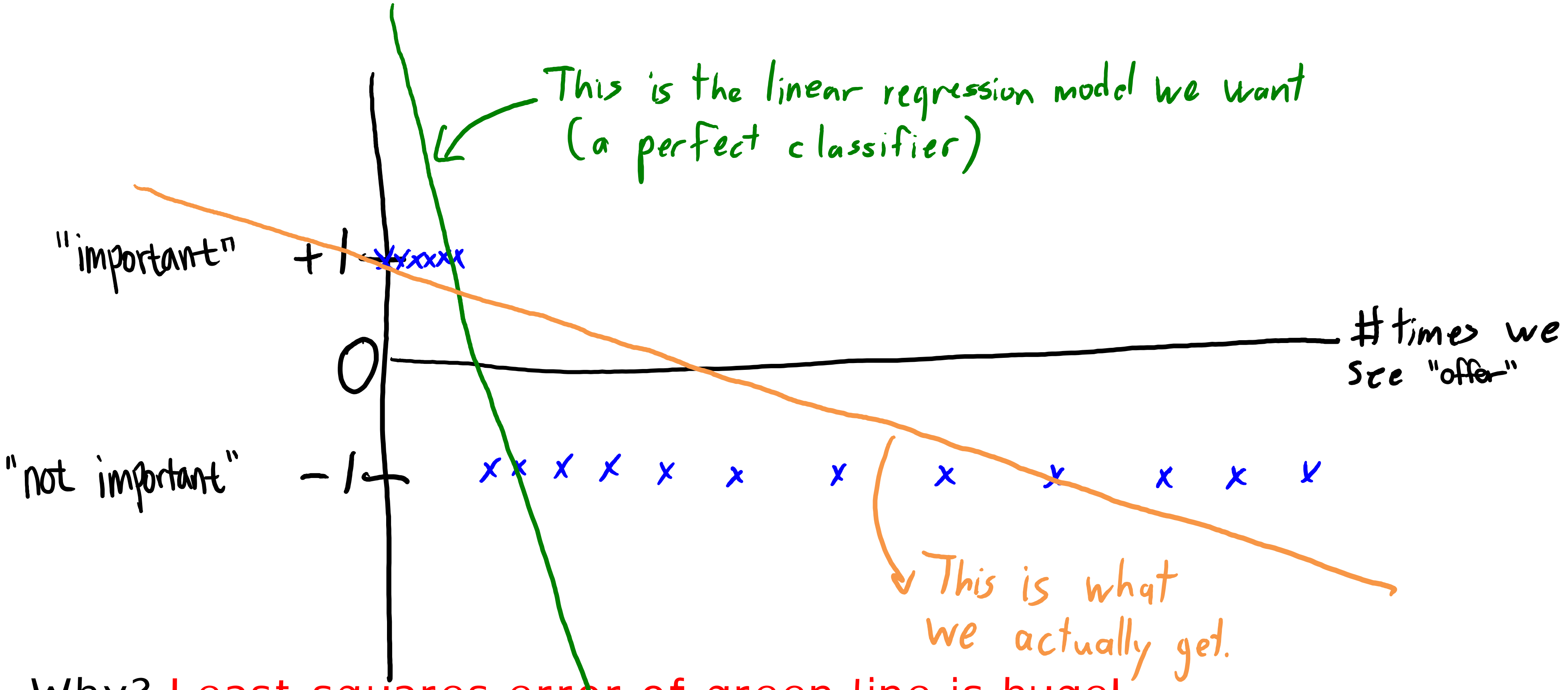
Issues with Least Squares Error

When least squares penalizes
my example far from 0

- x_i far from 0 means $w^T x_i$ will be ______________
- sign($w^T x_i$) is correct but $(w^T x_i - y_i)^2$ is huge
  - Penalizes for examples that are "too correct"
- Also, which examples get 0 error is arbitrary

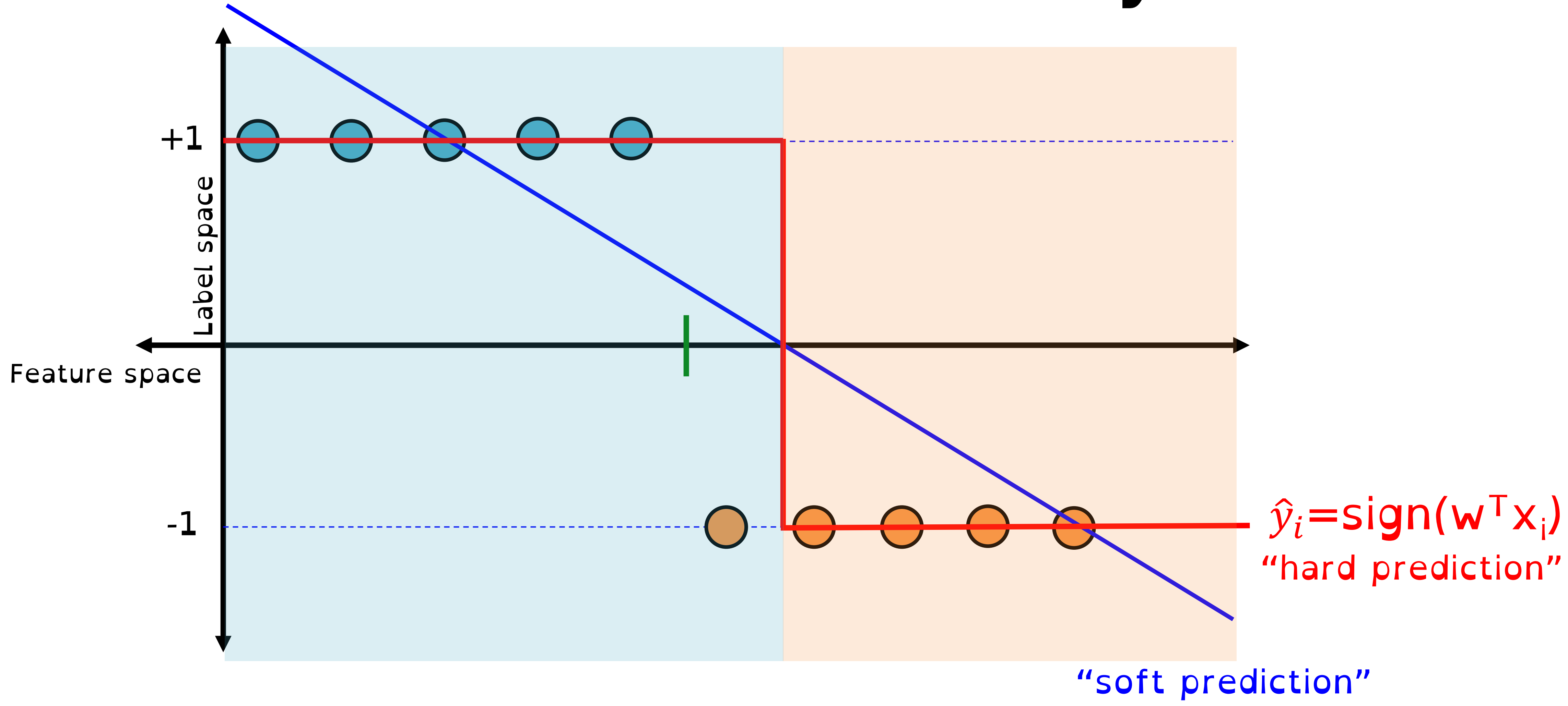
Should we use least squares for classification?

- Least squares can behave weirdly when applied to classification:



- Why? Least squares error of green line is huge!
  - The green line achieves 0 training classification error.

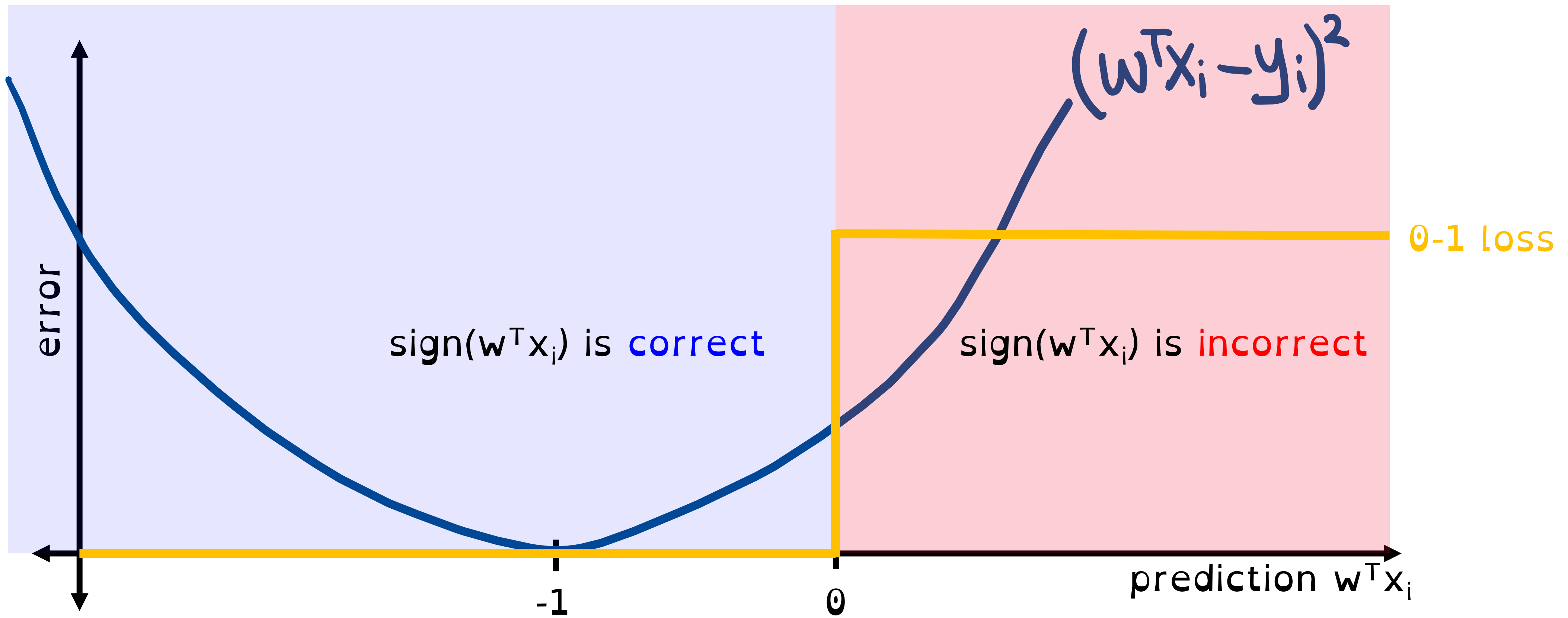
0-1 Loss: What We Really Want



- We want to minimize classification error based on "hard predictions"!

0-1 Loss: What We Really Want

Given example (x_i, -1)



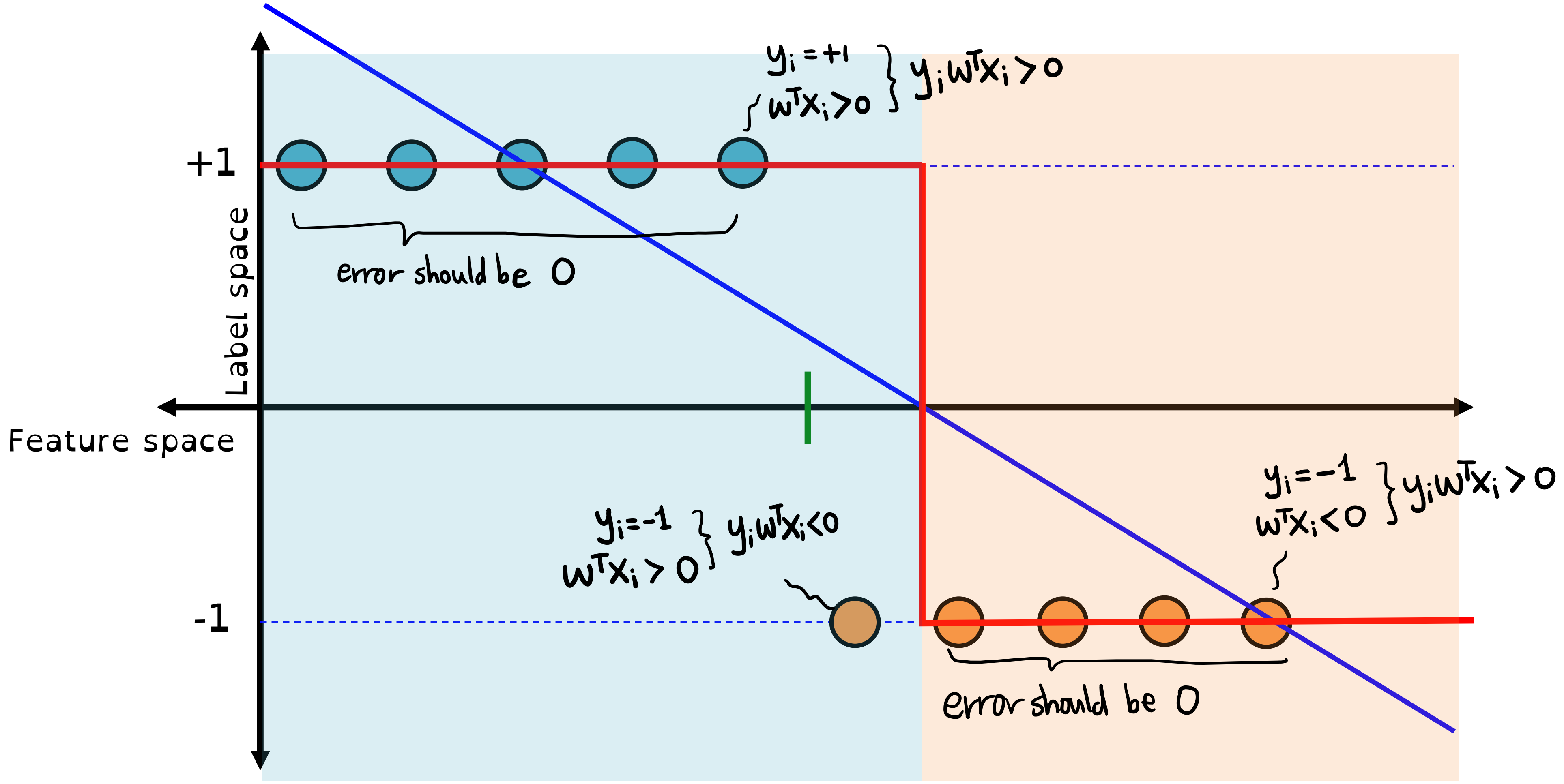
Q: What's wrong with the 0-1 loss?

0-1 Loss Function

- We can write using the L0-norm as $||\hat{y} - y||_0$.
  - In classification it's reasonable that $\hat{y}_i = y_i$ (it's either +1 or -1).
- 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: don't even know "which way to go".
  - NOT the same type of problem we had with using the squared loss.
    - We can minimize the squared error,
      but it might give a bad model for classification.
- 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 _________________.



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.

Let's count # times $y_i w^T x_i < 0$

"indicator"

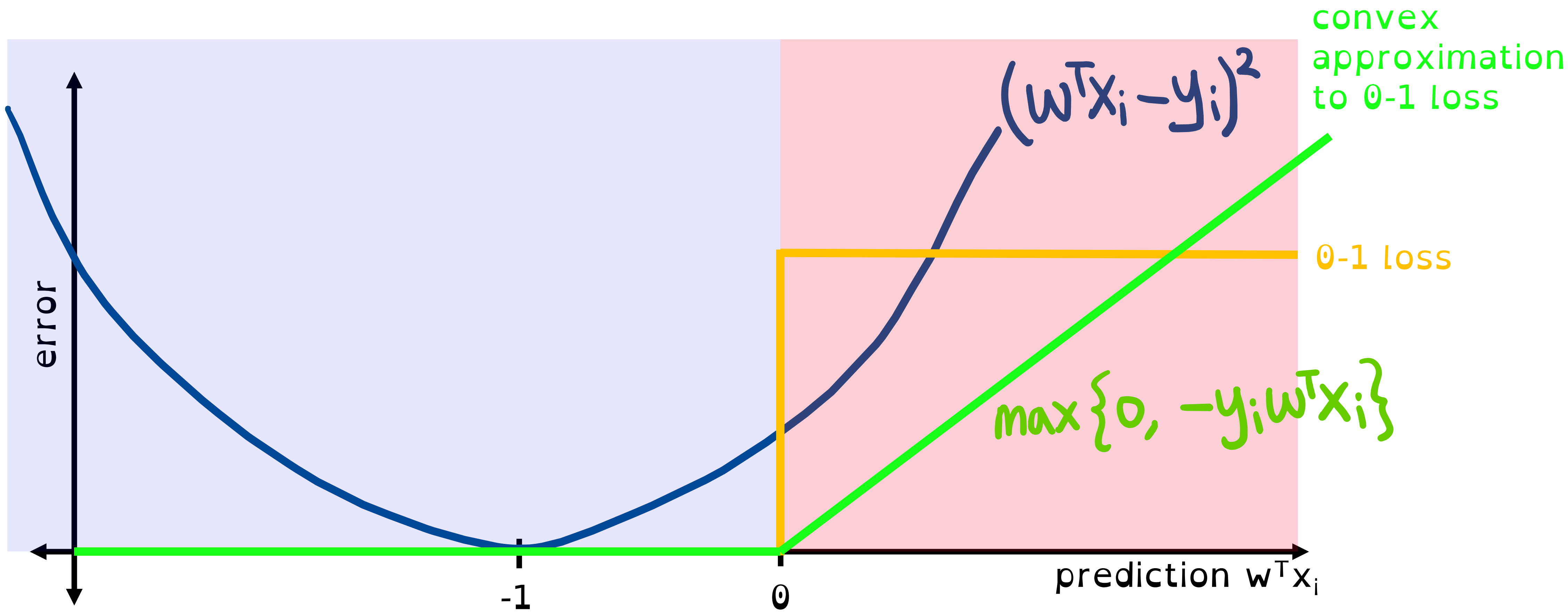
$$I(\text{stmt}) = \begin{cases} 1 & \text{if stmt is true} \\ 0 & \text{otherwise} \end{cases}$$

[1] $$f(w) = \sum_{i=1}^{n} I(y_i w^T x_i < 0) = \sum_{i=1}^{n} I(0 < -y_i w^T x_i) \quad (\text{0-1 loss})$$

[2] $$\approx \sum_{i=1}^{n} \max\{0, -y_i w^T x_i\} \quad (\text{Convex approximation})$$

0-1 Loss: What We Really Want

Given example (x_i, -1)



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:

Q: When is f(0) = 0?

- There are two standard fixes: hinge loss and logistic loss.

Summary

- Feature standardization:
  - Change the unit of every feature into "z-score"
- Radial basis functions:
  - Non-parametric bases that can model any function.
- Binary classification using regression:
  - Encode using y_i in {-1,1}.
  - Use $\text{sign}(w^T x_i)$ as prediction.
  - "Linear classifier" (a hyperplane splitting the space in half).
- Least squares is a weird error for classification.
- Perceptron algorithm: finds a perfect classifier (if one exists).
- 0-1 loss is the ideal loss, but is non-smooth and non-convex.

- Next time: logistic regression and support vector machine

Gaussian RBFs: Pseudo-Code

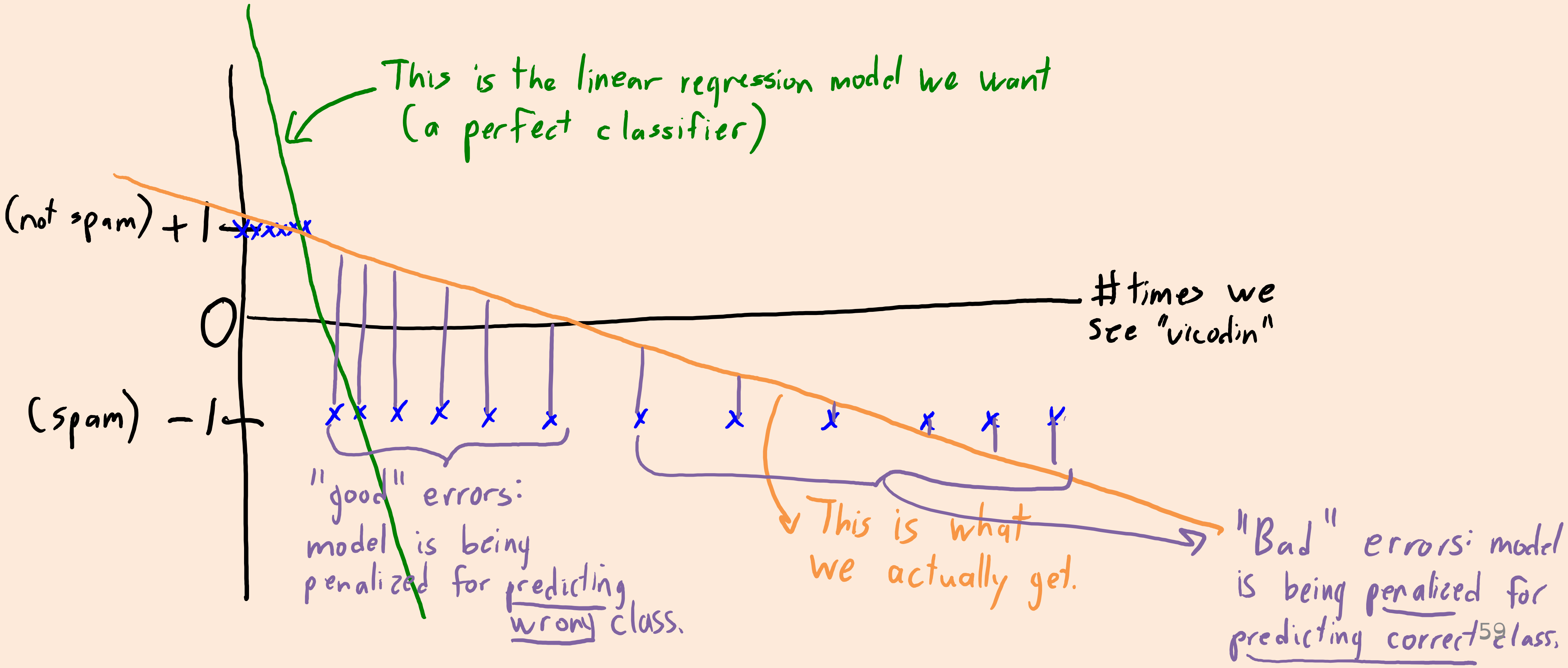
Constructing Gaussian RBFs given data 'X' and hyper-parameter σ:

```
Z = zeros(n,n)
for i1 in 1:n
    for i2 in 1:n
        Z[i1, i2] = $\exp(-\text{norm}(X[i1,:] - X[i2,:])^2 / 2\sigma^2)$
```

With test data $\tilde{X}$: form $\tilde{Z}$ based on distances to training examples.

Can we just use least squares??

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

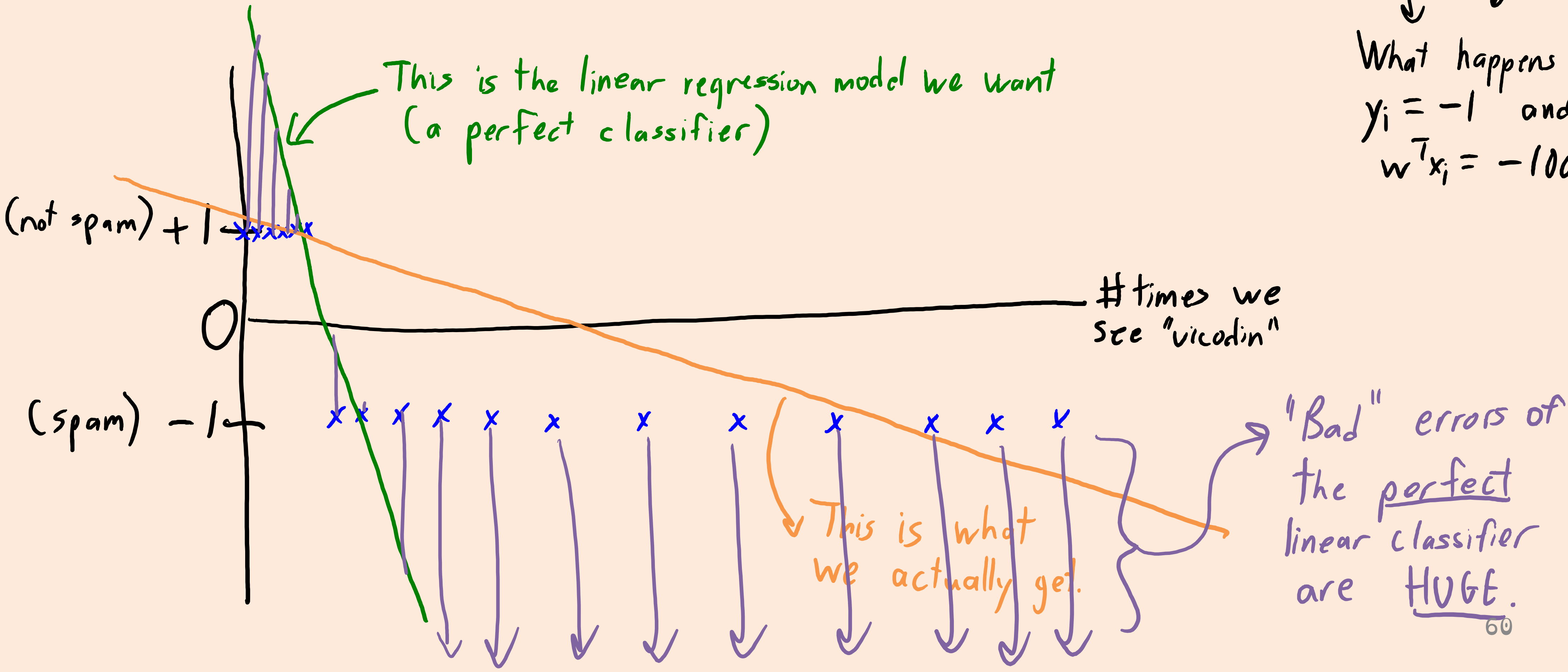

This is the linear regression model we want (a perfect classifier)
(not spam) +1
0
(spam) −1
# times we see "vicodin"
"good" errors: model is being penalized for predicting wrong class.
This is what we actually get.
"Bad" errors: model is being penalized for predicting correct class.

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 $w^T x_i = -1000$?


This is the linear regression model we want (a perfect classifier)
(not spam) +1
0
(spam) -1
# times we see "vicodin"
This is what we actually get.
"Bad" errors of the perfect linear classifier are HUGE.