Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

Topic 3 continued Linear Classification & Logistic Regression

PROF. LINDA SELLIE

Some slides/approaches used are from Prof. Rangan
Many approaches used are from CMU 18-661

Finding Parameters via Optimization
A general ML recipe

General ML problem	Multiple linear regression
❑Get data	→ Data: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1,2,\ldots,N$
❑Pick a model with parameters	→ **Logistic function** applied to $\mathbf{w}^T\mathbf{x}$ $$p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$
❑Pick a Objective/ loss function ◦ Measures goodness of fit model to data ◦ Function of the parameters	→ **Objective function:** $$\mathbf{w}* = \arg\max_{\mathbf{w}} \left(\frac{1}{N} \sum_{i=1}^{N} y^{(i)} \ln(h(\mathbf{x})) + (1 - y^{(i)}) \ln(1 - h(\mathbf{x})) \right)$$ where $h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$
❑Find parameters that maximizes likelihood	→ Select $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_d]^T$ to maximize $\frac{1}{N}\ell(\mathbf{w})$ (Or minimize $-\ell(\mathbf{w})$.)

Gradient Ascent Algorithm

$$\frac{1}{N}\ell(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\ln\sigma(\mathbf{w}^T\mathbf{x}^{(i)}) + (1-y^{(i)})\ln\left(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)})\right)\right]$$

for k = 1 to num_iter

$$temp0 = w_0 + \alpha\frac{\partial\ell(\mathbf{w})/N}{\partial w_0} = w_0 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_0^{(i)}$$

$$temp1 = w_1 + \alpha\frac{\partial\ell(\mathbf{w})/N}{\partial w_1} = w_1 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_1^{(i)}$$

$$\vdots$$

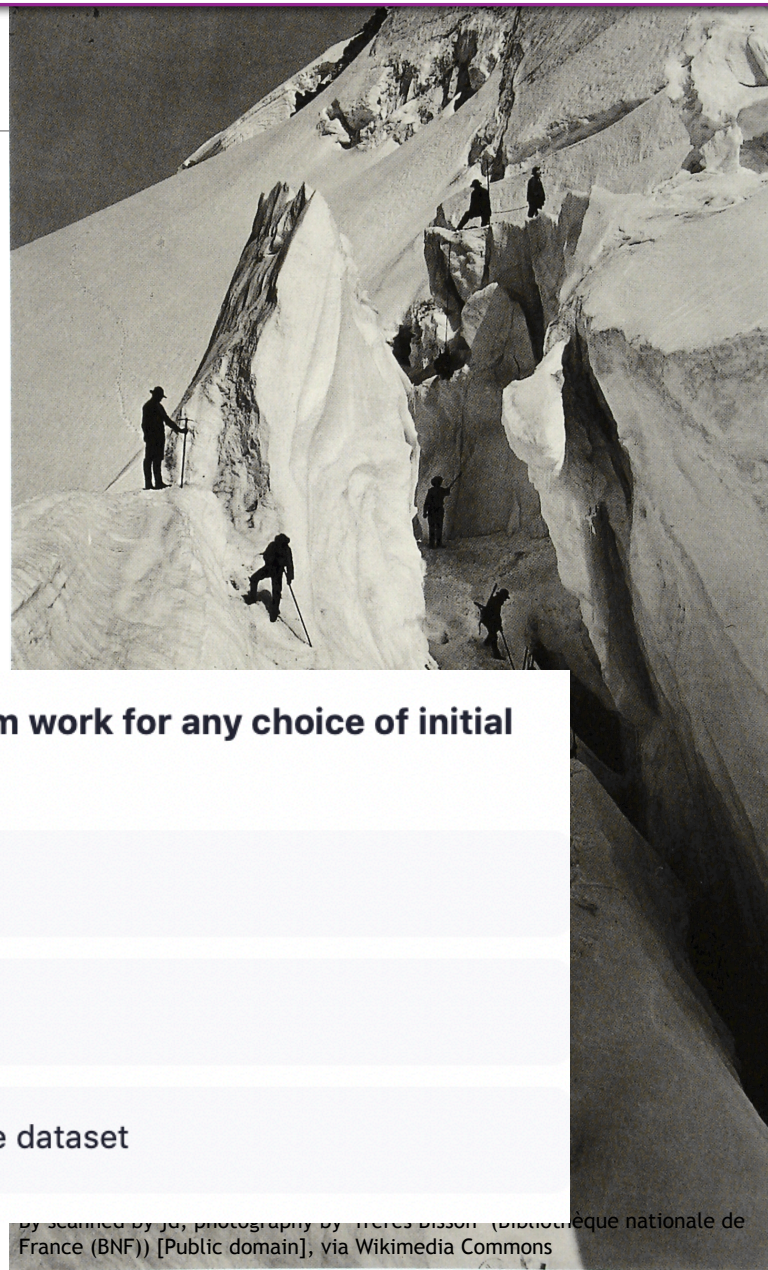
$$tempd = w_d + \alpha\frac{\partial\ell(\mathbf{w})/N}{\partial w_d} = w_d + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_d^{(i)}$$

$x^{(i)}_0=1$

$$w_0 = temp0$$
$$w_1 = temp1$$
$$\vdots$$
$$w_d = tempd$$

Simultaneous update

Running time:
$O(Nd \cdot$ # iter$)$



1. Does this algorithm work for any choice of initial values for w?

- Yes
- No
- It depends on the dataset

Gradient Ascent Step

$$X = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow X\mathbf{w} = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \hat{\mathbf{y}} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

for k = 1 to num_iter

$$temp0 = w_0 + \frac{\alpha}{N}\frac{\partial \ell(\mathbf{w})}{\partial w_0} = w_0 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_0^{(i)} = w_0 + \frac{0.2}{4}\left[\left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(1)}\right)\right) + \left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(2)}\right)\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(3)}\right)\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(4)}\right)\right)\right]$$

-0.5 -0.5 0.5 0.5

$$temp1 = w_1 + \frac{\alpha}{N}\frac{\partial \ell(\mathbf{w})}{\partial w_1} = w_1 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_1^{(i)} = w_1 + \frac{0.2}{4}\left[\left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(1)}\right)\right)x_1^{(1)} + \left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(2)}\right)\right)x_1^{(2)} + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(3)}\right)x_1^{(3)}\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(4)}\right)x_1^{(4)}\right)\right]$$

(-0.5)3 (-0.5)3 (0.5)4 (0.5)3

$$temp2 = w_2 + \frac{\alpha}{N}\frac{\partial \ell(\mathbf{w})}{\partial w_2} = w_2 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_2^{(i)} = w_2 + \frac{0.2}{4}\left[\left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(1)}\right)\right)x_2^{(1)} + \left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(2)}\right)\right)x_2^{(2)} + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(3)}\right)x_2^{(3)}\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(4)}\right)x_2^{(4)}\right)\right]$$

(-0.5)4 (-0.5)5 (0.5)1 (0.5)(1.5)

$w_0^{new} = temp0$ 0

$w_1^{new} = temp1$ 0.025

$w_2^{new} = temp2$ -0.1625

Gradient Ascent Example

$$X = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} 0 \\ 0.025 \\ -0.1625 \end{bmatrix} \Rightarrow X\mathbf{w} = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0.025 \\ -0.1625 \end{bmatrix} = \begin{bmatrix} -0.575 \\ -0.7375 \\ -0.0625 \\ -0.16875 \end{bmatrix} \Rightarrow \hat{\mathbf{y}} = \begin{bmatrix} 0.3601 \\ 0.3236 \\ 0.4844 \\ 0.4579 \end{bmatrix}$$

for k = 1 to num_iter

$$temp0 = w_0 + \frac{\alpha}{N}\frac{\partial \ell(\mathbf{w})}{\partial w_0} = w_0 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_0^{(i)} = w_0 + \frac{0.2}{4}\left[\left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(1)}\right)\right) + \left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(2)}\right)\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(3)}\right)\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(4)}\right)\right)\right]$$

(0-0.3601) (0-0.3236) (1-0.4844) (1-0.4579)

$$temp1 = w_1 + \frac{\alpha}{N}\frac{\partial \ell(\mathbf{w})}{\partial w_1} = w_1 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_1^{(i)} w_1 + \frac{0.2}{4}\left[\left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(1)}\right)\right)x_1^{(1)} + \left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(2)}\right)\right)x_1^{(2)} + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(3)}\right)x_1^{(3)}\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(4)}\right)x_1^{(4)}\right)\right]$$

(0-0.3601)*3 (0-0.3236)*3 (1-0.4844)*4 (1-0.4579)*3

$$temp2 = w_2 + \frac{\alpha}{N}\frac{\partial \ell(\mathbf{w})}{\partial w_2} = w_2 + \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - \sigma(\mathbf{w}^T\mathbf{x}^{(i)}))x_2^{(i)} = w_2 + \frac{0.2}{4}\left[\left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(1)}\right)\right)x_2^{(1)} + \left(0 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(2)}\right)\right)x_2^{(2)} + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(3)}\right)x_2^{(3)}\right) + \left(1 - \sigma\left(\mathbf{w}^T\mathbf{x}^{(4)}\right)x_2^{(4)}\right)\right]$$

(0-0.3601)*4 (0-0.3236)*5 (1-0.4844)*1 (1-0.4579)*(1.5)

$w_0^{new} = temp0$ 0.019

$w_1^{new} = temp1$ 0.107

$w_2^{new} = temp2$ -0.249

Vectorized Prediction

For this slide, we want $\hat{y} = p(y \mid \mathbf{x}; \mathbf{w})$

For $X = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix}$ $\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$ and initial weights $\mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$,

we predict $\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \\ \hat{y}^{(4)} \end{bmatrix} = \begin{bmatrix} \sigma(\mathbf{w}^T \mathbf{x}^{(1)}) \\ \sigma(\mathbf{w}^T \mathbf{x}^{(2)}) \\ \sigma(\mathbf{w}^T \mathbf{x}^{(3)}) \\ \sigma(\mathbf{w}^T \mathbf{x}^{(4)}) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} = \text{sigmoid}(X\mathbf{w})$

Vectorized Gradient

$$\frac{\partial \ell(\mathbf{w})}{\partial w_j} = \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)} = \sum_{i=1}^{N} (y^{(i)} - \sigma(\mathbf{w}^T \mathbf{x}^{(i)})) x_j^{(i)}$$

$$\nabla \ell(\mathbf{w}) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ x_d^{(1)} & x_d^{(2)} & \cdots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} y^{(1)} - \hat{y}^{(1)} \\ y^{(2)} - \hat{y}^{(2)} \\ \vdots \\ y^{(N)} - \hat{y}^{(N)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)}) x_0^{(i)} \\ \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)}) x_1^{(i)} \\ \vdots \\ \sum_{i=1}^{N} (y^{(i)} - \hat{y}^{(i)}) x_d^{(i)} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell(\mathbf{w})}{\partial w_0} \\ \frac{\partial \ell(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial \ell(\mathbf{w})}{\partial w_d} \end{bmatrix} = X^T (y - \text{sigmoid}(X\mathbf{w}))$$

Example: $X = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix}$ $\mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

$$\nabla \ell(\mathbf{w}) = \begin{bmatrix} \frac{\partial \ell(\mathbf{w})}{\partial w_0} \\ \frac{\partial \ell(\mathbf{w})}{\partial w_1} \\ \frac{\partial \ell(\mathbf{w})}{\partial w_2} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 3 & 4 & 3 \\ 4 & 5 & 1 & 1.5 \end{bmatrix}}_{X^T} \underbrace{\begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \\ \hat{y}^{(4)} \end{bmatrix} \end{bmatrix}}_{(\mathbf{y} - \hat{\mathbf{y}})} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 3 & 4 & 3 \\ 4 & 5 & 1 & 1.5 \end{bmatrix}}_{X^T} \underbrace{\begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \text{sigmoid} \left(\begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \right) \end{bmatrix}}_{(\mathbf{y} - \text{sigmoid}(X\mathbf{w}))}$$

Vectorized Gradient Ascent Algorithm

for k = 1 to num_iter

$$\mathbf{w} = \mathbf{w} + \frac{\alpha}{N}\left(X^T(\mathbf{y} - \text{sigmoid}(X\mathbf{w}))\right)$$

Example where $\alpha = 0.2$

for k = 1 to num_iter

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} + \frac{0.2}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 3 & 4 & 3 \\ 4 & 5 & 1 & 1.5 \end{bmatrix} \left[\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \text{sigmoid}\left(\begin{bmatrix} 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 1 \\ 1 & 3 & 1.5 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \right) \right]$$

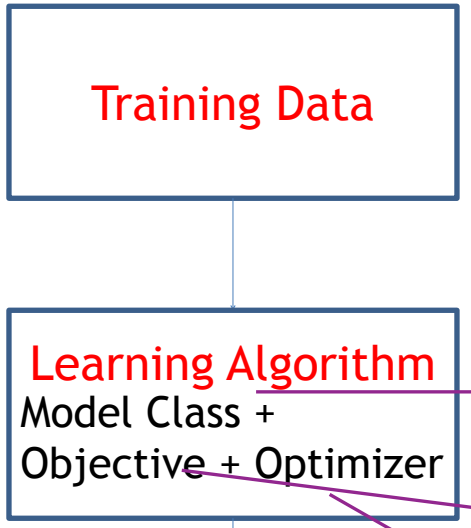
Prediction

After 200 iterations of gradient with a learning rate of 0.2, $\mathbf{w} = \begin{bmatrix} 0.551 \\ 2.011 \\ -2.48 \end{bmatrix}$

Given a feature vector for a new flower $\mathbf{x} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix}$ we predict the probability of being a Setosa Iris is $\sigma(-0.856)$, i.e. the flower has a 0.3 chance of being a Setosa Iris.

If we had to predict which flower we would predict it is a Versicolor Iris.

Supervised Learning



$$p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

$$\frac{1}{N}\ell(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)} \ln \sigma(\mathbf{w}^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \ln\left(1 - \sigma(\mathbf{w}^T \mathbf{x}^{(i)})\right)\right]$$

Vectorized gradient ascent

Input (say) patient data

$\mathbf{x}$

Hypothesis or function

Output 0/1. 0 no cancer, 1 cancer

(Estimated probability of cancer)

Finding Parameters via Optimization
A general ML recipe

General ML problem	Multiple linear regression
❑Get data	→ Data: $(\mathbf{x}^{(i)}, y^{(i)}), i = 1,2,\ldots,N$
❑Pick a model with parameters	→ **Logistic function** applied to $\mathbf{w}^T\mathbf{x}$ $$p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$
❑Pick a Objective/ loss function ◦ Measures goodness of fit model to data ◦ Function of the parameters	→ **Objective function:** $$\mathbf{w}* = \arg\max_{\mathbf{w}} \left(\frac{1}{N} \sum_{i=1}^{N} y^{(i)} \ln(h(\mathbf{x})) + (1 - y^{(i)}) \ln(1 - h(\mathbf{x})) \right)$$ where $h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$
❑Find parameters that maximizes likelihood	→ Select $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_d]^T$ to maximize $\frac{1}{N}\ell(\mathbf{w})$ (Or minimize $-\ell(\mathbf{w})$.)

Outline

- Motivating example: How can we classify? — How can we use a hyperplane for a classification problem?
- Estimating probabilities — Can we predict not only which class an example belongs to - but a confidence score of that classification
- Maximum likelihood — How can we find the most likely hyperplane? Could we write a function to describe how likely a hyperplane was to have generated the dataset?
  - Iterative approach - gradient ascent — Maximizing the function
- Thinking about different types of error — Some errors are more costly than other errors. Can we modify our predictions to decrease one type of error (and perhaps increase another type of error?
  - Accuracy
  - Motivating example
- Transformation of the features — Extending our algorithm to nonlinear decision boundaries
- Multiple classes — What if we have more than two classes?

Most Data not Perfectly Separable

- Generally cannot find a separating hyperplane
- Always, some points that will be mis-classified
- Algorithms attempt to find "good" hyper-planes
  - Reduce the number of mis-classified points
  - Or, some similar metric

What type of measurements do you think we should calculate?

Ideally, the training objective function should be the metric - but that is not always possible.

Training objective might not be a proxy for the real-world objective (secondary measure - use validation set to choose model)

Evaluating Performance

- Classification error: # mistakes made on the training set $\frac{\#\text{mistakes}}{\#\text{examples}}$
- Accuracy: fraction of correct examples $\frac{\#\text{correct}}{\#\text{examples}} = 1 - \frac{\#\text{mistakes}}{\#\text{examples}}$
  - Problem: Unbalanced datasets. Eg. Suppose your data consist of 99% positive examples, and 1% negative examples.

1. Given an unbalanced dataset containing 99% positive examples (class 1) and 1% negative examples (class 0). Do you think it is possible to find a classifier that has 99% accuracy?

- ○ No
- ○ Yes
- ○ Maybe - depends on the algorithm

Evaluating Performance

- Classification error: # mistakes made on the training set $\frac{\#\,\text{mistakes}}{\#\,\text{examples}}$
- Accuracy: fraction of correct examples $\frac{\#\,\text{correct}}{\#\,\text{examples}} = 1 - \frac{\#\,\text{mistakes}}{\#\,\text{examples}}$
  - Problem: Unbalanced datasets. Eg. Suppose your data consist of 99% positive examples, and 1% negative examples.

    If you predict positive (without looking at **x**), your accuracy is 99%!

What other error measurements make sense?

Hard Decisions

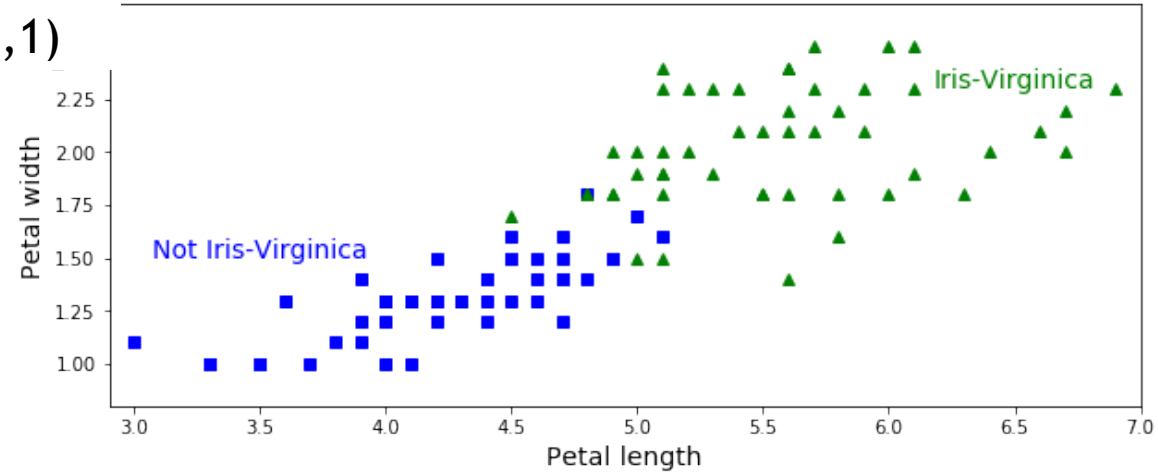

Iris virginica

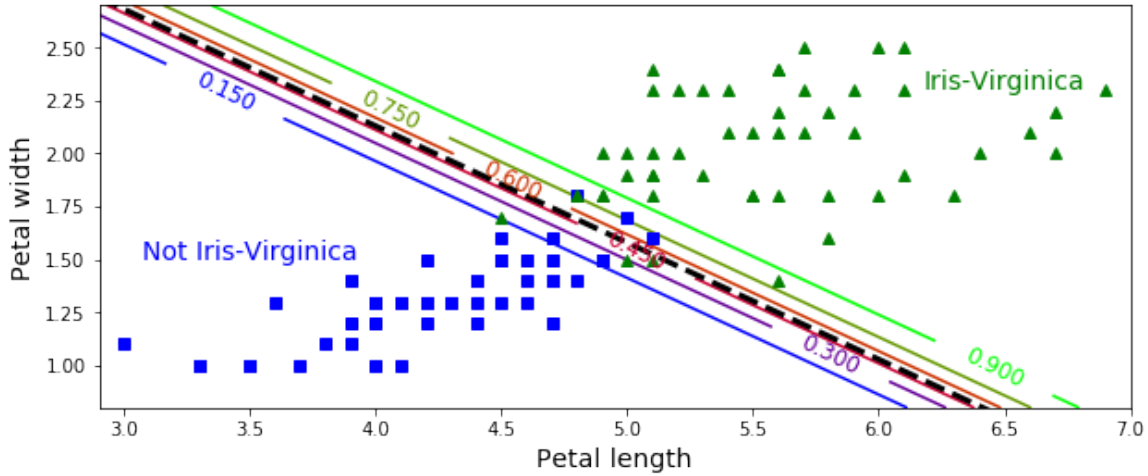

Iris versicolor


Iris setosa

iris pictures from https://en.wikipedia.org/wiki/Iris_flower_data_set

- Logistic classifier outputs a soft label: $P(y = 1|x) \in (0,1)$
  - $P(y = 1|x) \approx 1 \Rightarrow y = 1$ more likely
  - $P(y = 0|x) \approx 1 \Rightarrow y = 0$ more likely
- We obtain a hard label by thresholding:
  - Set $\hat{y} = 1 \Leftrightarrow P(y = 1|x) > t$
  - t = Threshold
- How to set threshold?
  - Set $t = \frac{1}{2} \Rightarrow$ Minimizes cross entropy loss
  - Increasing $t \Rightarrow$ Decreases false positives
  - Decreasing $t \Rightarrow$ Decreases missed detections



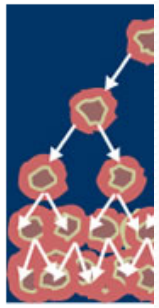


New example

Breast Cancer Wisconsin (Diagnostic) Data Set

Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	442524

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

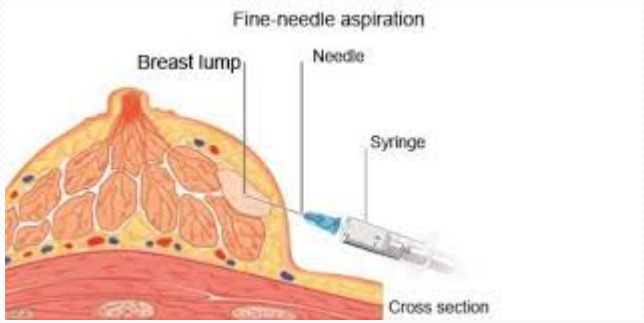
Ten real-valued features are computed for each cell nucleus:

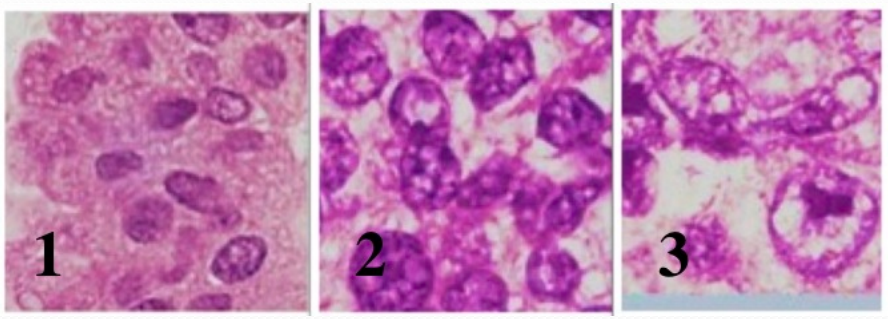
a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

- First publication: O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995

Diagnosing Breast Cancer

- Fine needle aspiration of suspicious lumps
- Cytopathologist visually inspects cells
  - Sample is stained and viewed under microscope
- Determines if cells are benign or malignant and furthermore provides grading if malignant
- Uses many features:
  - Size and shape of cells, degree of mitosis, differentiation, …
- Diagnosis is not exact
- If uncertain, use a more comprehensive biopsy
  - Additional cost and time
  - Stress to patient
- Can machine learning provide better rules?





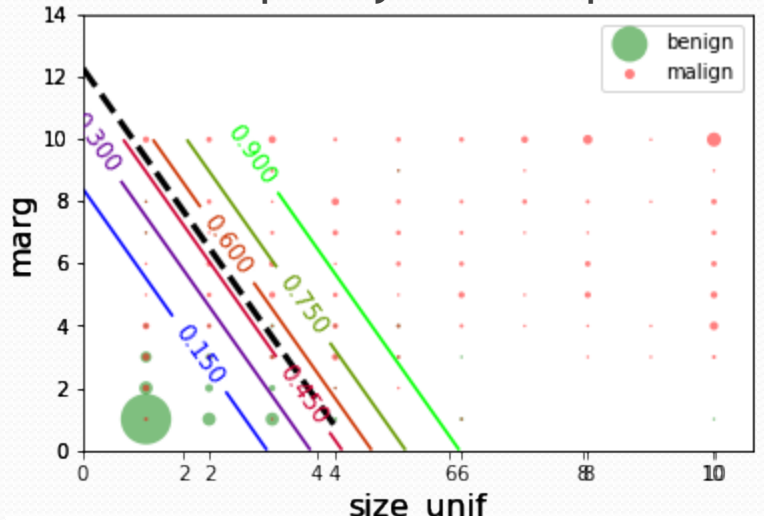
Grades of carcinoma cells
http://breast-cancer.ca/5a-types/

Example - predicting breast cancer

We have 683 examples

	id	thick	size_unif	shape_unif	marg	cell_size	bare	chrom	normal	mit	class
0	1000025	5	1	1	1	2	1.0	3	1	1	2
1	1002945	5	4	4	5	7	10.0	3	2	1	2
2	1015425	3	1	1	1	2	2.0	3	1	1	2
3	1016277	6	8	8	1	3	4.0	3	7	1	2
4	1017023	4	1	1	3	2	1.0	3	1	1	2
5	1017122	8	10	10	8	7	10.0	9	7	1	4

- Scatter plot of points from each class
- Plot not informative
  - Many points overlap
  - Relative frequency at each point

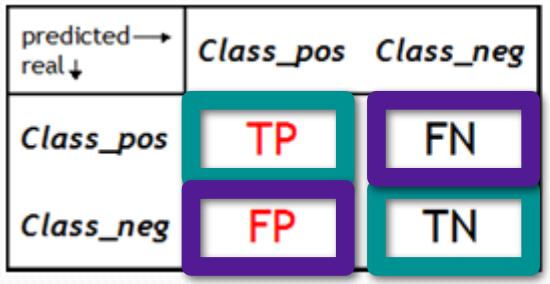


`w=[-5.50017433,1.29037381 0.44738685]`

Plot is using only 2 of the features.

Errors in Binary Classification

- Doctors want to know if a patient has breast cancer
- This is a binary classification problem
- There are two types of **errors** an example can have:
  - Type 1 error. The doctor predicts the patient has cancer but the patient doesn't. **False positive (FP)**
  - Type 2 error. The doctor predicts the patient doesn't have cancer, but the patient does. **False negative (FN)**
- Their are two types of **correctly** classified examples:
  - The doctor *correctly* predicts the patient has cancer. **True positive (TP)**
  - The doctor *correctly* predicts the patient doesn't have cancer. **True negative (TN)**
- Confusion matrix
  - The diagonal contains TP and TN
  - The off diagonal contains the FP and FN
- Uses of a confusion matrix:
  - Shows if the classes are imbalanced
  - Shows the different types of errors



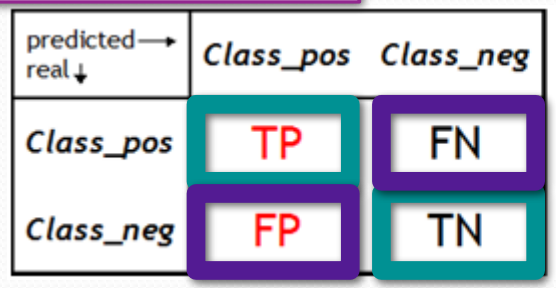
$$\text{TPR (sensitivity)} = \frac{TP}{TP + FN}$$

$$\text{FPR (1-specificity)} = \frac{FP}{TN + FP}$$

Errors in Binary Classification

- Doctors want to know if a patient has breast cancer
- This is a binary classification problem
- There are two types of **errors** an example can have:
  - Type 1 error. The doctor predicts the patient has cancer but the patient doesn't. **False positive (FP)**
  - Type 2 error. The doctor predicts the patient doesn't have cancer, but the patient does. **False negative (FN)**
- Their are two types of **correctly** classified examples:
  - The doctor *correctly* predicts the patient has cancer. **True positive (TP)**
  - The doctor *correctly* predicts the patient doesn't have cancer. **True negative (TN)**
- Confusion matrix
  - The diagonal contains TP and TN
  - The off diagonal contains the FP and FN
- Uses of a confusion matrix:
  - Shows if the classes are imbalanced
  - Shows the different types of errors

There is often a trade off between the TPR and the FPR.



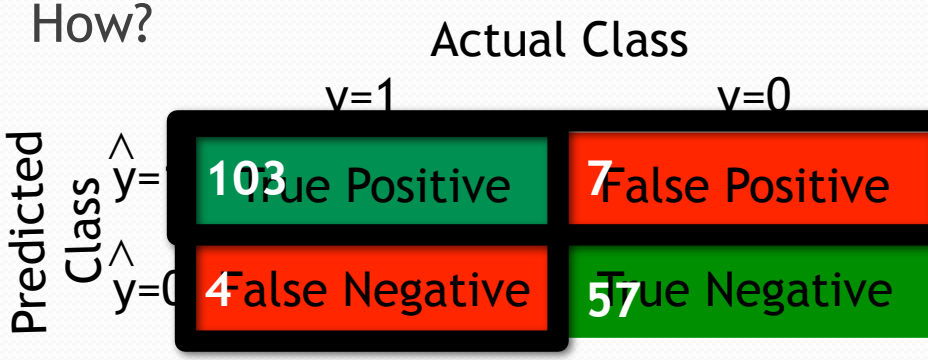
We can plot the FPR and TPR for different cutoff values - e.g. 0.2, 0.3, 0.4, 0.5, …, 0.9

By choosing a value different than 0.5 as our cutoff we can change the number of FP and FN

Evaluating Errors: Precision and Recall

Notice we changed the labels on the conclusion matrix Always check the labels

- How can we evaluate the performance of our classifier?
- For binary classification, we can easily get half the answers right! How?



- Precision and Recall
  - Precision- fraction of predicted positive answers that were really positive
  - Recall - fraction of positive answers you correctly calculated (Aka *TPR*, *sensitivity*)

Best is 1 and worst is 0

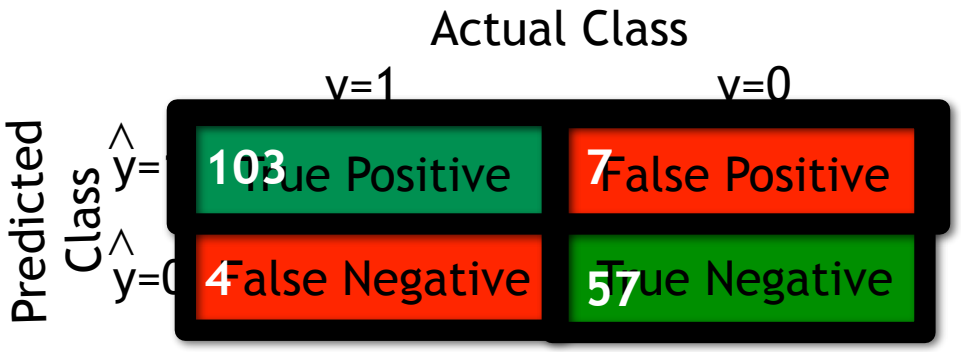
$$\frac{\#\text{ True Positive}}{\#\text{ True Positive} +\#\text{ False Positive}} \qquad \frac{103}{103+7}$$

$$\frac{\#\text{ True Positive}}{\#\text{ True Positive} +\#\text{False Negative}} \qquad \frac{103}{103+4}$$

Pair share: how could you get 100% recall?

Overview

$$\text{Accuracy} = \frac{\#\,\text{correct}}{\#\,\text{examples}} = \frac{103 + 57}{103 + 57 + 4 + 7}$$



Precision $\frac{\#\,\text{True Positive}}{\#\,\text{True Positive} + \#\,\text{False Positive}}$ $\frac{103}{103+7}$

Recall, *TPR*, *sensitivity*

$$\frac{\#\,\text{True Positive}}{\#\,\text{True Positive} + \#\text{False Negative}}$$

$$\frac{103}{103+4}$$

Specificity

$$\frac{\#\text{true negative}}{\#\text{true negative} + \#\text{false positive}}$$

$$\frac{57}{57 \quad +7}$$

Pair share:

If you raise the threshold, how do you expect recall to change?

If you raise the threshold, how do you expect precision change?

Toy example (numbers are made up)

True Positive	False Positive
False Negative	True Negative

t	TP	TN	FP	FN	ACC	Pr	Recall (TPR)	Specificity	1-Specificity (FPR)
0.5	4	6	1	2	10/13	4/5	4/6	6/7	1/7

Recall, *TPR*, *sensitivity*

$$\frac{\#\,\text{True Positive}}{\#\,\text{True Positive} + \#\text{False Negative}}$$

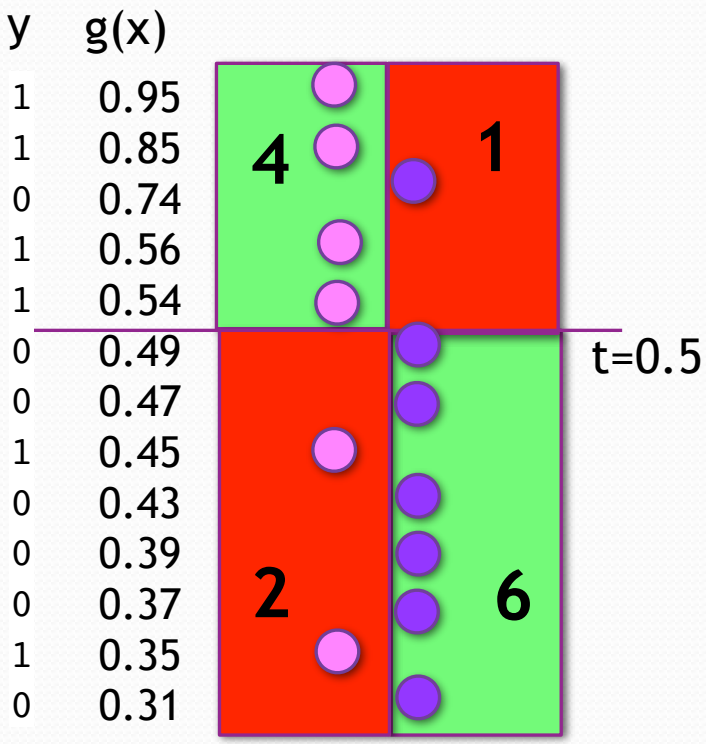
Specificity

$$\frac{\#\text{true negative}}{\#\text{true negative} + \#\text{false positive}}$$

Precision

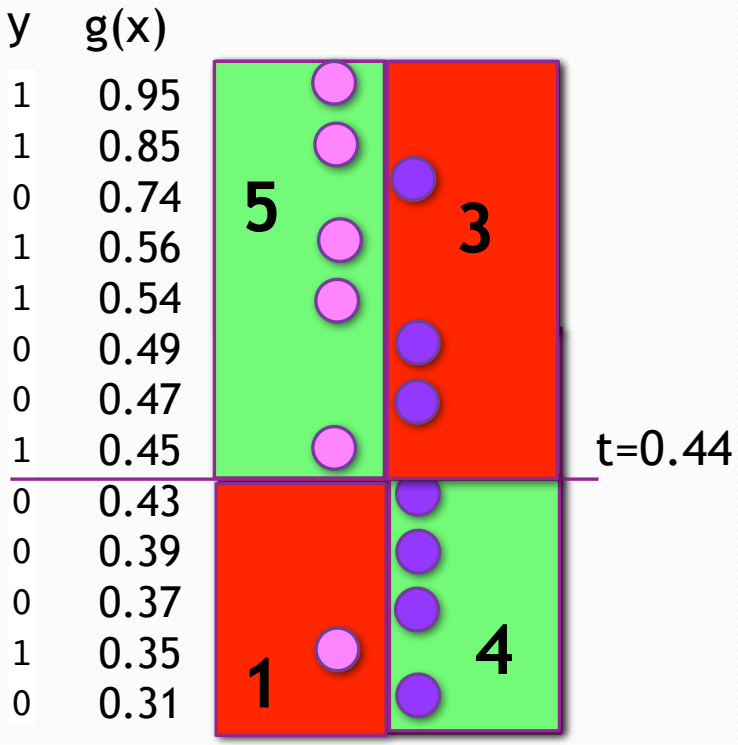
$$\frac{\#\,\text{True Positive}}{\#\,\text{True Positive} + \#\,\text{False Positive}}$$

$$FPR = \frac{\#\,FP}{\#\,FP + \#\,TN}$$



Toy example (numbers are made up)

True Positive	False Positive
False Negative	True Negative

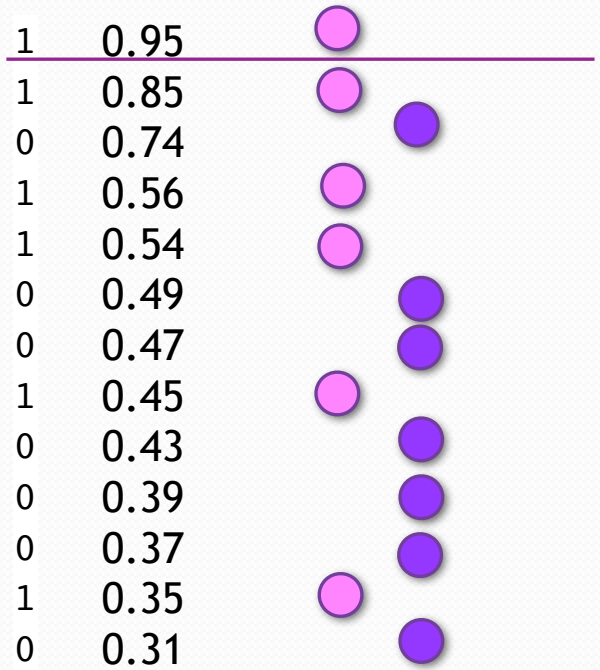


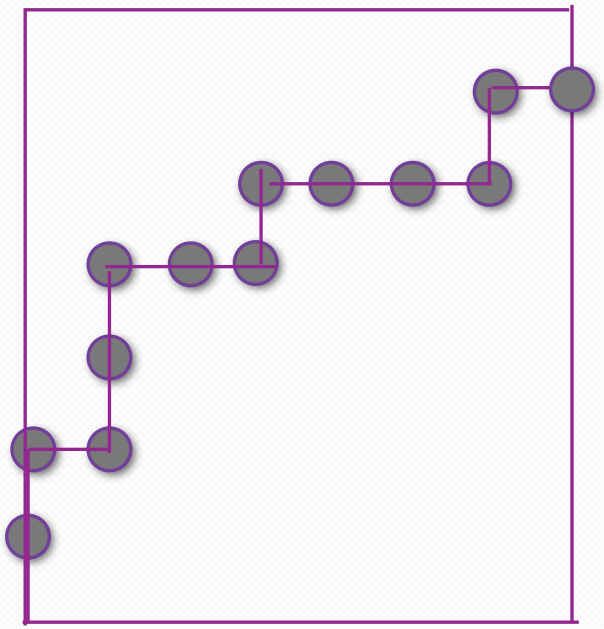
Receiver Operating Characteristic (ROC) curve

t	TPR	FPR
0.9	1/6	0
0.8	2/6	0
0.7	2/6	1/7
0.55	3/6	1/7
0.5	4/6	1/7
0.48	4/6	2/7
0.46	4/6	3/7
0.44	5/6	3/7
0.4	5/6	4/7
0.38	5/6	5/7
0.36	5/6	6/7
0.33	6/6	6/7
0.3	6/6	7/7

$$TPR = \frac{\#TP}{\#FN + \#TP} \qquad FPR = \frac{\#FP}{\#FP + \#TN}$$

y	g(x)
1	0.95
1	0.85
0	0.74
1	0.56
1	0.54
0	0.49
0	0.47
1	0.45
0	0.43
0	0.39
0	0.37
1	0.35
0	0.31





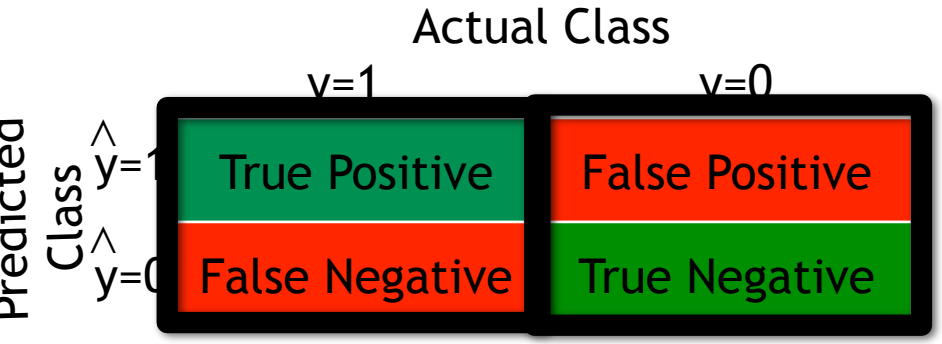
https://www.youtube.com/watch?v=21Igj5Pr6u4
https://www.youtube.com/watch?v=4jRBRDbJemM

Receiver Operating Characteristic (ROC) curve

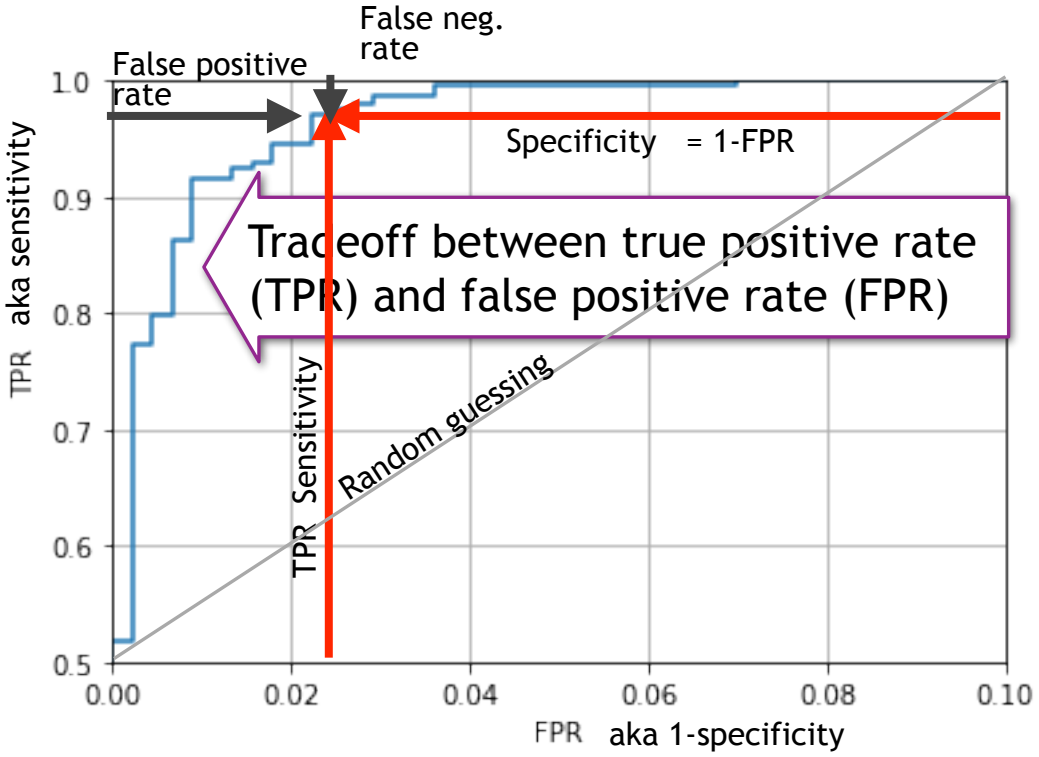
How many false positives we tolerate to get a certain number of true positives

- Varying the threshold obtains a set of classifiers.
- Trades off FPR and TPR for different hyperparameter settings (in this case, the threshold t)

Useful for unbalanced classes.



$$TPR = \frac{\#TP}{\#FN + \#TP} \qquad FPR = \frac{\#FP}{\#FP + \#TN}$$



https://www.youtube.com/watch?v=21Igj5Pr6u4
https://www.youtube.com/watch?v=4jRBRDbJemM

Receiver Operating Characteristic (ROC

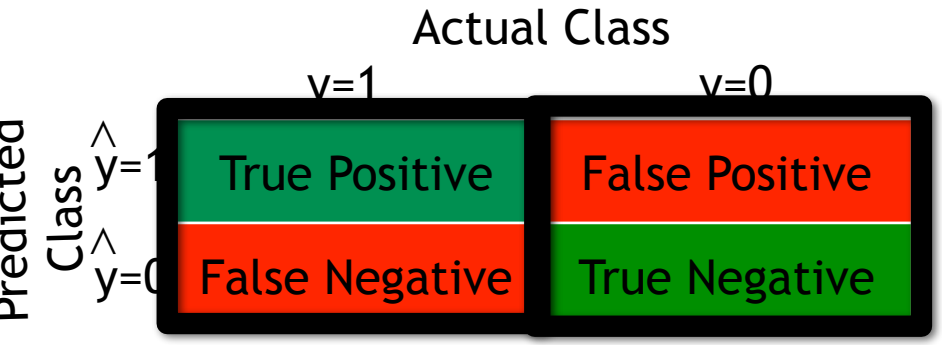
How many false positives we tolerate to get a certain number of true positives

If the data set is unbalanced, Precision recall curve is better than using accuracy

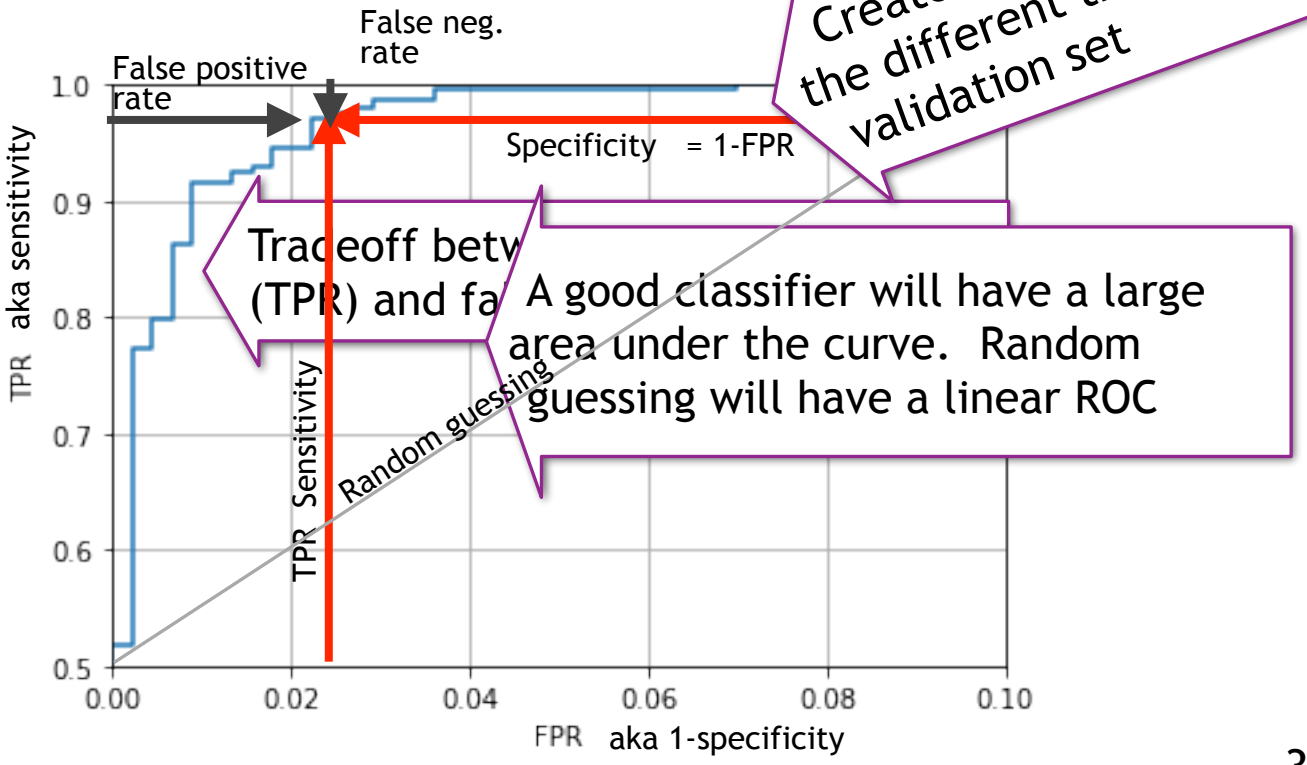
- Varying the threshold obtains a set of classifiers.
- Trades off FPR and TPR for different hyperparameter settings (in this case, the threshold t)

Useful for unbalanced classes.

Create plot by evaluating the different thresholds using validation set



$$TPR = \frac{\#TP}{\#FN + \#TP} \qquad FPR = \frac{\#FP}{\#FP + \#TN}$$



F1 Score

https://en.wikipedia.org/wiki/F1_score

$$\text{Precision} = \frac{\#\,\text{True Positive}}{\#\,\text{True Positive} + \#\,\text{False Positive}}$$

$$\text{Recall} = \frac{\#\,\text{True Positive}}{\#\,\text{True Positive} + \#\text{False Negative}}$$

The F1 score balances precision and recall

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The advantage of using F1 is we have a single number to evaluate the performance of an algorithm. The F1 score is between 0 and 1

Unlike accuracy, which can be misleading if the classes are unbalanced. F1 score gives one number indicating a score that is a balance between precision and recall

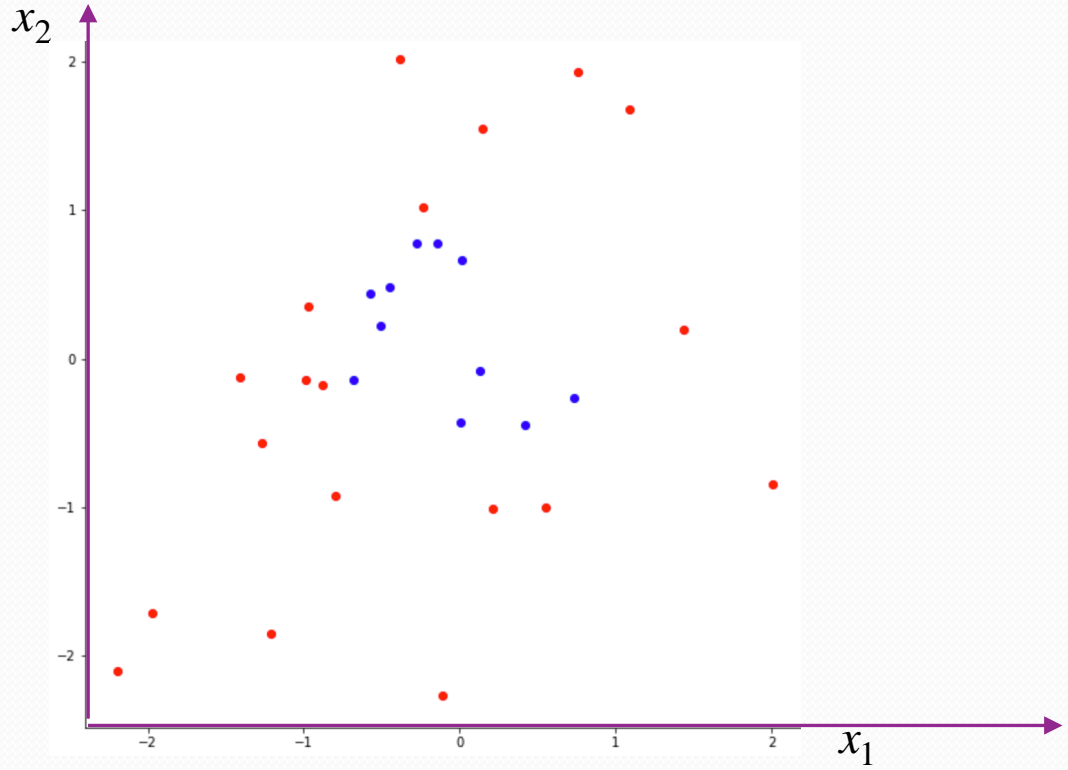
This may not be what you want if you care more about precision than recall, or if you care more about recall than precision

Outline

- Motivating example: How can we classify? — How can we use a hyperplane for a classification problem?
- Estimating probabilities — Can we predict not only which class an example belongs to - but a confidence score of that classification
- Maximum likelihood — How can we find the most likely hyperplane? Could we write a function to describe how likely a hyperplane was to have generated the dataset?
  - Iterative approach - gradient ascent — Maximizing the function
- Thinking about different types of error — Some errors are more costly than other errors. Can we modify our predictions to decrease one type of error (and perhaps increase another type of error?
  - Accuracy
  - Motivating example
- Transformation of the features — Extending our algorithm to nonlinear decision boundaries
- Multiple classes — What if we have more than two classes?

Non-linear Decision Boundary

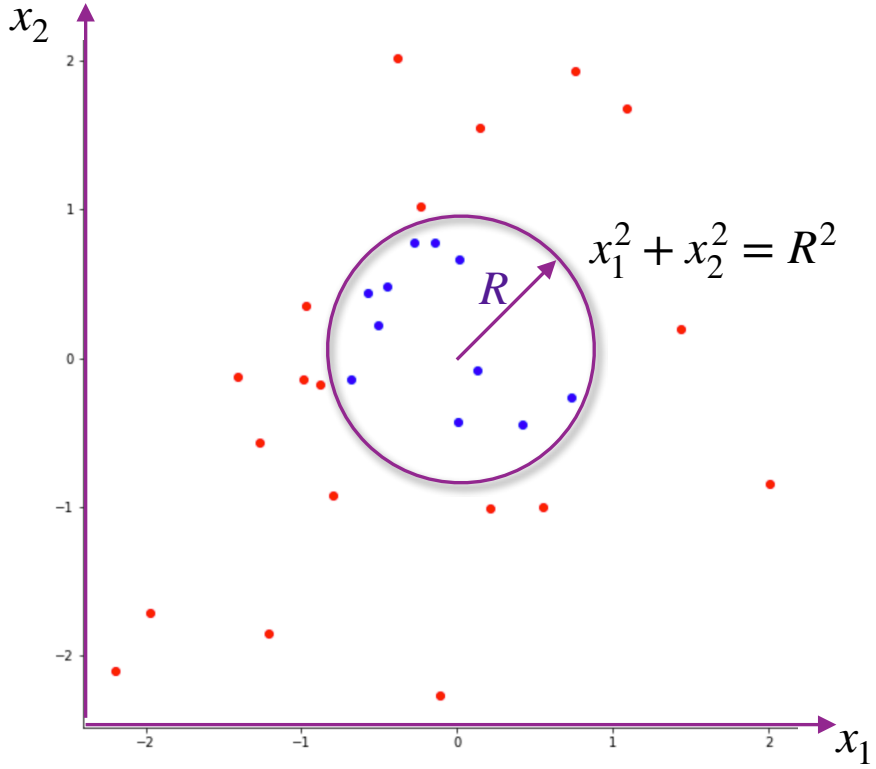
Could our logistic regression function do well on the dataset?

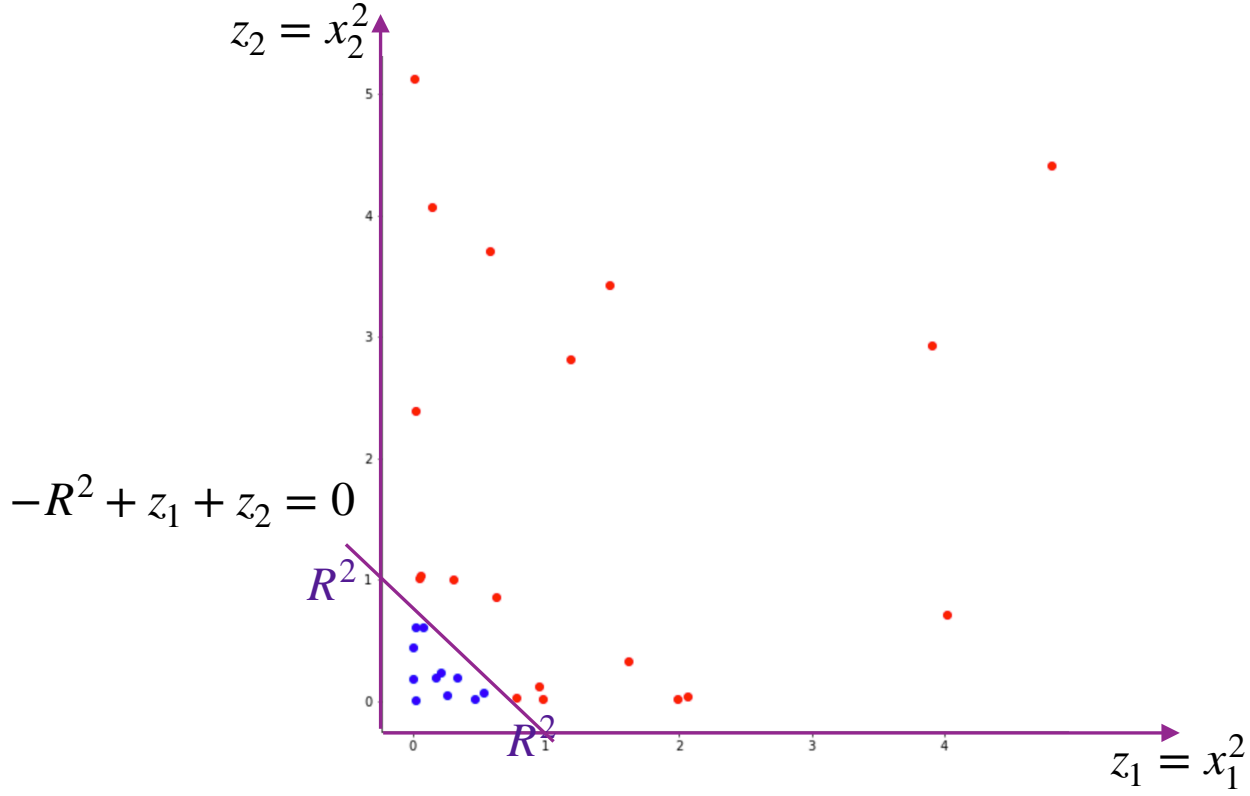


What do we do if our data had a more complex decision boundary?

Just as we did in linear regression, we can expand the model logistic model by transforming the features:

If we add features, can we separate the data?

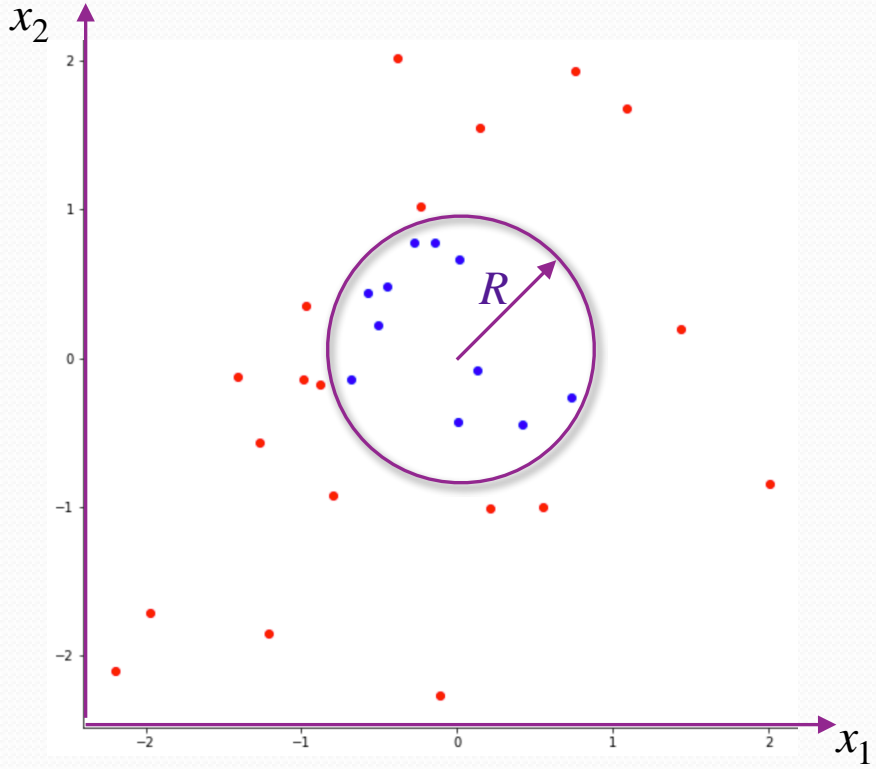


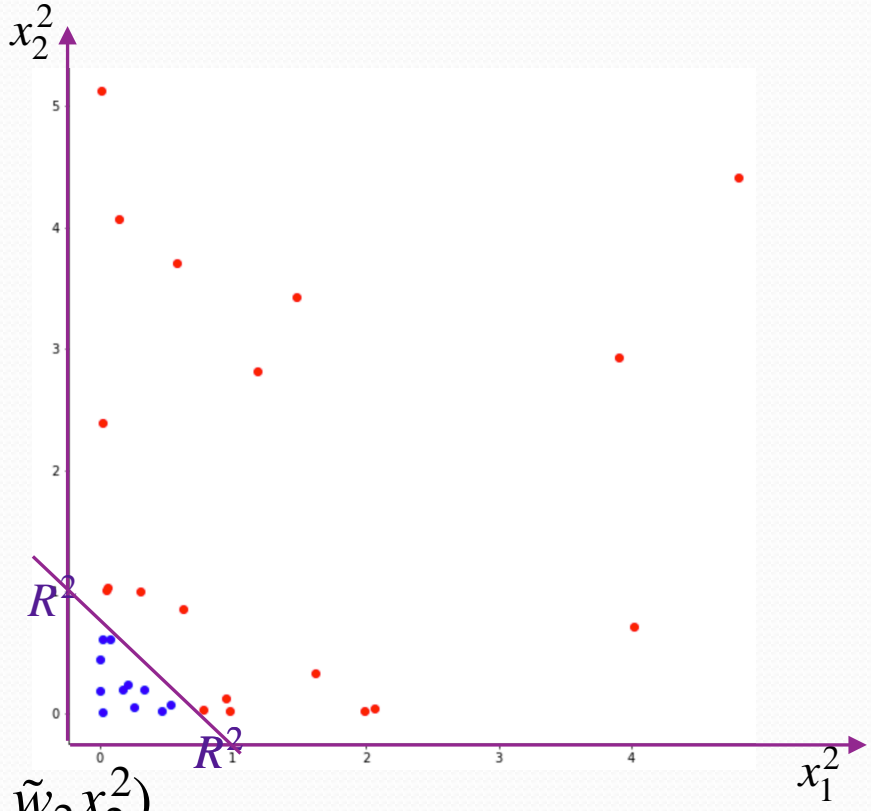


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ (x_1)^2 \\ (x_2)^2 \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix} \qquad \tilde{\mathbf{w}} = \begin{bmatrix} -R^2 \\ 1 \\ 1 \end{bmatrix}$$

Prediction





$$pr(y = 1 \mid \mathbf{x}) = \sigma(\tilde{w}_0 + +\tilde{w}_2 x_1^2 + \tilde{w}_2 x_2^2)$$

If $pr(y = 1 \mid \mathbf{x}) = \sigma(-R^2 + x_1^2 + x_2^2) \leq 0$ .5 predict blue

If $pr(y = 1 \mid \mathbf{x}) = \sigma(-R^2 + x_1^2 + x_2^2)$ > 0.5 predict red

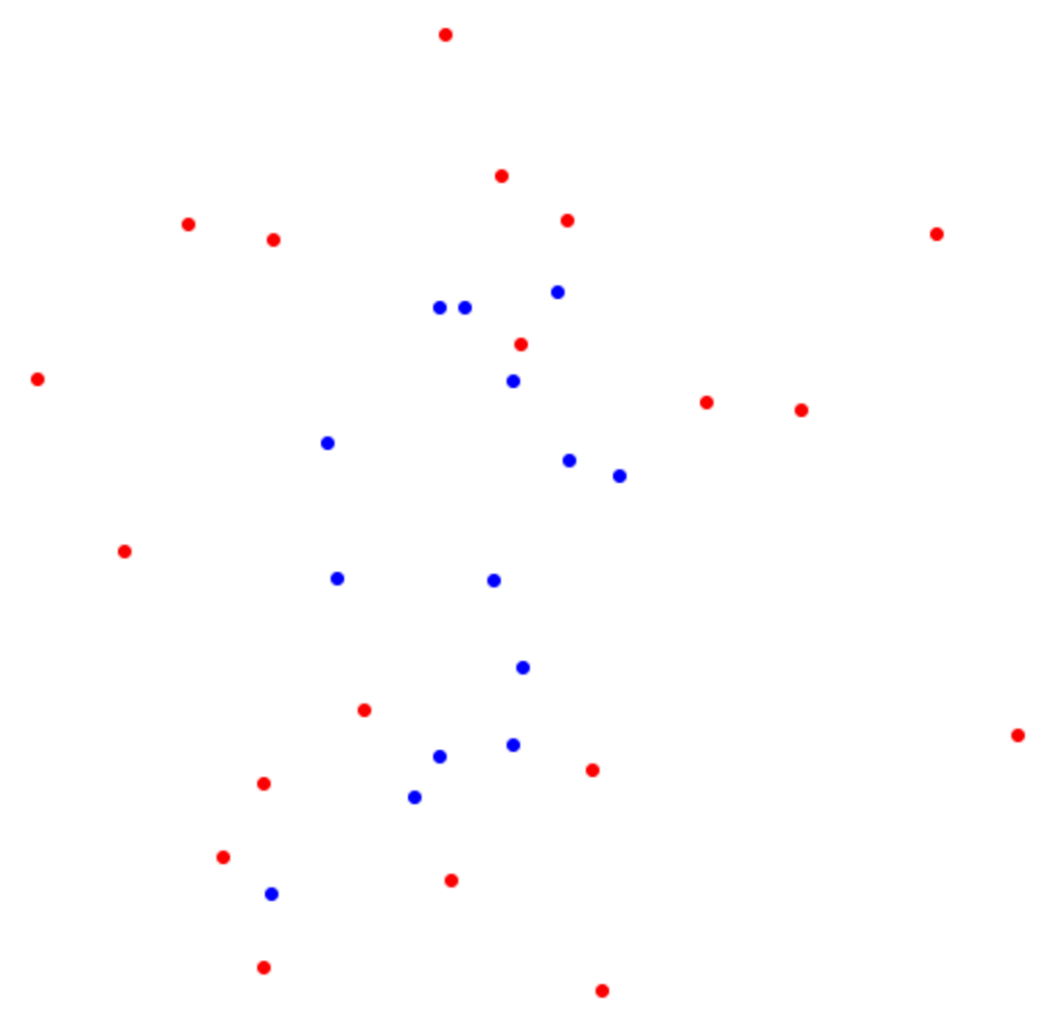
Pair share: what can we do to prevent overfitting?

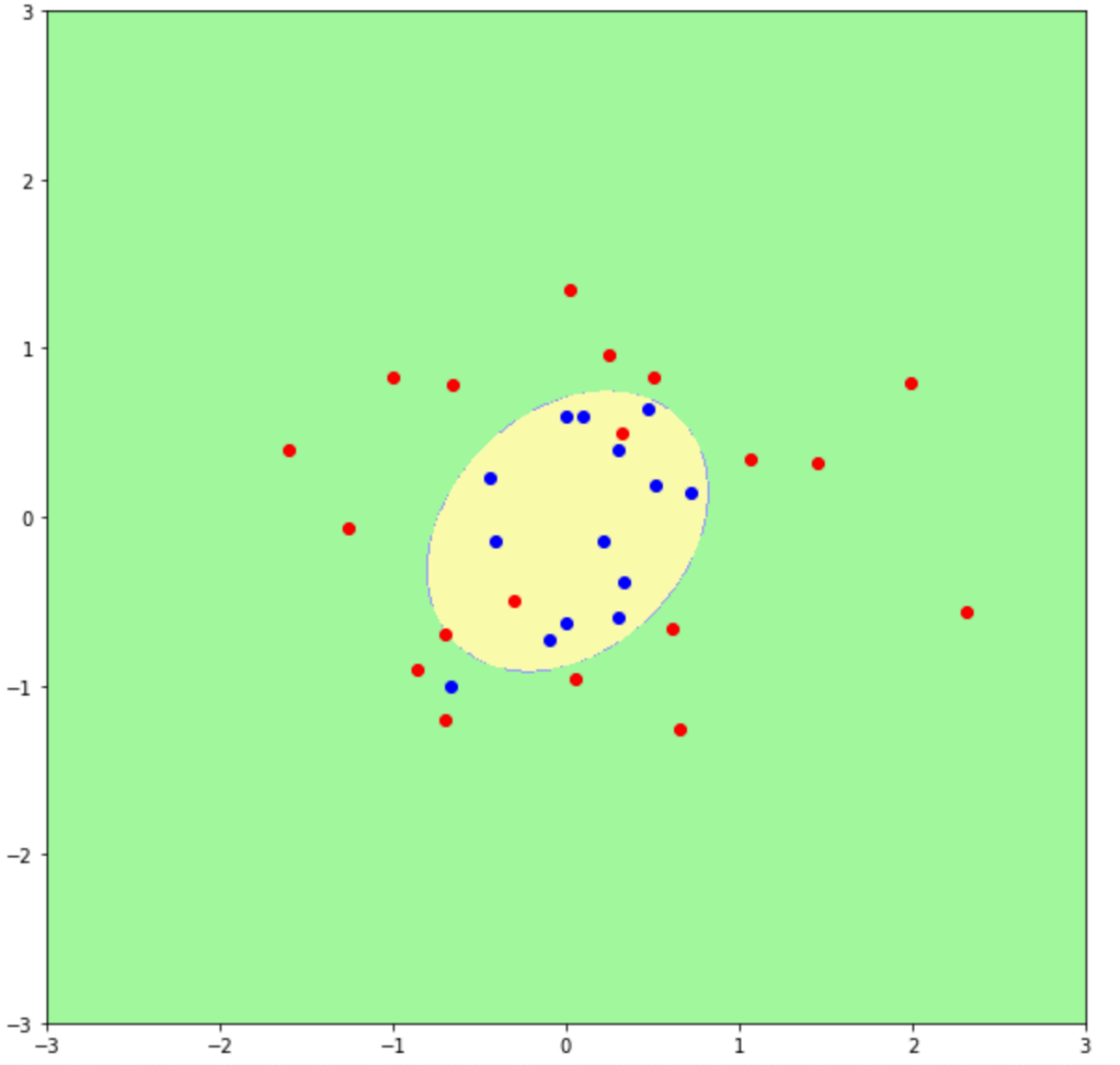
$$\Phi_2(\mathbf{x}) = \Phi_2([1, x_1, x_2]^T) = [1, x_1, x_1^2, x_2, x_2^2, x_1 x_2]^T$$

$$\Phi_3(\mathbf{x}) = \Phi_3([1, x_1, x_2]^T) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 & x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 \end{bmatrix}^T$$

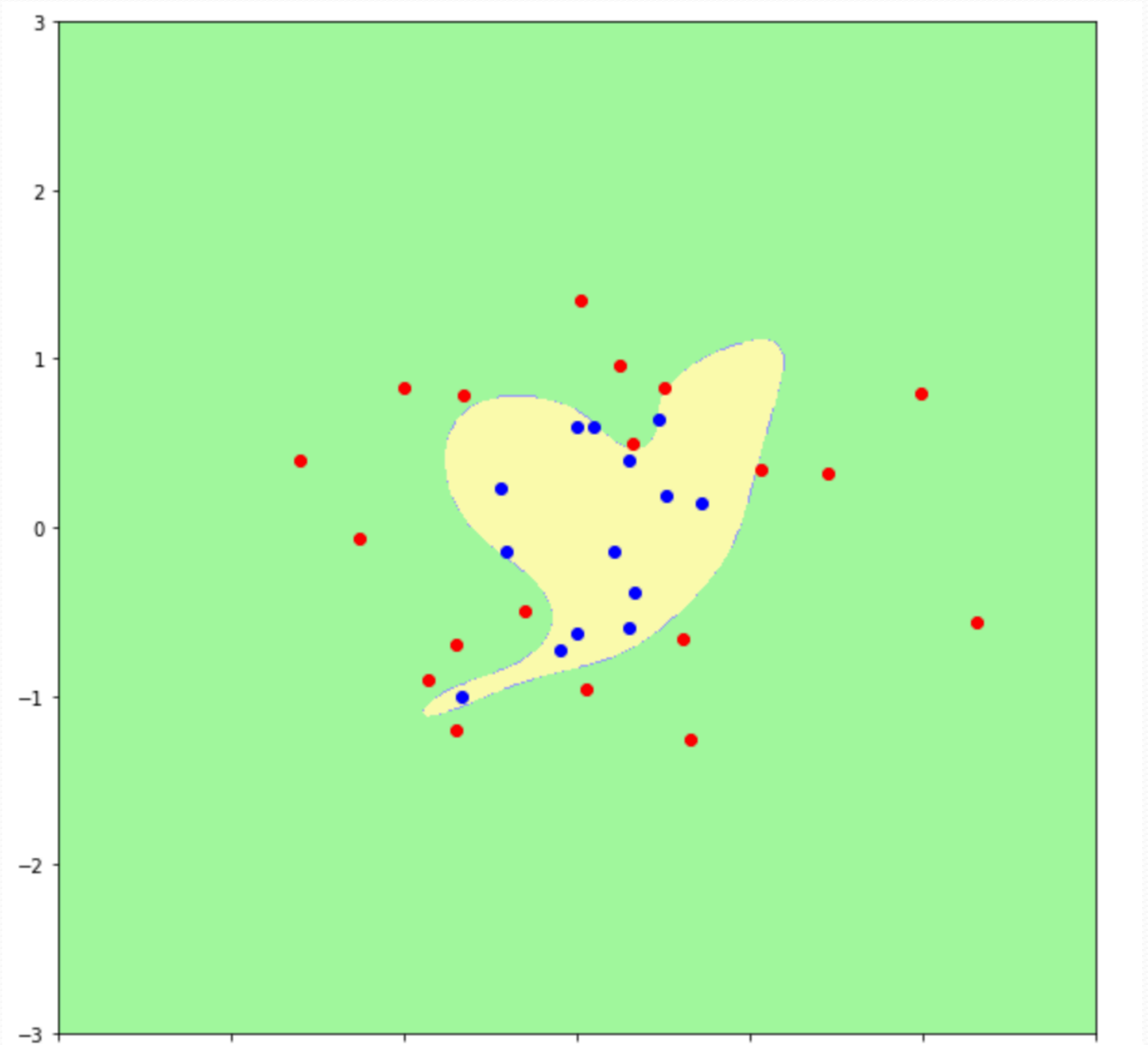
$$\Phi_4(\mathbf{x}) = \Phi_4([1, x_1, x_2]^T) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 & x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 & x_1^4 & x_1^3 x_2 & x_1^2 x_2^2 & x_1 x_2^3 & x_2^4 \end{bmatrix}^T$$

And so on




3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

```
-173.15294712]
92.67112466
-74.55278319
83.17160126
1216.06486182
508.53790679
-485.42371117
-1029.20177792
-268.94528728
129.60475013
144.9742966
-636.40288052
549.56073451
-887.16725134
-637.23456062
-323.39361867
-441.59715309
-87.41914152
-92.33185672
365.50986342
354.98813381
350.25543078
-225.00864839
284.77916437
-514.00666183
14.7827076
-996.79833852
-232.63112172
-135.31667883
-145.24288341
-87.81422812
```



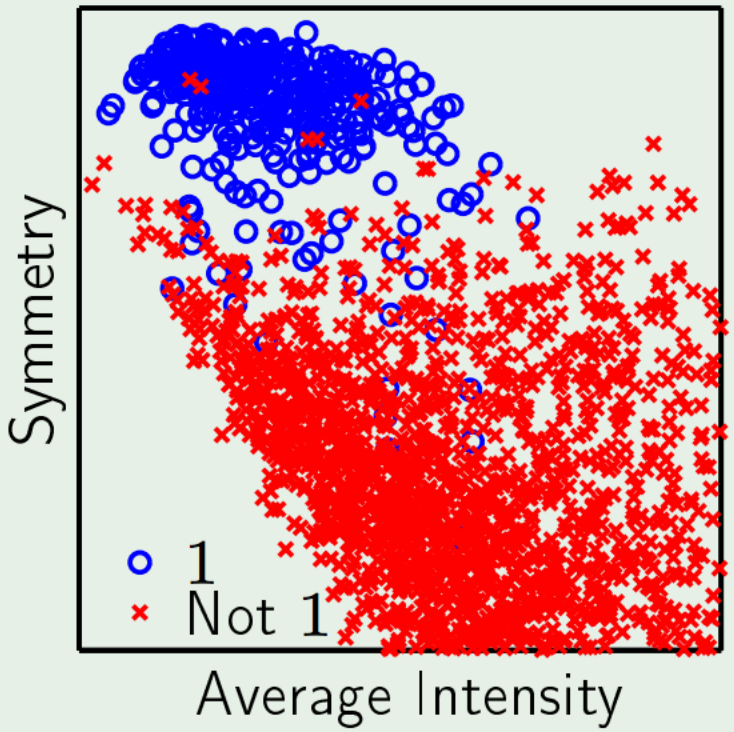
Too much flexibility in the model for quality and quantity of data → overfitting

Any transformation - not just polynomial

What should we do to choose the right transformation?

Cross validation in action

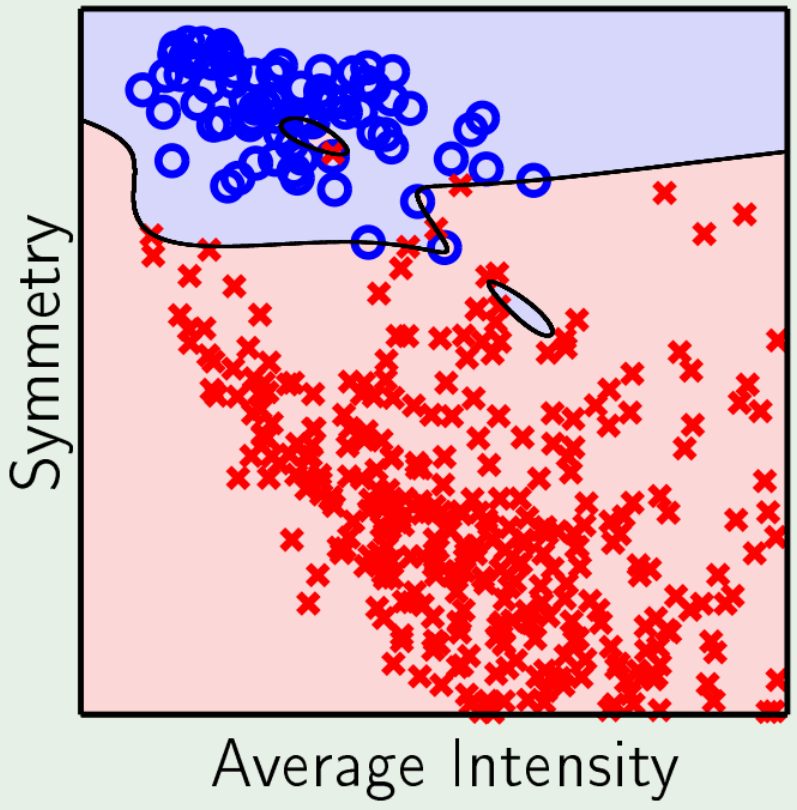
Digits classification task



Different errors

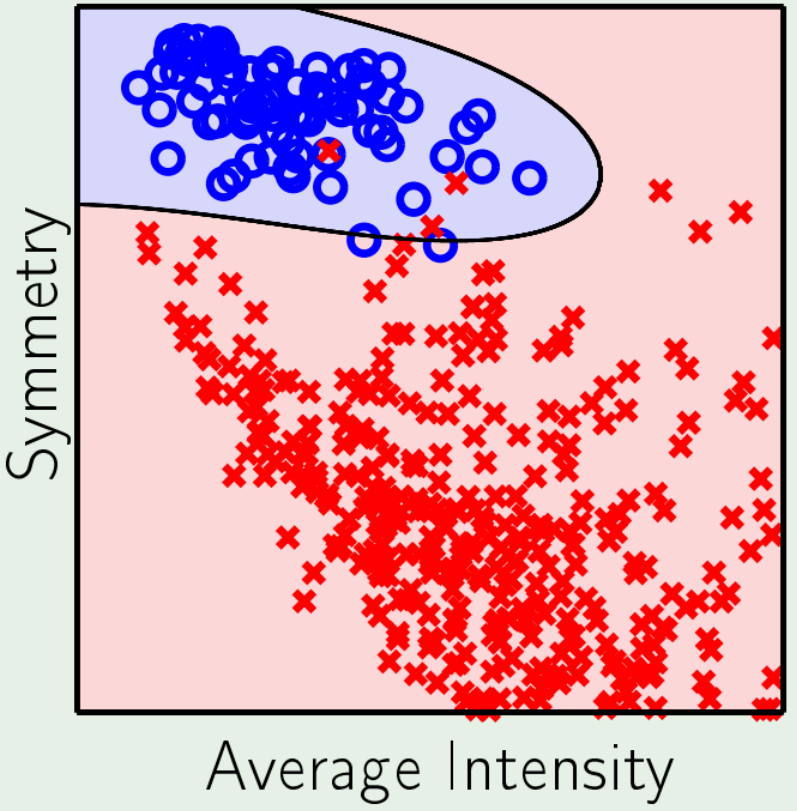
The result

without validation

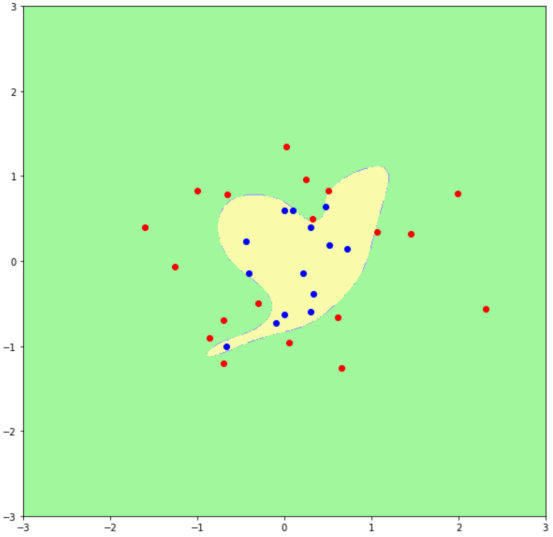


$E_{\text{in}} = 0\%$ $E_{\text{out}} = 2.5\%$

with validation



$E_{\text{in}} = 0.8\%$ $E_{\text{out}} = 1.5\%$



How can we keep the coefficients small (perform constrained optimization)?

```
[-173.15294712]
[[   92.67112466   -74.55278319    83.17160126  1216.06486182
    508.53790679  -485.42371117 -1029.20177792  -268.94528728
    129.60475013   144.9742966   -636.40288052   549.56073451
   -887.16725134  -637.23456062  -323.39361867  -441.59715309
    -87.41914152   -92.33185672   365.50986342   354.98813381
    350.25543078  -225.00864839   284.77916437  -514.00666183
     14.7827076   -996.79833852  -232.63112172  -135.31667883
   -145.24288341   -87.81422812   -57.65899988    83.89333379
    226.1990253    519.98132067   136.65298851   386.84309338
     12.18753244   230.21341886  -163.04847865    89.66107927
   -361.87286805   -80.1085153   -654.02299793   422.21633067
    -43.94900921    -7.91699718   -97.80530435   -39.42376437
    -28.36048529    54.79613101   104.74082391   211.75655274
    330.80830005  -464.99851205   275.23559716   114.38763131
    179.18806538   -24.00846735   110.51462689  -103.95211102
     43.47582249  -183.30214877    49.99173714  -160.95912587
   1207.05583987]]
```