Do not distribute course material

Topic 3 continued
Linear Classification & Logistic Regression

PROF. LINDA SELLIE

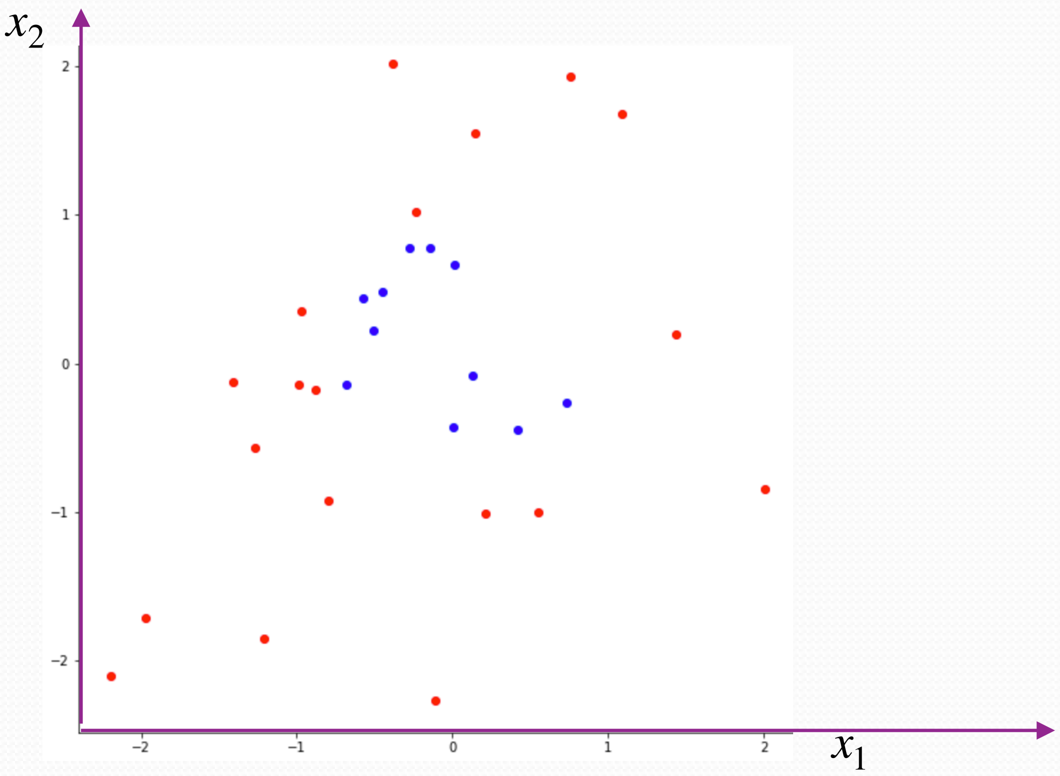
Some slides/approaches used are from Prof. Rangan
Many approaches used are from CMU 18-661

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
- Preventing overfitting
- Multiple classes — What if we have more than two classes?

Non-linear Decision Boundary

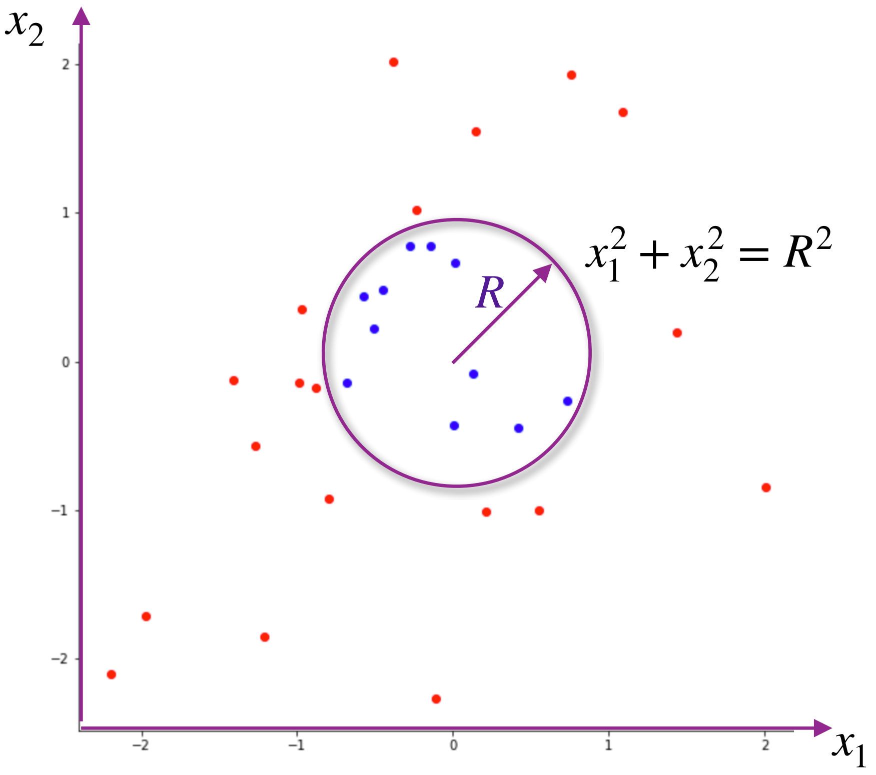
Could our logistic regression function do well on the dataset?

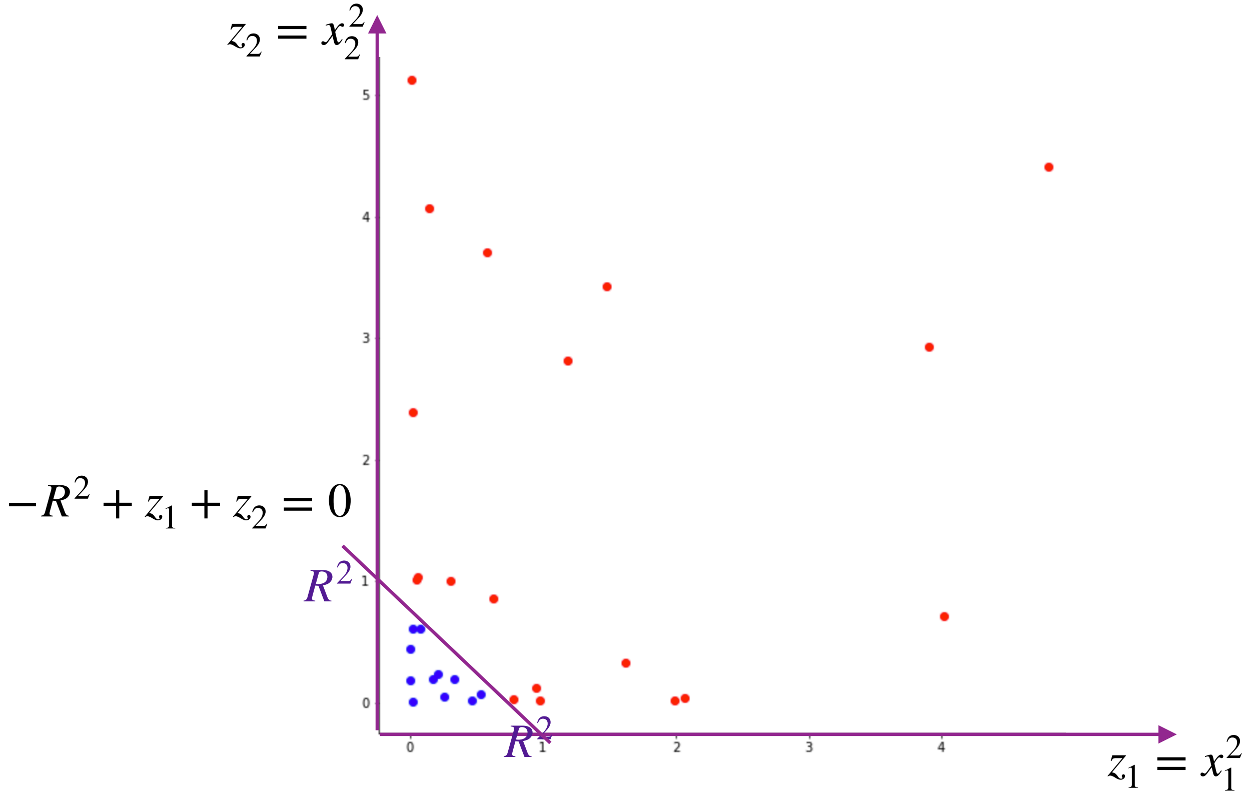


What do we do if our data had a more complex decision boundary?

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?

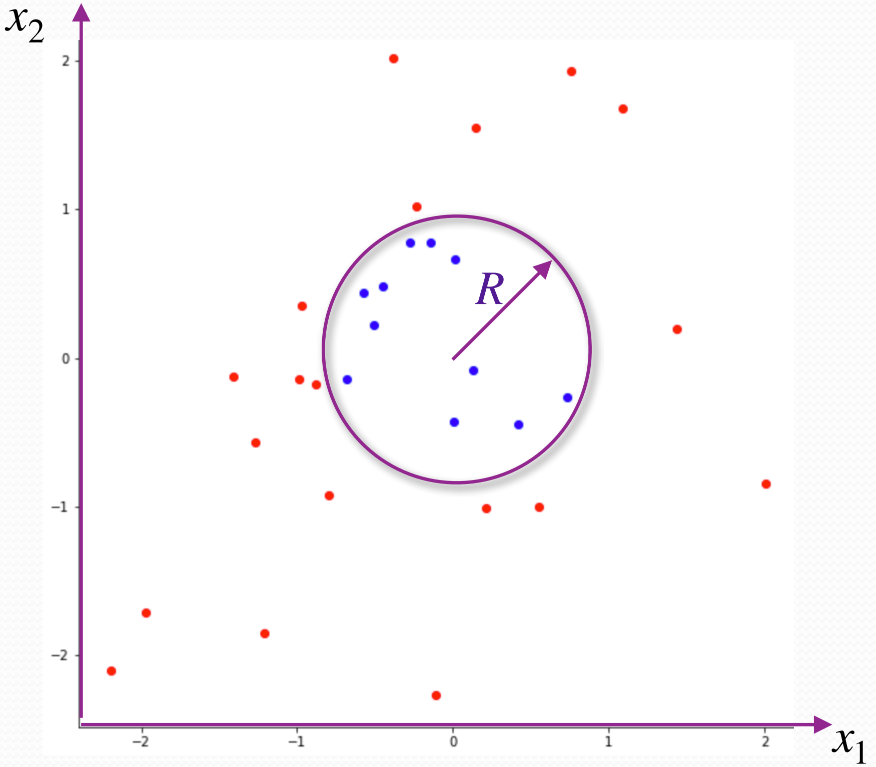


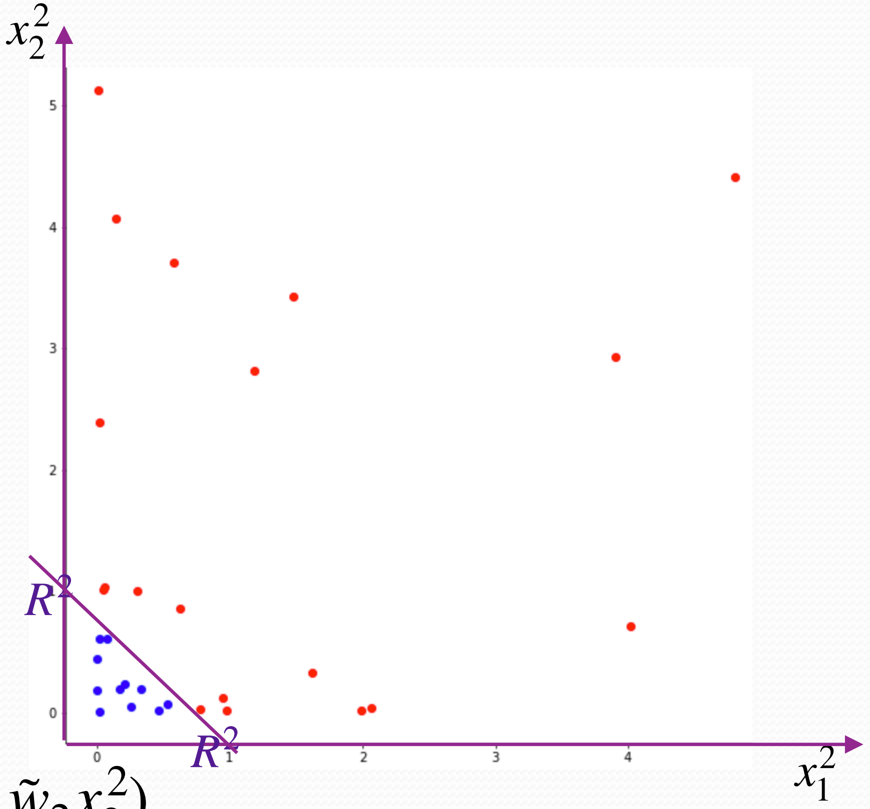


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

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

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?

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

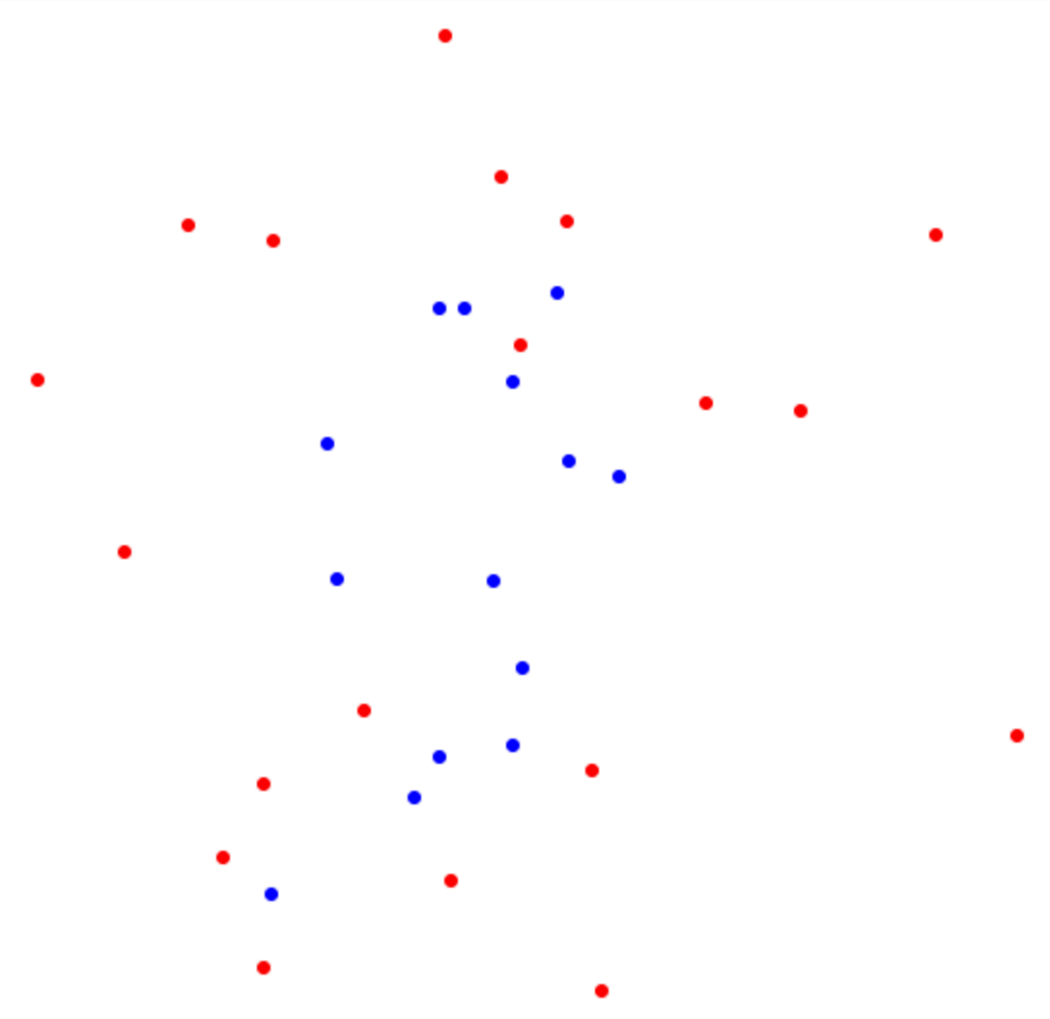
$$\Phi_3(\mathbf{x}) = \Phi_3([1, x_1, x_2]^T) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 & x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 \end{bmatrix}^T$$

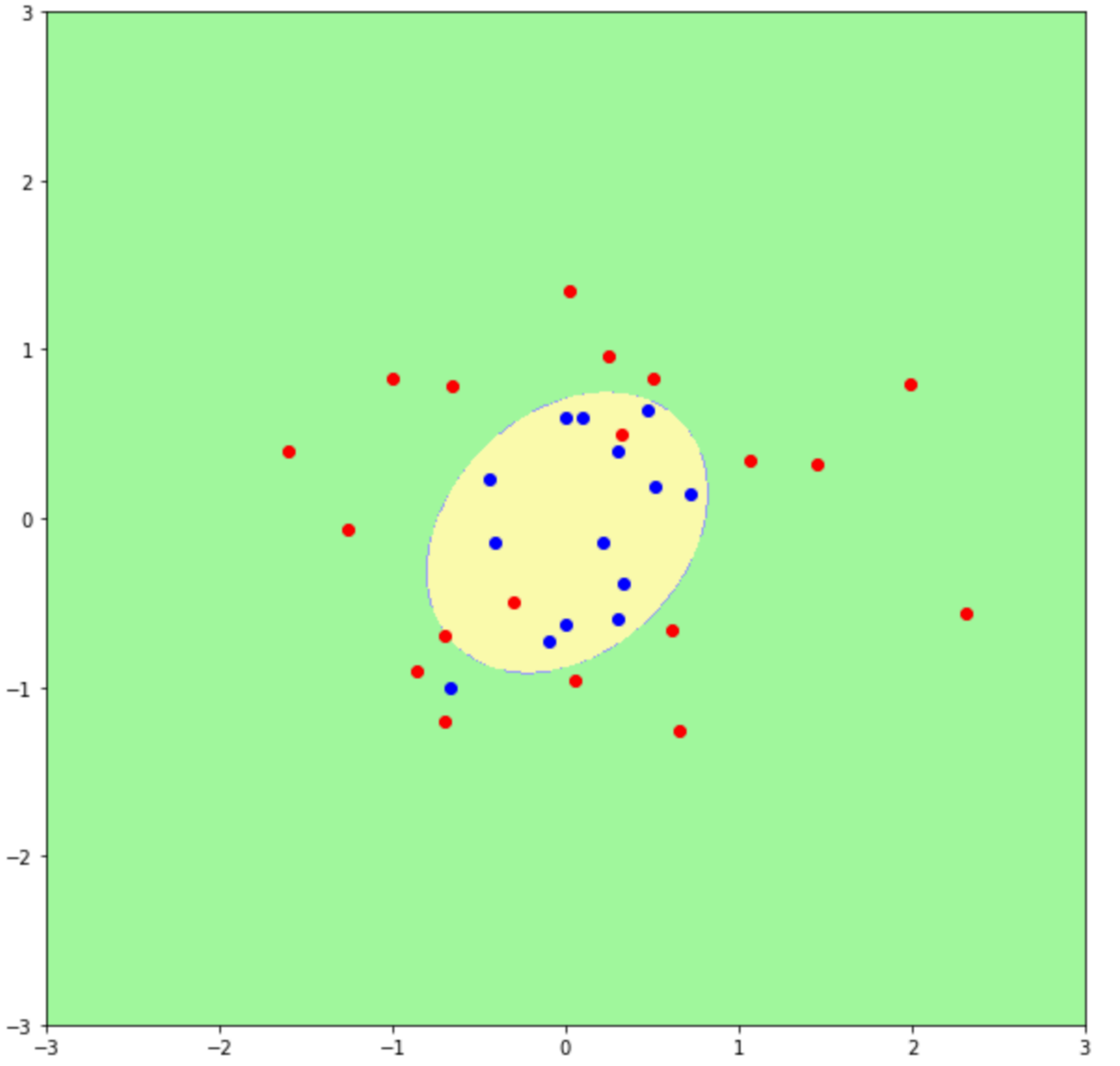
$$\Phi_4(\mathbf{x}) = \Phi_4([1, x_1, x_2]^T) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1 x_2 & x_2^2 & x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 & x_1^4 & x_1^3 x_2 & x_1^2 x_2^2 & x_1 x_2^3 & x_2^4 \end{bmatrix}^T$$

And so on

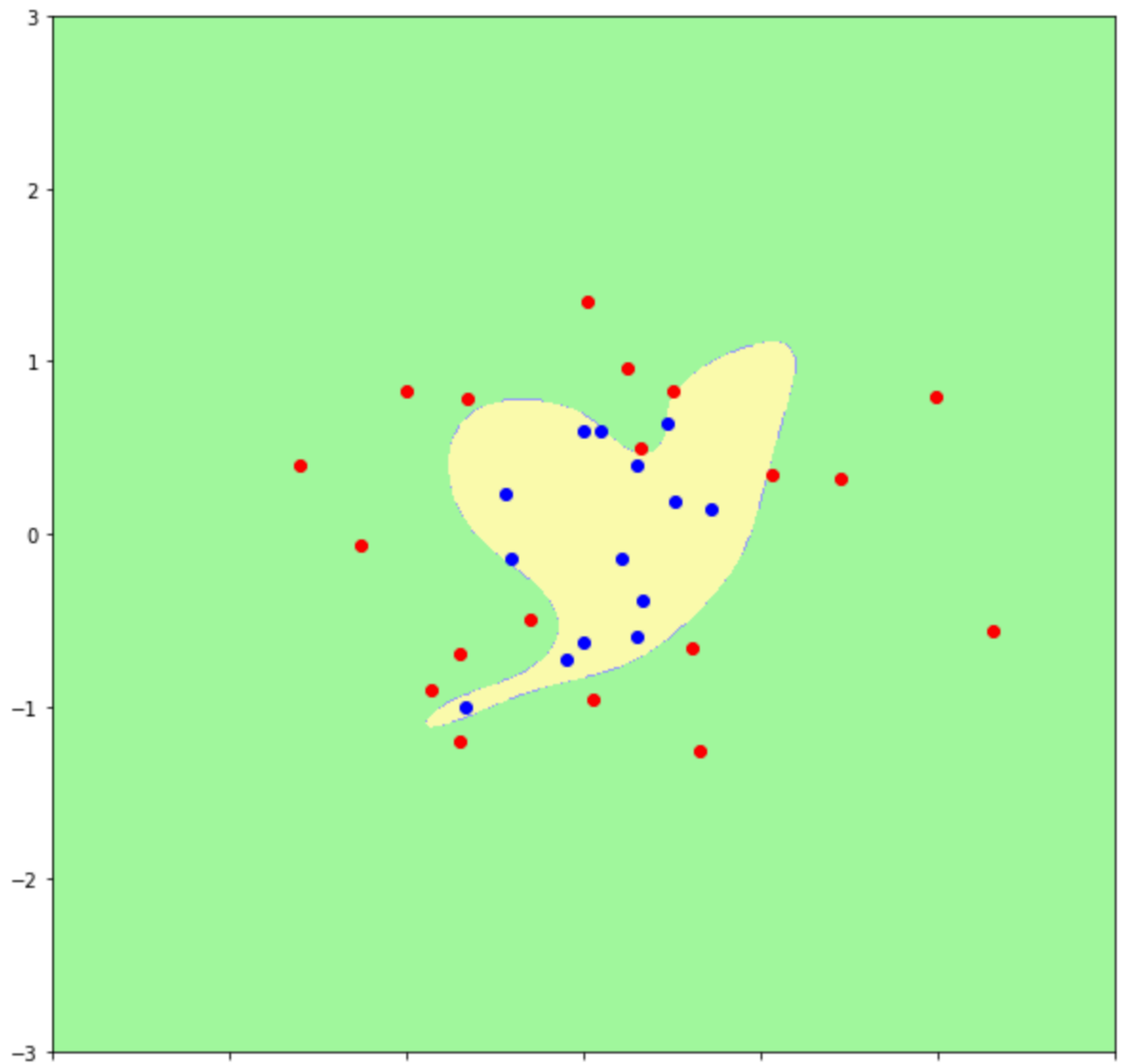
Outline

- 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
- Preventing overfitting
- Multiple classes — What if we have more than two classes?




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
```

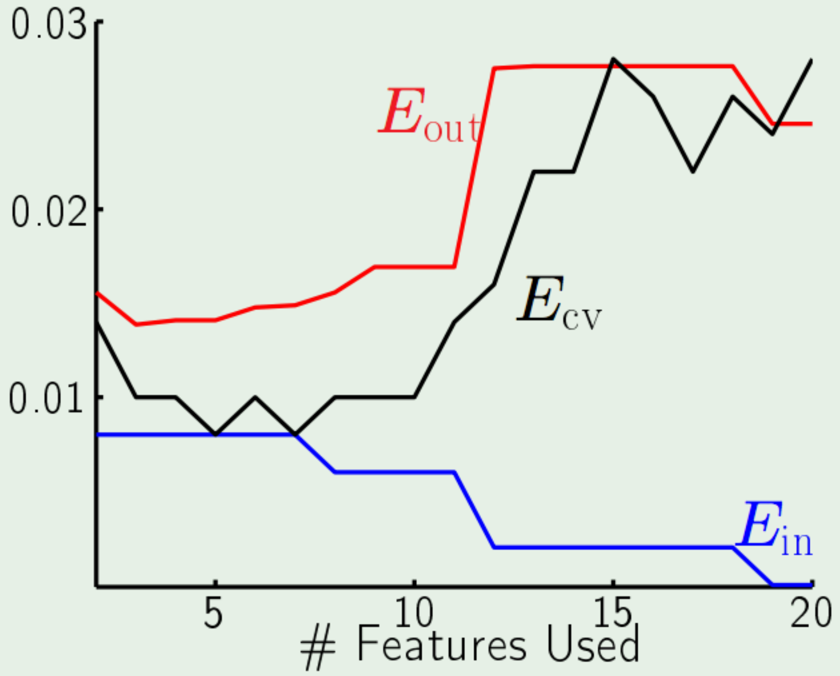


Too much flexibility in the model for quality and quantity of data → overfitting

What should we do to choose the right model?

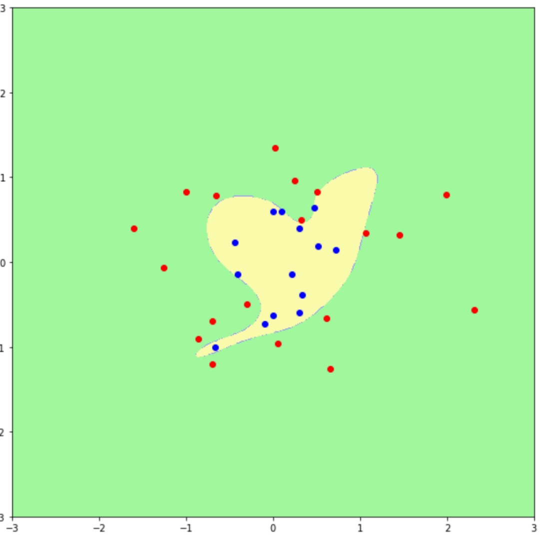
validation in action

Different errors



$$(1, x_1, x_2) \to (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, \ldots, x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5)$$

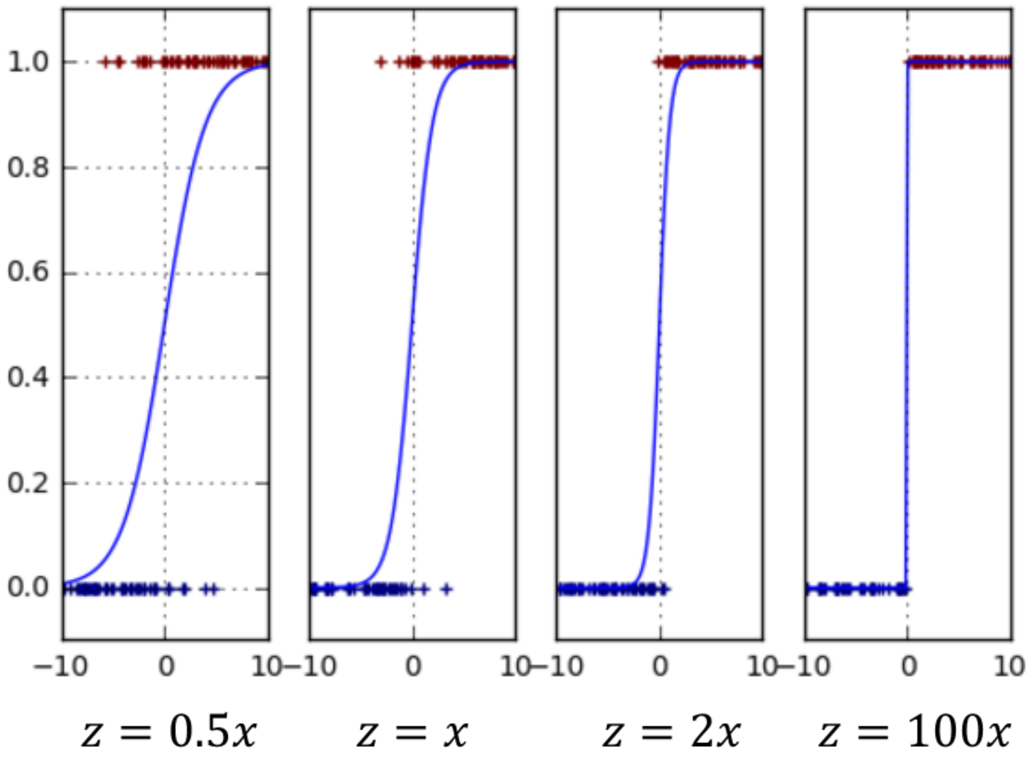
Slide modified from



How can we do 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]]
```

Logistic Model as a "Soft" Classifier



- Plot of

$$P(y = 1|x) = \frac{1}{1 + e^{-z}}, \qquad z = w_1 x$$

  - Markers are random samples
- Higher w_1: prob transition becomes sharper
  - Fewer samples occur across boundary
- As $w_1 \to \infty$ logistic becomes "hard" rule

$$P(y = 1|x) \approx \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

For linear regression:

- $E_{in}(\mathbf{w}) + \lambda(\|\mathbf{w}\|_2^2)$
- $E_{in}(\mathbf{w}) + \lambda(\|\mathbf{w}\|_1)$

Pair share:

How you would you add regularization to logistic regression? Use $\ell(w)$ to be the log likelihood function

Adding L2 regularization: $\underbrace{\frac{1}{N}\ell(\mathbf{w})}_{\text{Fit}} - \underbrace{\lambda(\|\mathbf{w}_{1:d}\|_2^2)}_{\text{Regularization term}}$

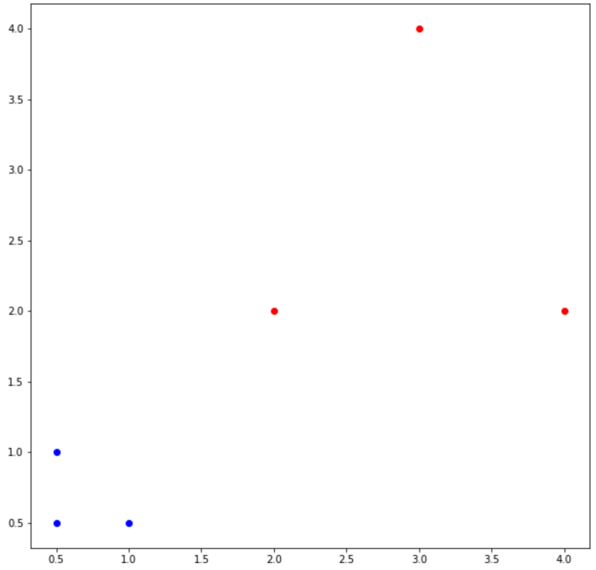
- The log likelihood function for logistic regression

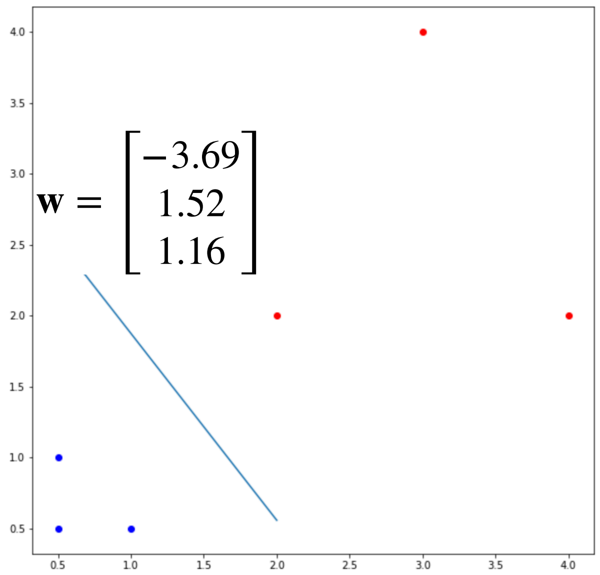
$$\ell_{\text{ridge}}(\mathbf{w}) = \frac{1}{N}\log L(\mathbf{w}) - \lambda(\|\mathbf{w}_{1:d}\|_2^2)$$

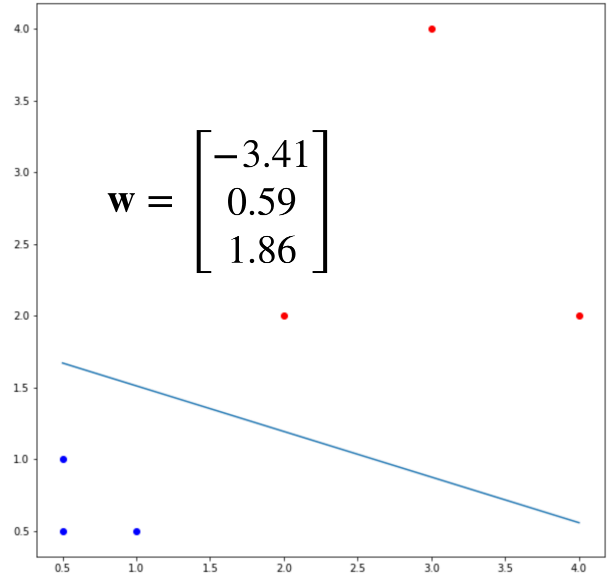
$$= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)}\log\sigma(\mathbf{w}^T\mathbf{x}^{(i)}) + (1-y^{(i)})\log(1-\sigma(\mathbf{w}^T\mathbf{x}^{(i)}))\right) - \lambda(\|\mathbf{w}_{1:d}\|_2^2)$$

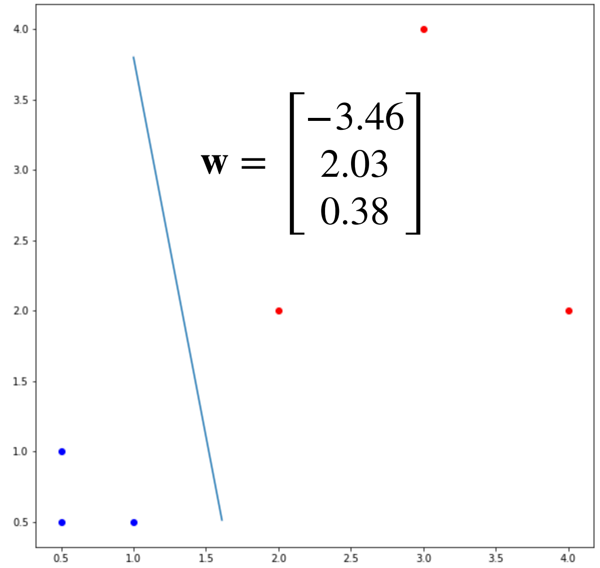
- Perform gradient ascent on this regularized function

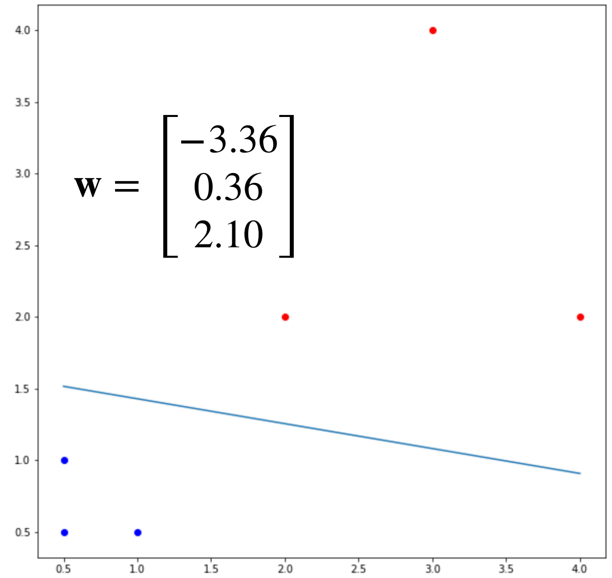
- Typically, we don't want to restrict the size of the intercept term

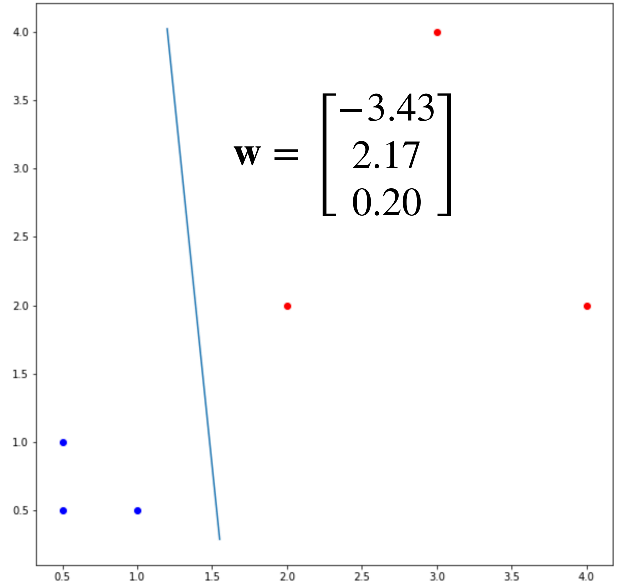












Pair share: What happens if I only regularize w_1?

Pair share: What happens if I only regularize w_2?

Outline

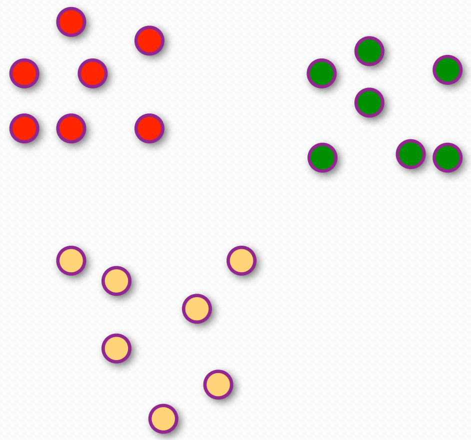
- Motivating example: How can we classify? — How can we use a hyperplane for a classification problem?
- Estimating probabilities — Can we predict not only which class an example belongs to - but a confidence score of that classification
- Maximum likelihood — How can we find the most likely hyperplane? 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
- Preventing overfitting
- Multiple classes — What if we have more than two classes?

- Multi-class: every example belongs to **<u>one</u>** class.
- Multi-label: every example can belong to **<u>one or more</u>** classes

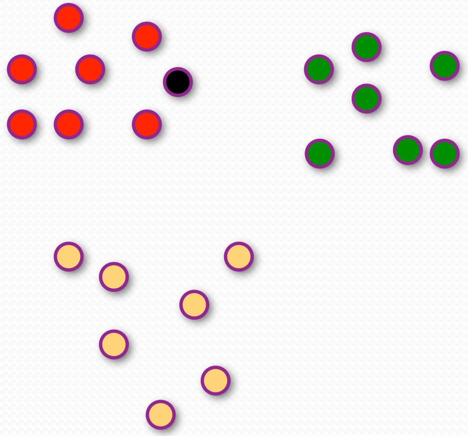
What if there is more than two classes?

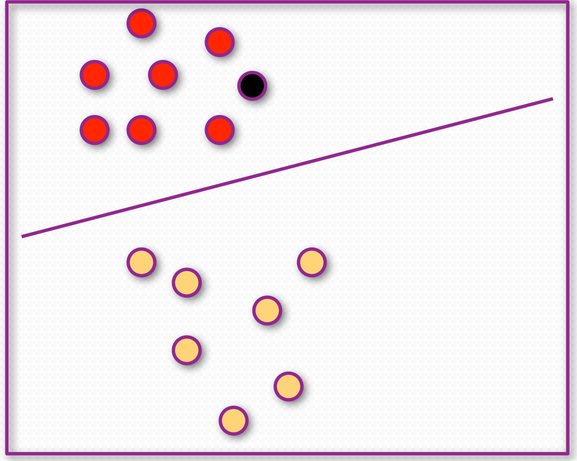
Multi-class classification

- Versicolor Iris vs Setosa Iris vs Veronica Iris
- 0 vs 1 vs 2 vs 3 … vs 9
- Dog vs Cat vs Mouse
- …



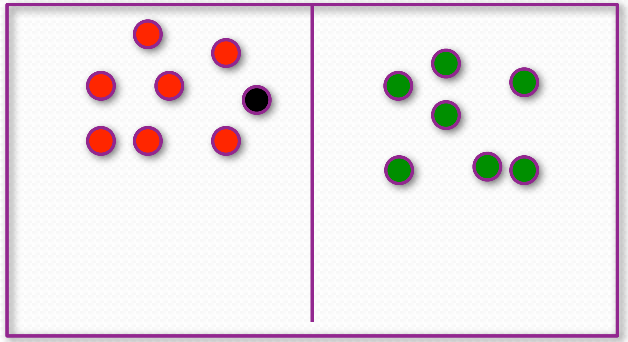
Turn multiple classes problem $C_1, C_2, \ldots, C_K$ into $\frac{K(K-1)}{2}$ binary classification problems





For each pair C_i, C_j of classes

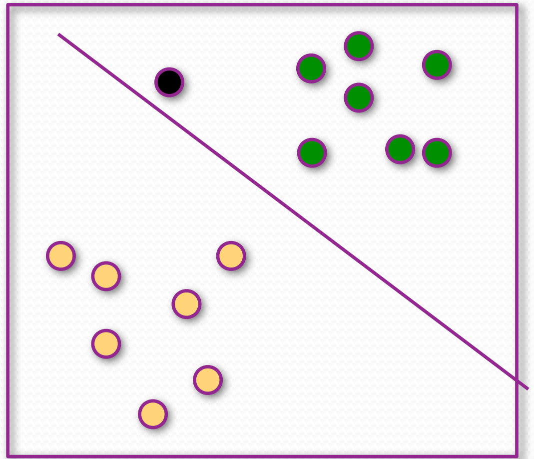
- Relabel training examples with label C_i into '1'
- Relabel training examples with label C_j into '0'
- Remove all other training examples



How do we predict given $\frac{K(K-1)}{2}$ binary classifiers? (i.e. K decision boundaries)

Step 1) Given a new example $\mathbf{x}$, predict the class that wins "majority of votes"

Step 2) If there is a tie, use confidence scores to resolve ties

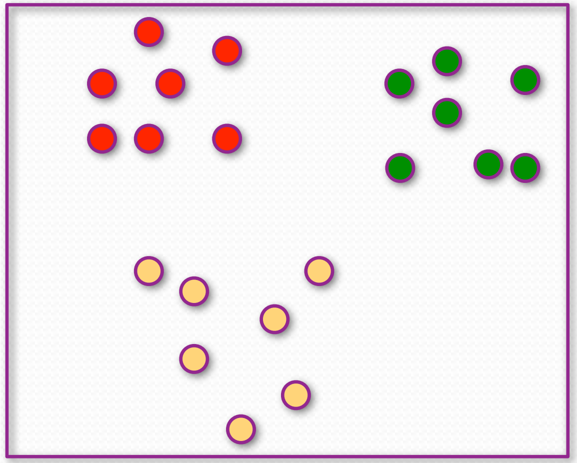


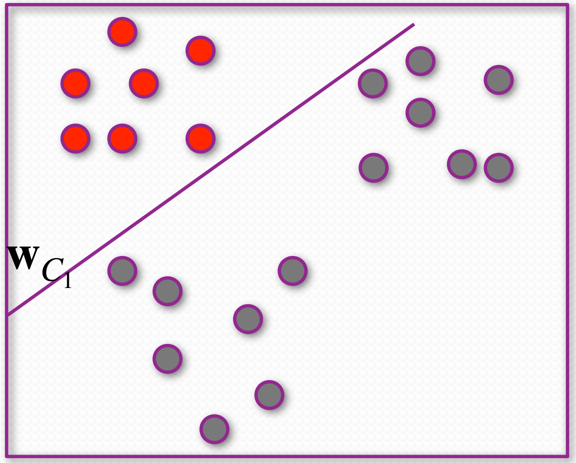
One-versus-Rest or One-Versus All Approach

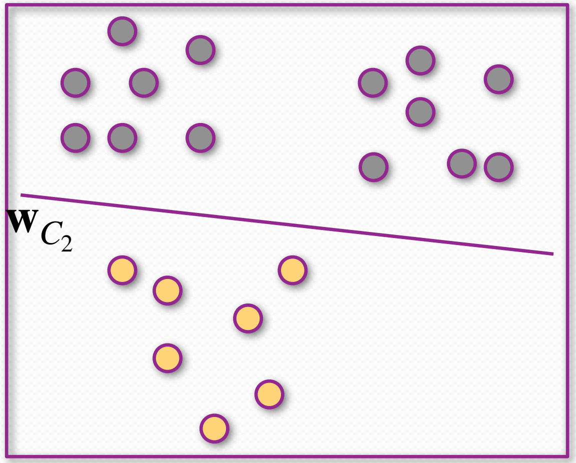
Turn multiple classes problem $C_1, C_2, \ldots, C_K$ into K binary classification problems

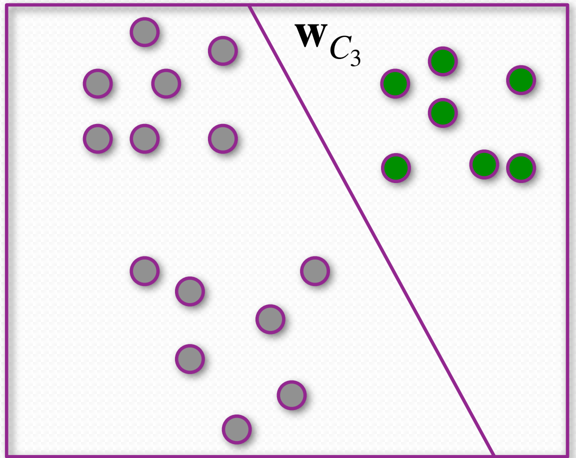
- Relabel training examples with label C_i into ‘1’
- Relabel all other training examples into ‘0’

Repeat K times using logistic regression to classify









How do we predict given K binary classifiers? (i.e. K decision boundaries)

Use confidence estimates $p(y = C_i \mid \mathbf{x})$ for $i = 1 \ldots K$

Predict C_i^* where

$$i = \arg \max_{i \in 1,\ldots,K} p(y = C_i \mid \mathbf{x}) = \sigma(\mathbf{w}_{C_i}^T \mathbf{x})$$

Works well in practice

Learning over local correctness - does not guarantee good global performance

One-versus-One Approach

K(K-1)/2 classifiers. Slow if K is large

Trained on a smaller subset of data which can result in variance

One-versus-Rest or One-Versus All Approach

K classifiers

In-balance in number of class 1 and class 0 training examples

For midterm 2, you will not be responsible for the material on the following slides

- Multi-class: every example belongs to **<u>one</u>** class.
- Multi-label: every example can belong to **<u>one or more</u>** classes

SOFTMAX- A GENERALIZED FORM OF LOGISTIC REGRESSION (*MULTI-CLASS*)

Called:
multiclass logistic regression, *softmax regression* and *multinomial regression*

Could our algorithm directly estimate the probability of label belonging to each of the K classes? (i.e. don't resort to a binary classification problem)

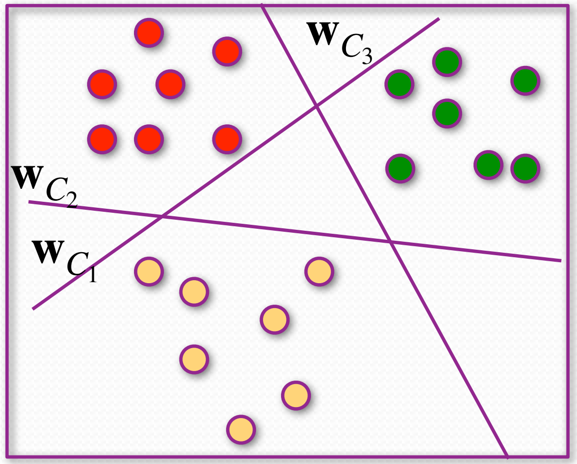
Computing $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$, ..., $\mathbf{w}_K$ separately doesn't work since $\sum_{i=1}^{K} p(y = C_i \mid \mathbf{x}) = \sum_{i=1}^{K} \sigma(\mathbf{w}_i^T \mathbf{x}) \neq 1$

Idea: Learn the probabilities jointly (to keep the sum =1)

Intuition

$$\begin{bmatrix} \mathbf{w}_{C_1}^T \mathbf{x} \\ \mathbf{w}_{C_2}^T \mathbf{x} \\ \mathbf{w}_{C_3}^T \mathbf{x} \end{bmatrix}$$

One vs. rest decision boundary for each class.



To create prob distribution:

- Nonnegative
- Sum to one

Pair share: How could we turn these numbers into probabilities that sum to 1?

Intuition

- Setosa - C_1
- Versicolor -C_2
- Veronica - C_3

If $\mathbf{w}_{C_1}^T\mathbf{x} = -18$, $\mathbf{w}_{C_2}^T\mathbf{x} = 12$, and $\mathbf{w}_{C_3}^T\mathbf{x} = 6$

We predict **Versicolor**

We could turn these numbers to probabilities using softmax (well-formed conditional probabilities)

$$p(y = C_1 \mid \mathbf{x}) = \frac{e^{-18}}{e^{-18} + e^{12} + e^{6}} < 1$$

$$p(y = C_2 \mid \mathbf{x}) = \frac{e^{12}}{e^{-18} + e^{12} + e^{6}} < 1$$

$$p(y = C_3 \mid \mathbf{x}) = \frac{e^{6}}{e^{-18} + e^{12} + e^{6}} < 1$$

Normalization term makes the probabilities sum to 1

Keeps relative ordering!

One hot encoding

Not all approaches do a one hot encoding (1-of-K encoding) of the target

$$h(\mathbf{x}) = \begin{bmatrix} p(y = C_1 \mid \mathbf{x}) \\ p(y = C_2 \mid \mathbf{x}) \\ p(y = C_3 \mid \mathbf{x}) \end{bmatrix}$$

We will predict K different probabilities: $y^{(i)}$ becomes $\mathbf{y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \ldots, y_K^{(i)}]^T$

Using one hot encoding (1-of-K encoding).

$$y_j^{(i)} = \begin{cases} 1 & y^{(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

If we have 3 classes:

class C_1 → $[1 \quad 0 \quad 0]^T$

class C_2 → $[0 \quad 1 \quad 0]^T$

class C_3 → $[0 \quad 0 \quad 1]^T$

Soft-max

W has dimension k x (d+1), we stacked the transpose of the coefficient vectors

$$W = \begin{bmatrix} -\mathbf{w}_1^T- \\ -\mathbf{w}_2^T- \\ -\mathbf{w}_3^T- \\ \cdots \\ -\mathbf{w}_K^T- \end{bmatrix}$$

$$\begin{bmatrix} -\mathbf{w}_1^T- \\ -\mathbf{w}_2^T- \\ -\mathbf{w}_3^T- \\ \cdots \\ -\mathbf{w}_K^T- \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_d \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^T\mathbf{x} \\ \mathbf{w}_2^T\mathbf{x} \\ \mathbf{w}_3^T\mathbf{x} \\ \cdots \\ \mathbf{w}_K^T\mathbf{x} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \cdots \\ z_K \end{bmatrix}$$

score for each coefficient Vector

$$h(\mathbf{x}) = \begin{bmatrix} p(y=1 \mid \mathbf{x}) \\ p(y=2 \mid \mathbf{x}) \\ p(y=3 \mid \mathbf{x}) \\ \cdots \\ p(y=K \mid \mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\mathbf{w}_j^T\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_1^T\mathbf{x}} \\ e^{\mathbf{w}_2^T\mathbf{x}} \\ \\ \\ e \end{bmatrix}$$

normalizing the score

Soft-max

W has dimension
k x (d+1), we stacked
the transpose of the
coefficient vectors

$$W = \begin{bmatrix} -\mathbf{w}_1^T- \\ -\mathbf{w}_2^T- \\ -\mathbf{w}_3^T- \\ \cdots \\ -\mathbf{w}_K^T- \end{bmatrix}$$

$$\begin{bmatrix} -\mathbf{w}_1^T- \\ -\mathbf{w}_2^T- \\ -\mathbf{w}_3^T- \\ \cdots \\ -\mathbf{w}_K^T- \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ x_d \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^T\mathbf{x} \\ \mathbf{w}_2^T\mathbf{x} \\ \mathbf{w}_3^T\mathbf{x} \\ \cdots \\ \mathbf{w}_K^T\mathbf{x} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \cdots \\ z_K \end{bmatrix}$$

score for
each
coefficient
Vector

$$h(\mathbf{x}) = \begin{bmatrix} p(y=1 \mid \mathbf{x}) \\ p(y=2 \mid \mathbf{x}) \\ p(y=3 \mid \mathbf{x}) \\ \cdots \\ p(y=K \mid \mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\mathbf{w}_j^T\mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_1^T\mathbf{x}} \\ e^{\mathbf{w}_2^T\mathbf{x}} \\ \\ \\ e \end{bmatrix}$$

normalizing
the score

You will not responsible for the material on this slide for midterm 2

We will predict K different probabilities: $y^{(i)}$ becomes $\mathbf{y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \ldots, y_K^{(i)}]^T$

Using 1-of-K encoding (one hot encoding).

$$y_j^{(i)} = \begin{cases} 1 & y^{(i)} = j \\ 0 & \textbf{otherwise} \end{cases}$$

Approach is maximize Log likelihood

Only one of these is 1

Maximize $\log L = \sum_{i=1}^{N} \log p(y^{(i)} \mid \mathbf{x}^{(i)}) = \sum_{i=1}^{N} \log \prod_{k=1}^{K} p(C_k^{(i)} \mid \mathbf{x}^{(i)})^{y_k^{(i)}} = \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^{(i)} \log p(C_k^{(i)} \mid \mathbf{x}^{(i)})$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^{(i)} \log \frac{e^{\mathbf{w}_k^T \mathbf{x}^{(i)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}$$

Negative Log likelihood for softmax is the cross entropy error

Concave thus parameters can be found using gradient ascent

You will not responsible for the material on this slide for midterm 2

Using the indicator function $I\{\cdot\}$.
$I\{\text{true statement}\} = 1$
$I\{\text{false statement}\} = 0$

$$J(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log p(y^{(i)} = k \mid \mathbf{x}^{(i)}; \mathbf{w})$$

$$J(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \log \frac{e^{\mathbf{w}_k^T\mathbf{x}^{(i)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}$$

$$J(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \left[\log e^{\mathbf{w}_k^T\mathbf{x}^{(i)}} - \log \sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right]$$

$$J(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \left[\mathbf{w}_k^T\mathbf{x}^{(i)} - \log \sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right]$$

A very complicated derivation creates

$$\nabla_{\mathbf{w}_k} J(\mathbf{w}) = \sum_{i=1}^{N} \mathbf{x}^{(i)} \left[1\{y^{(i)} = k\} - \frac{e^{\mathbf{w}_k^T\mathbf{x}^{(i)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}\right]$$

Randomly initialize w
For I = 1 to num_iters:
For each $\mathbf{w}_k$:
$\mathbf{w}_k = \mathbf{w}_k + \alpha \nabla_{\mathbf{w}_k} J(\mathbf{w})$

http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/

You will not responsible for the material on this slide for midterm 2

Here we are only updating the kth coefficient vector

Using the indicator function $I\{\cdot\}$.
$I\{\text{true statement}\} = 1$
$I\{\text{false statement}\} = 0$

$$\nabla_{\mathbf{w}_r} J(\mathbf{w}) = \sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y^{(i)} = k\}\, \nabla_{\mathbf{w}_r}\left[\mathbf{w}_k^T\mathbf{x}^{(i)} - \log\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right]$$

$$\nabla_{\mathbf{w}_r} J(\mathbf{w}) = \sum_{i=1}^{N} 1\{y^{(i)} = r\}\, \nabla_{\mathbf{w}_r}\left[\mathbf{w}_r^T\mathbf{x}^{(i)} - \log\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right] + \sum_{i=1}^{N}\sum_{k=1,k\neq r}^{K} 1\{y^{(i)} = k\}\, \nabla_{\mathbf{w}_r}\left[\mathbf{w}_k^T\mathbf{x}^{(i)} - \log\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right]$$

$$\nabla_{\mathbf{w}_r} J(\mathbf{w}) = \sum_{i=1}^{N} 1\{y^{(i)} = r\}\left[\mathbf{x}^{(i)} - \nabla_{\mathbf{w}_r}\log\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right] + \sum_{i=1}^{N}\sum_{k=1,k\neq r}^{K} 1\{y^{(i)} = k\}\left[-\nabla_{\mathbf{w}_r}\log\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}\right]$$

Only one term in the sum has $\mathbf{w}_k$

$$\nabla_{\mathbf{w}_r} J(\mathbf{w}) = \sum_{i=1}^{N} 1\{y^{(i)} = r\}\left[\mathbf{x}^{(i)} - \frac{e^{\mathbf{w}_r^T\mathbf{x}^{(i)}}\mathbf{x}^{(i)}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}\right] + \sum_{i=1}^{N}\sum_{k=1,k\neq r}^{K} 1\{y^{(i)} = k\}\left[-\frac{e^{\mathbf{w}_r^T\mathbf{x}^{(i)}}\mathbf{x}^{(i)}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}\right]$$

$$\nabla_{\mathbf{w}_r} J(\mathbf{w}) = \sum_{i=1}^{N} 1\{y^{(i)} = r\}\mathbf{x}^{(i)}\left[1 - \frac{e^{\mathbf{w}_r^T\mathbf{x}^{(i)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}\right] + \sum_{i=1}^{N}\sum_{k=1,k\neq r}^{K} 1\{y^{(i)} = k\}\mathbf{x}^{(i)}\left[-\frac{e^{\mathbf{w}_r^T\mathbf{x}^{(i)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}\right]$$

1 or 0

$$\nabla_{\mathbf{w}_r} J(\mathbf{w}) = \sum_{i=1}^{N} \mathbf{x}^{(i)}\left[1\{y^{(i)} = r\} - \frac{e^{\mathbf{w}_r^T\mathbf{x}^{(i)}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}^{(i)}}}\right]$$

How can we modify our existing model ?

Our existing model gave a probability of belonging to a class.

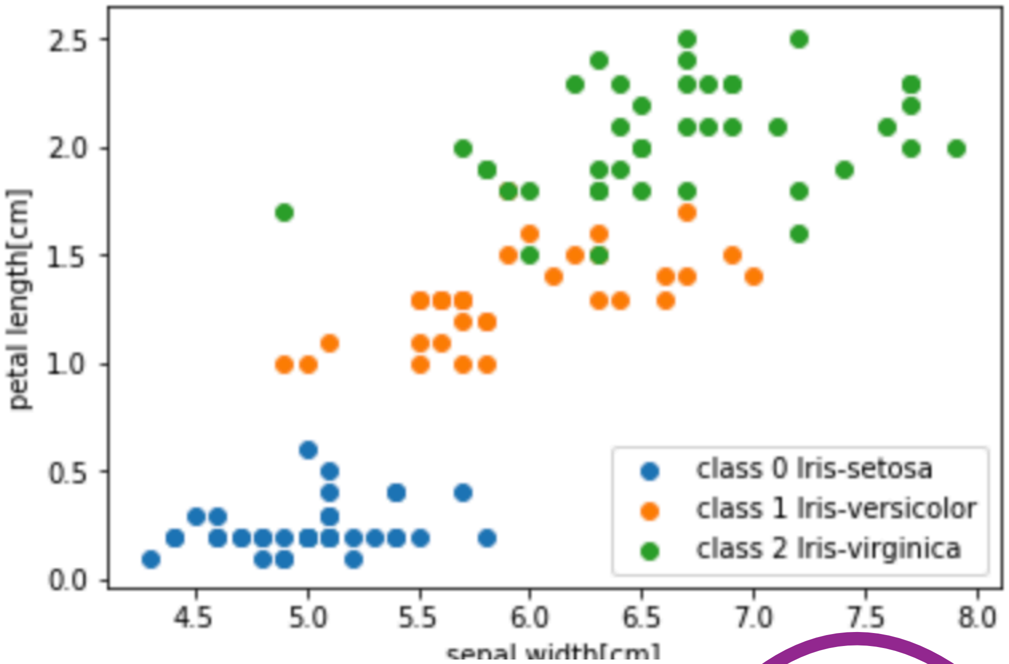
How can we provide an estimate of K classes? If K = 3 we can represent each class as follows:

$[1 \quad 0 \quad 0]^T$ $[0 \quad 1 \quad 0]^T$ $[0 \quad 0 \quad 1]^T$

class Setosa

class Versicolor

class Virginia



$$
X = \begin{bmatrix} 1 & 5.8 & 2.4 \\ 1 & 6 & 1 \\ 1 & 5.5 & 0.2 \\ 1 & 7.3 & 1.8 \\ 1 & 5 & 0.2 \\ 1 & 6.3 & 2.5 \\ 1 & 5 & 0.3 \\ 1 & 6.7 & 1.5 \\ 1 & 6.8 & 1.4 \\ 1 & 6.1 & 1.3 \end{bmatrix}
\qquad
y = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \\ 2 \\ 2 \end{bmatrix}
\rightarrow
\begin{bmatrix} [0,1,0] \\ [0,1,0] \\ [0,0,1] \\ [1,0,0] \\ [0,0,1] \\ [1,0,0] \\ [1,0,0] \\ [0,1,0] \\ [0,0,1] \\ [0,0,1] \end{bmatrix}
$$

Example

Suppose we are clarifying which iris out of 3 choices. If we have fit the model and thus have created **W**

$$[1 \quad 0 \quad 0]^T \qquad [0 \quad 1 \quad 0]^T \qquad [0 \quad 0 \quad 1]^T$$

class Setosa

class Versicolor

class Virginia

Code to create graphics adapted from Hands-On Machine Learning with Scikit-Learn & TensorFlow

$$(P(y = i \mid \mathbf{x})) = \left(\frac{e^{\mathbf{w}_i^T X_i}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T X_i}} \right)$$

W

41.09	-8.08	-18.02
2.09	1.16	3.79
-43.18	6.92	14.23

x

1
4
1.5

=

Z score

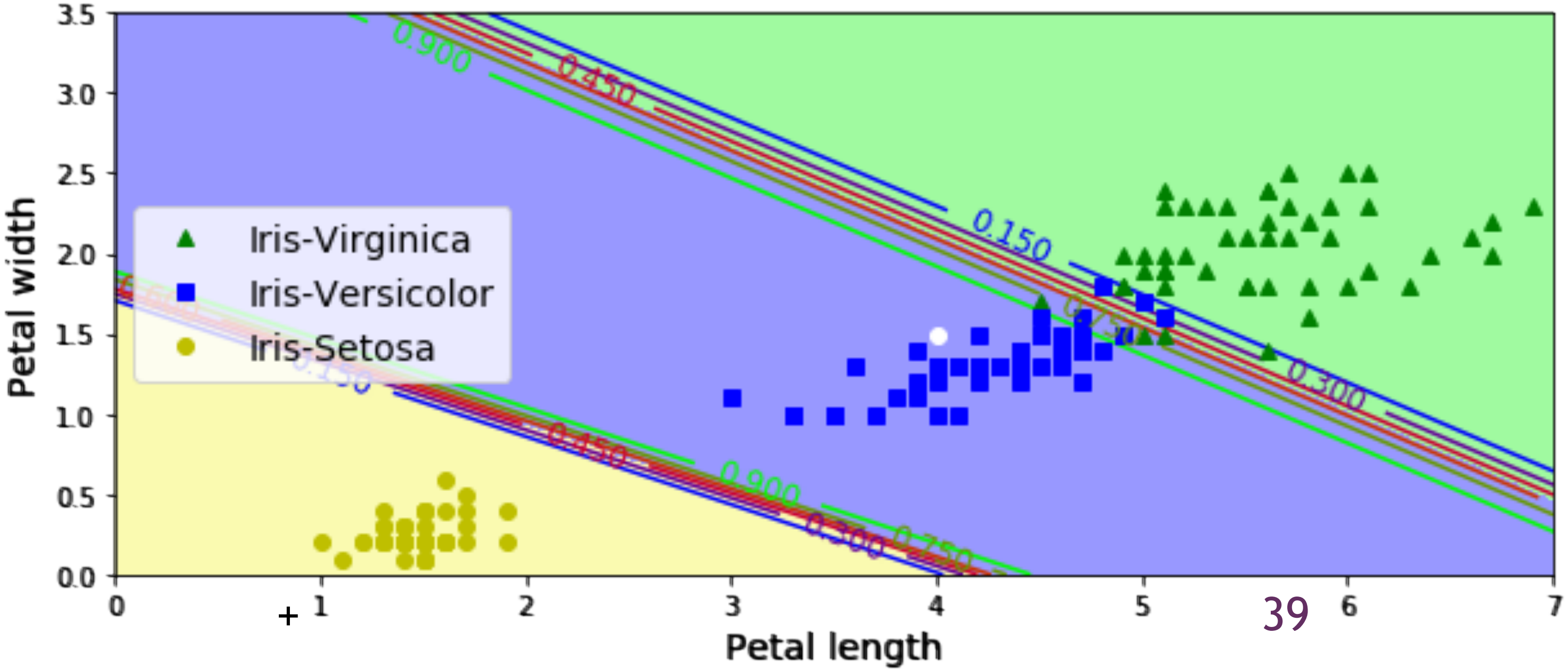
-18.28
12.43
5.84

e^{score}

1.15E-08
250404
346

normalize divide by 250750

4.60E-14
0.9986
0.00137



Example

Suppose we are clarifying which iris out of 3 choices. If we have fit the model and thus have created **W**

$$[1 \quad 0 \quad 0]^T \quad [0 \quad 1 \quad 0]^T \quad [0 \quad 0 \quad 1]^T$$

class Setosa

class Versicolor

class Virginia

$$(P(y=i \mid \mathbf{x})) = \left(\frac{e^{\mathbf{w}_i^T X_i}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T X_i}} \right)$$

W

41.09	-8.08	-18.02
2.09	1.16	3.79
-43.18	6.92	14.23

x

1
5.2
1.5

=

Z score

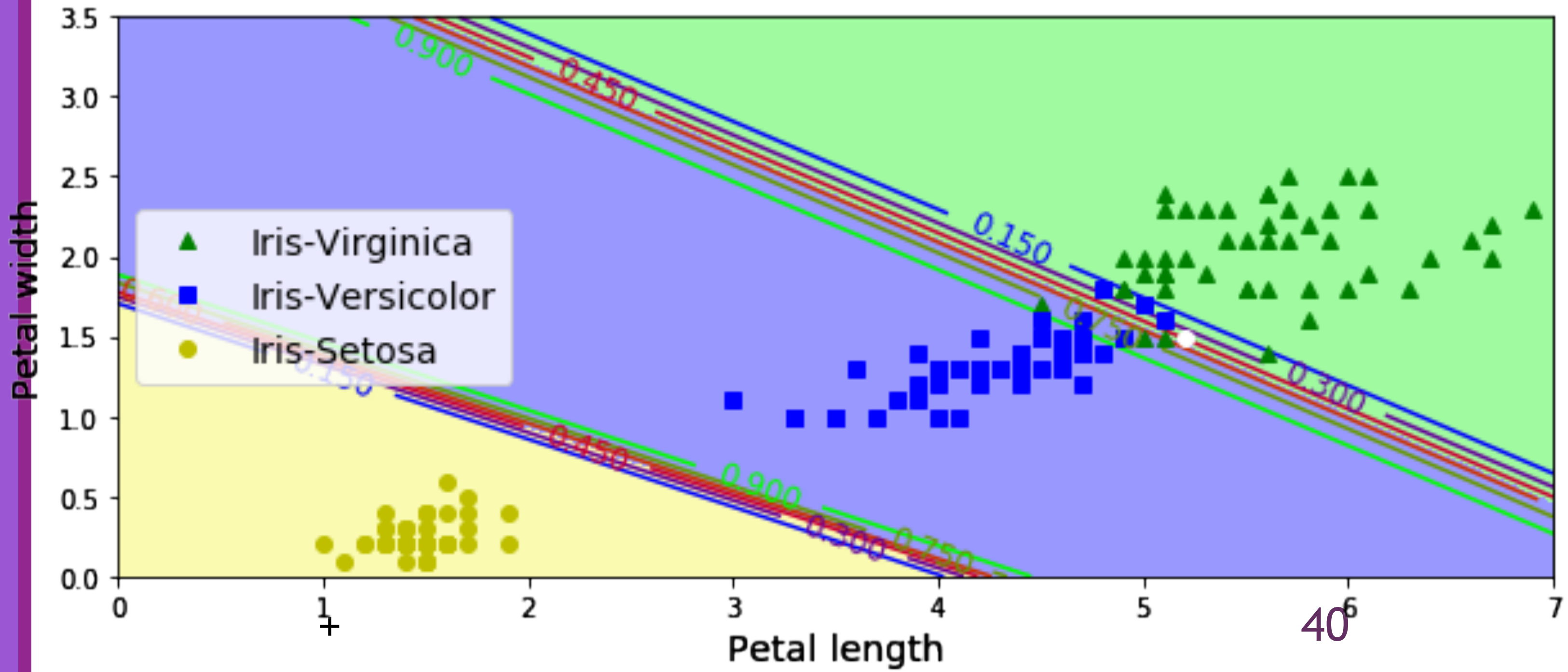
-27.98
13.83
14.15

e^{score}

7.06E-13
1013275.8
1397137.3

normalize divide by 2410413.2

2.93E-19
0.420
0.580



Example

Suppose we are clarifying which iris out of 3 choices. If we have fit the model and thus have created **W**

$$\left(P(y=i \mid \mathbf{x})\right) = \left(\frac{e^{\mathbf{w}_i^T X_i}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T X_i}}\right)$$

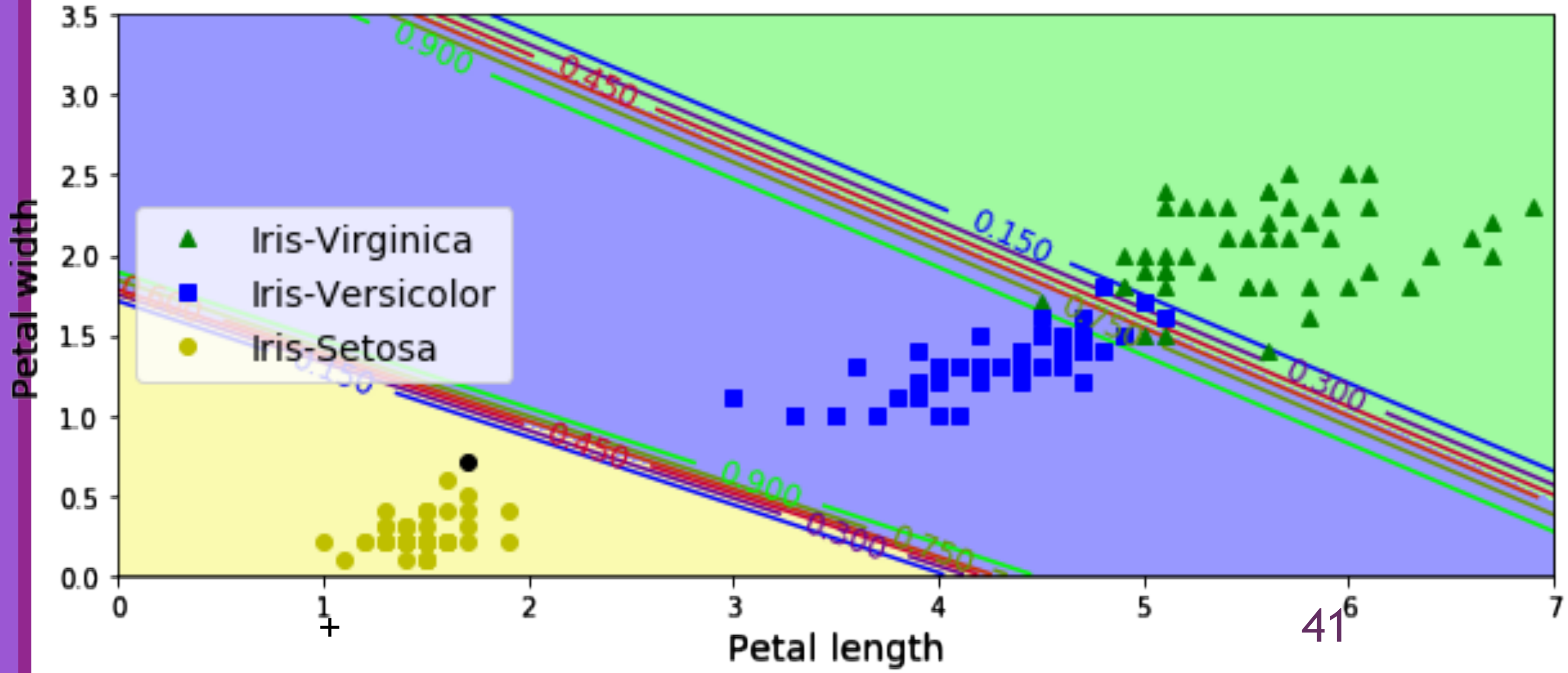
$$\underset{W}{\begin{bmatrix} 41.09 & -8.08 & -18.02 \\ 2.09 & 1.16 & 3.79 \\ -43.18 & 6.92 & 14.23 \end{bmatrix}} \underset{\mathbf{x}}{\begin{bmatrix} 1 \\ 1.7 \\ 0.7 \end{bmatrix}} = \underset{\text{Z score}}{\begin{bmatrix} 14.73 \\ 6.72 \\ -21.46 \end{bmatrix}} \rightarrow \underset{e^{\text{score}}}{\begin{bmatrix} 2501744.5 \\ 830.99 \\ 4.81E-10 \end{bmatrix}} \rightarrow \begin{bmatrix} 0.999679 \\ 0.000332 \\ 1.92E-16 \end{bmatrix}$$

normalize divide by 2502575.5

$[1 \quad 0 \quad 0]^T$ class Setosa

$[0 \quad 1 \quad 0]^T$ class Versicolor

$[0 \quad 0 \quad 1]^T$ class Virginia



You will not responsible for the material on this slide for midterm 2

If K=2 Softmax ➜ Logistic Regression

We can subtract any fixed K dimensional vector from each of the K coefficient vectors w_j

$$p(y_i = j \mid \mathbf{x}; \mathbf{w}) = \frac{e_i^{(\mathbf{w}_j - \theta)^T \mathbf{x}}}{\sum_{i=1}^{K} e^{(\mathbf{w}_i - \theta)^T \mathbf{x}}}$$

$$p(y_i = j \mid \mathbf{x}; \mathbf{w}) = \frac{e_i^{(\mathbf{w}_j)^T \mathbf{x}} e_i^{(-\theta)^T \mathbf{x}}}{\sum_{i=1}^{K} e_i^{(\mathbf{w}_i)^T \mathbf{x}} e_i^{(-\theta)^T \mathbf{x}}}$$

$$p(y_i = j \mid \mathbf{x}; \mathbf{w}) = \frac{e_i^{(\mathbf{w}_j)^T \mathbf{x}}}{\sum_{i=1}^{K} e^{(\mathbf{w}_i)^T \mathbf{x}}}$$

For y ∈ {1,2}

$$h(\mathbf{x}) = \begin{bmatrix} p(y=1 \mid \mathbf{x}) \\ p(y=2 \mid \mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{2} e^{\mathbf{w}_j^T \mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_1^T \mathbf{x}} \\ e^{\mathbf{w}_2^T \mathbf{x}} \end{bmatrix}$$

$$= \frac{1}{e^{\vec{\mathbf{0}}^T \mathbf{x}} + e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}} \begin{bmatrix} e^{\vec{0}^T \mathbf{x}} \\ e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}} \end{bmatrix}$$

$$= \frac{1}{1 + e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}} \begin{bmatrix} 1 \\ e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{1}{1 + e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}} \\ \dfrac{e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}}{1 + e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1 + e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}} \\ 1 - \dfrac{1}{1 + e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1 + e^{(\mathbf{w}')^T \mathbf{x}}} \\ 1 - \dfrac{1}{1 + e^{(\mathbf{w}')^T \mathbf{x}}} \end{bmatrix}$$

You will not responsible for the material on this slide for midterm 2

If K=2 Softmax → Logistic Regression

For y ∈ {1,2}

$$h(\mathbf{x}) = \begin{bmatrix} p(y=1 \mid \mathbf{x}) \\ p(y=2 \mid \mathbf{x}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{2} e^{\mathbf{w}_j^T \mathbf{x}}} \begin{bmatrix} e^{\mathbf{w}_1^T \mathbf{x}} \\ e^{\mathbf{w}_2^T \mathbf{x}} \end{bmatrix}$$

$$\frac{1}{+ e^{(\mathbf{w_2} - \mathbf{w_1})^T \mathbf{x}}} \begin{bmatrix} e^{\vec{0}^T \mathbf{x}} \\ e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}} \end{bmatrix}$$

$$\frac{1}{+ e^{(\mathbf{w_2} - \mathbf{w_1})^T \mathbf{x}}} \begin{bmatrix} 1 \\ e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{1}{1 + e^{(\mathbf{w_2} - \mathbf{w_1})^T \mathbf{x}}} \\ \dfrac{e^{(\mathbf{w}_2 - \mathbf{w}_1)^T \mathbf{x}}}{1 + e^{(\mathbf{w_2} - \mathbf{w_1})^T \mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1 + e^{(\mathbf{w_2} - \mathbf{w_1})^T \mathbf{x}}} \\ 1 - \dfrac{1}{1 + e^{(\mathbf{w_2} - \mathbf{w_1})^T \mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1 + e^{(\mathbf{w}')^T \mathbf{x}}} \\ 1 - \dfrac{1}{1 + e^{(\mathbf{w}')^T \mathbf{x}}} \end{bmatrix}$$

We can subtract any fixed K dimensional vector from each of the K coefficient vectors w_j

$$p(y_i = j \mid \mathbf{x}; \mathbf{w}) = \frac{e_i^{(\mathbf{w}_j - \theta)^T \mathbf{x}}}{\sum_{i=1}^{K} e^{(\mathbf{w}_i - \theta)^T \mathbf{x}}}$$

$$p(y_i = j \mid \mathbf{x}; \mathbf{w}) = \frac{e_i^{(\mathbf{w}_j)^T \mathbf{x}} e_i^{(-\theta)^T \mathbf{x}}}{\sum_{i=1}^{K} e_i^{(\mathbf{w}_i)^T \mathbf{x}} e_i^{(-\theta)^T \mathbf{x}}}$$

$$p(y_i = j \mid \mathbf{x}; \mathbf{w}) = \frac{e_i^{(\mathbf{w}_j)^T \mathbf{x}}}{\sum_{i=1}^{K} e^{(\mathbf{w}_i)^T \mathbf{x}}}$$

over-parameterized
there are multiple parameters that give the same answer

Using Sklearn's Logistic Regression

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(solver='sag',multi_class='multinomial')

clf.fit(X_train, y_train)

yhat_test = clf.predict(X_test)

score = clf.score(X_test, y_test)
```

make an instance of the model
Parameters set for this instance:
multi_class = 'multinomial'

Use the coefficient vector to predict

compute the accuracy

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Using Sklearn’s Logistic Regression

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(solver=‘sag',multi_class='multinomial')

clf.fit(X_train, y_train)

yhat_test = clf.predict(X_test)

score = clf.score(X_test, y_test)
```

import the model
in Sklearn this will
implemented as a class

make an instance of the model
Parameters set for this instance:
multi_class = ‘multinomial'

Train the model. (i.e. run the
algorithm using X_train and y_train
to create the coefficient vector

Use the coefficient vector to predict

compute the accuracy

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)#default is 1/4 test

scaler = StandardScaler()
X_train= scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
X_train = [[ 7.   6. 10.   5.   3. 10.   9. 10.   2.]
           [ 8.   3.   8.   3.   4.   9.   8.   9.   8.]
           [ 8. 10. 10. 10.   6. 10. 10. 10.   1.]
           [ 1.   1.   1.   1.   2.   1.   1.   1.   1.]]
```

y_train = [1 1 1 0]

```
X_test = [[ 1.   1.   1.   1.   2.   5.   1.   1.   1.]
          [ 3.   1.   1.   1.   2.   1.   2.   1.   1.]
          [ 5.   5.   5.   2.   5. 10.   4.   3.   1.]
          [ 4.   7.   8.   3.   4. 10.   9.   1.   1.]]
```

```
y_test = [0 0 1 1]
```

```
[[ 0.92  0.94  2.31  0.77 -0.1   1.81  2.23  2.27  0.25]
 [ 1.28 -0.04  1.63  0.07  0.34  1.53  1.82  1.95  3.75]
 [ 1.28  2.25  2.31  2.53  1.23  1.81  2.63  2.27 -0.34]
 [-1.22 -0.7  -0.74 -0.64 -0.55 -0.68 -0.99 -0.62 -0.34]]
```

```
[[-1.22 -0.7  -0.74 -0.64 -0.55  0.43 -0.99 -0.62 -0.34]
 [-0.51 -0.7  -0.74 -0.64 -0.55 -0.68 -0.59 -0.62 -0.34]
 [ 0.21  0.61  0.62 -0.28  0.78  1.81  0.22  0.02 -0.34]
 [-0.15  1.27  1.63  0.07  0.34  1.81  2.23 -0.62 -0.34]]
```

Next, we create a logistic regression object. By default, the logistic regresssion object will use $L2$ regularization. Here we have selected not to have regularization by using *penalty*='*none*'. If regularization is used, the parameter `C` states the level of regularization. Higher values of `C` have `less` regularization. We can chose to set explicity the regularization, or have the optimal value determined for us.

Note that $\lambda = 1/C$.

```
logreg = linear_model.LogisticRegression(penalty='none')
```

Penalty choices: L1,L2, L1 and L2.
Default is L2

Then, we fit the model.

```
logreg.fit(X_train, y_train)
```

```
LogisticRegression(penalty='none')
```

	feature	slope
0	thick	1.194634
1	size_unif	0.260756
2	shape_unif	0.839409
3	marg	0.615559
4	cell_size	0.592045
5	bare	1.601667
6	chrom	0.843135
7	normal	0.740983
8	mit	0.340539

```
print(logreg.intercept_)
print(logreg.coef_)
```

```
yhat = logreg.predict(X_test)# the predict method will return 0 or 1
```

```
[0 0 1 1]
```

```
yhat_proba = logreg.predict_proba(X_test)
```

```
[[0.99 0.01]
 [1.   0.  ]
 [0.06 0.94]
 [0.01 0.99]]
```

score = clf.score(X_test, y_test)

"Return the mean accuracy"

```
0.947368
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(yhat, y_test)
```

"Confusion matrix whose i-th row and j-th column entry indicates the number of samples with true label being i-th class and predicted label being j-th class."

```
logreg = linear_model.LogisticRegression(solver='liblinear', penalty='l1', C=0.01)
```

```
logreg = linear_model.LogisticRegression( penalty='l2', C=0.08)
```

	feature	slope
0	thick	1.194634
1	size_unif	0.260756
2	shape_unif	0.839409
3	marg	0.615559
4	cell_size	0.592045
5	bare	1.601667
6	chrom	0.843135
7	normal	0.740983
8	mit	0.340539

```
[0 0 1 1]

[[0.99 0.01]
 [1.   0.  ]
 [0.06 0.94]
 [0.01 0.99]]
```

	feature	slope
0	thick	0.000000
1	size_unif	0.436027
2	shape_unif	0.191333
3	marg	0.000000
4	cell_size	0.000000
5	bare	0.484421
6	chrom	0.000000
7	normal	0.000000
8	mit	0.000000

```
[0 0 1 1]

[[0.56 0.44]
 [0.68 0.32]
 [0.22 0.78]
 [0.15 0.85]]
```

	feature	slope
0	thick	0.622906
1	size_unif	0.500051
2	shape_unif	0.560409
3	marg	0.431736
4	cell_size	0.453271
5	bare	0.892948
6	chrom	0.550934
7	normal	0.503239
8	mit	0.226096

```
[0 0 1 1]

[[0.97 0.03]
 [0.98 0.02]
 [0.17 0.83]
 [0.05 0.95]]
```