

Day 4: Linear Classifiers

Summer STEM: Machine Learning

Department of Electrical and Computer Engineering
NYU Tandon School of Engineering
Brooklyn, New York

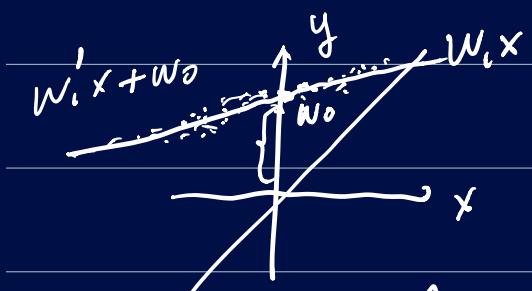
June 25, 2020

Linear Regression

- scalar-valued features x_i

$$\hat{y}_i = w_i x_i + w_0$$

w_0 : intercept / bias



$$\text{Loss} : \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

- matrix-vector form

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} 1 & | & | & | & x_1 \\ | & | & | & | & x_2 \\ | & | & | & | & \vdots \\ | & | & | & | & x_N \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \mathbf{x} \mathbf{w}$$

$$= \begin{bmatrix} w_0 + w_1 x_1 \\ \vdots \\ w_0 + w_1 x_N \end{bmatrix}$$

Alternatively, we can write

$$\hat{\mathbf{y}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} [w_1] + \begin{bmatrix} w_0 \\ w_0 \\ \vdots \\ w_0 \end{bmatrix} \left\{ \begin{array}{l} \text{intercept} \\ \text{vector} \end{array} \right\}$$

. fit(X, y)
 ↑ design matrix X

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix} = Xw$$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$\| Y - \hat{Y} \|^2 = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$= \sum_{i=1}^N (y_i - (w_1 x_i + w_0))^2$$

• Matrix multiplication

$$(AB)_{ij} = \sum_{k=1} A_{ik} B_{kj}$$

The element of the product
at the i -th row and the j -th
column is the inner product
of the i -th row of A and
 j -th column of B.

- Inner product (dot product)
of vectors.

Given two vectors u, v of the
same size D

$$u = (u_1, u_2, \dots, u_D)$$

$$v = (v_1, v_2, \dots, v_D)$$

$$\begin{aligned} u \cdot v &= u_1 v_1 + u_2 v_2 + \dots + u_D v_D \\ &= \sum_{j=1}^D u_j v_j \end{aligned}$$

$$\text{Ex: } u = (1, 2, 3)$$

$$v = (4, 5, 6)$$

$$\begin{aligned} u \cdot v &= 1 \times 4 + 2 \times 5 + 3 \times 6 \\ &= 4 + 10 + 18 \\ &= \underline{32} \quad \text{a scalar} \end{aligned}$$

Ex. matrix - multiplication

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 4 \\ 5 & 7 \end{bmatrix} \quad B = \begin{bmatrix} 1 & -2 \\ 0 & 3 \end{bmatrix}$$

$$(AB)_{11} = 3$$

$$AB = \begin{bmatrix} (AB)_{11} \\ (AB)_{21} \\ (AB)_{31} \end{bmatrix} = \begin{bmatrix} (AB)_{11} \\ (AB)_{21} \\ (AB)_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

$$\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix}$$

$$\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \left\| \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_N - \hat{y}_N \end{bmatrix} \right\|^2$$

$$= \sqrt{(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \dots + (y_N - \hat{y}_N)^2}$$

$$= (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \dots \\ + (y_N - \hat{y}_N)^2$$

$$= \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = w_0 + w_1 x_i$$

$$\cdot \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\cdot \frac{1}{N} \| y - \hat{y} \|^2$$

$$\text{where } \hat{Y} = Xw$$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots \\ 1 & x_N \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

design matrix
(feature matrix)

- Vector-valued features

$$x_i = \begin{bmatrix} x_{i1} \\ \tilde{x}_{i2} \\ x_{i2} \end{bmatrix} \quad \text{both of } x_{i1} \text{ and } x_{i2} \text{ are scalars}$$

$$\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2}$$

Design matrix in this case

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & & \\ 1 & x_{N1} & x_{N2} \end{bmatrix}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \mathbf{X} w$$

↑ design

matrix

for x_i of size D

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{bmatrix}$$

the design matrix X is

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1D} \\ 1 & x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & & & & \\ 1 & x_{N1} & x_{N2} & \dots & x_{ND} \end{bmatrix}$$

$$\hat{y} = Xw \quad w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{bmatrix}$$

$$\text{Loss} : \frac{1}{N} \| y - \hat{y} \|^2$$

$$= \frac{1}{N} \| y - Xw \|^2$$

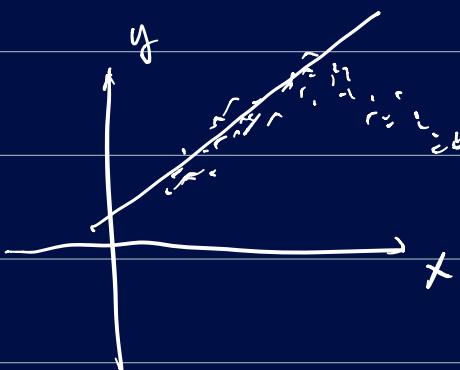
Our task is to find optimal parameters w^* such that

$$\text{the loss } \frac{1}{N} \|Y - Xw^*\|^2$$

is minimized (has the least value for all possible w)

For linear regression with MSE loss

$$w^* = (X^T X)^{-1} X^T Y$$



$\hat{y} = w_0 + w_1 x$
is not appropriate
to fit this data set

Instead, use polynomials

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M$$

• P-norm np.linalg.norm(u, ord=2)

Given a vector $u = \begin{bmatrix} u_1 \\ \vdots \\ u_D \end{bmatrix}$

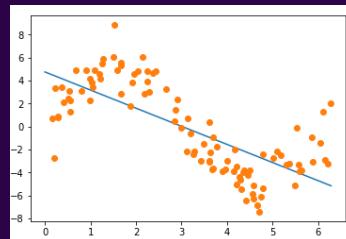
$$\|u\| = \sqrt{|u_1|^2 + |u_2|^2 + \dots + |u_D|^2} \quad (\text{L2-norm})$$

$$\begin{aligned} \|u\|_P &= \left(|u_1|^P + |u_2|^P + \dots + |u_D|^P \right)^{\frac{1}{P}} \\ &= \left(\sum_{j=1}^D |u_j|^P \right)^{\frac{1}{P}} \quad \left(2^{\frac{1}{2}} = \sqrt{2} \quad 2^{\frac{1}{3}} = \sqrt[3]{2} \right) \end{aligned}$$

$$\begin{aligned} \text{L1 norm} &\quad |u_1| + |u_2| + \dots + |u_D| \\ \text{(L0 norm} &\quad \max_j |u_j| \end{aligned}$$

Polynomial Fitting

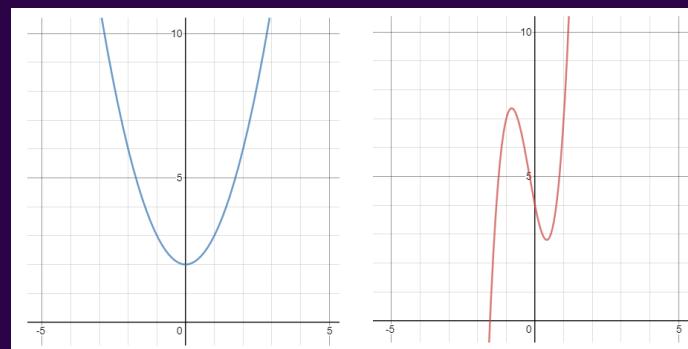
- We have been using straight lines to fit our data. But it doesn't work well every time
 - Some data have more complex relation that cannot be fitted well using a straight line



- Can we use some other model to fit this data?

Polynomial Fitting

- Can we use a polynomial to fit our data?
- Polynomial: A sum of different powers of a variable
 - Examples: $y = x^2 + 2$, $y = 5x^3 - 3x^2 + 4$



Review
oooooooooooo

Regularization
oooooo

Opt
oooooooo

Logistic
ooooooooooooooo

Demo
oo

Multiclass
oooo

Lab
o

Polynomial Fitting

- Polynomials of x : $\hat{y} = w_0 + w_1x + w_2x^2 + w_3x^3 + \dots + w_Mx^M$
- M is called the order of the polynomial.
- The process of fitting a polynomial is similar to linearly fitting multivariate data.

Polynomial fitting

- Rewrite in matrix-vector form

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \vdots & \ddots & \ddots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$

- This can still be written as

$$Y \approx X\mathbf{w}$$

- Loss $J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2$
- The i -th row of the design matrix X is simply a transformed feature $\phi(x_i) = (1, x_i, x_i^2, \dots, x_i^M)$

Polynomial Fitting

■ Original design matrix:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

■ Design matrix after feature transformation:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^M \\ 1 & x_2 & x_2^2 & \cdots & x_2^M \\ \vdots & \ddots & \vdots & & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix}$$

■ For the polynomial fitting, we just added columns of features that are powers of the original feature

Review
oooooooo●ooooo

Regularization
ooooo

Opt
ooooooo

Logistic
oooooooooooooo

Demo
oo

Multiclass
oooo

Lab
o

Linear Regression

- Model $\hat{y} = \mathbf{w}^T \phi(\mathbf{x})$
- Loss $J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2$
- Find \mathbf{w} that minimizes $J(\mathbf{w})$

Review
oooooooo●oooo

Regularization
ooooo

Opt
ooooooo

Logistic
oooooooooooooo

Demo
oo

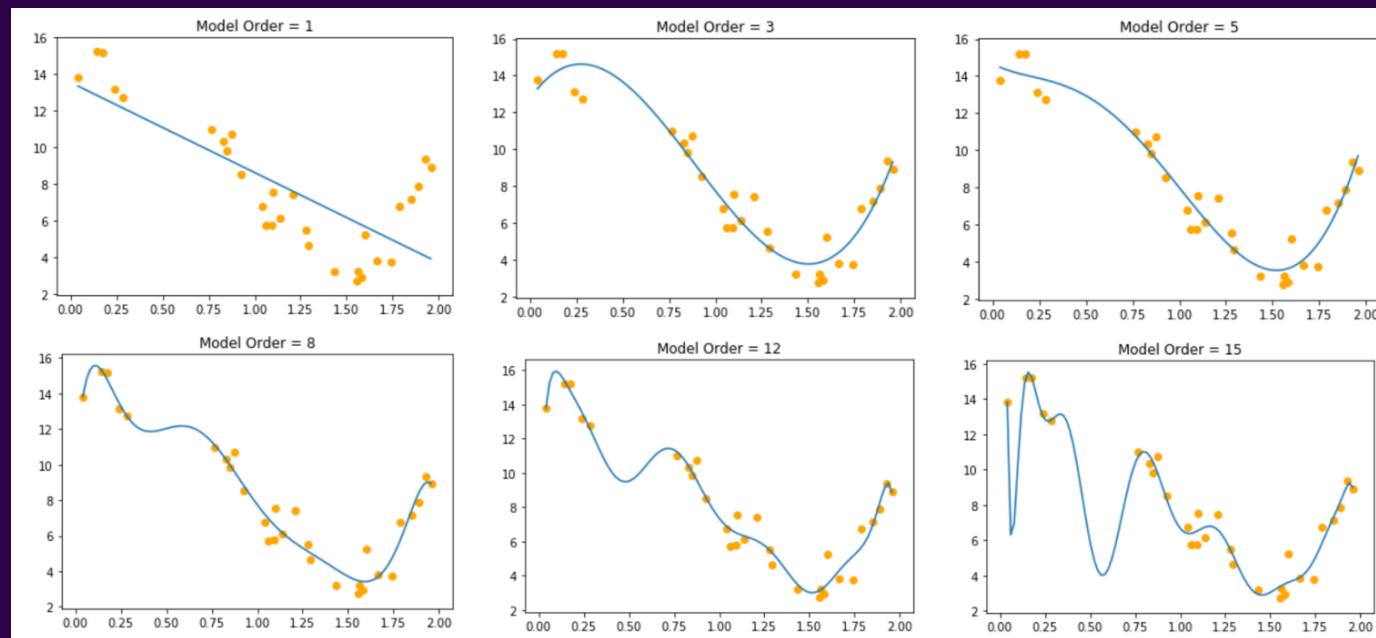
Multiclass
oooo

Lab
o

Overfitting

- We learned how to fit our data using polynomials of different order
- With a higher model order, we can fit the data with increasing accuracy
- As you increase the model order, at certain point it is possible find a model that fits your data perfectly (ie. zero error)
- What could be the problem?

Overfitting

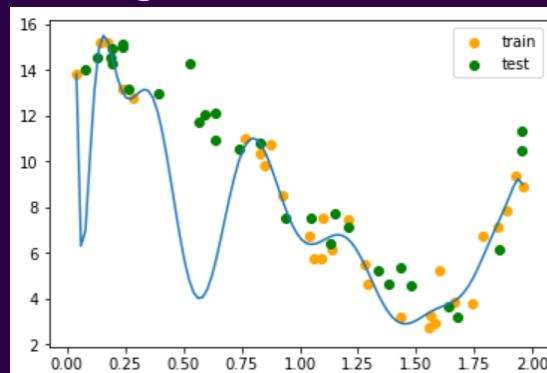


- Which of these model do you think is the best? Why?



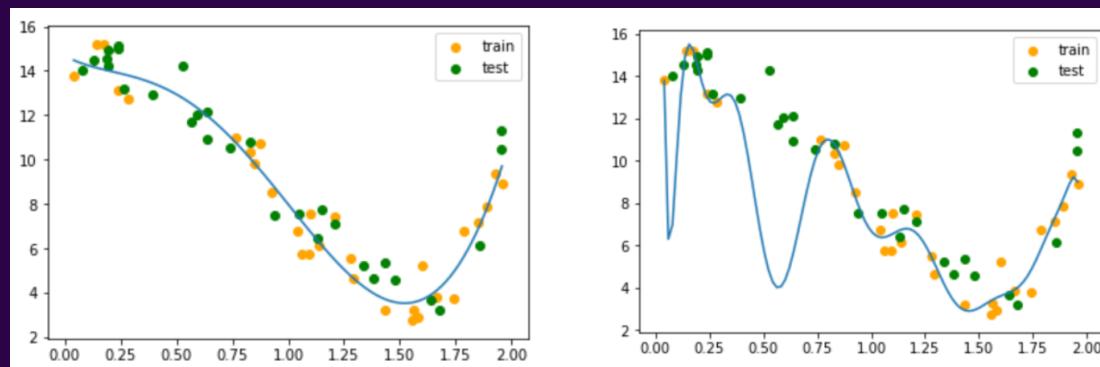
Overfitting

- The problem is that we are only fitting our model using data that is given
- Data usually contains noise
- When a model becomes too complex, it will start to fit the noise in the data
- What happens if we apply our model to predict some data that the model has never seen before? It will not work well.
- This is called over-fitting



Overfitting

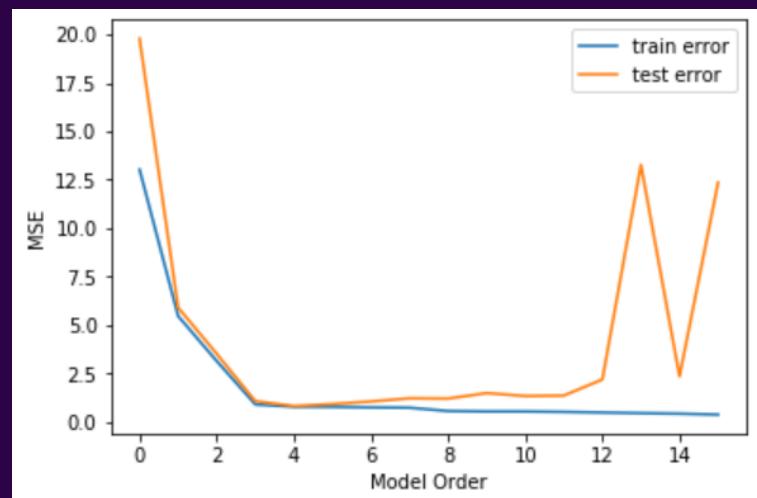
- Split the data set into a train set and a test set
- Train set will be used to train the model
- The test set will not be seen by the model during the training process
- Use test set to evaluate the model when a model is trained



- With the training and test sets shown, which one do you think is the better model now?

Train and Test Error

- Plot of train error and test error for different model order
- Initially both train and test error go down as model order increase
- But at a certain point, test error start to increase because of overfitting



Review
oooooooooooo

Regularization
●oooo

Opt
ooooooo

Logistic
oooooooooooo

Demo
oo

Multiclass
oooo

Lab
o

Outline

1 Leftovers from Day 3

2 Regularization

3 Non-linear Optimization

4 Logistic Regression

5 Lab: Diagnosing Breast Cancer

6 Multiclass Classificaiton

7 Lab: Iris Dataset

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization:** methods to prevent overfitting

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization:** methods to prevent overfitting

- We just covered regularization by model order selection

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization:** methods to prevent overfitting

- We just covered regularization by model order selection
- Is there another way? Talk among your classmates.

How can we prevent overfitting without knowing the model order before-hand?

- **Regularization:** methods to prevent overfitting

- We just covered regularization by model order selection
- Is there another way? Talk among your classmates.
- Solution: We can change our cost function.

Weight Based Regularization

- Looking back at the polynomial overfitting
- Notice that weight-size increases with overfitting

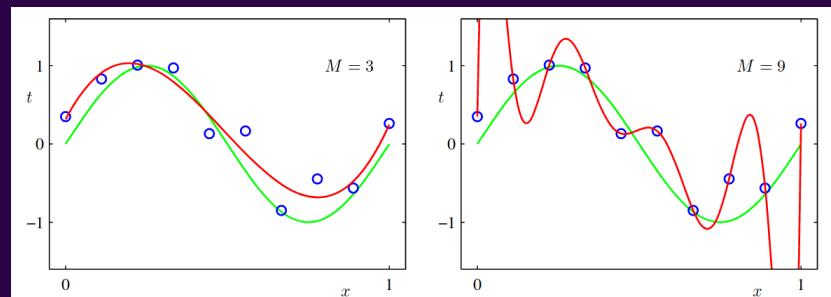


Table 1.1 Table of the coefficients w^* for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
w_0^*	0.19		0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

New Cost Function

$$J(\mathbf{w}) = \frac{1}{N} \|Y - X\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2$$

- Penalize complexity by simultaneously minimizing weight values.
- We call λ a **hyper-parameter**
 - λ determines relative importance

Table 1.2 Table of the coefficients w^* for $M = 9$ polynomials with various values for the regularization parameter λ . Note that $\ln \lambda = -\infty$ corresponds to a model with no regularization, i.e., to the graph at the bottom right in Figure 1.4. We see that, as the value of λ increases, the typical magnitude of the coefficients gets smaller.

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

Tuning Hyper-parameters

- Motivation: never determine a hyper-parameter based on training data
- **Hyper-Parameter:** a parameter of the algorithm that is not a model-parameter solved for in optimization.
 - Ex: λ weight regularization value vs. model weights (w)
- Solution: split dataset into three
 - **Training set:** to compute the model-parameters (w)
 - **Validation set:** to tune hyper-parameters (λ)
 - **Test set:** to compute the performance of the algorithm (MSE)

Outline

- 1** Leftovers from Day 3
 - 2** Regularization
 - 3** Non-linear Optimization
 - 4** Logistic Regression
 - 5** Lab: Diagnosing Breast Cancer
 - 6** Multiclass Classification
 - 7** Lab: Iris Dataset

Motivation

- Cannot rely on closed form solutions
 - Computation efficiency: operations like inverting a matrix is not efficient
 - For more complex problems such as neural networks, a closed-form solution is not always available
 - Need an optimization technique to find an optimal solution
 - Machine learning practitioners use **gradient**-based methods

Gradient Descent Algorithm

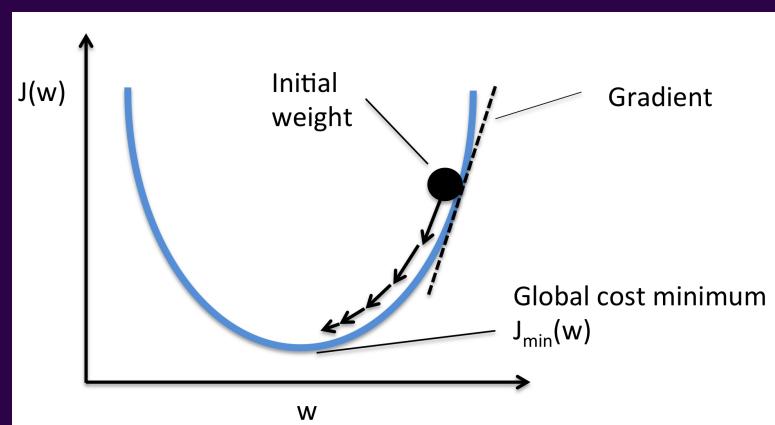
■ Update Rule

Repeat{

$$\mathbf{w}_{new} = \mathbf{w} - \alpha \nabla_{\mathbf{w}} J$$

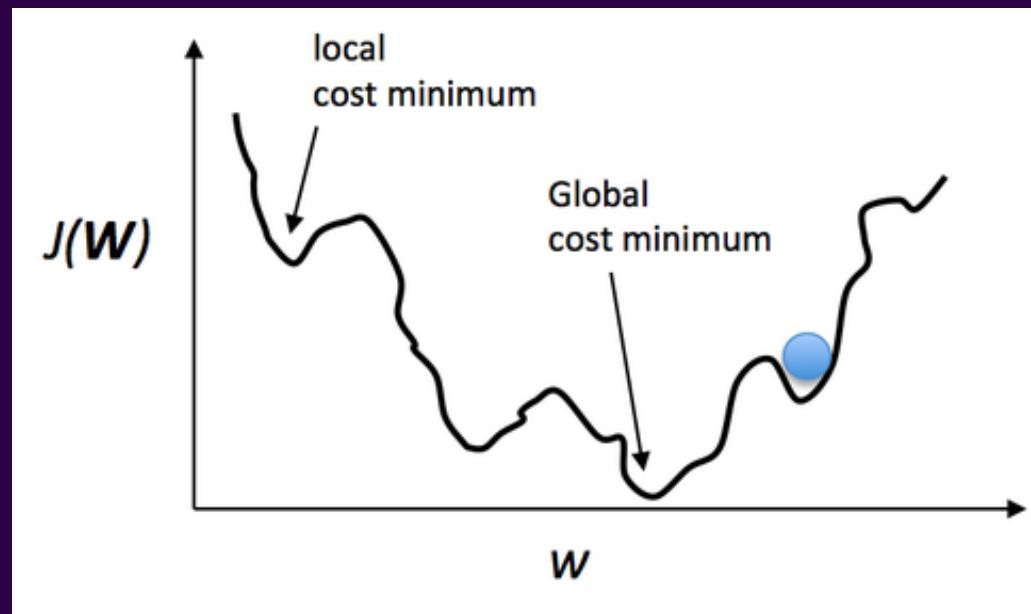
}

α is the learning rate

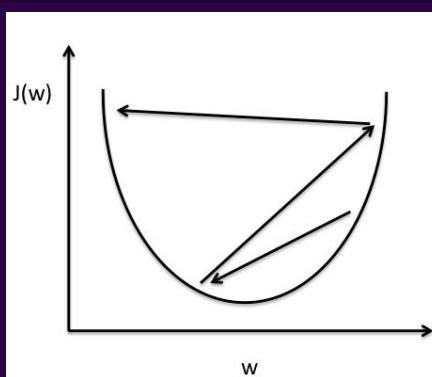


General Loss Function Contours

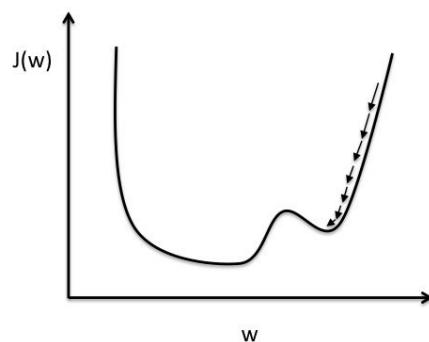
- Most loss function contours are not perfectly parabolic
- Our goal is to find a solution that is very close to global minimum by the right choice of hyper-parameters



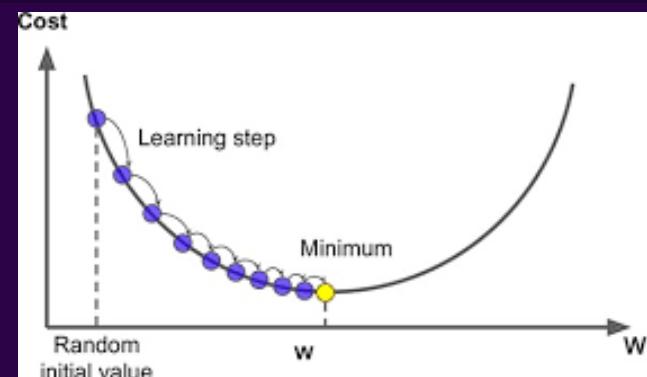
Understanding Learning Rate



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.



Correct learning rate

Review
oooooooooooo

Regularization
ooooo

Opt
oooooo•o

Logistic
oooooooooooo

Demo
oo

Multiclass
oooo

Lab
o

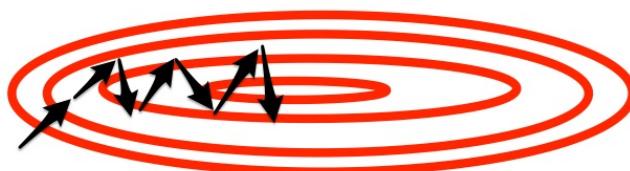
Some Animations

- Demonstrate gradient descent animation

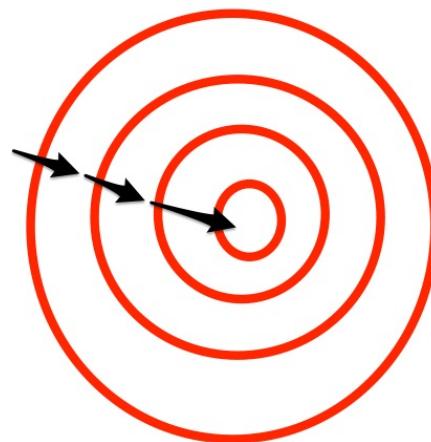
Importance of Feature Normalization (Optional)

- Helps improve the performance of gradient based optimization

Without feature scaling



With feature scaling



Review
oooooooooooo

Regularization
ooooo

Opt
ooooooo

Logistic
●oooooooooooo

Demo
oo

Multiclass
oooo

Lab
o

Outline

- 1 Leftovers from Day 3
- 2 Regularization
- 3 Non-linear Optimization
- 4 Logistic Regression
- 5 Lab: Diagnosing Breast Cancer
- 6 Multiclass Classificaiton
- 7 Lab: Iris Dataset

Classification Vs. Regression

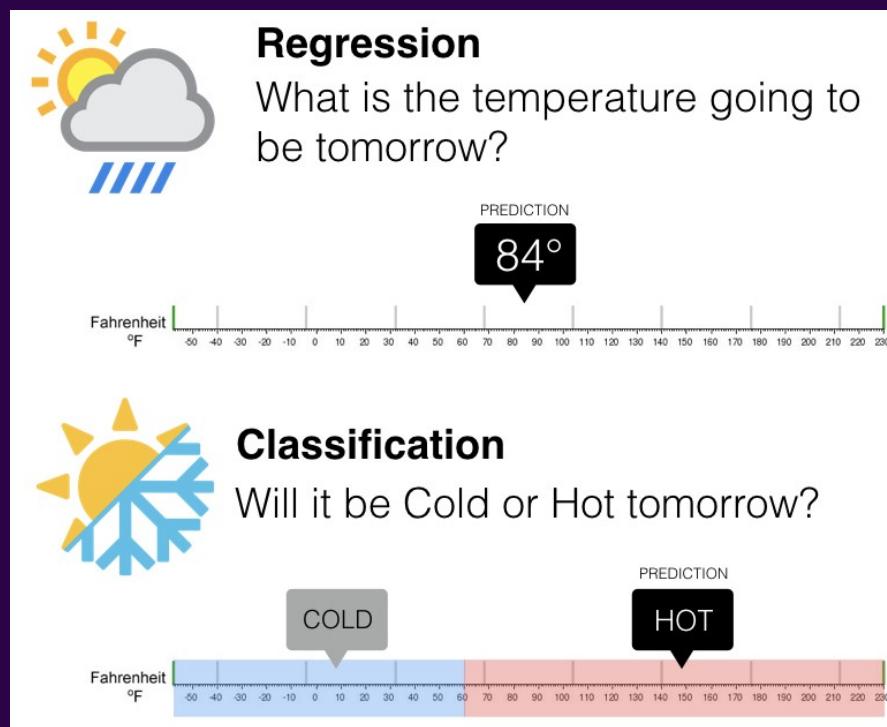
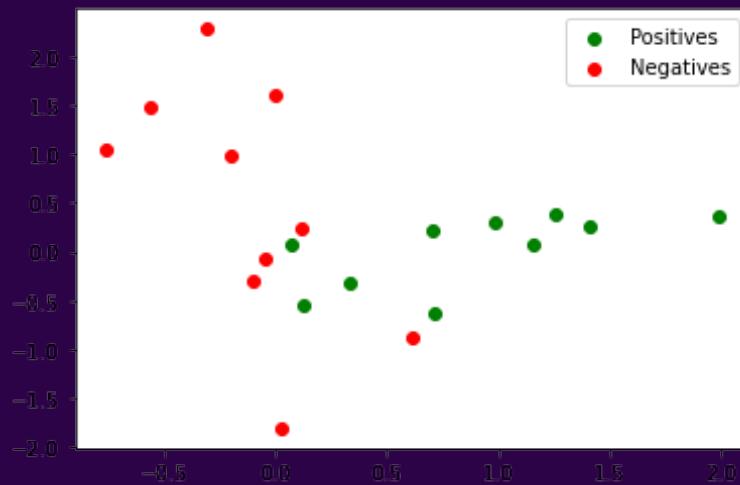


Figure: <https://www.pinterest.com/pin/672232681855858622/?lp=true>

Classification

Given the dataset (x_i, y_i) for $i = 1, 2, \dots, N$, find a function $f(x)$ (model) so that it can predict the label \hat{y} for some input x , even if it is not in the dataset, i.e. $\hat{y} = f(x)$.

- Positive : $y = 1$
- Negative : $y = 0$



Classification via regression

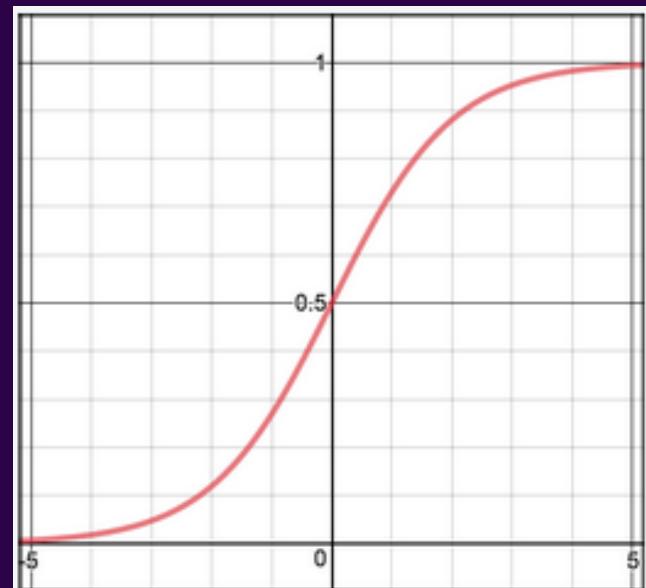
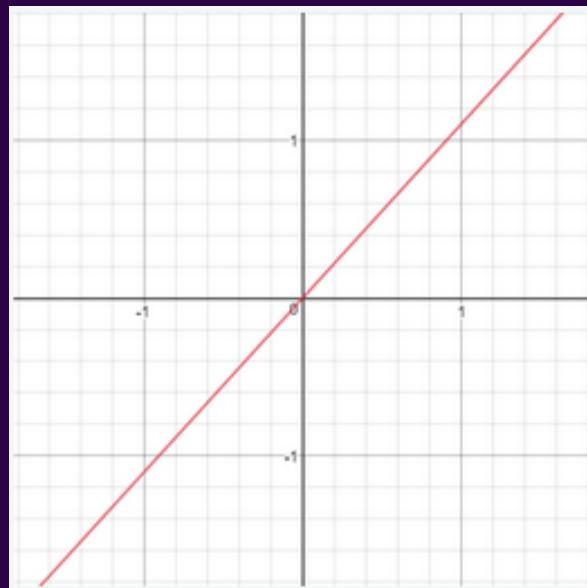
- Proposal: train a model to fit the data with linear regression (potentially with polynomial features)!

Classification via regression

- Proposal: train a model to fit the data with linear regression (potentially with polynomial features)!
- What could be the problem?

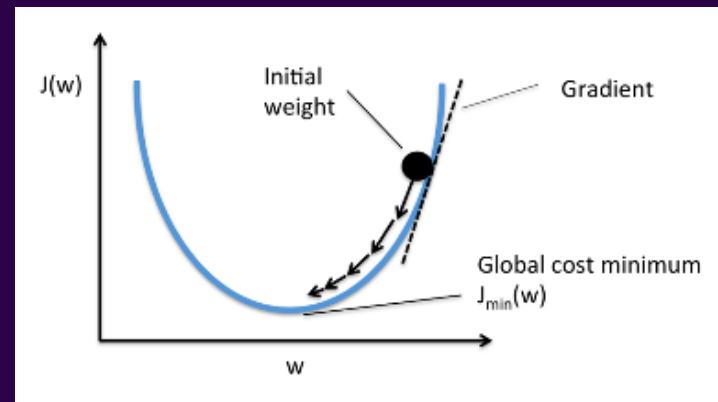
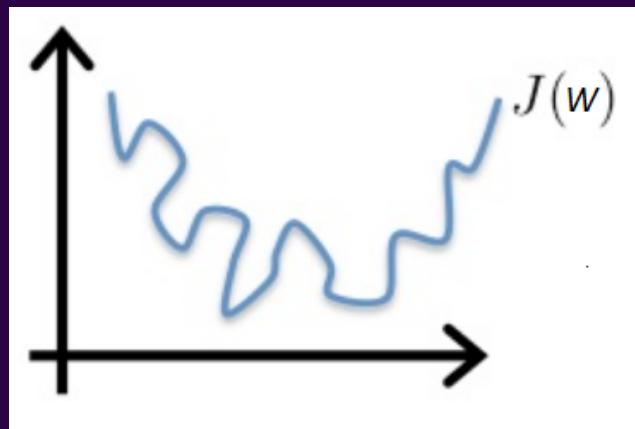
Sigmoid Function

- Recall from linear regression $z = w_0 + w_1x$
- By applying the sigmoid function to z , we enforce $0 \leq \hat{y} \leq 1$
 - $\hat{y} = \text{sigmoid}(z) = \frac{1}{1+e^{-z}}$

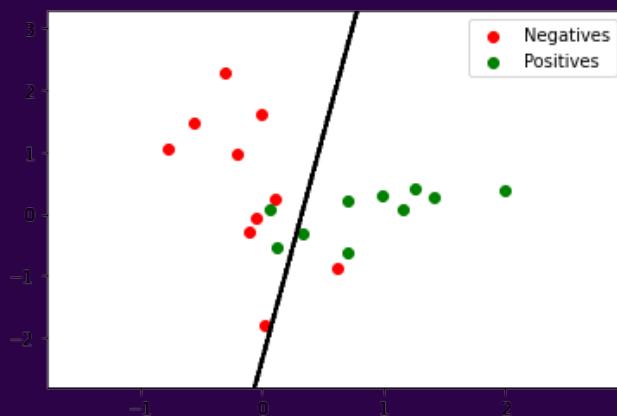


Classification Loss Function

- Cannot use the same cost function that we used for linear regression
 - MSE of a logistic function has many local minima
- Use $\frac{1}{N} \sum_{i=1}^N \left[-y \log(\hat{y}) - (1-y) \log(1-\hat{y}) \right]$
 - This loss function is called binary cross entropy loss
 - This loss function has only one minimum



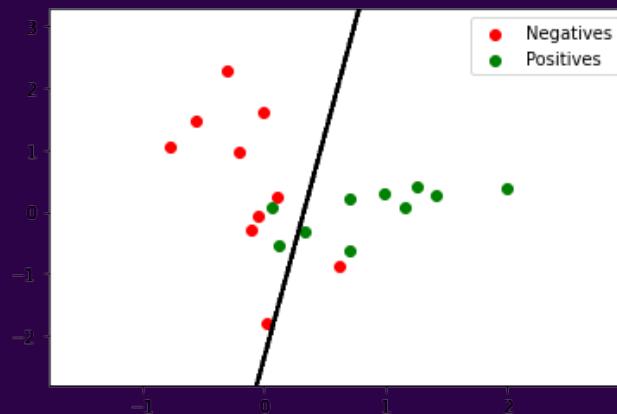
Decision Boundary



■ Evaluation metric :

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}}$$

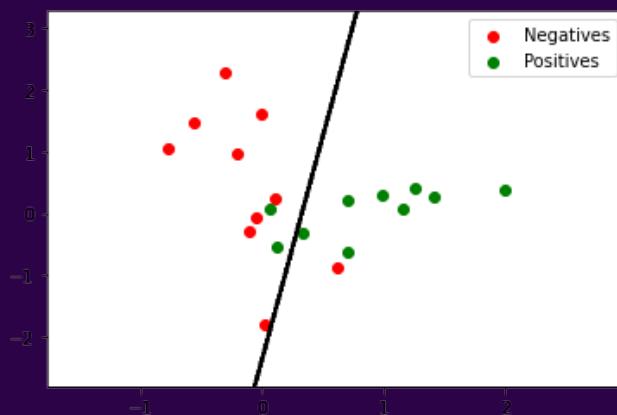
Decision Boundary



- Evaluation metric :

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}}$$

- What is the accuracy in this example ?



■ Evaluation metric :

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}} = \frac{17}{20} = 0.85 = 85\%$$

Classifier

- ## ■ How to deal with uncertainty ?

Review
oooooooooooo

Regularization
ooooo

Opt
ooooooo

Logistic
oooooooo●ooooo

Demo
oo

Multiclass
oooo

Lab
o

Classifier

- How to deal with uncertainty ?
 - $\hat{y} = f(x)$ should be between 0 and 1.

Classifier

- How to deal with uncertainty ?
 - $\hat{y} = f(x)$ should be between 0 and 1.
 - If \hat{y} is close to 0, the data is probably negative
 - If \hat{y} is close to 1, the data is probably positive
 - If \hat{y} is around 0.5, we are not sure.

Review

oooooooooooo

Regularization

○○○○○

Opt

1

Logistic

oooooooooooo●oooo

Demo

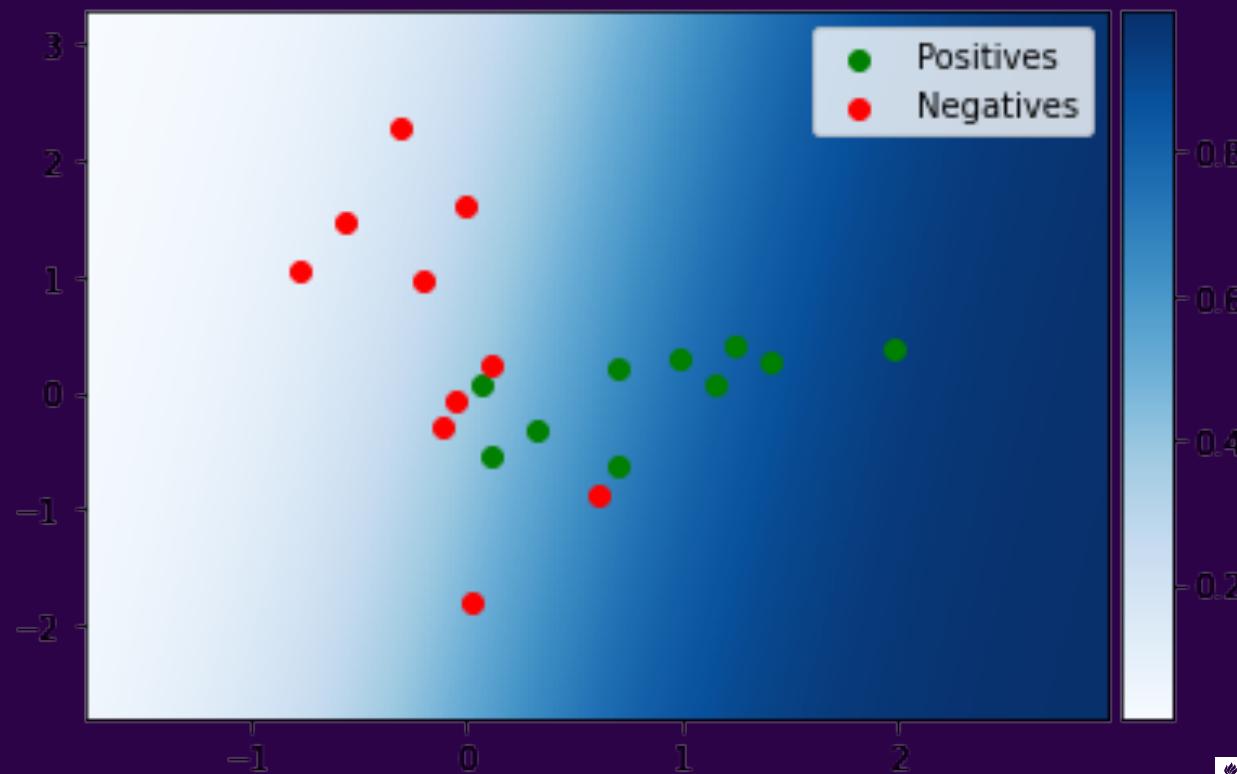
○○

Multiclass

0000

Lab

Classifier



Types of Errors in Classification

- Correct predictions:

- True Positive (TP) : Predict $\hat{y} = 1$ when $y = 1$
 - True Negative (TN) : Predict $\hat{y} = 0$ when $y = 0$

- Two types of errors:

- False Positive/ False Alarm (FP): $\hat{y} = 1$ when $y = 0$
 - False Negative/ Missed Detection (FN): $\hat{y} = 0$ when $y = 1$

Review
oooooooooooo

Regularization
ooooo

Opt
ooooooo

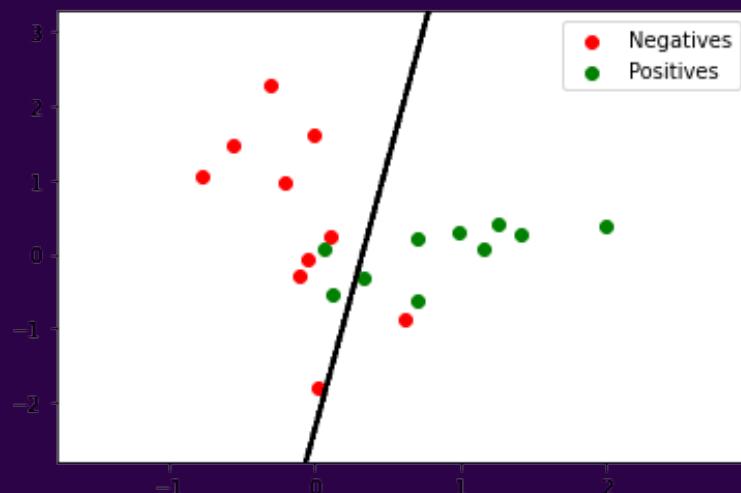
Logistic
oooooooooooo●oo

Demo
oo

Multiclass
oooo

Lab
o

Example



- How many True Positive (TP) are there ?
- How many True Negative (TN) are there ?
- How many False Positive (FP) are there ?
- How many False Negative (FN) are there ?

Performance metrics for a classifier

- Accuracy of a classifier:
 - $(TP + TN) / (TP + FP + TN + FN)$ (percentage of correct classification)
 - Why accuracy alone is not a good measure for assessing the model

Performance metrics for a classifier

- Accuracy of a classifier:

- $(TP + TN) / (TP + FP + TN + FN)$ (percentage of correct classification)

- Why accuracy alone is not a good measure for assessing the model

- There might be an overwhelming proportion of one class over another (unbalanced classes)
 - Example: A rare disease occurs 1 in ten thousand people
 - A test that classifies everyone as free of the disease can achieve 99.999% accuracy when tested with people drawn randomly from the entire population

Other metrics

Some other metrics

- Sensitivity/Recall/TPR = $TP/(TP+FN)$ (How many positives are detected among all positive?)
 - Precision = $TP/(TP+FP)$ (How many detected positives are actually positive?)
 - Specificity/TNR = $TN/(TN+FP)$ (How many negatives are detected among all negatives?)

Exercise: think of tasks for which sensitivity, precision, or specificity is a better metric.

Outline

- 1 Leftovers from Day 3
 - 2 Regularization
 - 3 Non-linear Optimization
 - 4 Logistic Regression
 - 5 Lab: Diagnosing Breast Cancer
 - 6 Multiclass Classification
 - 7 Lab: Iris Dataset

Lab: Diagnosing Breast Cancer

- We're going to use the breast cancer dataset to predict whether the patients' scans show a malignant tumour or a benign tumour.
- Let's try to find the best linear classifier using logistic regression.

Outline

- 1 Leftovers from Day 3
 - 2 Regularization
 - 3 Non-linear Optimization
 - 4 Logistic Regression
 - 5 Lab: Diagnosing Breast Cancer
 - 6 Multiclass Classification
 - 7 Lab: Iris Dataset

Multiclass Classification

- Previous model: $f(\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$
- Representing Multiple Classes:
 - One-hot / 1-of-K vectors, ex : 4 Class
 - Class 1 : $\mathbf{y} = [1, 0, 0, 0]$
 - Class 2 : $\mathbf{y} = [0, 1, 0, 0]$
 - Class 3 : $\mathbf{y} = [0, 0, 1, 0]$
 - Class 4 : $\mathbf{y} = [0, 0, 0, 1]$

Multiclass Classification

- Previous model: $f(\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$
- Representing Multiple Classes:
 - One-hot / 1-of-K vectors, ex : 4 Class
 - Class 1 : $\mathbf{y} = [1, 0, 0, 0]$
 - Class 2 : $\mathbf{y} = [0, 1, 0, 0]$
 - Class 3 : $\mathbf{y} = [0, 0, 1, 0]$
 - Class 4 : $\mathbf{y} = [0, 0, 0, 1]$
- Multiple outputs: $f(\mathbf{x}) = \text{softmax}(W^T \phi(\mathbf{x}))$
- Shape of $W^T \phi(\mathbf{x})$: $(K, 1) = (K, D) \times (D, 1)$
- $\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$

Multiclass Classification

- Multiple outputs: $f(\mathbf{x}) = \text{softmax}(\mathbf{z})$ with $\mathbf{z} = W^T \phi(\mathbf{x})$
 - $\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_i e^{z_j}}$

- Softmax example: If $\mathbf{z} = \begin{bmatrix} -1 \\ 2 \\ 1 \\ -4 \end{bmatrix}$ then,

$$\text{softmax}(z) = \begin{bmatrix} \frac{e^{-1}}{e^{-1}+e^2+e^1+e^{-4}} \\ \frac{e^2}{e^{-1}+e^2+e^1+e^{-4}} \\ \frac{e^1}{e^{-1}+e^2+e^1+e^{-4}} \\ \frac{e^{-4}}{e^{-1}+e^2+e^1+e^{-4}} \end{bmatrix} \approx \begin{bmatrix} 0.035 \\ 0.704 \\ 0.259 \\ 0.002 \end{bmatrix}$$

Cross-entropy

- Multiple outputs: $\hat{\mathbf{y}}_i = \text{softmax}(W^T \phi(\mathbf{x}_i))$
- Cross-Entropy: $J(W) = -\sum_{i=1}^N \sum_{k=1}^K \mathbf{y}_{ik} \log(\hat{\mathbf{y}}_{ik})$
- Example : $K = 4$

If, $\mathbf{y}_i = [0, 0, 1, 0]$ then, $\sum_{k=1}^K \mathbf{y}_{ik} \log(\hat{\mathbf{y}}_{ik}) = \log(\hat{\mathbf{y}}_{i3})$

Outline

- 1** Leftovers from Day 3
 - 2** Regularization
 - 3** Non-linear Optimization
 - 4** Logistic Regression
 - 5** Lab: Diagnosing Breast Cancer
 - 6** Multiclass Classification
 - 7** Lab: Iris Dataset