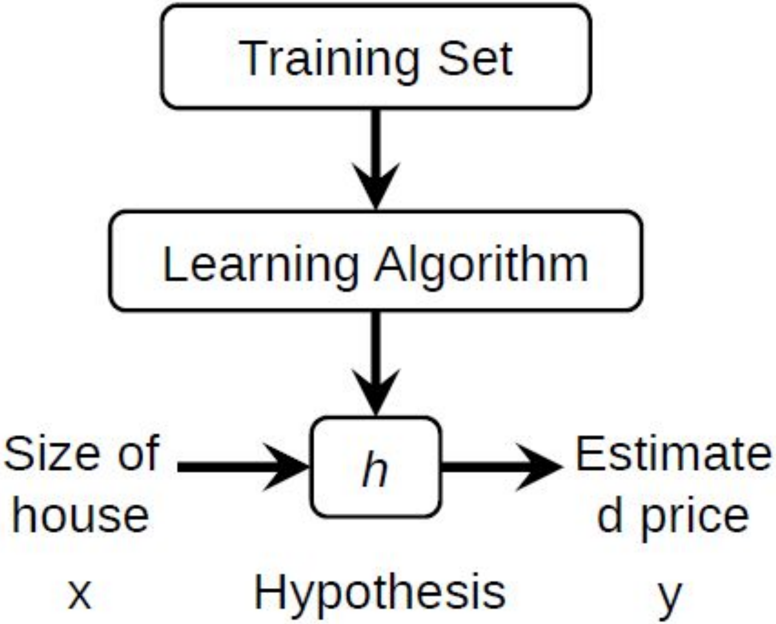
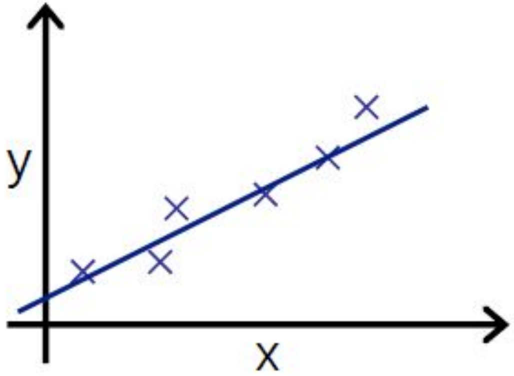
Text Classification II

Logistic Regression

Quick Review on Linear Regression



How do we represent *h* ?

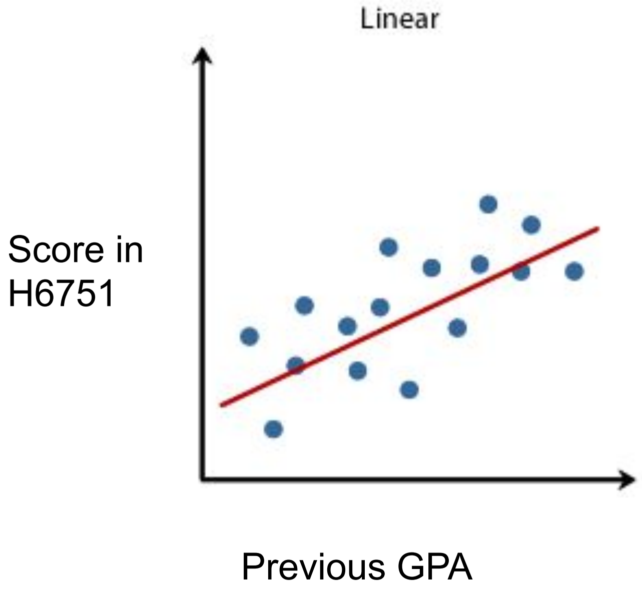


$$h(x) = w_0 + w_1 x$$

Linear regression with one variable.
"**Univariate Linear Regression**"

Continuous Target

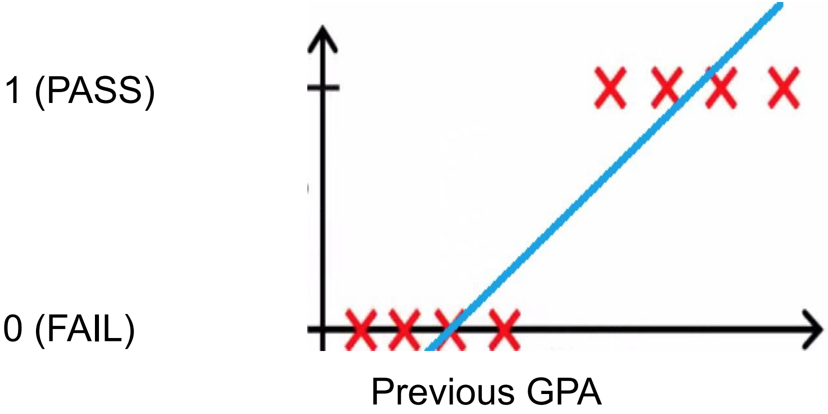
- Let us build an auto-grade algorithms
- Input feature is one scalar: your previous GPA
- Target value is: your score in H6751



Discrete Target

- We only want to predict whether you can pass H6751
- Input feature is one scalar: your previous GPA
- Target value is: a binary value (1: pass, 0: fail)

Classification Problem



Classification

Binary Classification

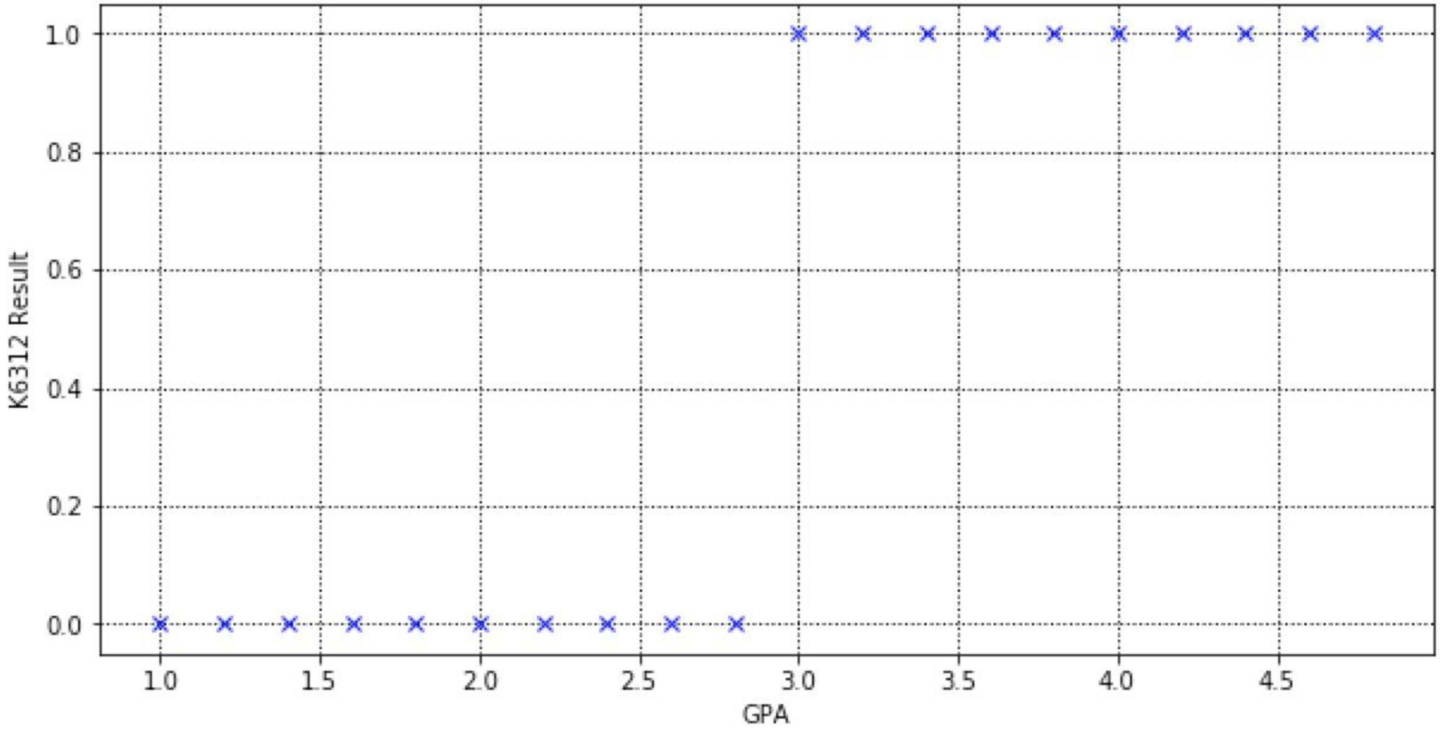
- Email: spam or not spam
- Online Transaction: fraud or not fraud
- ……

$$y = \begin{cases} -1 & \text{Negtive class, e. g. not spam and not fraud} \\ 1 & \text{Positive class, e. g. spam and fraud} \end{cases}$$

Machine Learning is to learn a function from data such that

$$f : X \rightarrow Y$$

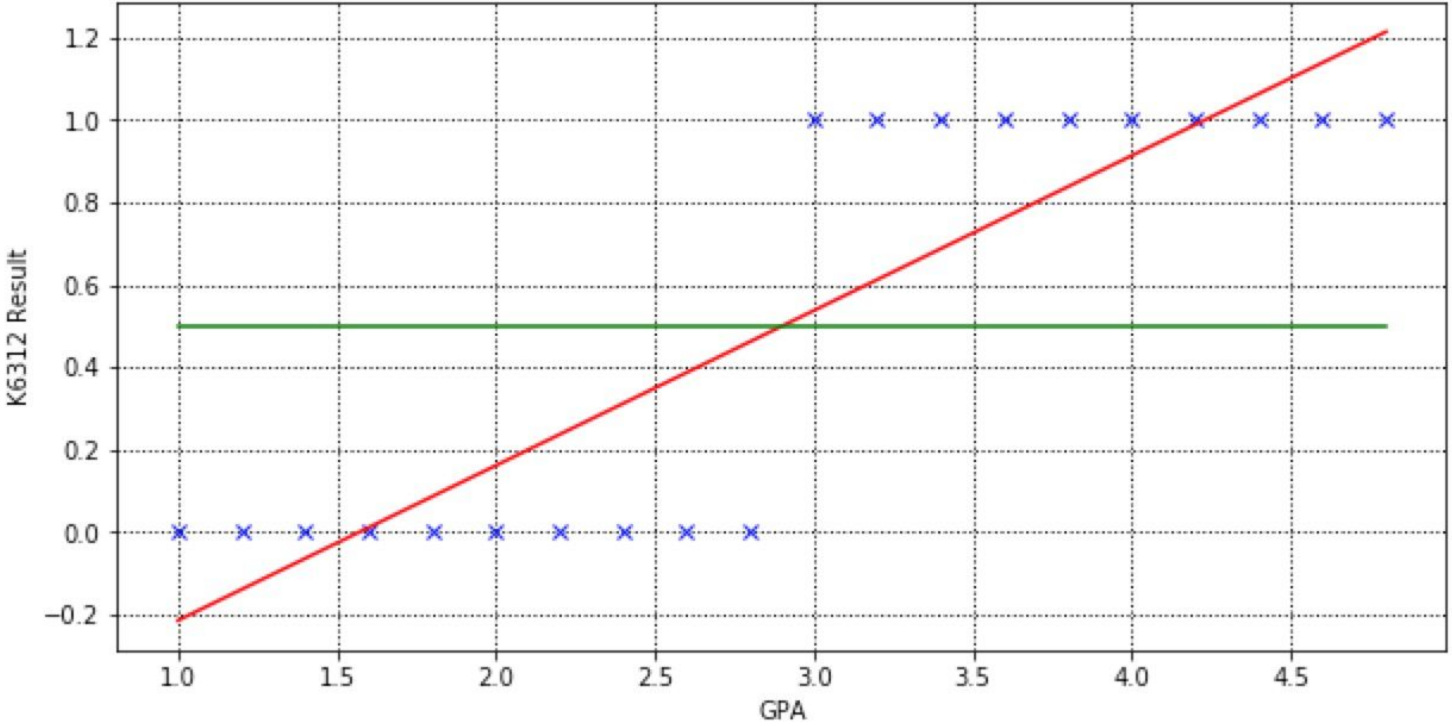
Can we use linear regression for classification?



After fitting,

```
print(lin_regression.coef_)
print(lin_regression.intercept_)
```

```
[0.37593985]
-0.5902255639097744
```



Output Value is continuous

- For classification problem, we want the output value to be probabilistic, which should be in range(0, 1).
- However, the output of linear regression is unbounded

```
print(lin_regression.coef_)
print(lin_regression.intercept_)
```

```
[0.37593985]
-0.5902255639097744
```

y=0.3759*x-0.59

When x = 5,   y=1.289

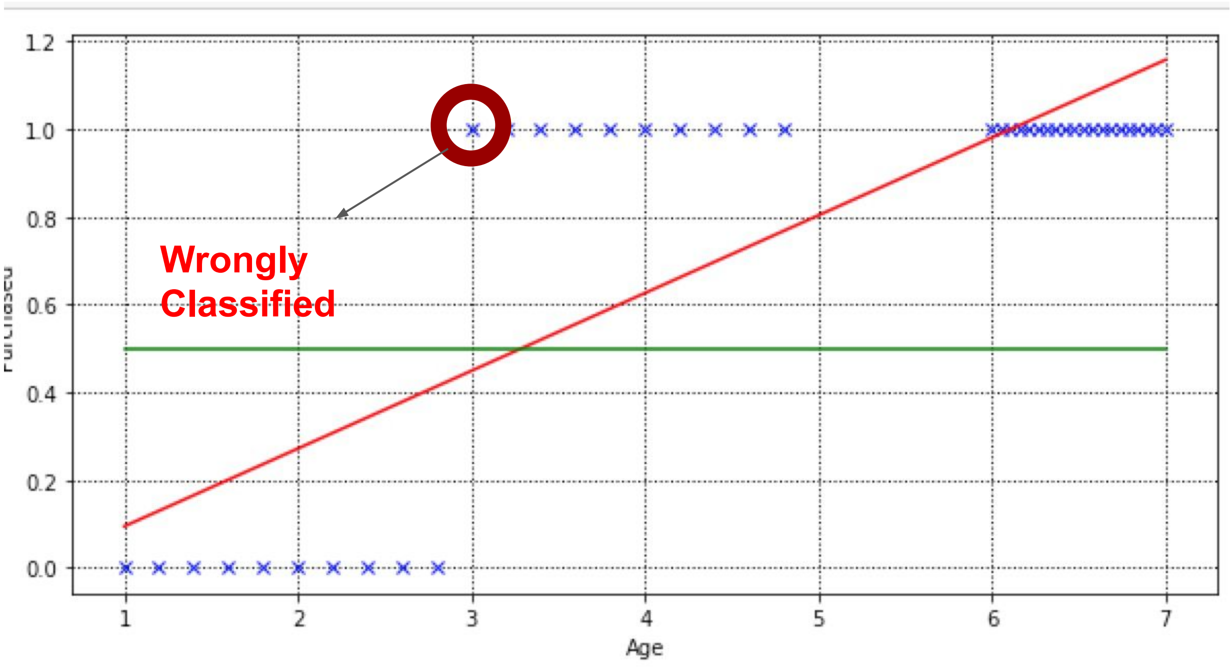
When x = 4.6, y=1.038

When x = 1.2, y=-0.13

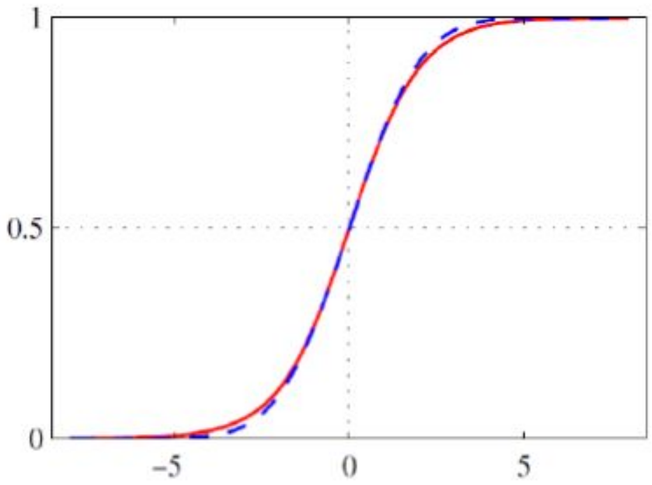
What we want is that the proba. Scores of the first two cases is close to 1 and the last case is close to 0.

Imbalanced data

- Let us add 20 students whose GPA are in the range(6, 7) and pass the H6751



Logistic Function



$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t}$$

t : (-infinite, +infinite)

$\sigma(t)$: probabilistic score from 0 to 1

Linear Regression

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t}$$

Logistic Regression

$$f(x) = w * x + b$$

$$f(x) = \frac{1}{1+e^{-(w*x+b)}}$$

Logistic Regression

Logistic Regression

- Uses logistic function to model binary target
- Model the distribution of *p(y = 1|**x**)* given ***x***
- The exact parametric formulation is:

$$p(y=1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$$

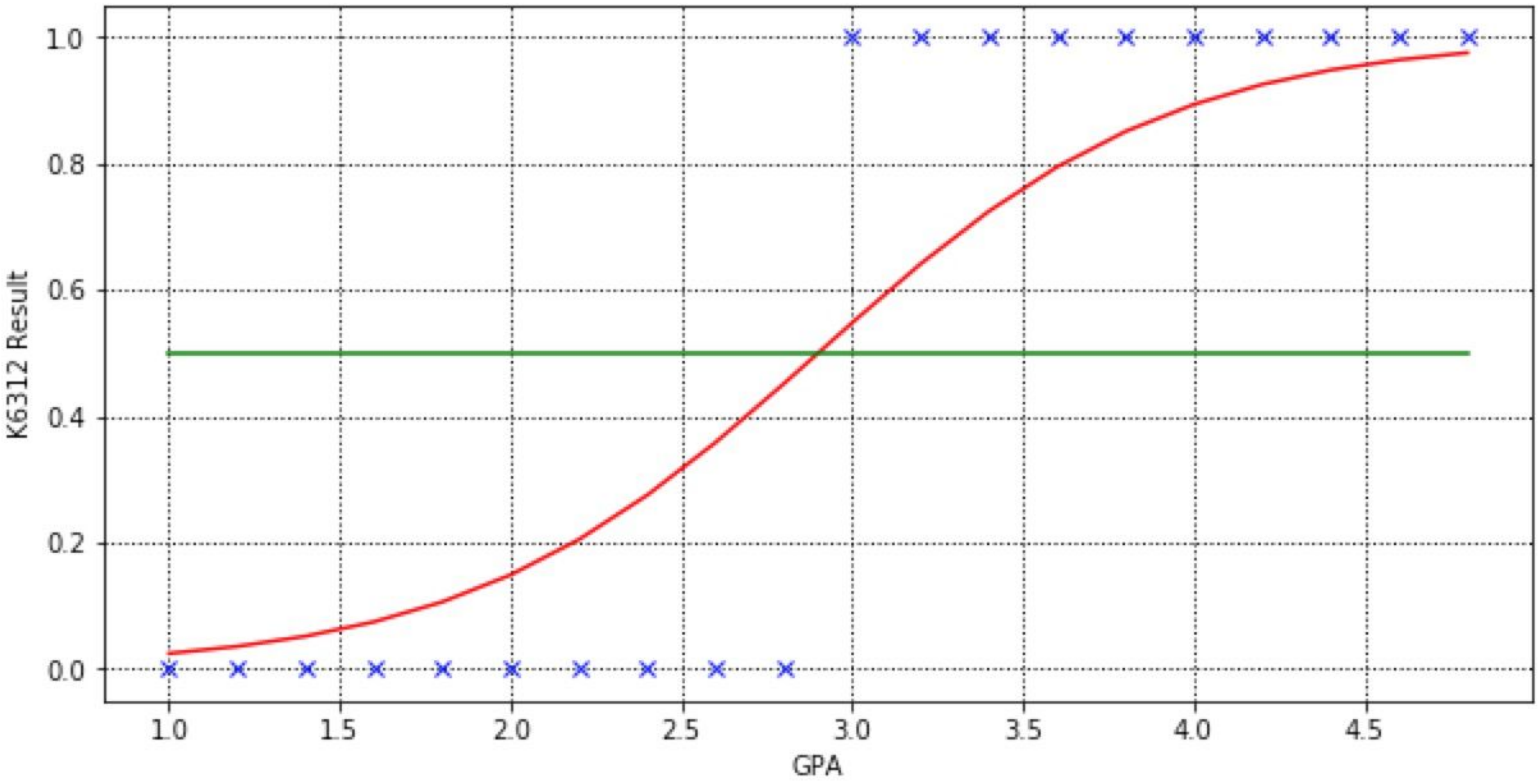
$$p(y=0|x) = 1 - \frac{1}{e^{(-wx+b)}+1} = \frac{1}{e^{(wx+b)}+1}$$

- Let us check the performance of Logistic Regression on H6751 auto-grade system

After fitting

```
print(log_regression.coef_)
print(log_regression.intercept_)
```

```
[[1.93582432]]
[-5.61388646]
```



Output Value is Prob.score

```
print(log_regression.coef_)
print(log_regression.intercept_)
```

```
[[1.93582432]]
[-5.61388646]
```

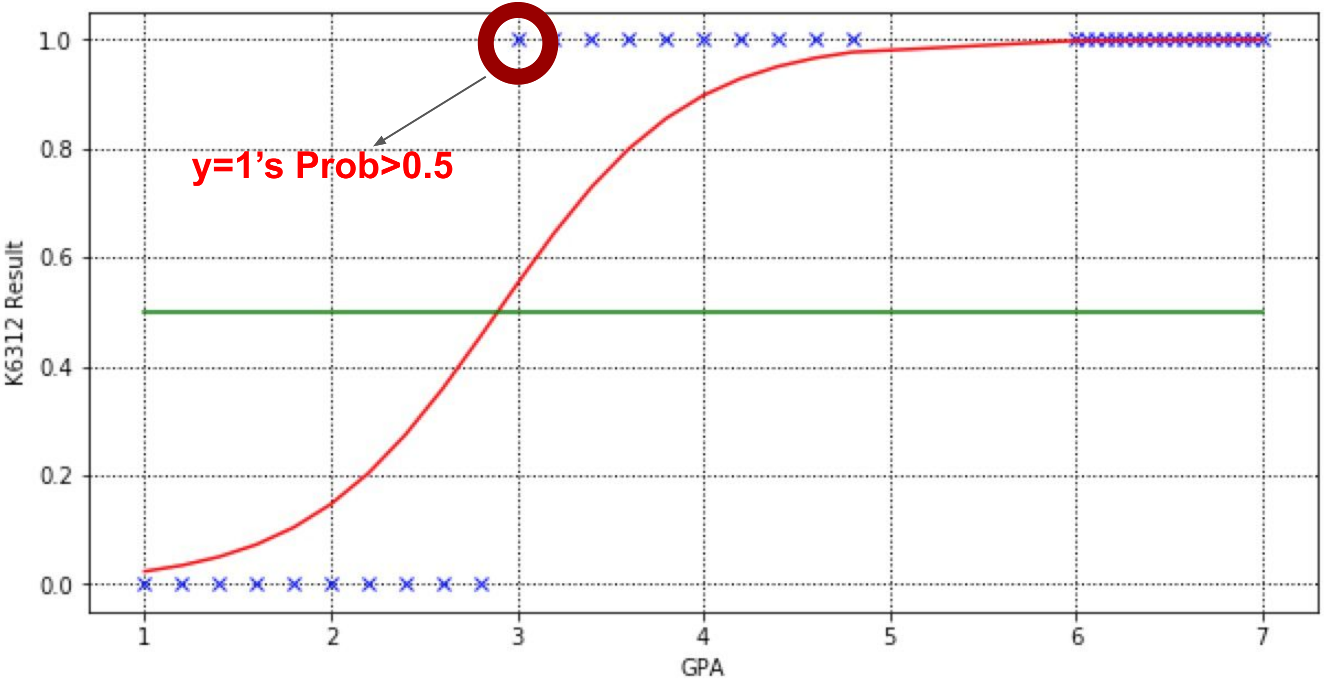
$$p(y=1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$$

p(y=1|x)=e^t/(1+e^t)
p(y=0|x)=1/(1+e^t)
t=1.93*x-5.6

	prob(y=0)	prob(y=1)
x = 5	[[0.01686949	0.98313051]
x = 4.6	[0.03588451	0.96411549]
x = 1.2	[0.96411521	0.03588479]]

Imbalanced data

- Let us add 20 students whose GPA are in the range(6, 7) and pass the H6751



How to learn parameters

- Fitting the data
- In the python code, it is simple

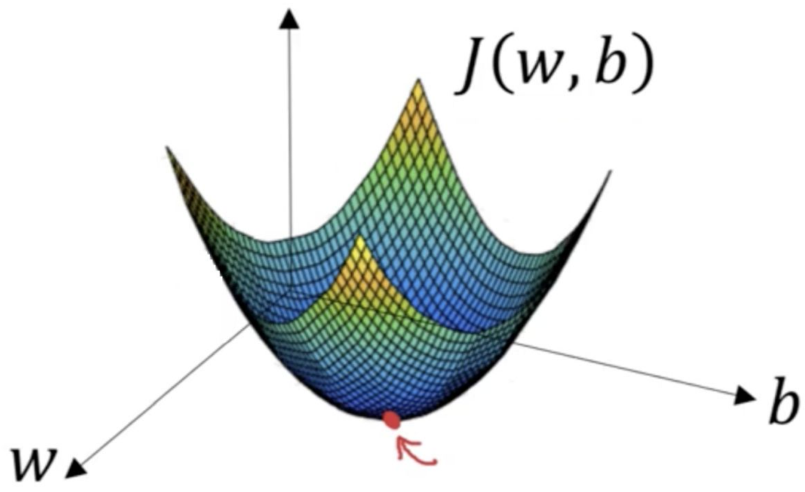
Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
```

- Actually, it is an optimization problem (in math perspective)

Optimization

- Fitting the data -> define a loss function, which reflects the fitness of the different model parameters over the parameters
- Optimization is the process to search the minimum point



Entropy Loss

- For each single data point: $\tilde{y} = p(y=1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$

$$Loss(y, \tilde{y}) = -[y log\tilde{y} + (1-y)log(1-\tilde{y})]$$

- To understand this loss function, compute the loss values for these following cases:
  - If y=1, predict prob of y=1 is 0.9,
  - If y=0, predict prob of y=1 is 0.2,

  - If y=1, predict prob of y=1 is 0.2,
  - If y=0, predict prob of y=1 is 0.9,

Whether the model prediction is good nor not?

Entropy Loss

- For each single data point: $\tilde{y} = p(y=1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$

$$Loss(y, \tilde{y}) = -[ylog\tilde{y} + (1-y)log(1-\tilde{y})]$$

- To understand this loss function, compute the loss values for these following cases:
  - **If y=1, predict prob of y=1 is 0.9, -log0.9=0.1**
  - **If y=0, predict prob of y=1 is 0.2, -log0.8=0.22**

Good Fitness, Low Loss

  - **If y=1, predict prob of y=1 is 0.2, -log0.2=1.6**
  - **If y=0, predict prob of y=1 is 0.9, -log0.1=2.3**

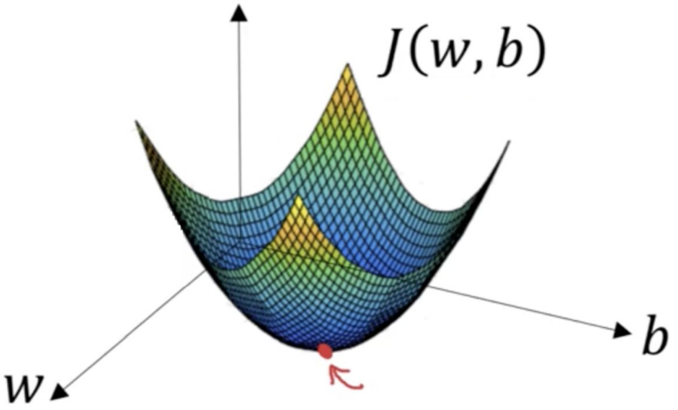
Bad Fitness, High Loss

Optimization is to reduce Loss

- For training data of m data points (x, y), the loss is the function of model parameters

$$J(w,b) = \frac{\sum_{i=1}^{m} Loss(\tilde{y}^{i}, y^{i})}{m}$$

$$\tilde{y} = p(y=1|x) = \frac{1}{e^{(-wx+b)}+1} = \frac{e^{(wx+b)}}{e^{(wx+b)}+1}$$



Gradient Descent Algorithm

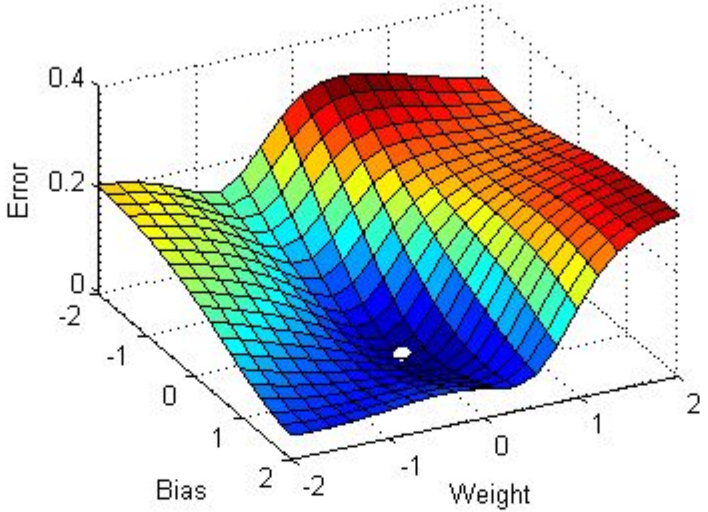
Gradient for loss function f for $\mathbf{x}_n$, which computed by BP algorithm

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \alpha \boxed{\nabla J(\mathbf{w})}$$

New Parameters Guess

Current Parameters Guess

Learning Rate



Like hiking down a mountain

Decision boundary of Logistic Regression

- Decision is made by comparing the probabilities

$$p(y=1|\mathbf{x}) > p(y=-1|\mathbf{x}) \Leftrightarrow \frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})} > 1$$

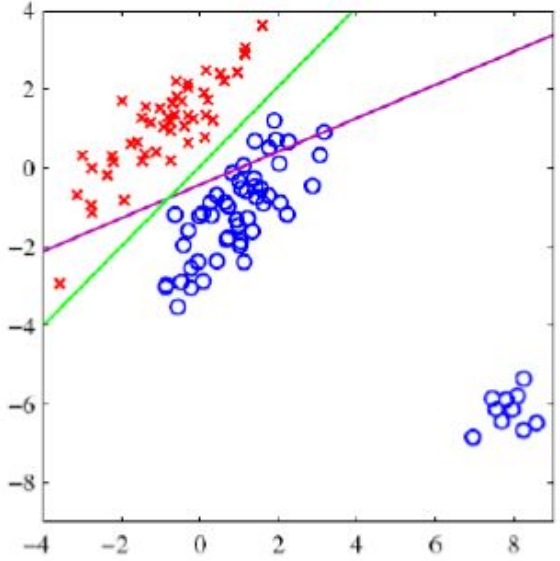
- Take the logarithm

$$\ln \frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})} = \mathbf{w}^\top \mathbf{x} + b \rightarrow \mathbf{w}^\top \mathbf{x}$$

- Decision boundary is linear

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

$$y = \begin{cases} +1 & \text{if } \mathbf{w}^\top \mathbf{x} + b > \boxed{0} \\ -1 & \text{otherwise} \end{cases}$$



* The threshold is tunable

We can have multiple w

- For simplicity, we only have one feature x therefore only one w and bias b in the example.
- In practice, each data sample is represented by a n-dimensional vector and the logistic regression model has n weights and one bias b.
- For text mining, the input vectors will be BoW vectors.

output: $\sigma(-1.2*(10) + 1.4*(5) + 2.2*(3) + 0.6*(5) + 0.2)$

Evaluation

How to do we evaluate the model performance?

i.e., how to quantify the matching degree between the ground truth y and the predicted labels y^.

Evaluation of Classification Problems

- Confusion Matrix

	Predicted Positive	Predicted Negative
Positive Label	**TP** True Positive	**FN** False Negative
Negative Label	**FP** False Positive	**TN** True Negative

- Accuracy: How accurate is the prediction?

$$\frac{\text{Correct Prediction}}{\text{Total \#-of-Samples}} = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision and Recall

	Predicted Positive	Predicted Negative
Positive Label	TP	FN
Negative Label	FP	TN

- $\text{Precision} = \dfrac{TP}{TP + FP}$
  - how accurate the positive prediction is?

- $\text{Recall} = \dfrac{TP}{TP + FN}$
  - how many positive cases are detected?

Example: H6751 Auto-grade

	Predicted Positive	Predicted Negative
Positive Label	27	4
Negative Label	1	18

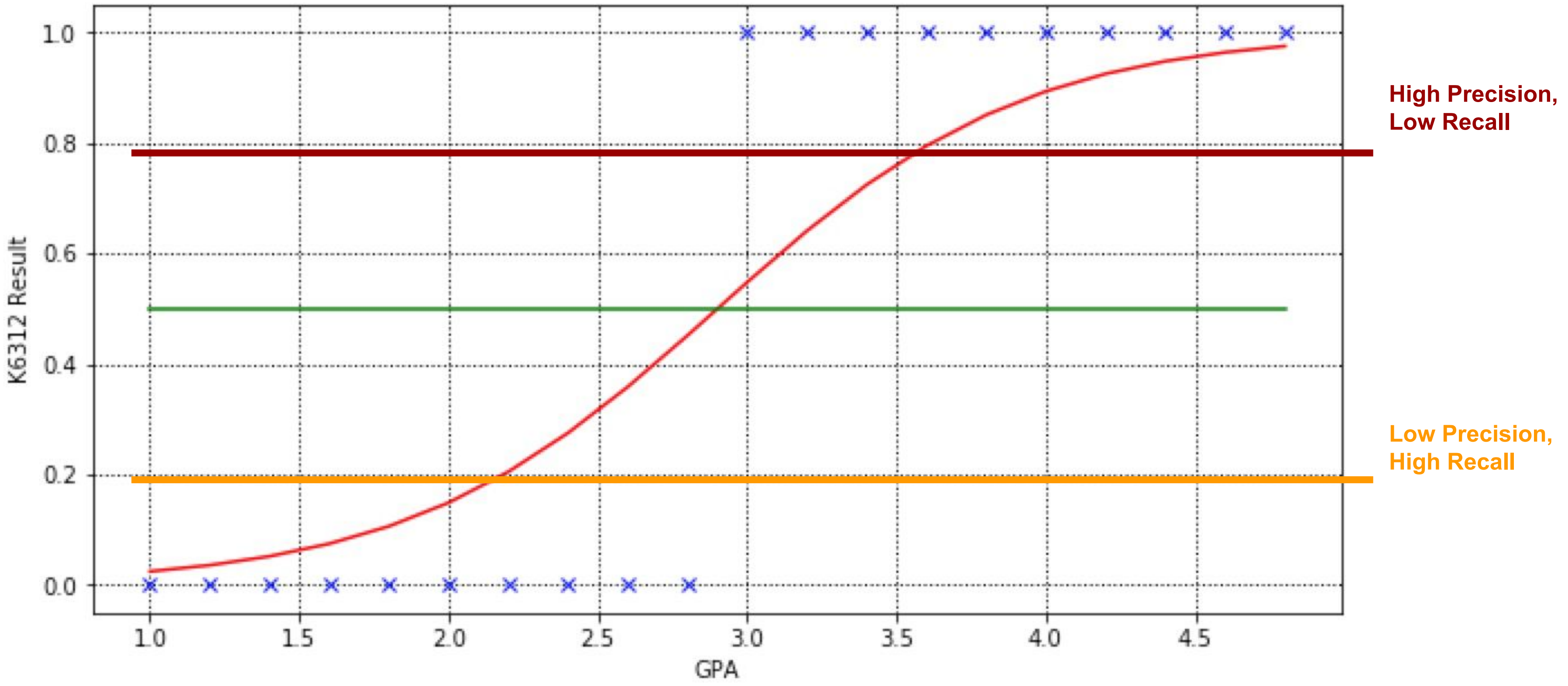
- Accuracy = (27 + 18) / (27 + 1 + 18 + 4) = 0.9
- Precision = 27 / (27 + 1) = 0.964
- Recall = 27/ (27 + 4) = 0.871

- Can we do better?

Precision vs Recall

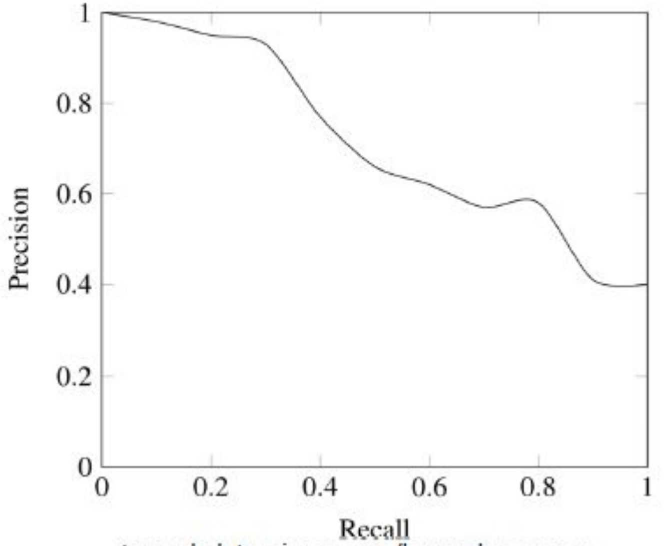
- Case 1: Accuracy is high, but recall is low.
  - Examples?
- Case 2: Accuracy is high, but precision is low.
  - Examples?

Precision vs Recall



Precision v.s. Recall

- Under non-trivial situation, precision and recall cannot be optimized at the same time
  - Which one to optimize depends on use cases

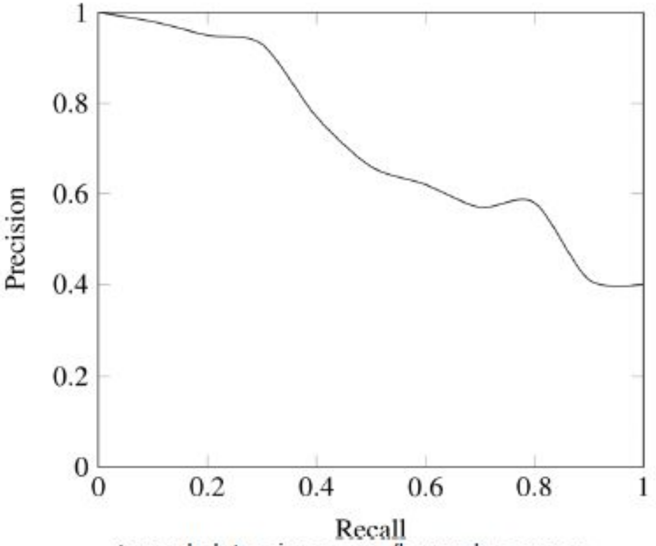


towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

F1 Score Vs Accuracy

- Accuracy does not perform well for imbalanced data sets
    - Assume we have 100 transaction, 90 are non-fraud cases and 10 fraud ones
    - High accuracy can be achieved by classifying every transaction as non-fraud
- Precision and Recall can give more insights
- F1 Score conveys the balance between the precision and recall.

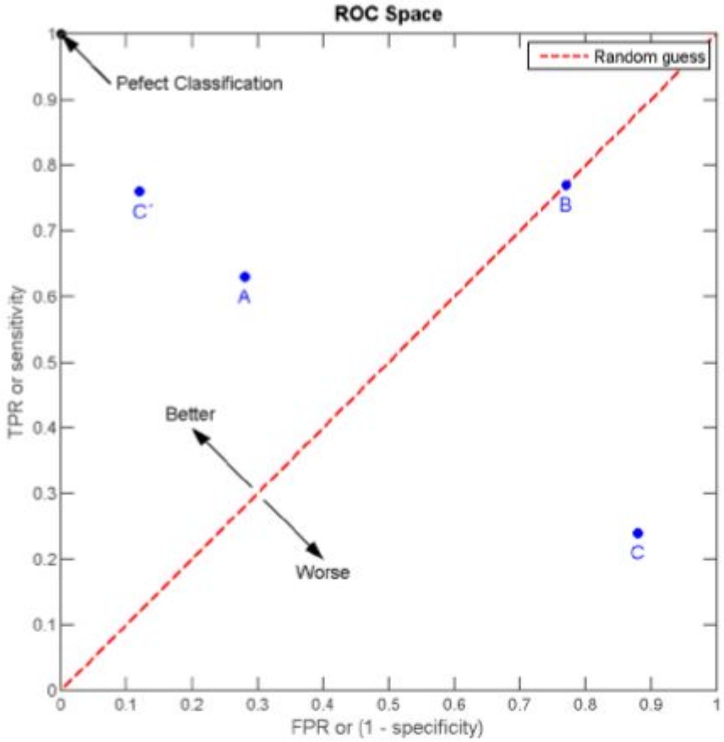
$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$



towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

Receiver Operation Characteristic

- Illustrates the diagnostic ability of a binary classifier as its discrimination threshold varies
- True positive rate (TPR) against the false positive rate (FPR) at various threshold



en.wikipedia.org/wiki/Receiver_operating_characteristic

Receiver Operation Characteristic

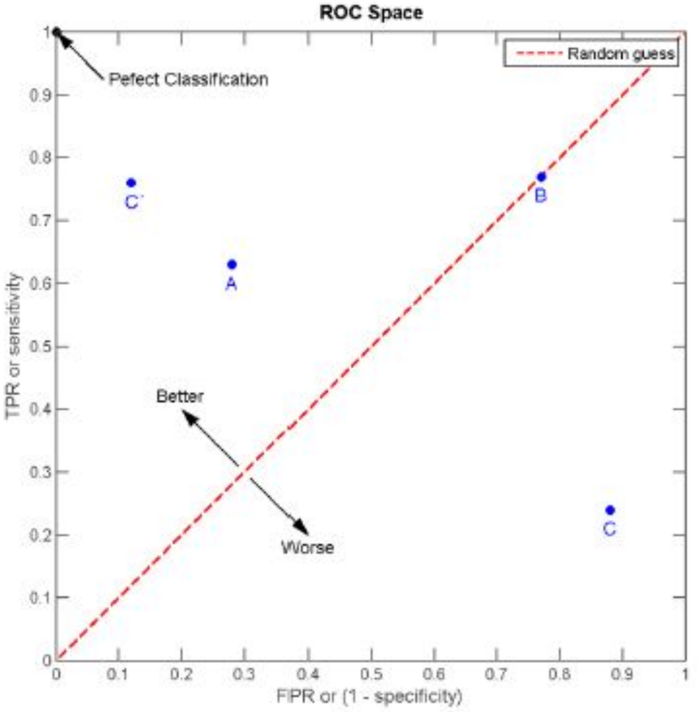
- **FPR**: Probability of False Alarm

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

- **TPR**: Probability of Detection

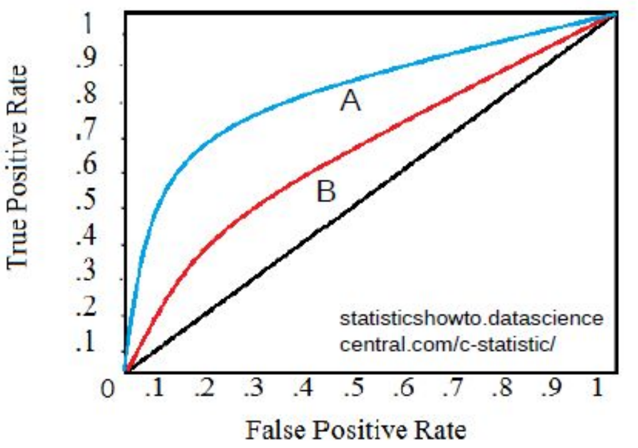
$$TPR = \frac{TP}{TP + FN}$$

- The best possible prediction method would yield a point in the upper left corner (0,1)
- The (0,1) point is also called a **perfect classification**
- A random guess would give a point along a diagonal



en.wikipedia.org/wiki/Receiver_operating_characteristic

Area Under the Curve



Area Under Curve (AUC) is the area under the ROC curve

AUC(A) > AUC(B)

Classifier A is better than Classifier B

- ROC curve plots parametrically TPR(T) versus FPR(T) with threshold T as the varying parameter
- AUC equals to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example
- AUC is one of the most widely used metrics for evaluation of binary classification problem

Multiclass Classification

Multiclass Classification

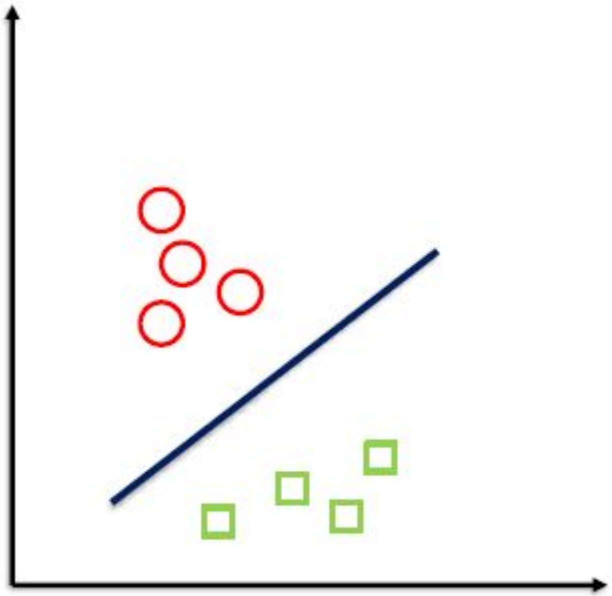
- Weather: Cloudy, Rain, Snow, ...

- Fruit: Apple, Orange, Peach, ...

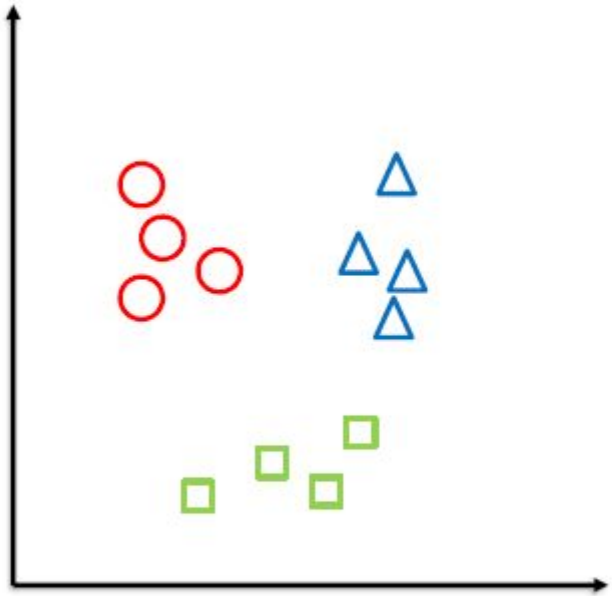
- Email tagging: Work, Ad, Friends, …

Multiclass Classification

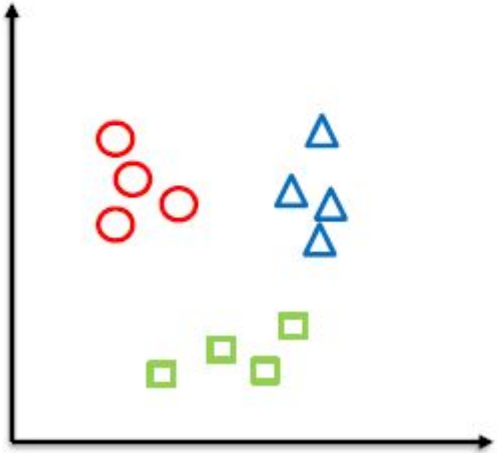
Binary classification

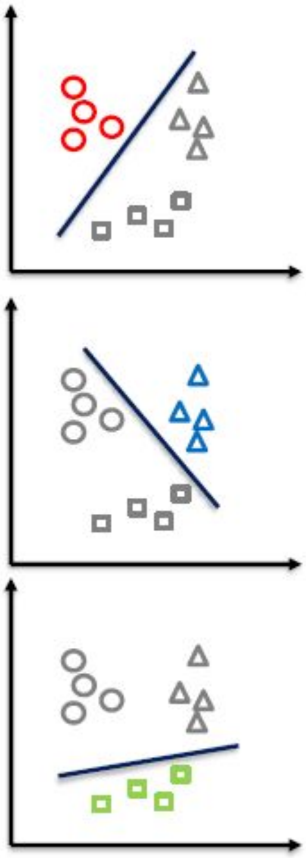


Multiclass classification



One-v.s.-All (One-v.s-Rest)





Train a LR classifier $p_i(y = 1|\boldsymbol{x})$ for each class i to predict the probability of $y = i$

$$i^* = \arg\max_i p_i(y = 1|\boldsymbol{x})$$

Softmax Classifier:

- Extend to 4-class classification
- Find 4 vectors w1, w2, w3, w4, such that
  - $P(C_1|x):P(C_2|x):P(C_3|x):P(C_4|x)$

$$= e^{w_1^T x} : e^{w_2^T x} : e^{w_3^T x} : e^{w_4^T x}$$

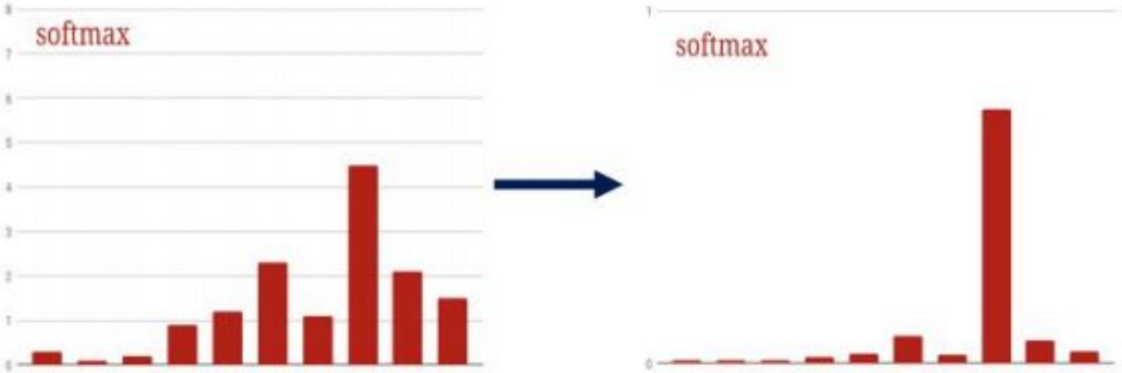
  - $P(C_1|x) + P(C_2|x) + P(C_3|x) + P(C_4|x) = 1$
  - $$P(C_1|x) = \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_2^T x} + e^{w_3^T x} + e^{w_4^T x}}$$

Softmax Function

Softmax Classifier:

- Extend from binary classification case
- Model the distribution of p(y=i|**x**), i=1,..., k, with softmax

$$P(y = i|\mathbf{x}) = \frac{e^{w_i^T x}}{\sum_j e^{w_j^T x}}$$

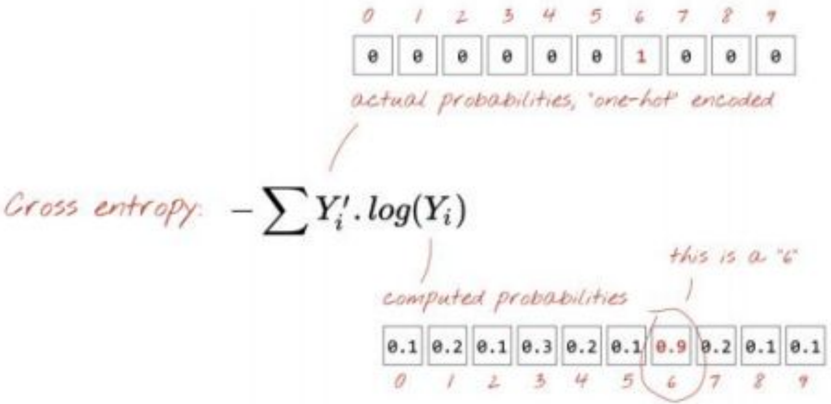


Softmax Classifier:

- The objective is to optimize cross entropy

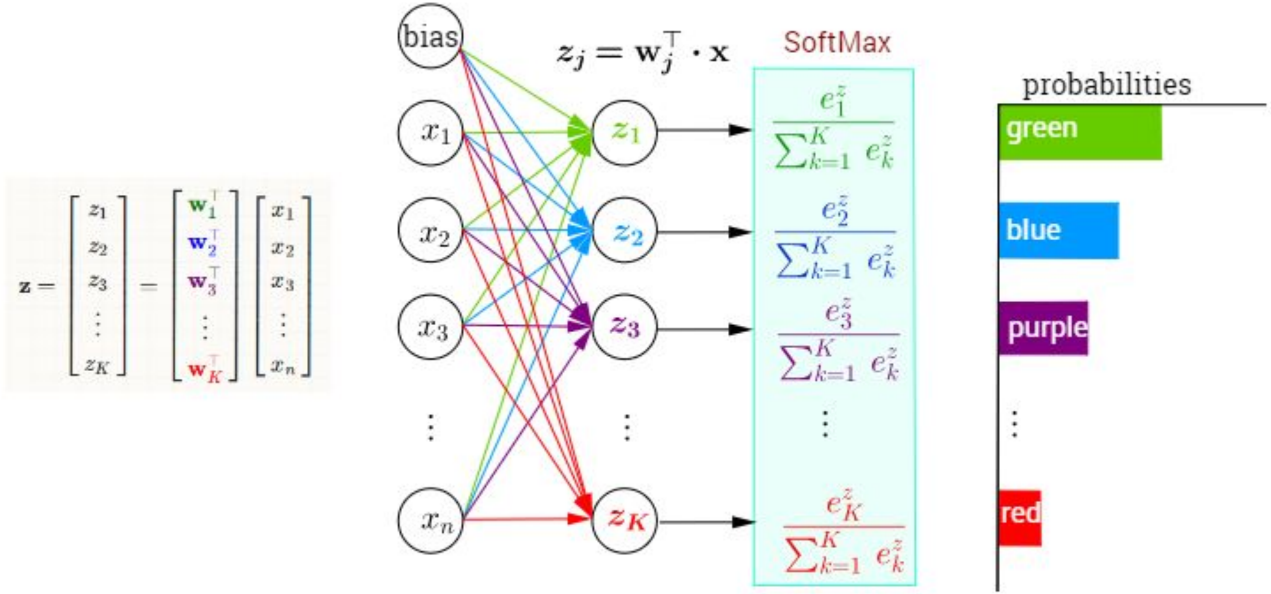
$$L(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik} logp(y=k|\mathbf{x})$$

Where tik=1 if yi=k, otherwise 0



github.com/GoogleCloudPlatform/tensorflow-without-a-phd

Softmax Classifier



https://stats.stackexchange.com/questions/265905/derivative-of-softmax-with-respect-to-weights

Generative vs Discriminative

For Classification

- Generative Approaches :
  - Given feature X and label Y, a generative model try to find the joint probability: P(X, Y)
  - How the data was generated?
  - From P(X,Y) -> P(Y|X), then categorize
  - Less Direct, More Probabilistic
- Discriminative Approaches :
  - Given feature X and label Y, a discriminative model try to find the joint probability: P(Y|X)
  - Distribution-free Approaches
  - Simply categorizes the data
  - More Direct, Less Probabilistic

Questions

- Generative and Discriminative?
  - Naive Bayes
  - Logistic Regression

Naive Bayes Model for Text Generation

- For the index of words in range(1, 2, 3, ….T)
  - Random sample the category h_i from p(h) or h_i is fixed
  - Sample the word from the distribution: $p(\boldsymbol{d}|h_i)$
- However, it does not consider the words' intrinsic dependency
  - E.g., Probability (read the paper) > Probability(read the movie)
  - The words at index T should depend on previous words (T-1, T-2, T-3,...)
- **Hidden Markov Model** ***partially*** solve the above issue