DATA MINING

/CISC 873 - Steven Ding
/Week #5/Lecture 1
Perceptron & SVM

Perceptron

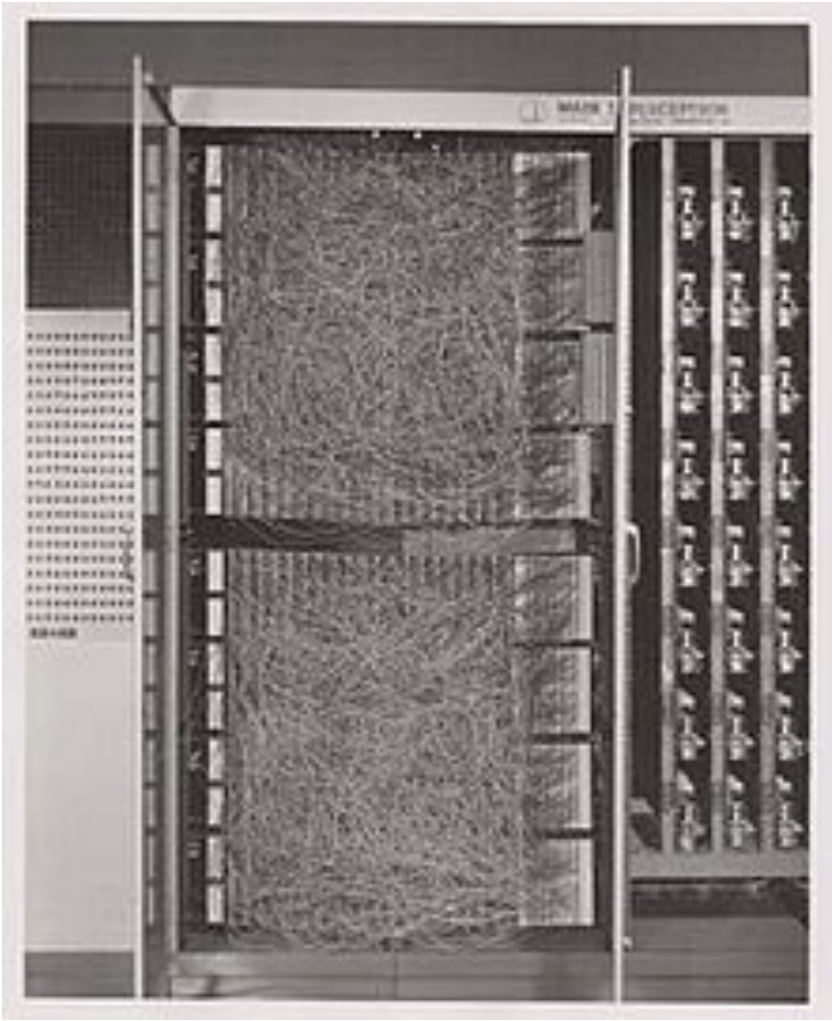
- Binary Classification:

$$x =< x_1, x_2, x_3, ..., x_m >$$
$$y \in \{-1, +1\}$$

- Perceptron:

$$h_w = \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$$
$$sign(z) = \begin{cases} +1, & \text{if } z \geq 1 \\ -1, & \text{otherwise} \end{cases}$$

Perceptron



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

Perceptron

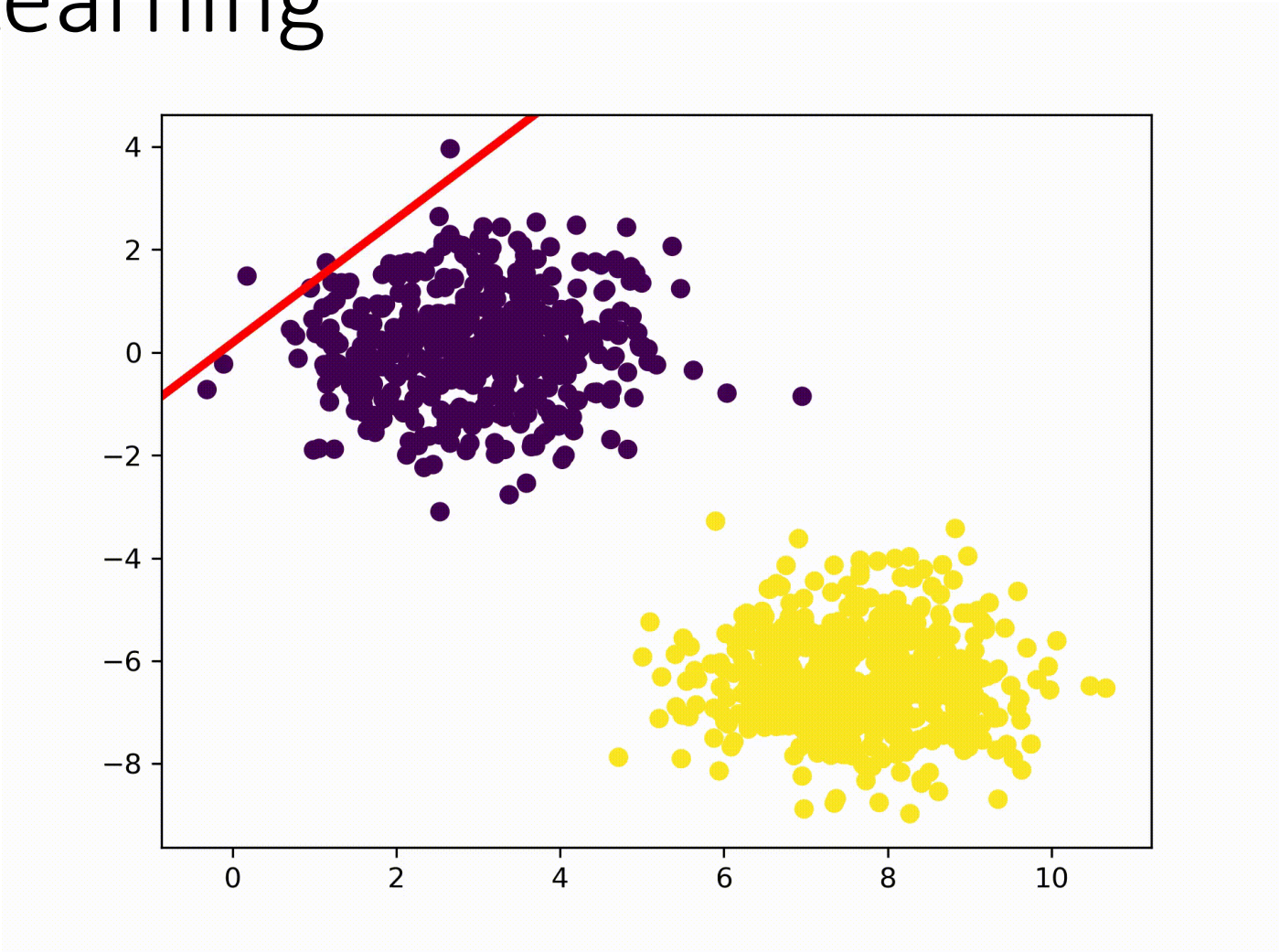
$$h_w = \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$$

$$sign(z) = \begin{cases} +1, & \text{if } z \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

- Training
  - For mis-classified positive sample, increase wx
  - For mis-classified negative sample, decrease wx
  - Loss:

$$\text{Err}(\boldsymbol{w}) = \sum_i^n \begin{cases} 0, & \text{if } y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 0 \\ -y_i \boldsymbol{w}^T \boldsymbol{x}_i, & \text{otherwise} \end{cases}$$

Learning

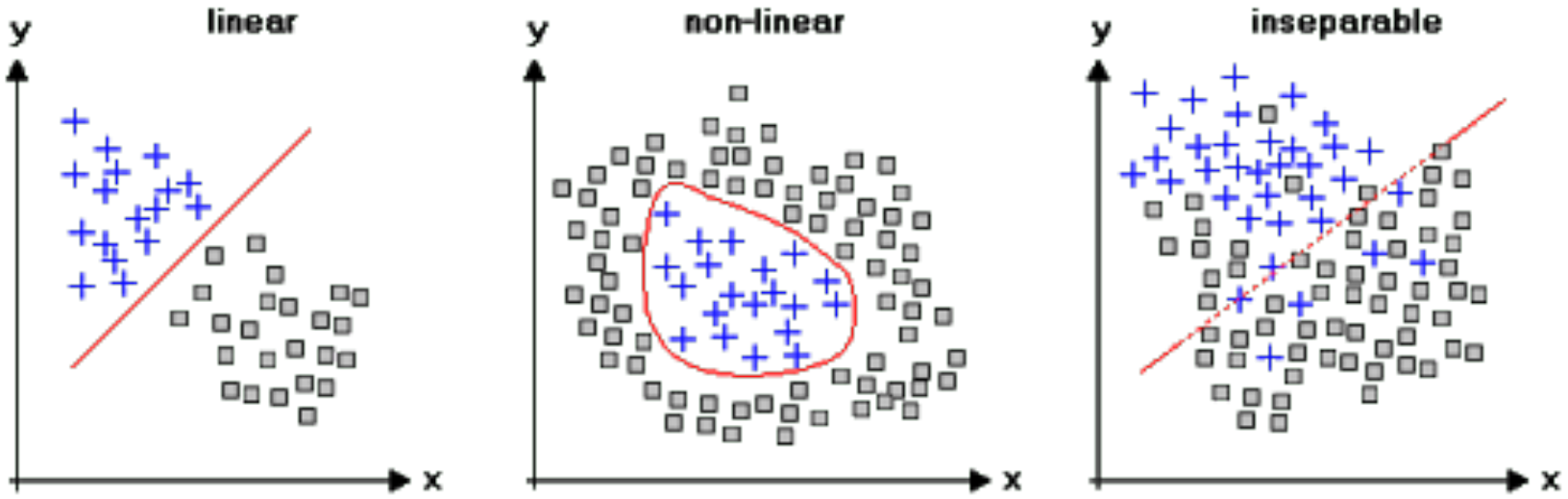


https://towardsdatascience.com/from-biology-to-ai-the-perceptron-81abfdc788bf

Linear Separable

- For any i of n, there exists such w that:

$$y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 0$$



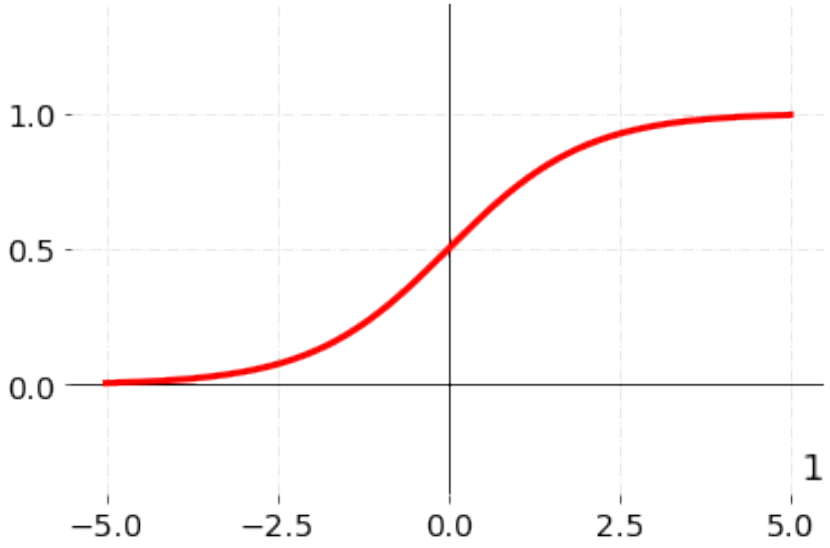
http://www.statistics4u.com/fundstat_eng/cc_data_structure.html

Perceptron convergence theorem

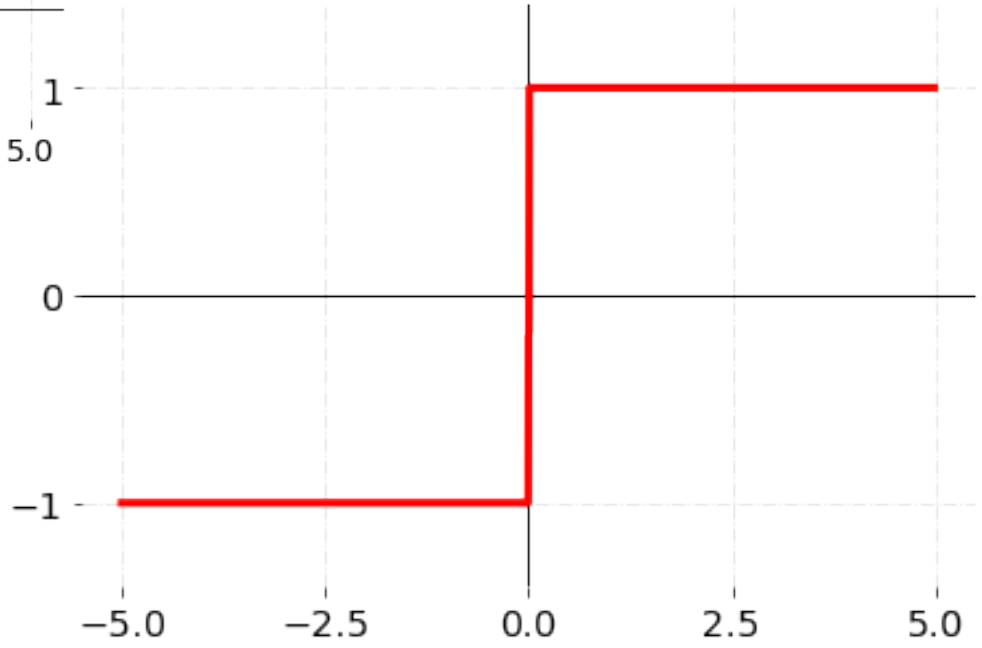
If the data set is linearly separable, a solution will be found after a finite number of iterations/updates.

- The number depends on many factors.
- Oscillation – if it is non-linearly separable.
- One solution: decreasing learning rate

Perceptron loss vs. logistic loss

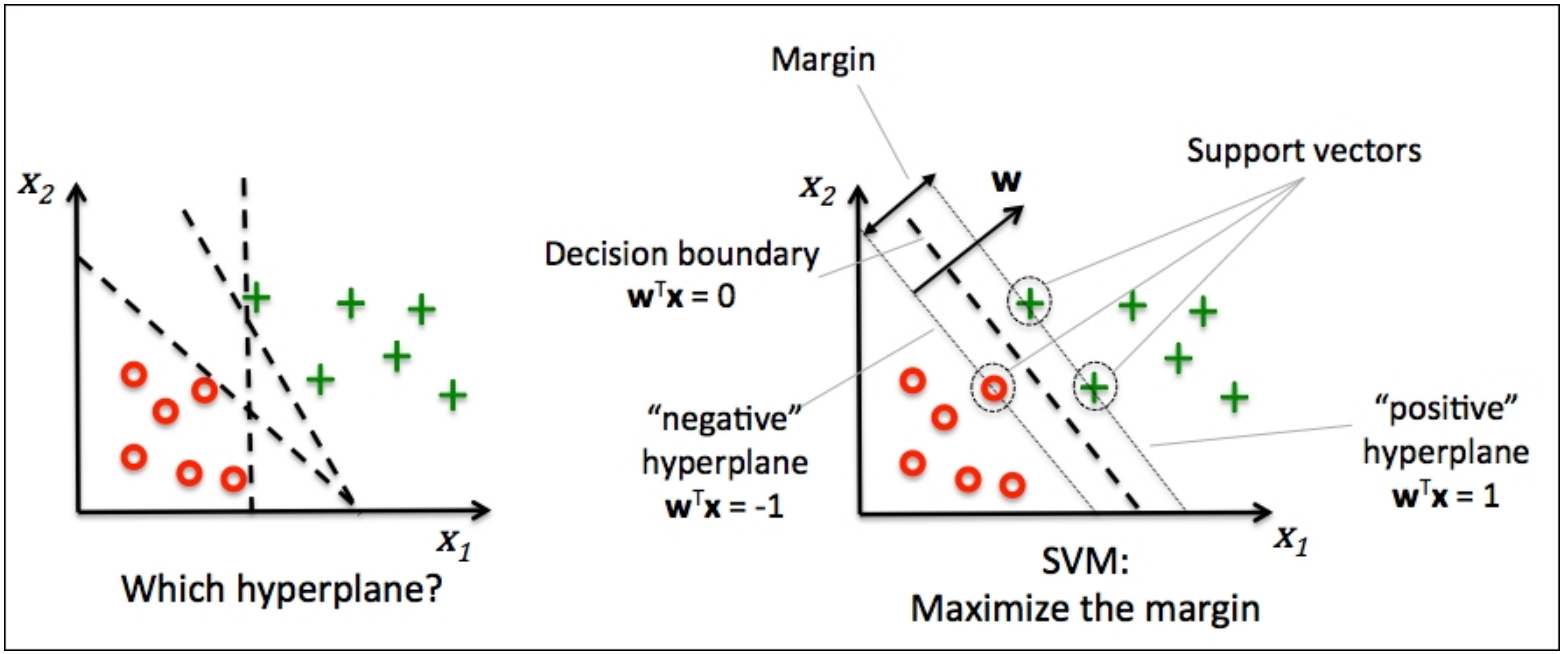


- Logistic – “smoothed” version



Issue

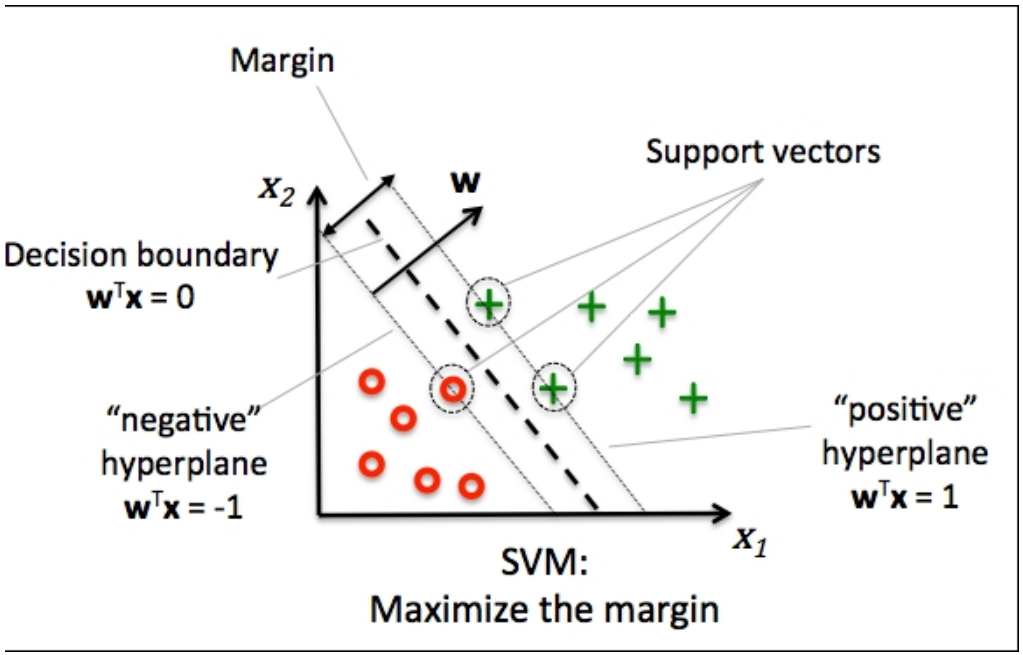
- Non-unique solutions/hyperplanes
  - Solution: max-margin

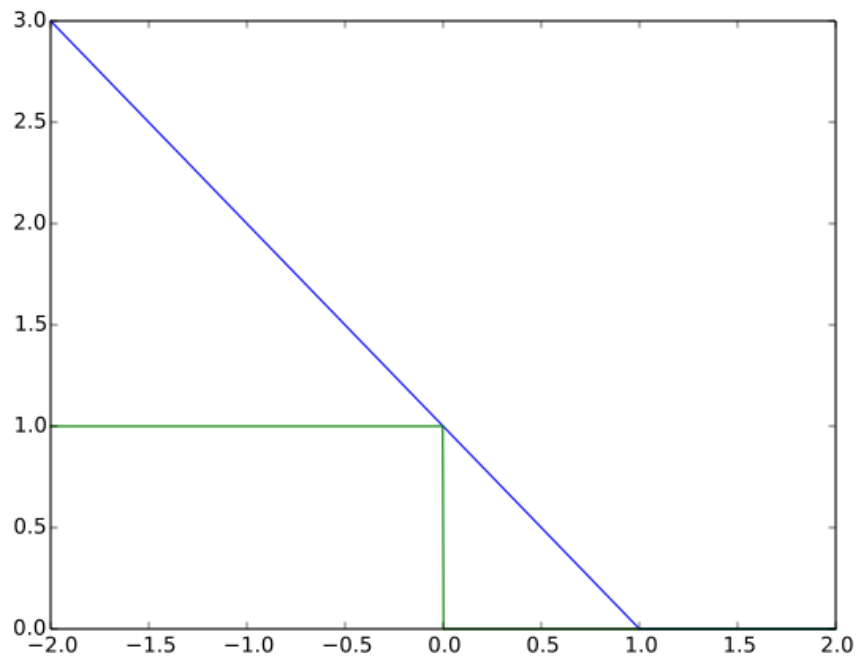


https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781783555130/3/ch03lvl1sec21/maximum-margin-classification-with-support-vector-machines

SVM Hing loss – Max Margin

$$\text{loss}(\boldsymbol{w}) = \sum_{i}^{n} \max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x})$$



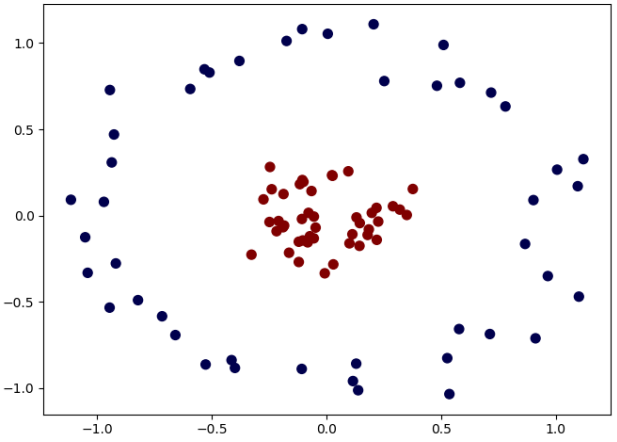


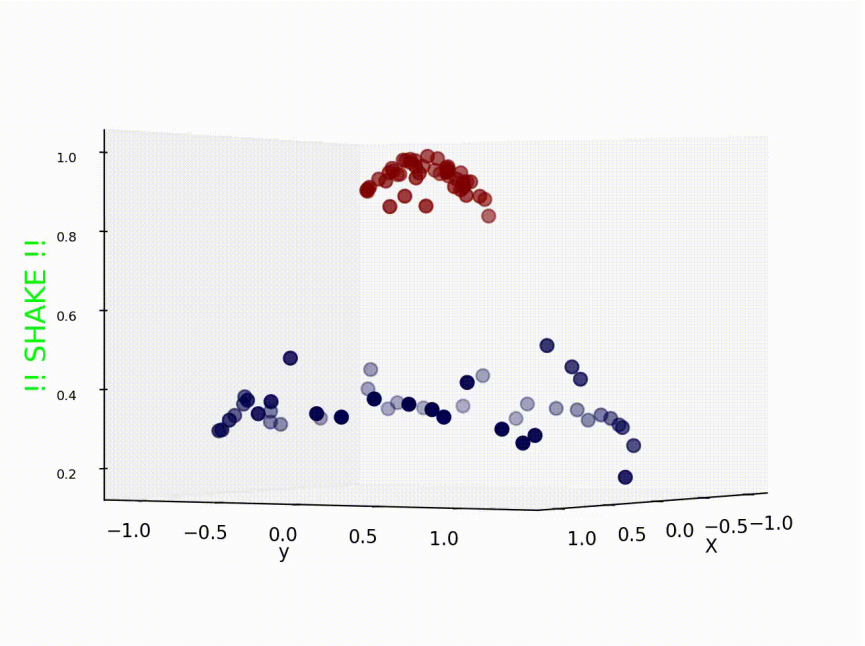
SVM Training

- Default: binary classification
  - For multi-class: but one-vs-all classifiers
- Constrained optimization
- Lagrange multipliers
- Primal optimization
- Dual optimization
- Need another 10+ slides
- [in-short, skipped]

Non-linear Optimization

- What happens if the data is not linearly separable?
  - Transform the space (introduce additional dimensions) to make it linearly separable.





https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d

Nonlinearity

- XOR?
  - Not linearly separable
    - Convexity (convex set)
- Feature Map
  - Aka basis function, kernels etc
  - a,b -> class
  - a,b,ab -> class
  - Now it is linearly separable
  - Difficult to find the right function
- Solution
  - Learn the mapping function from data!
  - Multi-layer perception

	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

	a	b	ab	class
1	0	0	0	0
2	0	1	0	1
3	1	0	0	1
4	1	1	1	0

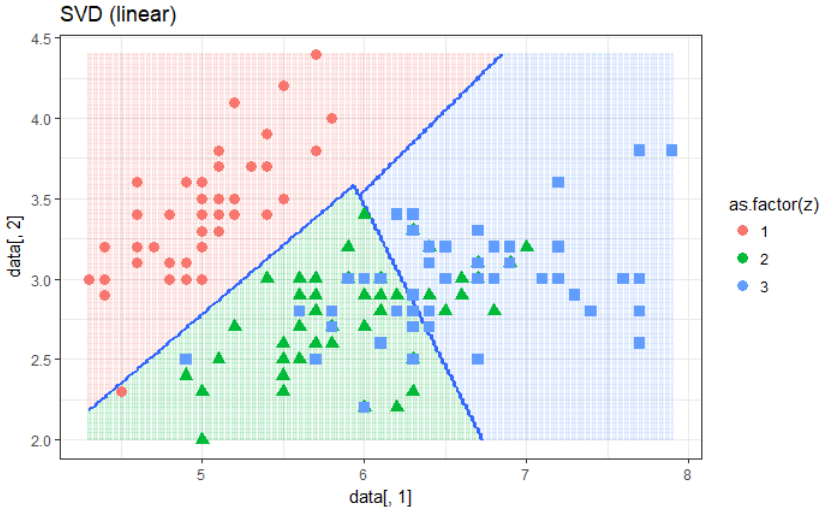
Kernel Functions

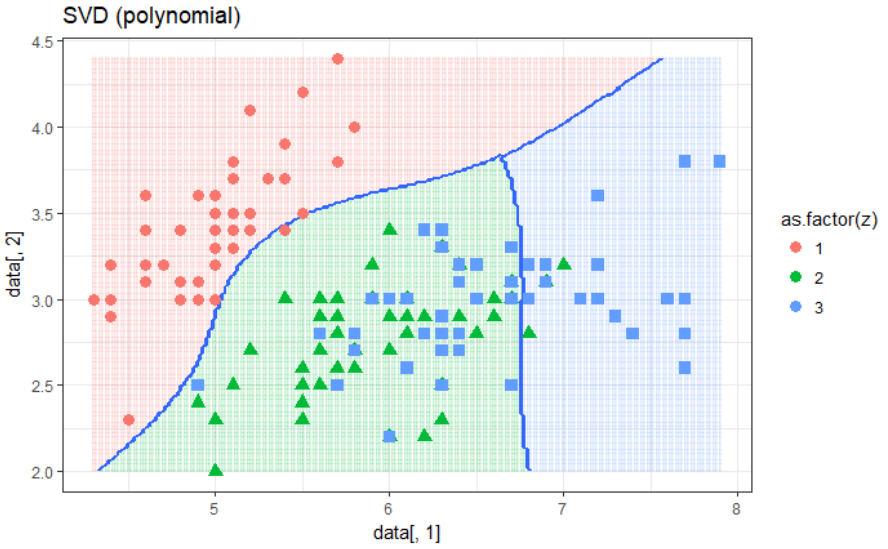
Kernel function	Expression	Parameter
Liner kernel function	$K(x_i, x_j) = x_i \cdot x_j$	
Polynomial kernel function	$K(x_i, x_j) = \left(x_i \cdot x_j + 1\right)^d$	d
Radial basis function (RBF) kernel function	$K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right)$	$\gamma > 0$
Sigmoid kernel function	$K\left(x_i, x_j\right) = \tanh\left(b\left(x_i, x_j\right) + c\right)$	b, c

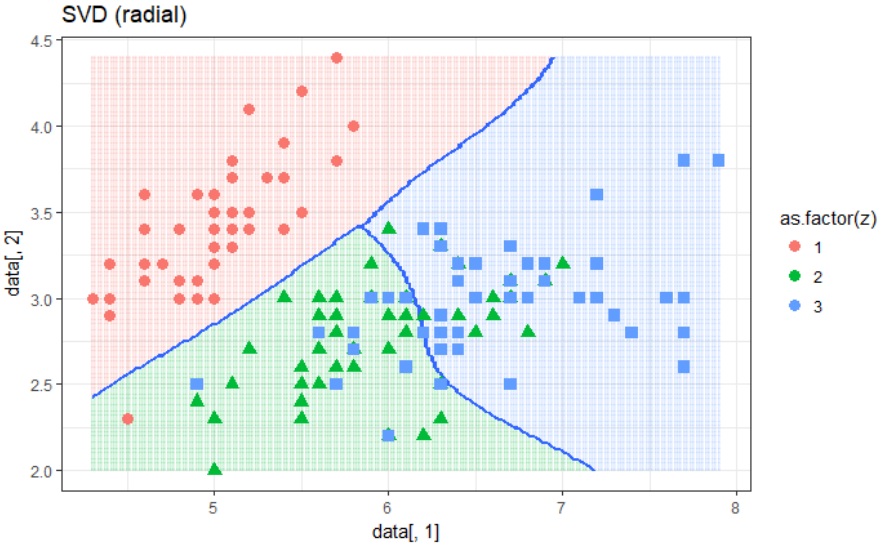
https://www.researchgate.net/figure/Common-kernel-function_tbl1_270033634

SVM Training

- Kernels





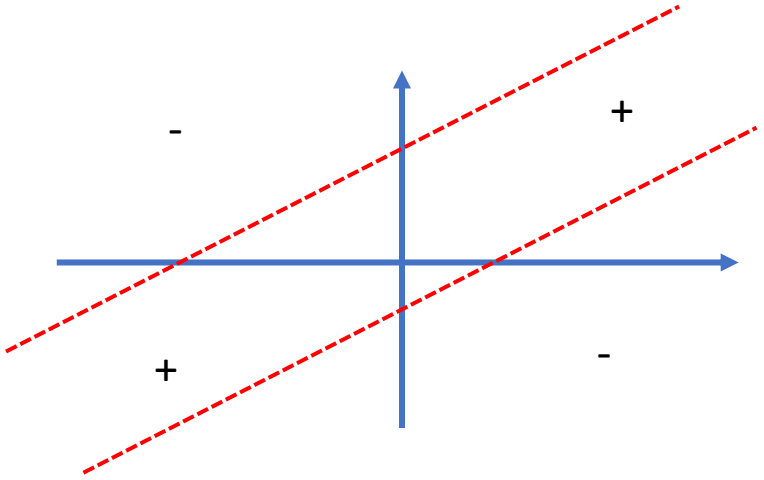


DATA MINING

/CISC 873 - Steven Ding
/Week #5/Lecture 2
Multi-layer Perceptron

Perceptron

- Binary classification on linearly separable data
  - Non-linear? Add kernel function.
  - But any other options?
- One perceptron models -> one linear hyperplane
- Two perceptron models -> two linear hyperplane
  - Two hyperplanes can model a non-linear problem
    - For example, XOR



Nonlinearity

- XOR?
  - Not linearly separable
    - Convexity (convex set)
- Feature Map
  - Aka basis function, kernels etc
  - a,b -> class
  - a,b,ab -> class
  - Now it is linearly separable
  - Difficult to find the right function
- Solution
  - Learn the mapping function from data!
  - Multi-layer perception

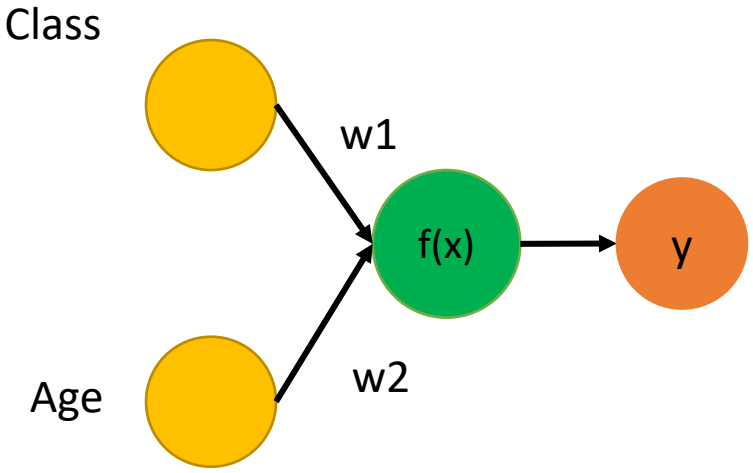
	a	b	class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

	a	b	ab	class
1	0	0	0	0
2	0	1	0	1
3	1	0	0	1
4	1	1	1	0

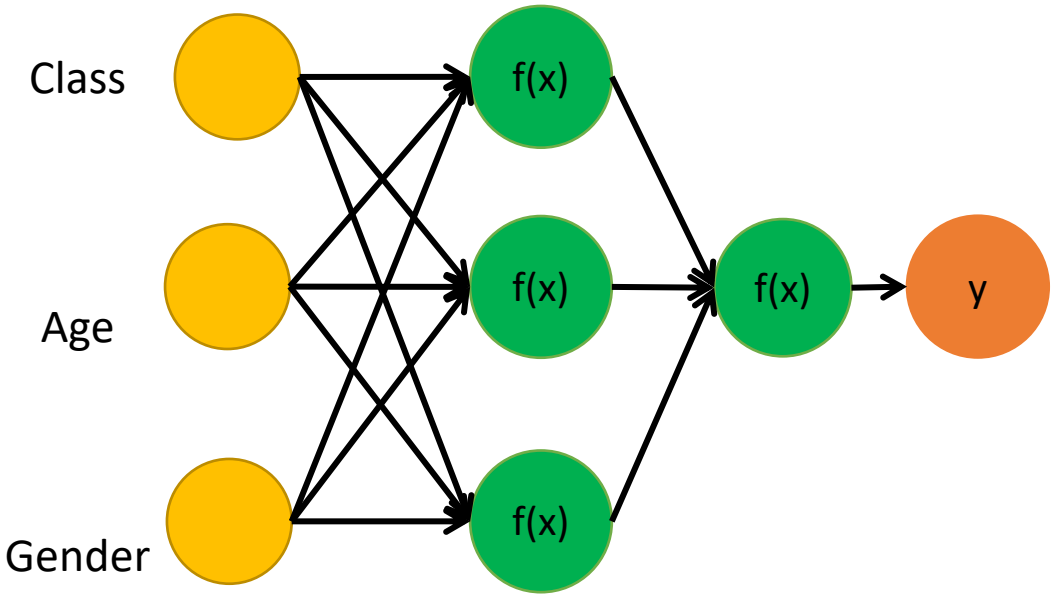
Neural Network

Passenger Class	Gender	age	Survived
1	1	29	1
1	2	2	0
2	2	21	1
2	1	19	1

- Added multiple layer of interconnected regression node.
- Following the same way of training.



A logistic regression model.

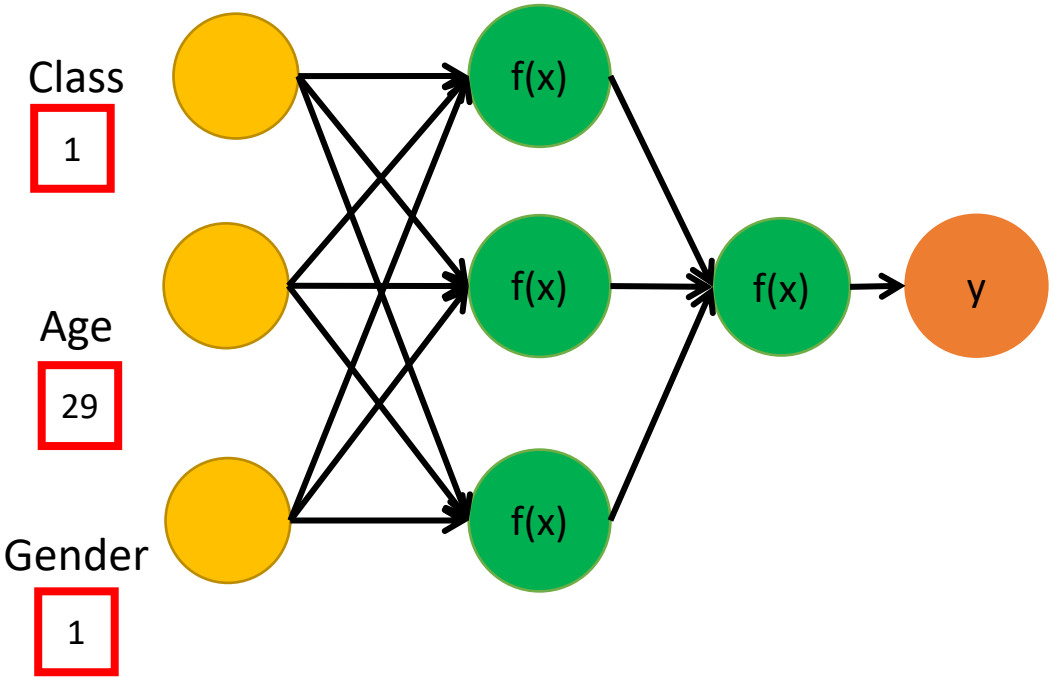


A neural network model.

Neural Network

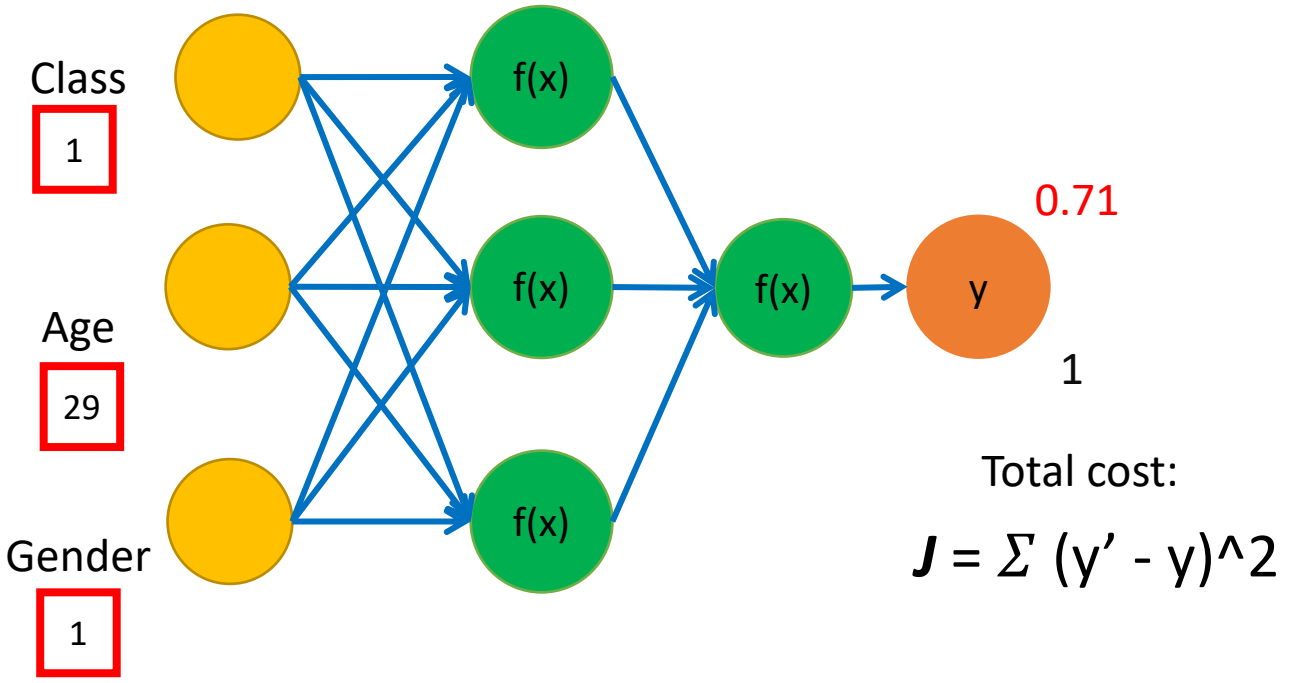
Passenger Class	Gender	age	Survived
1	1	29	1
1	2	2	0
2	2	21	1
2	1	19	1

Randomly initialize weights.



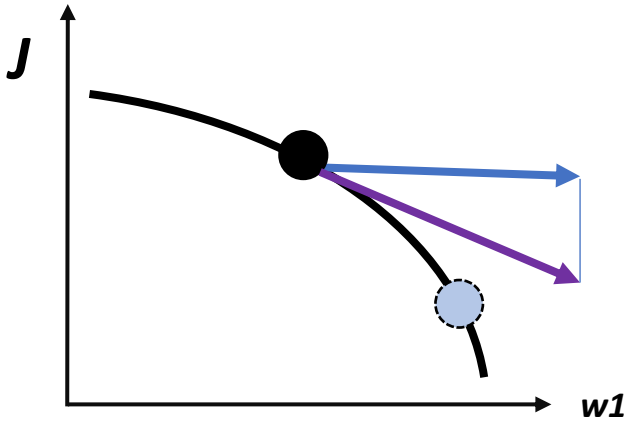
Neural Network

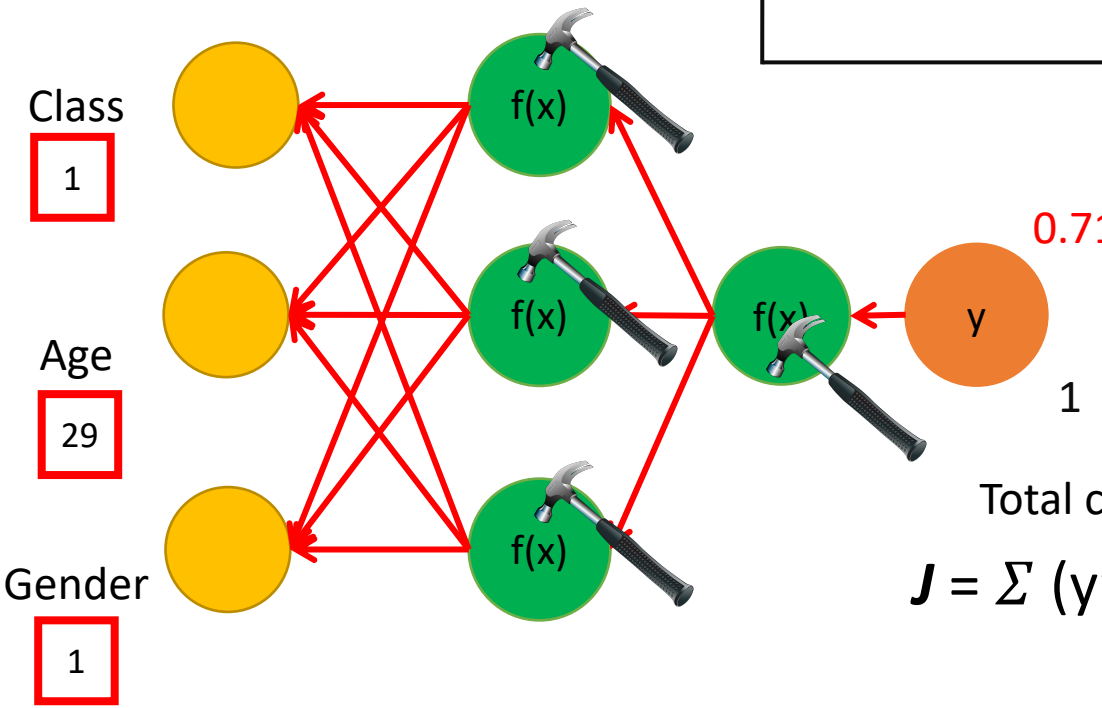
Passenger Class	Gender	age	Survived
1	1	29	1
1	2	2	0
2	2	21	1
2	1	19	1



Neural Network

Passenger Class	Gender	age	Survived
1	1	29	1
1	2	2	0
2	2	21	1
2	1	19	1



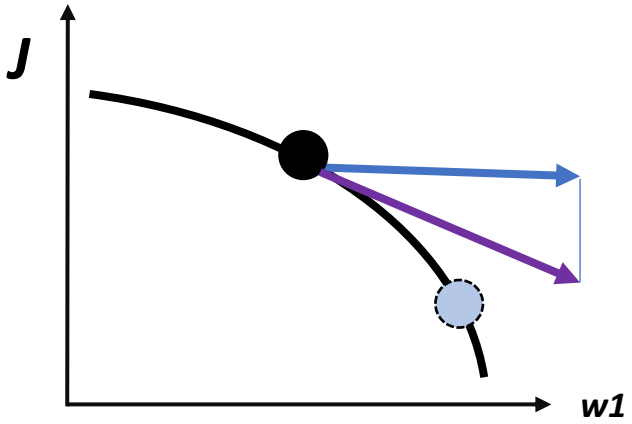


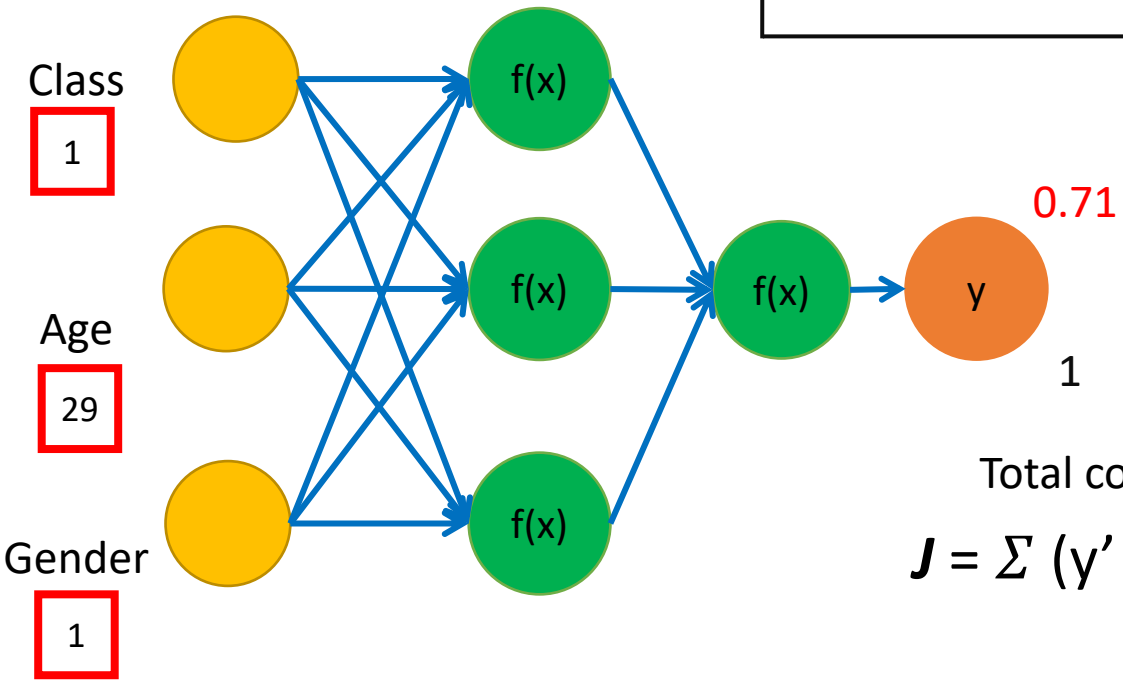
Total cost:

$J = \Sigma\ (y' - y)^2$

Neural Network

Passenger Class	Gender	age	Survived
1	1	29	1
1	2	2	0
2	2	21	1
2	1	19	1



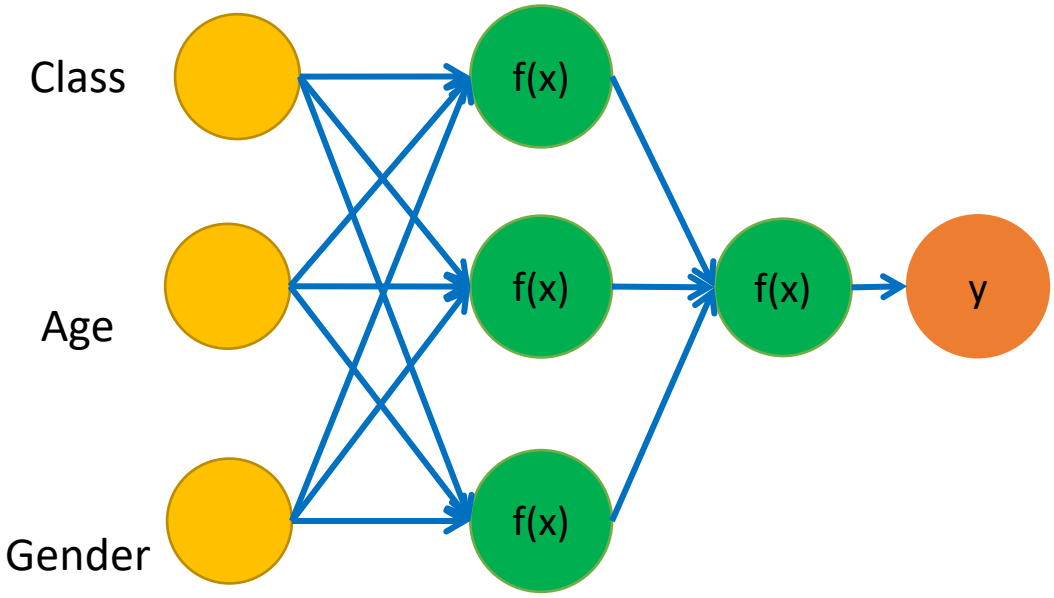


Total cost:

$J = \Sigma\ (y' - y)^2$

Neural Network

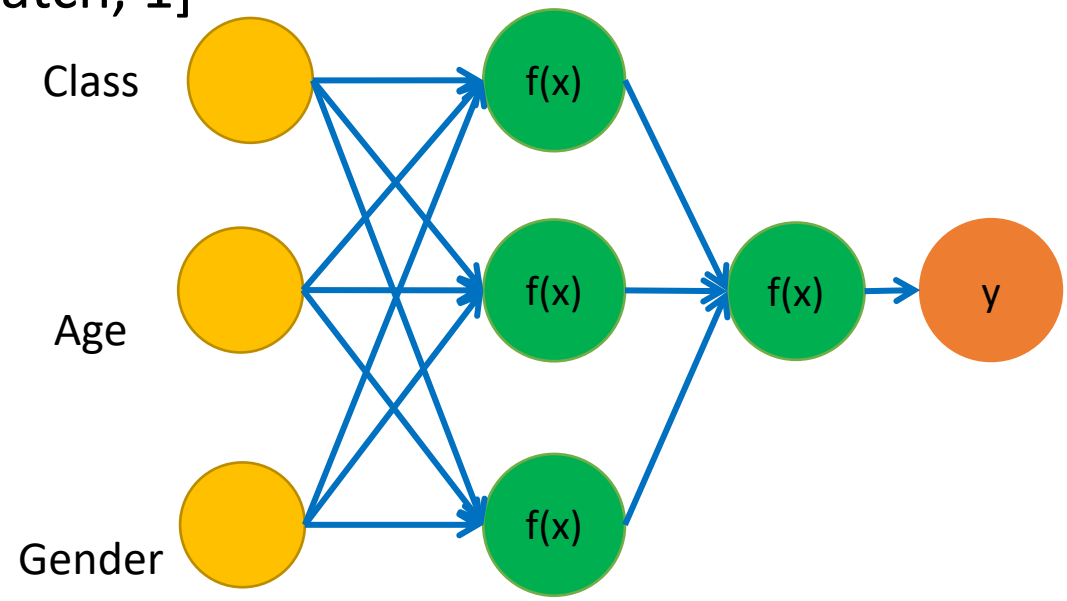
- Matrix/tensor notation (important)
- Vector (dim = 1)
- Matrix (dim = 2)
- Tensor (dim > 2)



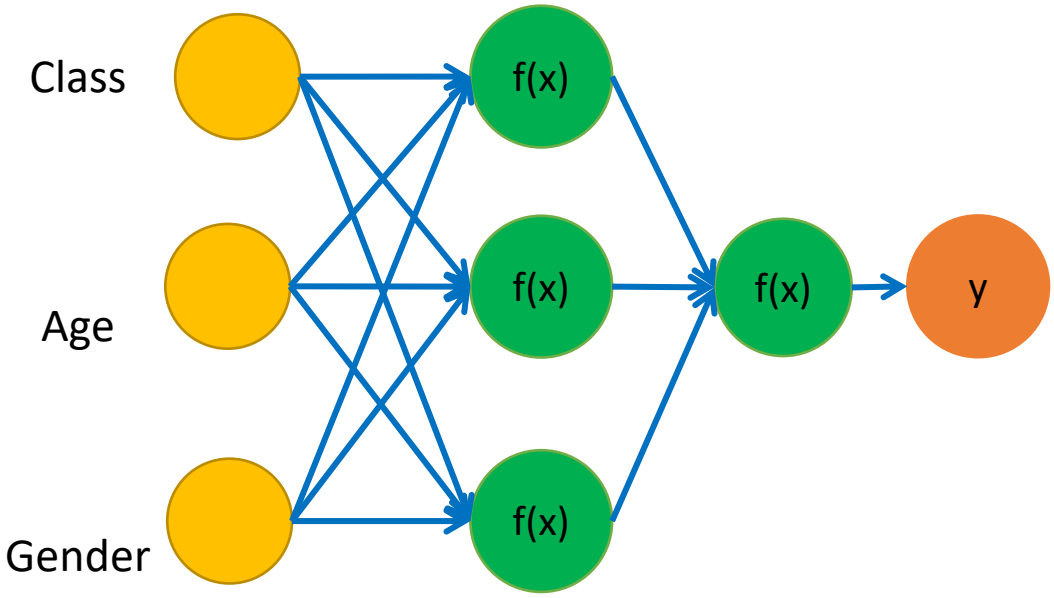
- Recall: we feed mini-batch of data rather than a single sample.

Neural Network

- Input
  - a matrix of shape [batch, 3]
- After first layer (transform into three units)
  - a matrix of shape [batch, 3]
- After the last layer
  - A matrix of shape [batch, 1]



Neural Network




```
1 from keras.layers import Dense, Input
2 from keras import Model
3
4 x = Input(shape=[None, 3], batch_size=True)
5 h0 = Dense(3, activation='sigmoid')(x)
6 y = Dense(1, activation='sigmoid')(h0)
7
8 model = Model(inputs=x, outputs=y)
9
10 model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

More Layers?

```
1 from keras.layers import Dense, Input
2 from keras import Model
3
4 x = Input(shape=[None, 3], batch_size=True)
5 h0 = Dense(64,  activation='sigmoid')(x)
6 h1 = Dense(128, activation='sigmoid')(h0)
7 h2 = Dense(128, activation='sigmoid')(h1)
8 h3 = Dense(128, activation='sigmoid')(h2)
9 h4 = Dense(32,  activation='sigmoid')(h3)
10 y  = Dense(1,   activation='sigmoid')(h4)
11
12 model = Model(inputs=x, outputs=y)
13
14 model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Multi-Layer Perception (MLP)

- A class of feedforward neural network
  - Characterized by multiple layers of linear activation functions, to model the non-linear data through interactions.

```
from keras.layers import Dense, Input
from keras import Model

x = Input(shape=[None, 3], batch_size=True)
h0 = Dense(64,  activation='sigmoid')(x)
h1 = Dense(128, activation='sigmoid')(h0)
h2 = Dense(128, activation='sigmoid')(h1)
h3 = Dense(128, activation='sigmoid')(h2)
h4 = Dense(32,  activation='sigmoid')(h3)
y  = Dense(1,   activation='sigmoid')(h4)

model = Model(inputs=x, outputs=y)

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```