Nonlinear Transformation

Wenlu Zhang

CECS@CSULB

Readings for Chapter 3

Excluding: 3.2.2; 3.4.2 (Generalization part)

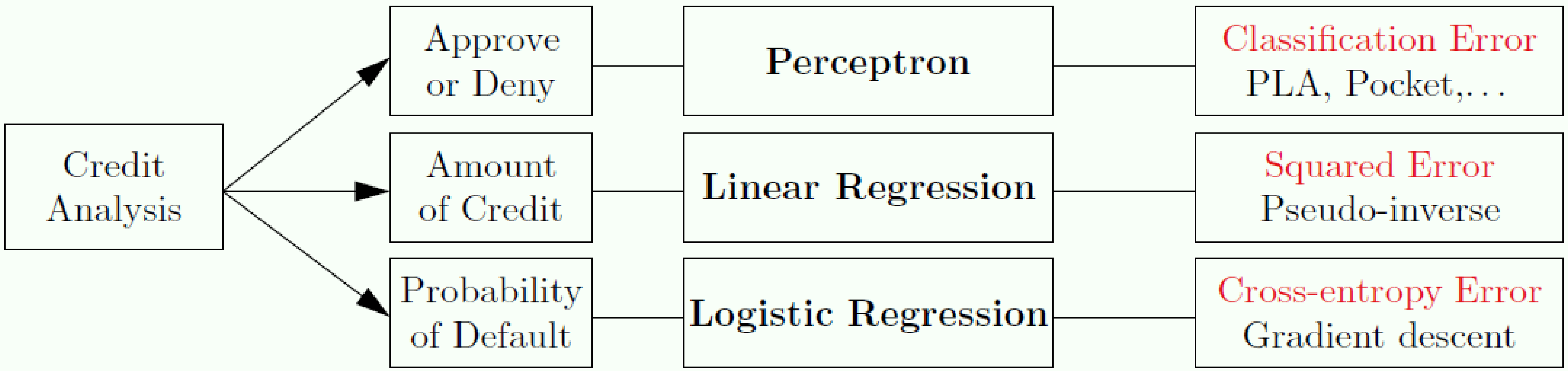
Ignore other discussions of generalization, VC dimension

RECAP: The Linear Model

linear in $\mathbf{x}$: gives the line/hyperplane separator

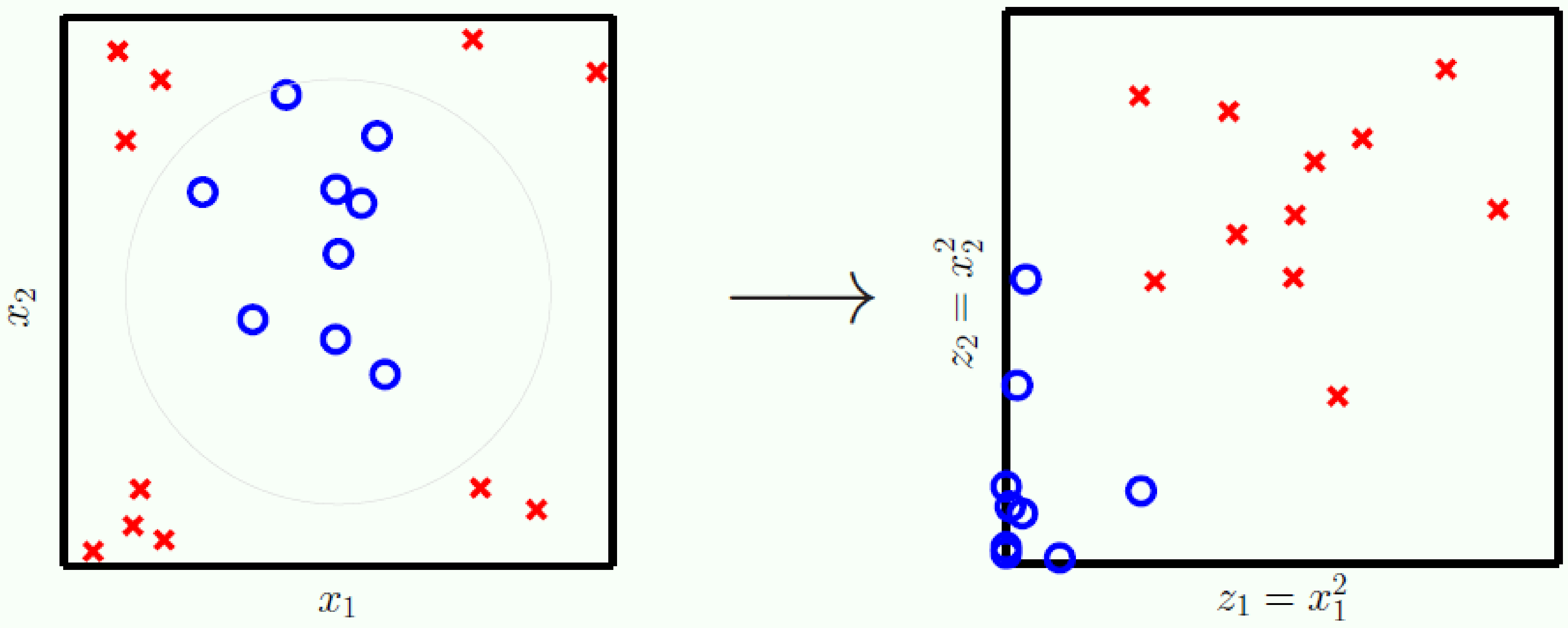
$$s = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

linear in $\mathbf{w}$: makes the algorithms work



Mechanics of the Feature Transform I

Transform the data to a $\mathcal{Z}$-space in which the data is separable.

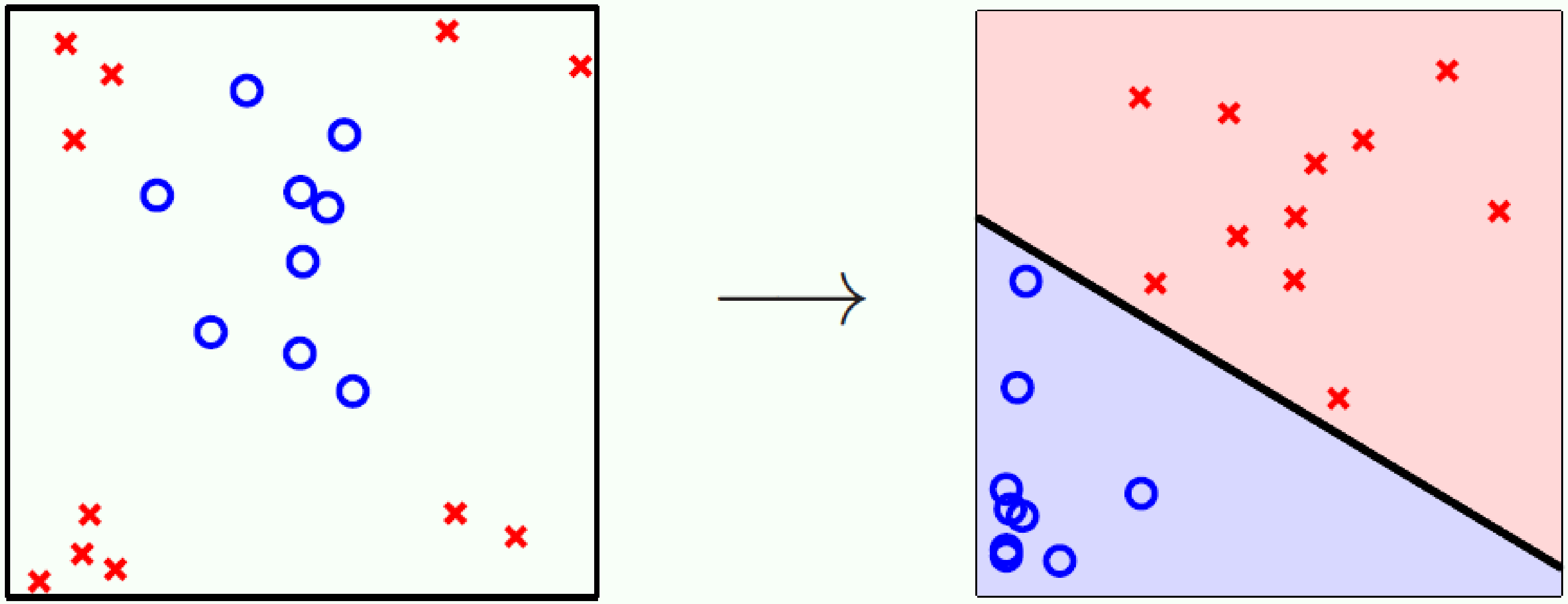


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \longrightarrow \mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \end{bmatrix}$$

Mechanics of the Feature Transform II

Separate the data in the $\mathcal{Z}$-space with $\tilde{\mathbf{w}}$:

$$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{z})$$

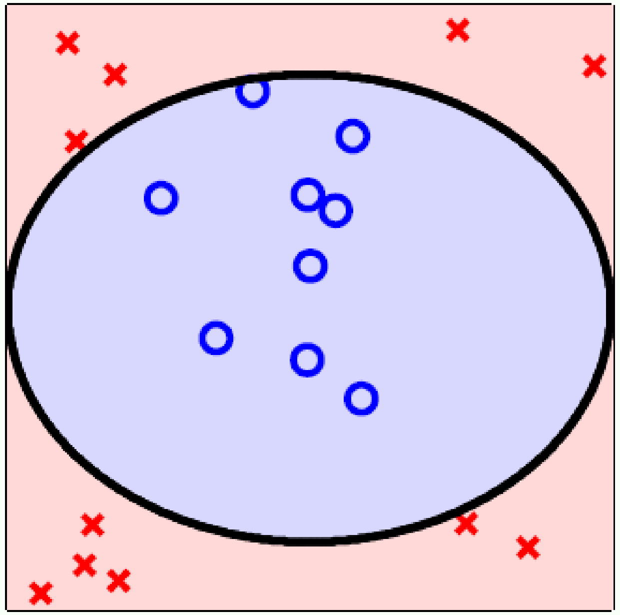


Mechanics of the Feature Transform III

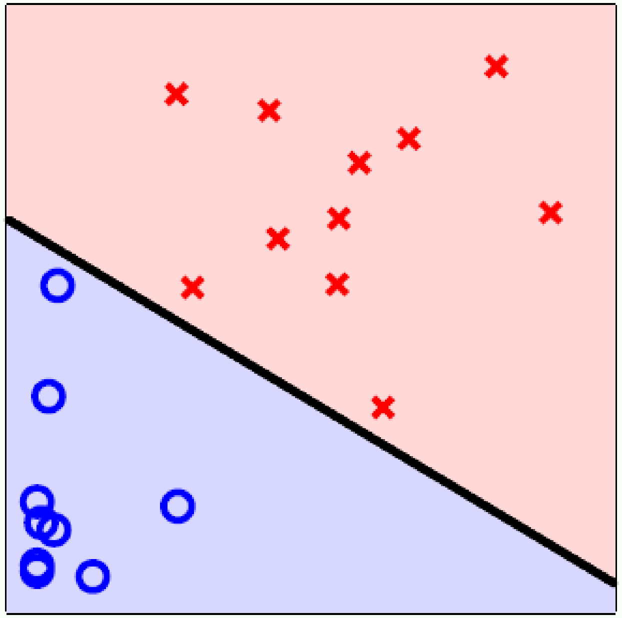
To classify a new $\mathbf{x}$, first transform $\mathbf{x}$ to $\mathbf{\Phi}(\mathbf{x}) \in \mathcal{Z}$-space and classify there with $\tilde{g}$.

$$\begin{aligned} g(\mathbf{x}) &= \tilde{g}(\mathbf{\Phi}(\mathbf{x})) \\ &= \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{x})) \end{aligned}$$

$$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{z})$$



$\longleftarrow$



The General Feature Transform

<u>$\mathcal{X}$-space is $\mathbb{R}^d$</u>

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

$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$

$y_1, y_2, \ldots, y_N$

no weights

$g(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{x}))$

<u>$\mathcal{Z}$-space is $\mathbb{R}^{\tilde{d}}$</u>

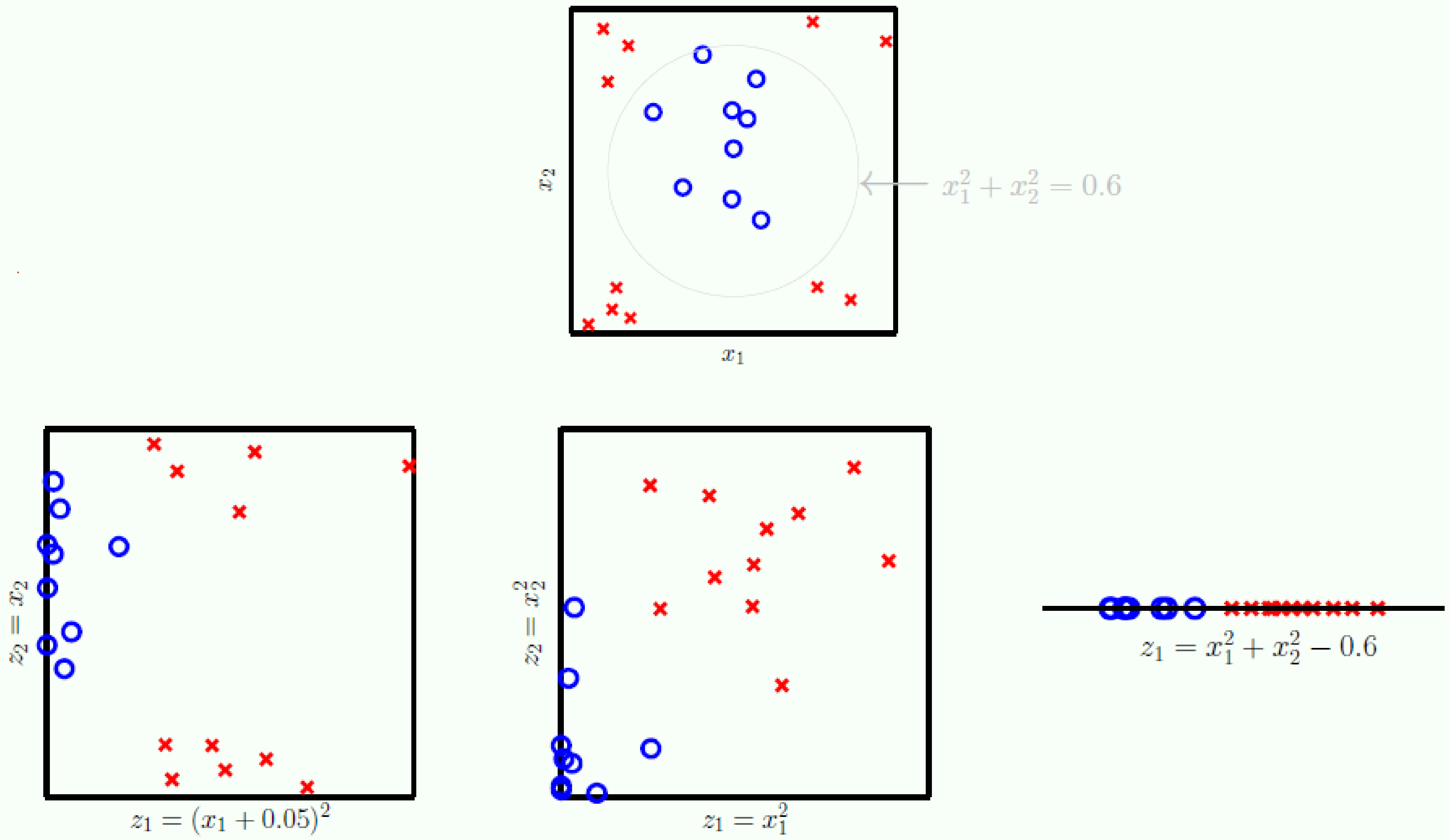
$$\mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_{\tilde{d}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ \vdots \\ z_{\tilde{d}} \end{bmatrix}$$

$\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$

$y_1, y_2, \ldots, y_N$

$$\tilde{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{\tilde{d}} \end{bmatrix}$$

Many Nonlinear Features May Work



A rat! A rat!

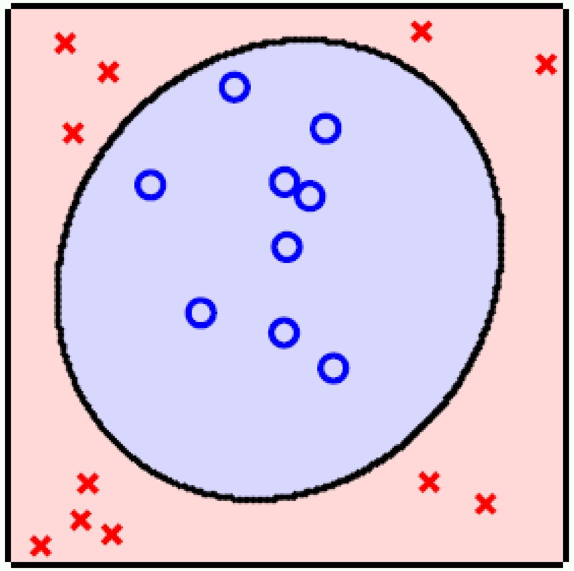
This is called data snooping: looking at your data and tailoring your $\mathcal{H}$.

Must Choose Φ BEFORE Your Look at the Data

After constructing features carefully, **before** seeing the data ...

...if you think linear is not enough, try **the 2nd order polynomial transform**.

$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = \mathbf{x} \longrightarrow \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \\ \Phi_3(\mathbf{x}) \\ \Phi_4(\mathbf{x}) \\ \Phi_5(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}$$



The General Polynomial Transform Φ_k

We can get even fancier: degree-k polynomial transform:

$$\begin{aligned}
\mathbf{\Phi}_1(\mathbf{x}) &= (1, x_1, x_2), \\
\mathbf{\Phi}_2(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2), \\
\mathbf{\Phi}_3(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3), \\
\mathbf{\Phi}_4(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4), \\
&\vdots
\end{aligned}$$

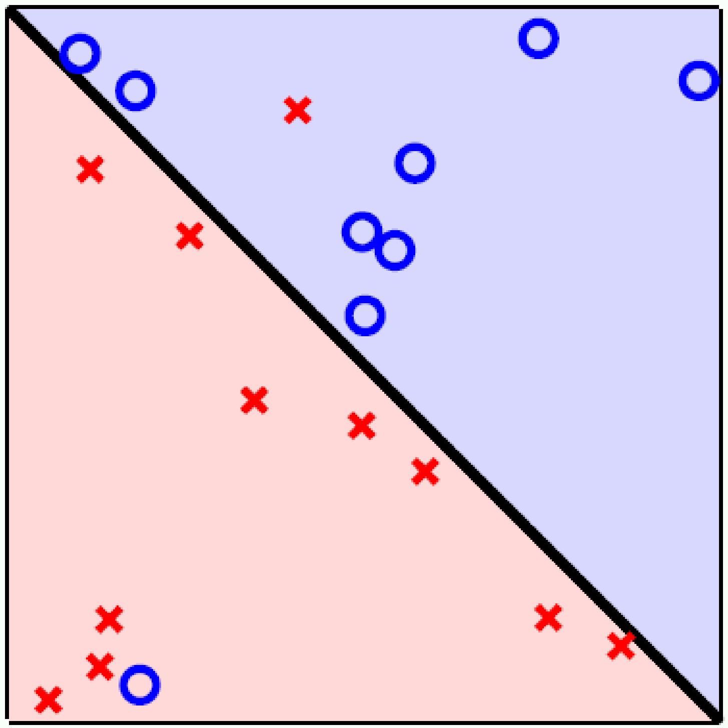
– Dimensionality of the feature space increases rapidly (d_{VC})!

– Similar transforms for d-dimensional original space.

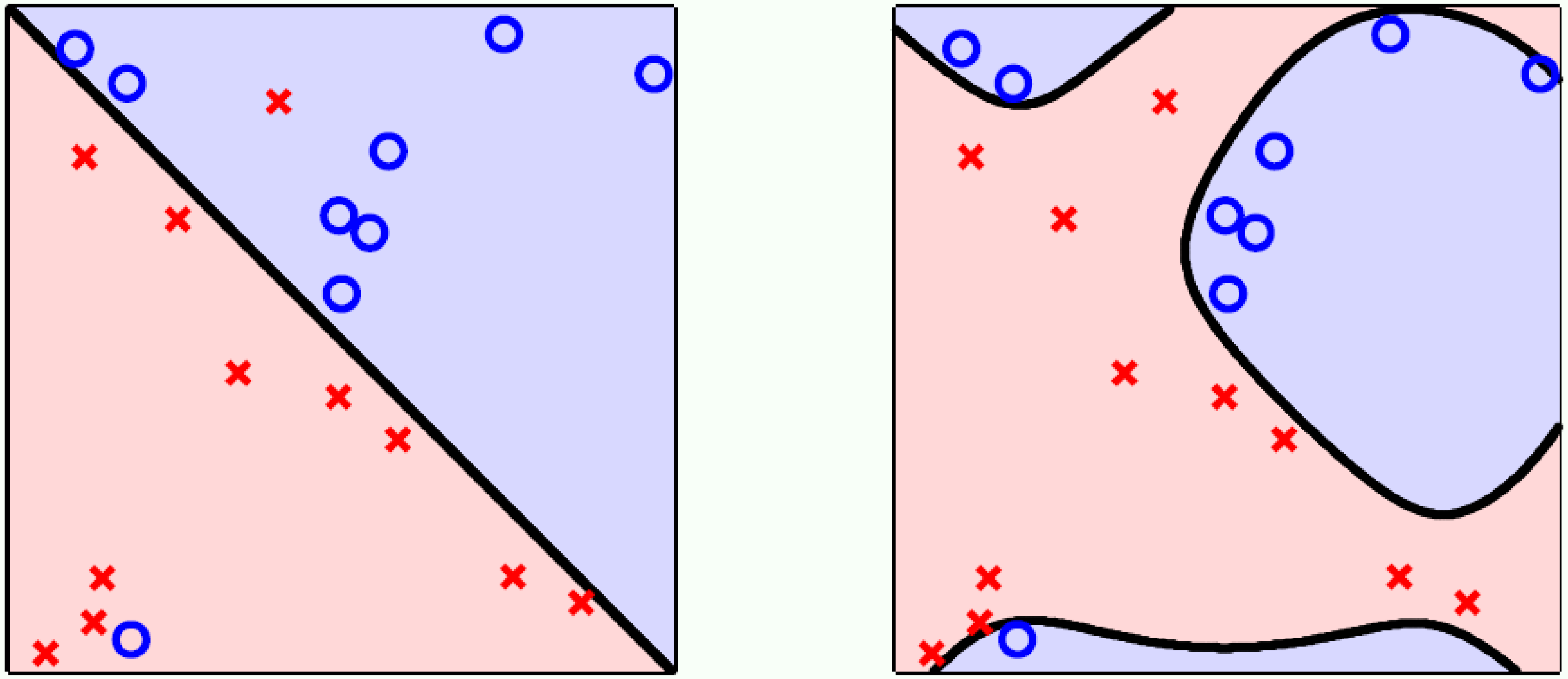
– Approximation-generalization tradeoff

Higher degree gives lower (even zero) E_{in} but worse generalization.

Be Careful with Feature Transforms



Be Careful with Feature Transforms



High order polynomial transform leads to “nonsense”.

Use the Linear Model!

- First try a linear model – simple, robust and works.

- Algorithms can tolerate error plus you have nonlinear feature transforms.

- Choose a feature transform *before* seeing the data. Stay simple.

  Data snooping is hazardous to your E_{out}.

- Linear models are fundamental in their own right; they are also the building blocks of many more complex models like support vector machines.

- Nonlinear transforms also apply to regression and logistic regression.

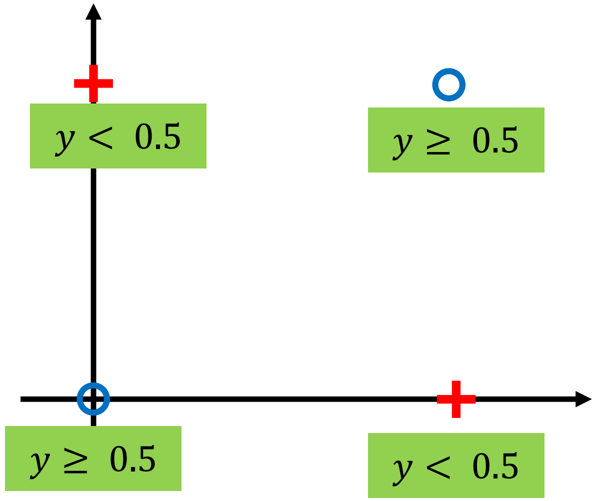
Limitation of Logistic Regression

Data		Target
x_1	x_2	t
0	0	1
0	1	0
1	0	0
1	1	1

Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

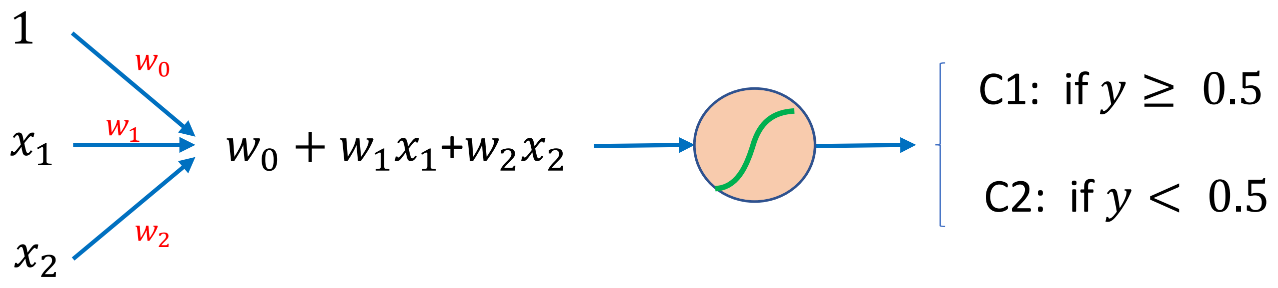
Data		Target
x_1	x_2	t
0	0	1
0	1	0
1	0	0
1	1	1

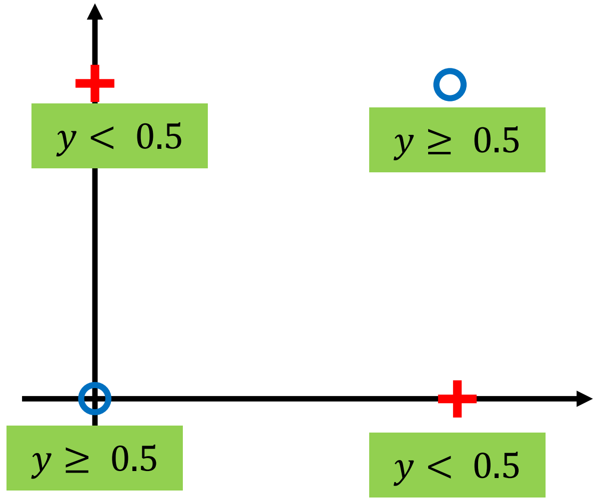


Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

Data		Target
x_1	x_2	t
0	0	1
0	1	0
1	0	0
1	1	1

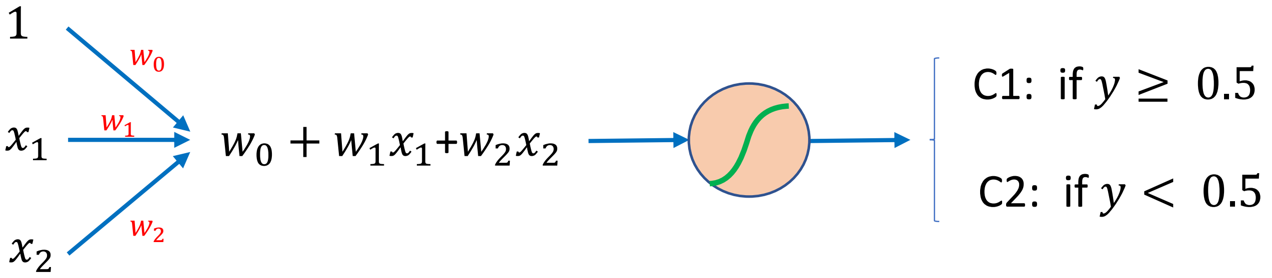


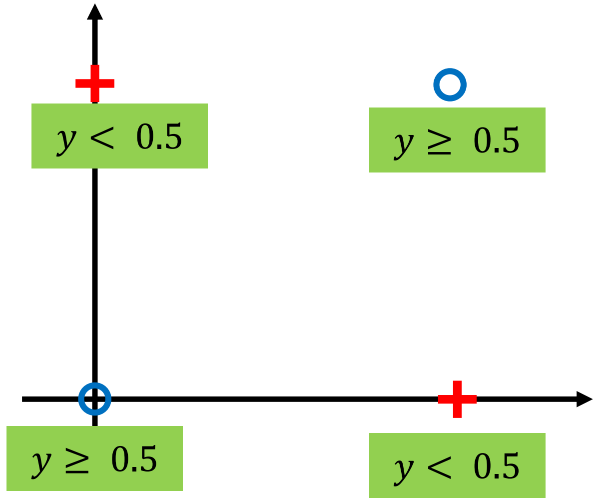


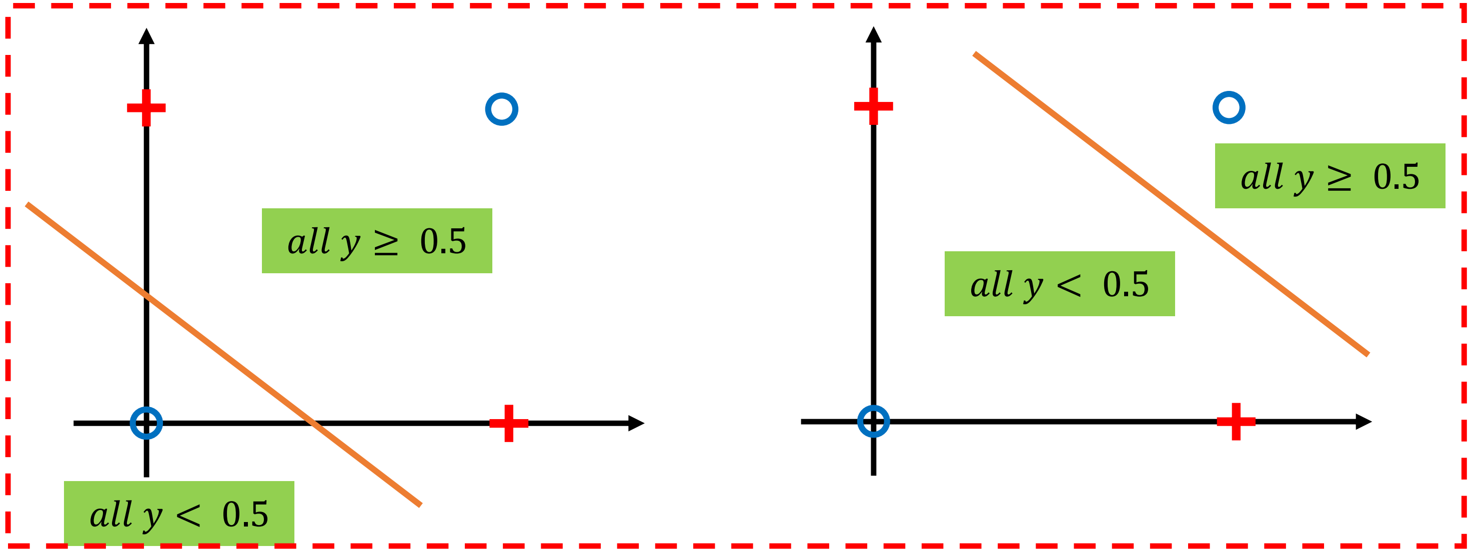
Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

Data		Target
x_1	x_2	t
0	0	1
0	1	0
1	0	0
1	1	1



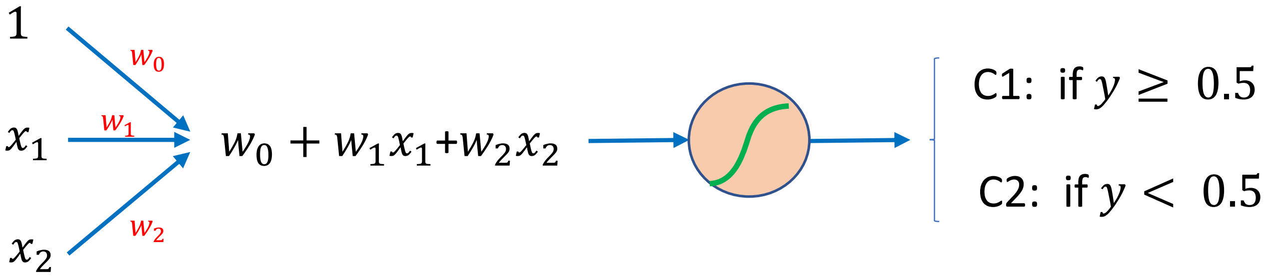




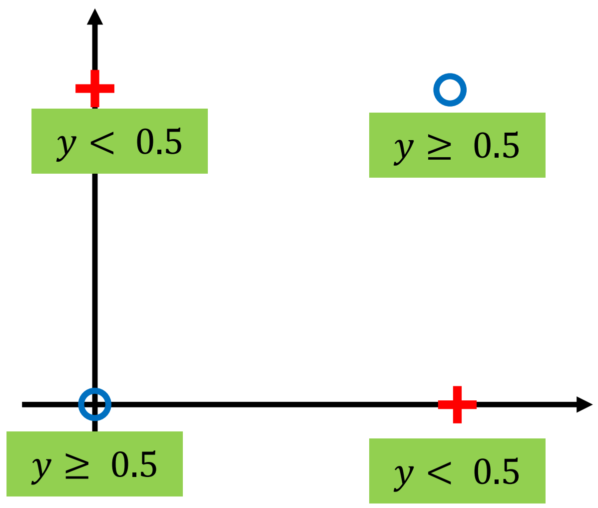
Courtesy of Dr. Hung-yi Lee

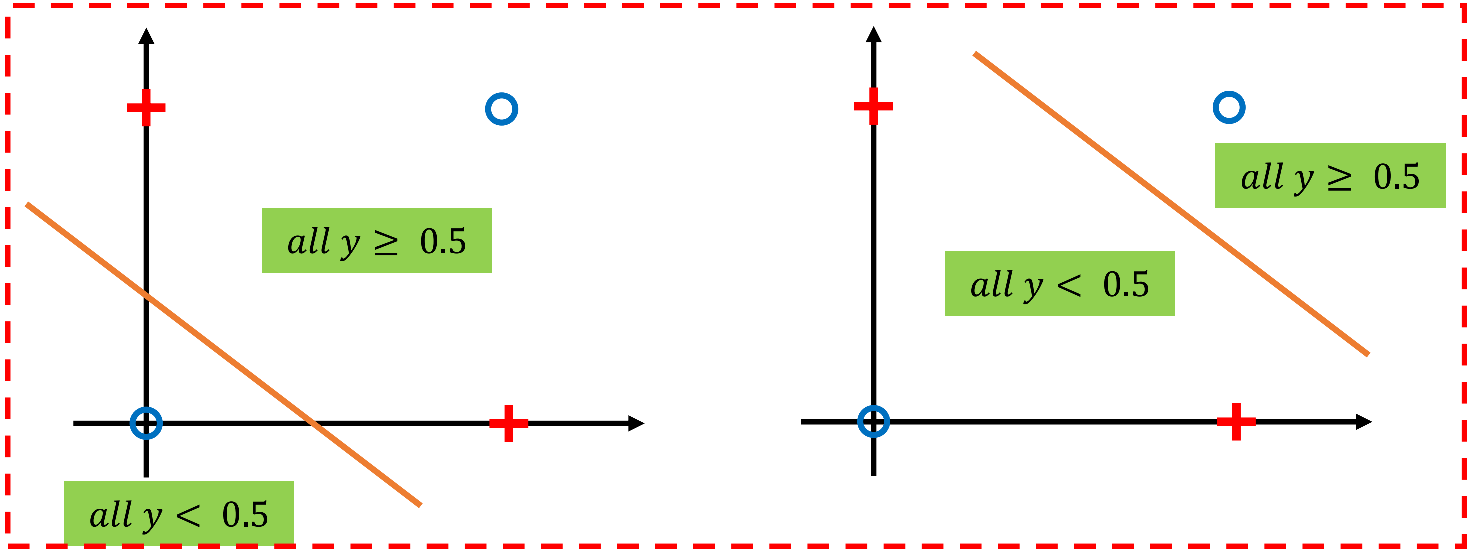
Limitation of Logistic Regression

Data		Target
x_1	x_2	t
0	0	1
0	1	0
1	0	0
1	1	1



We can't separate them well using a simple logistic regression (**not linearly separable**)



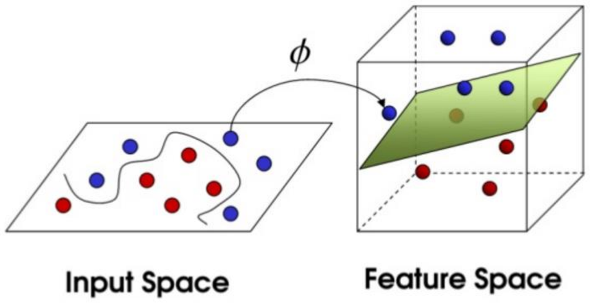


Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

Feature Transformation

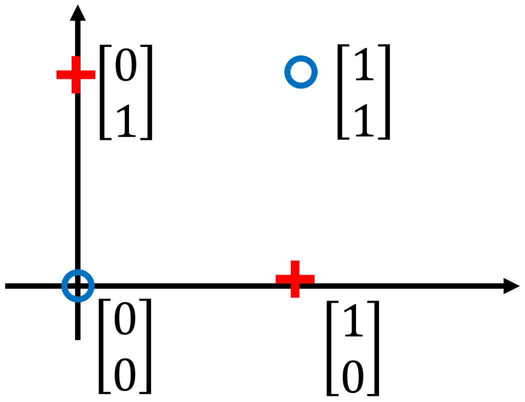
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

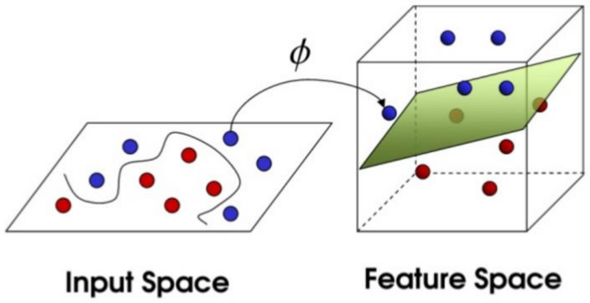
https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression



Feature Transformation

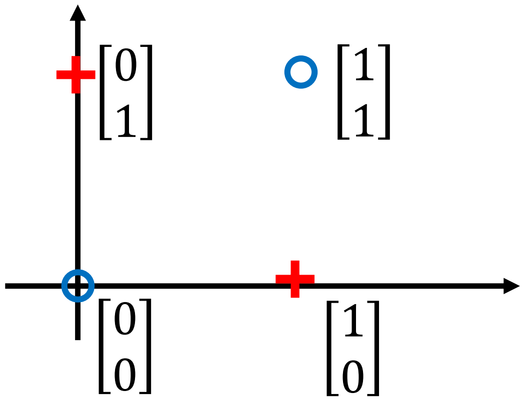
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

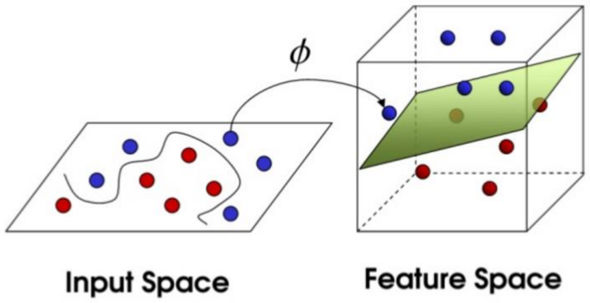


x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

Feature Transformation

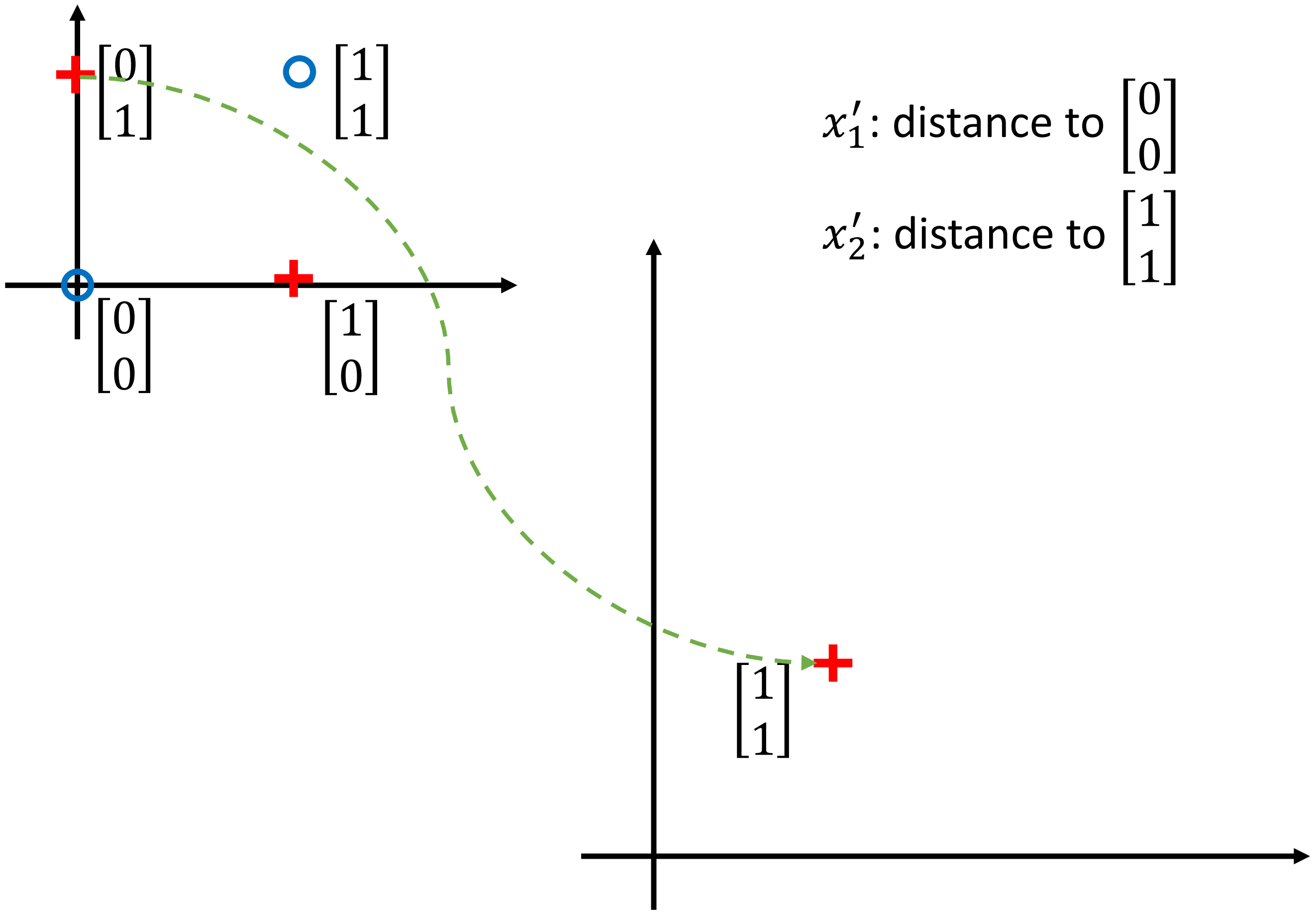
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression

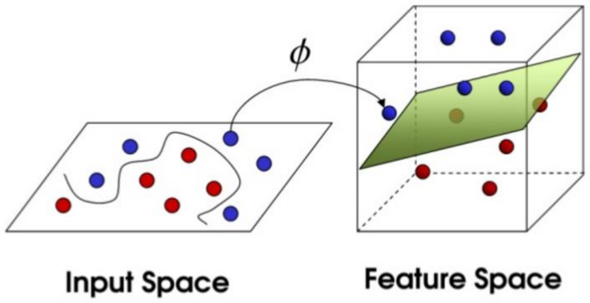


x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

Feature Transformation

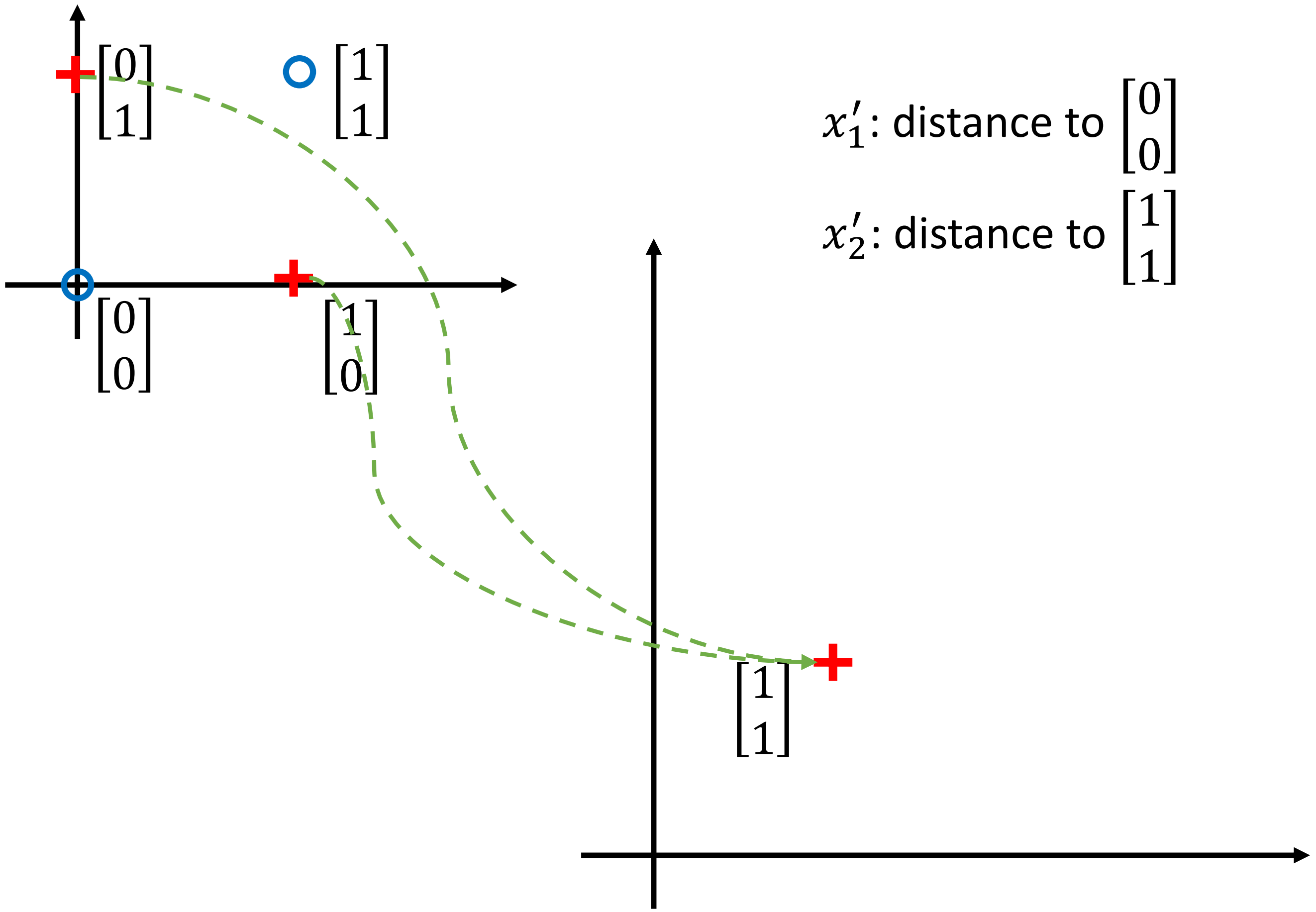
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression

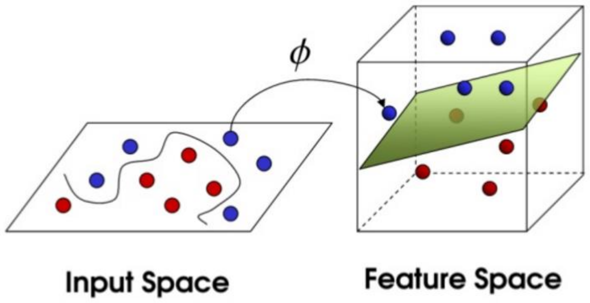


x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

Feature Transformation

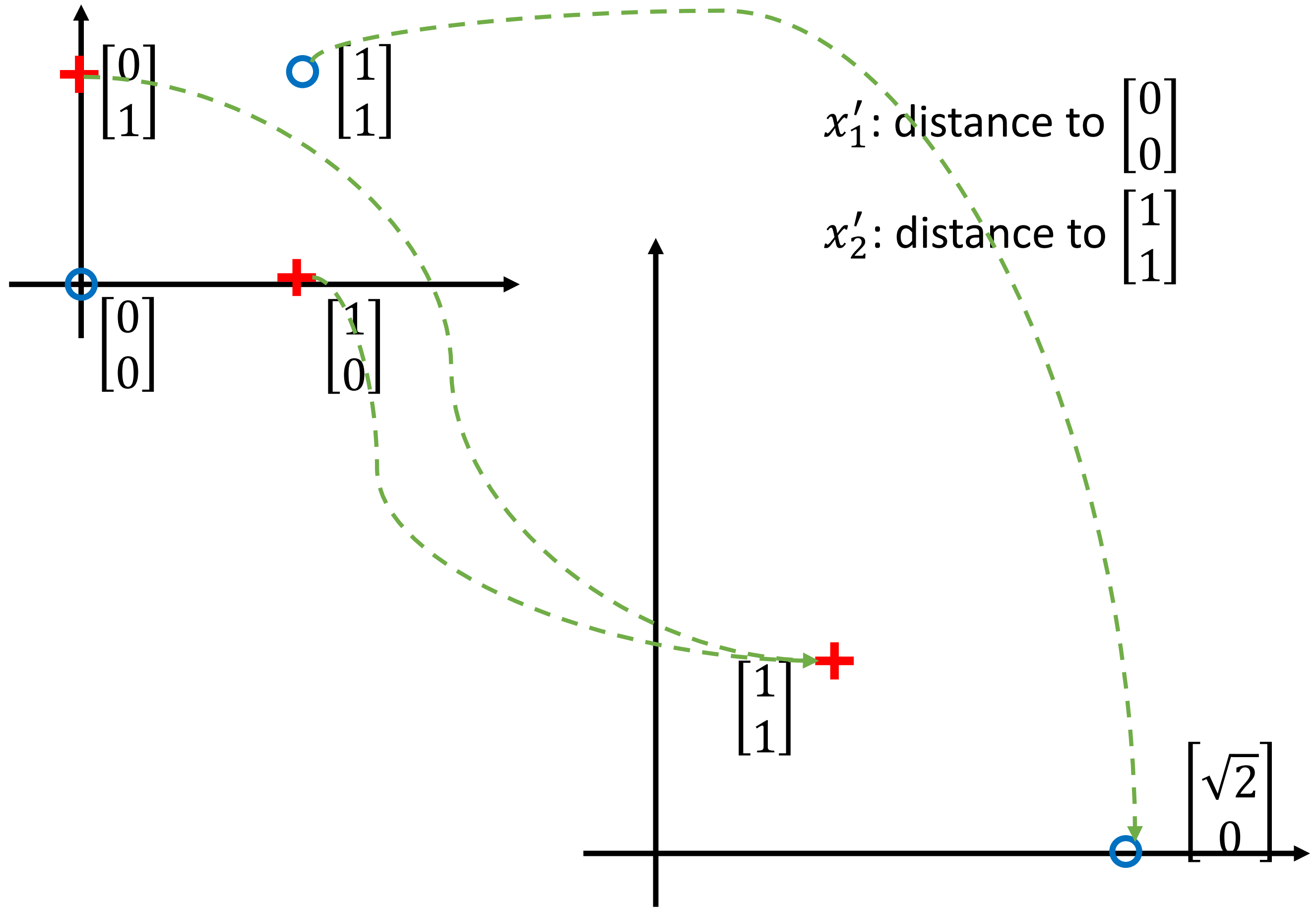
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression

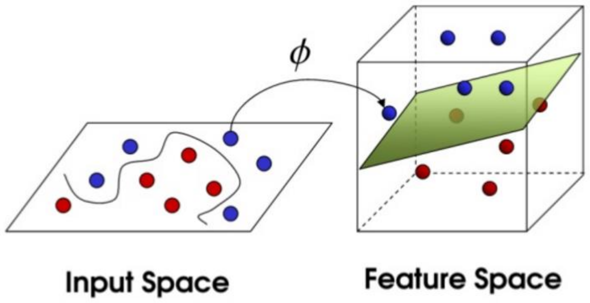


x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

Feature Transformation

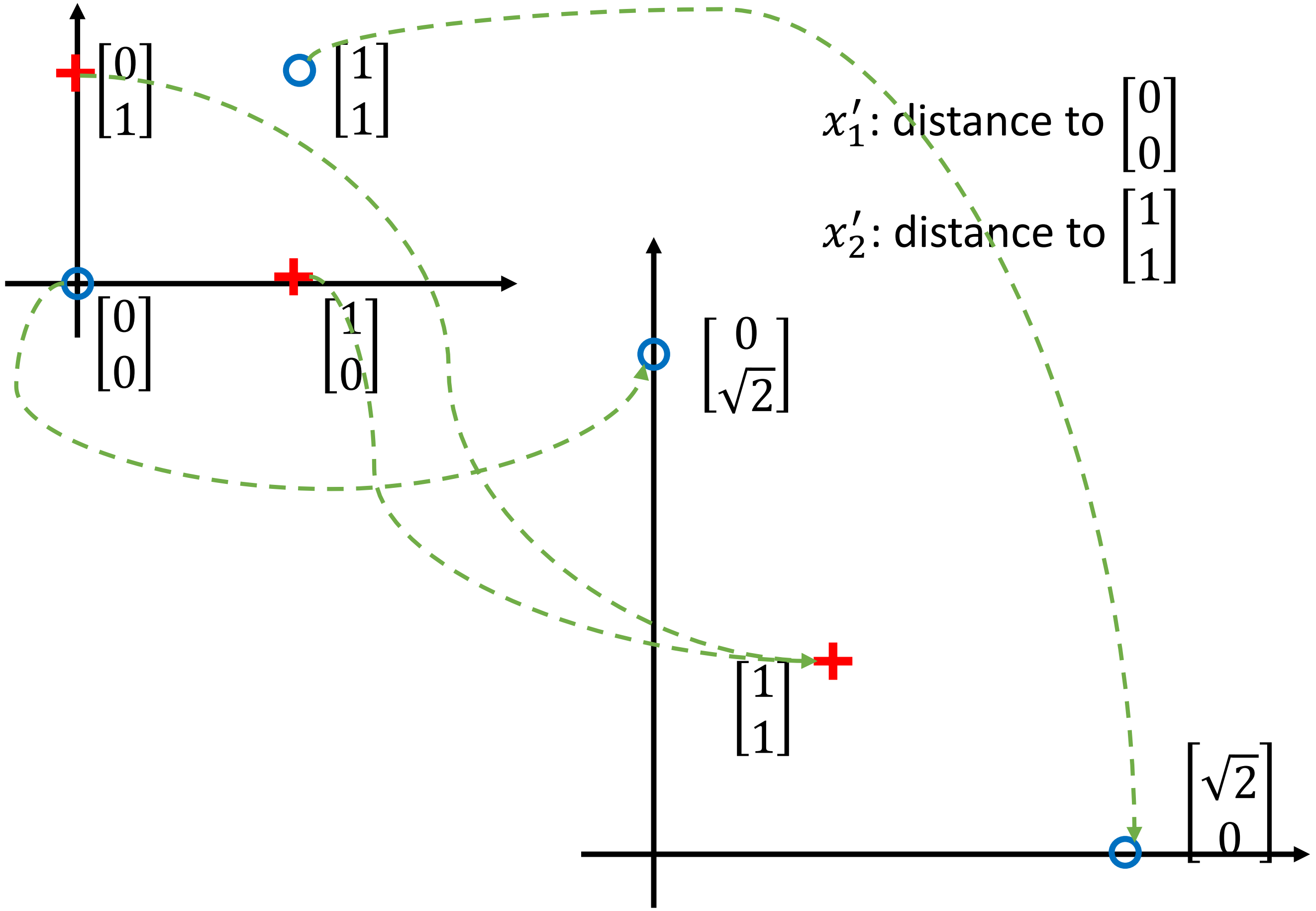
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression

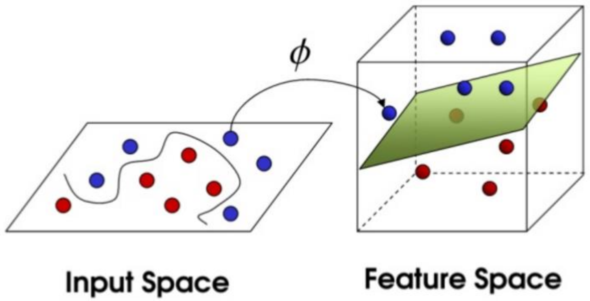


x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

Feature Transformation

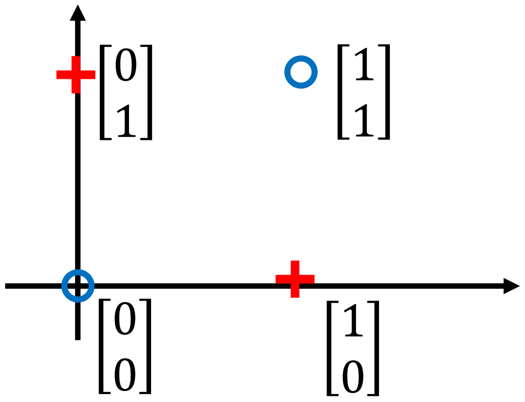
$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$



Courtesy of Dr. Hung-yi Lee

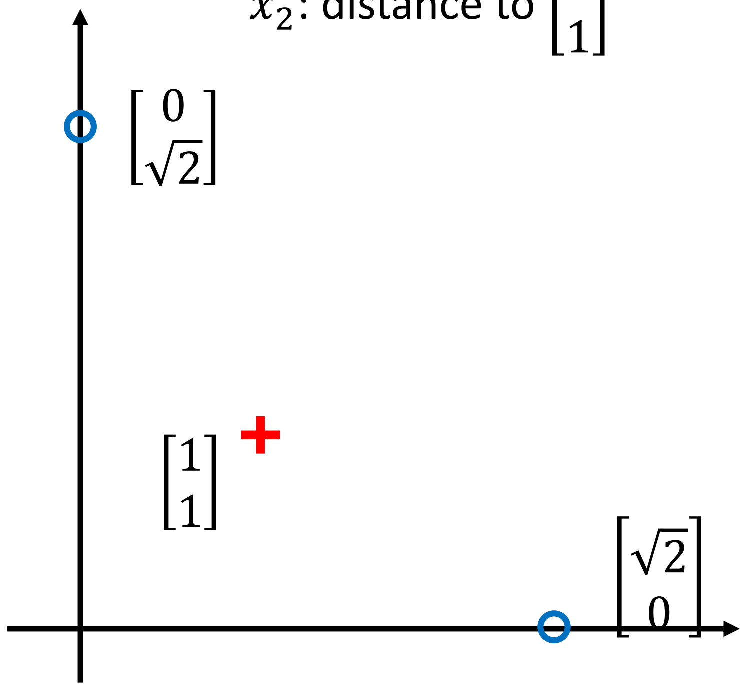
https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression



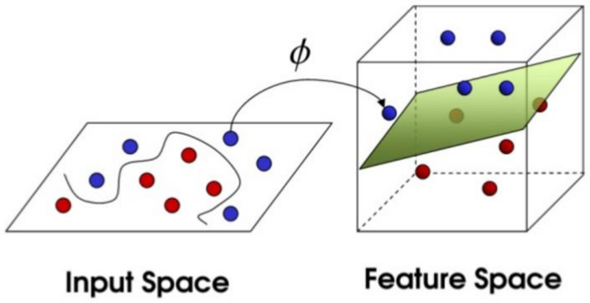
x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$



Feature Transformation

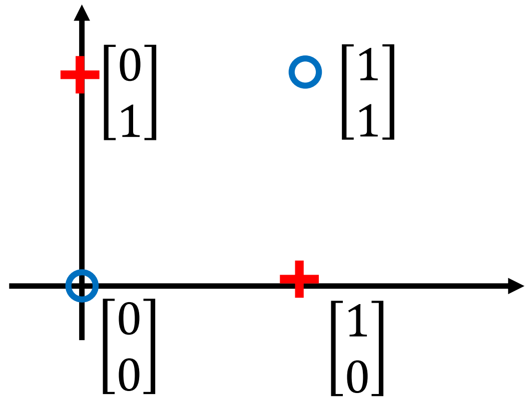
$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$



Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

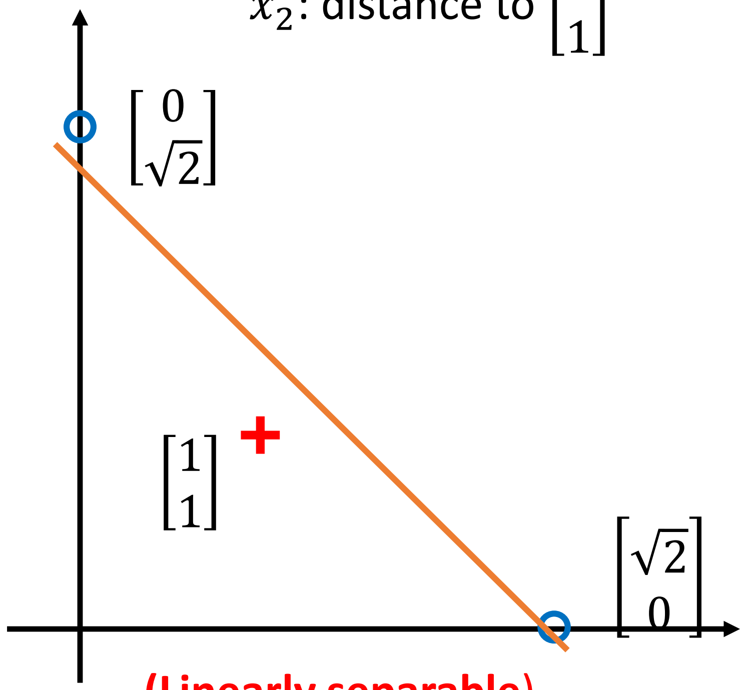
Limitation of Logistic Regression



(Not linearly separable)

x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

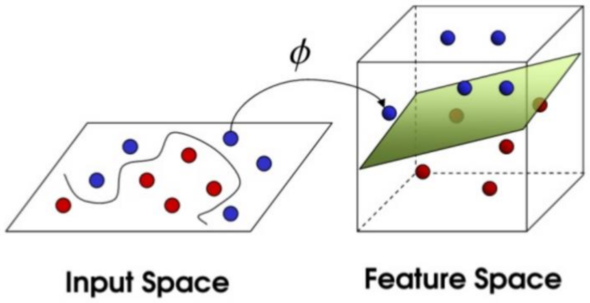
x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$



(Linearly separable)

Feature Transformation

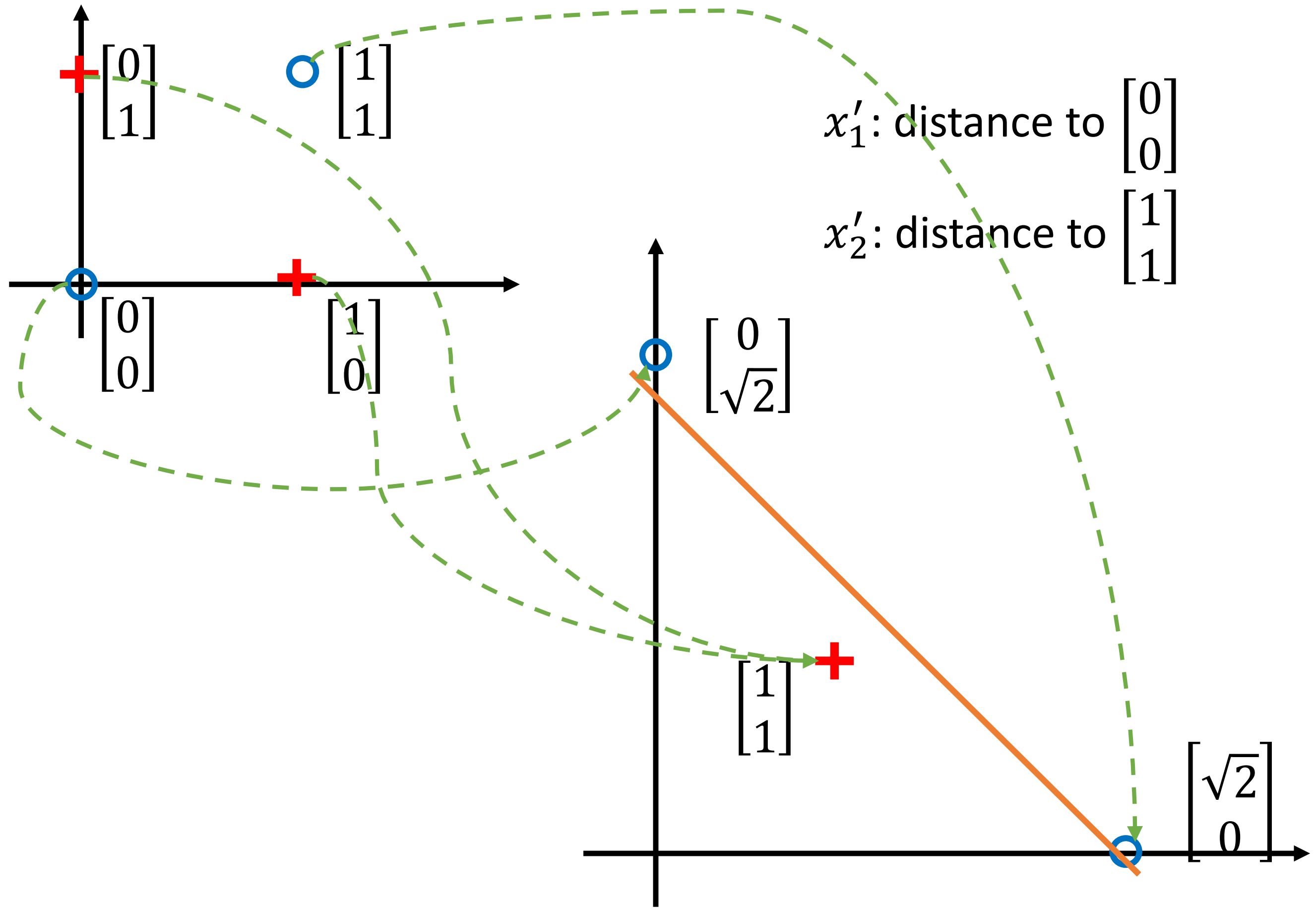
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

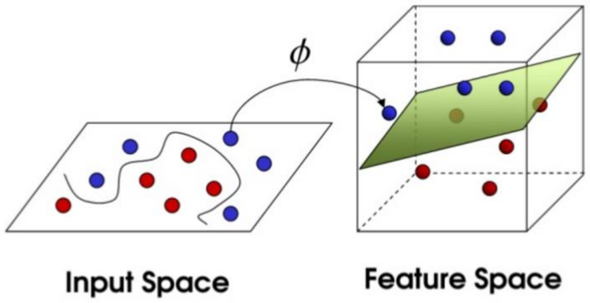
https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression



Feature Transformation

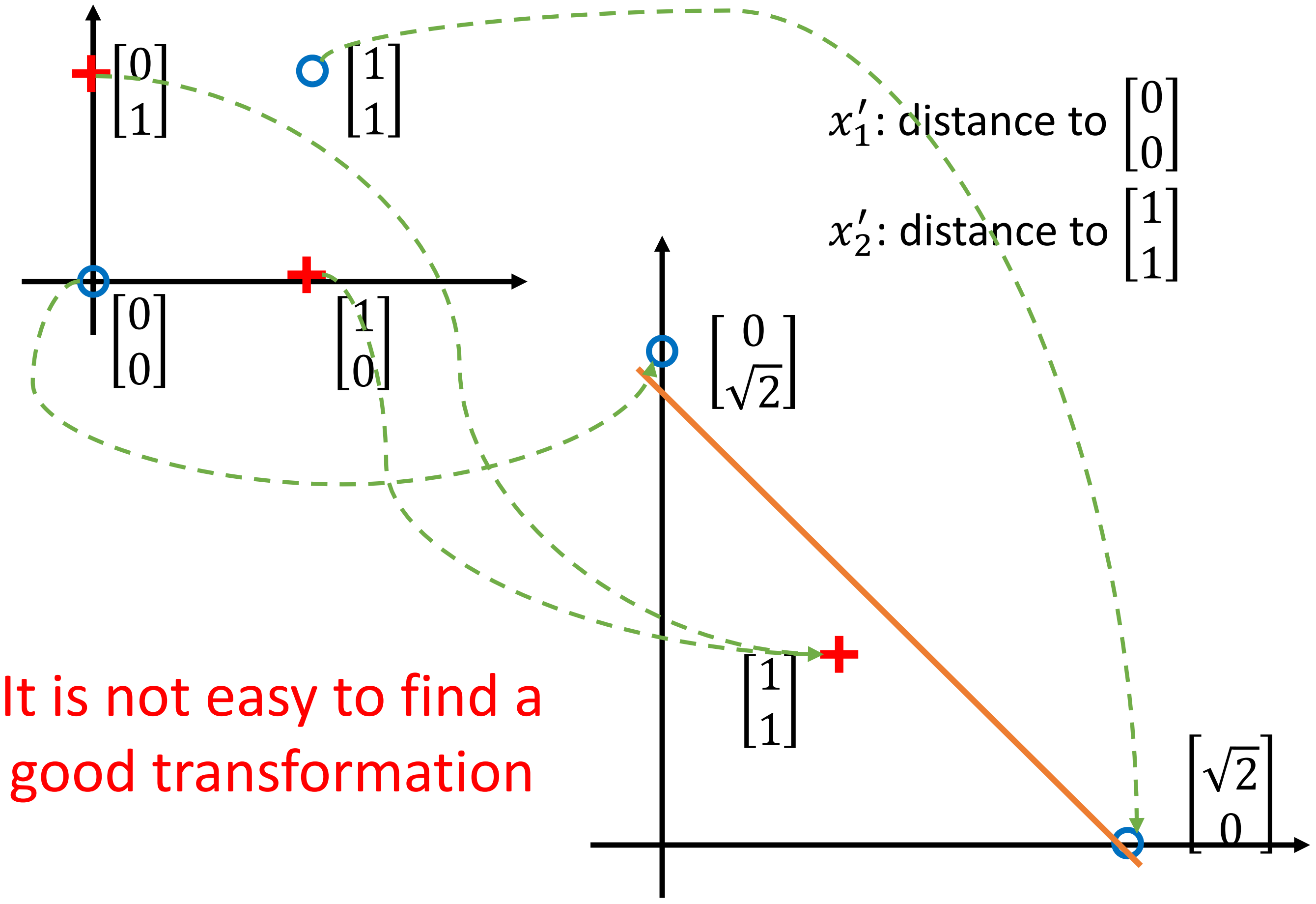
$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$



Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

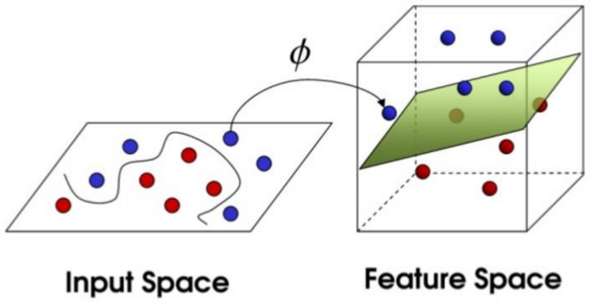
Limitation of Logistic Regression



It is not easy to find a good transformation

Feature Transformation

$$\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow \begin{bmatrix}x_1'\\x_2'\end{bmatrix}$$

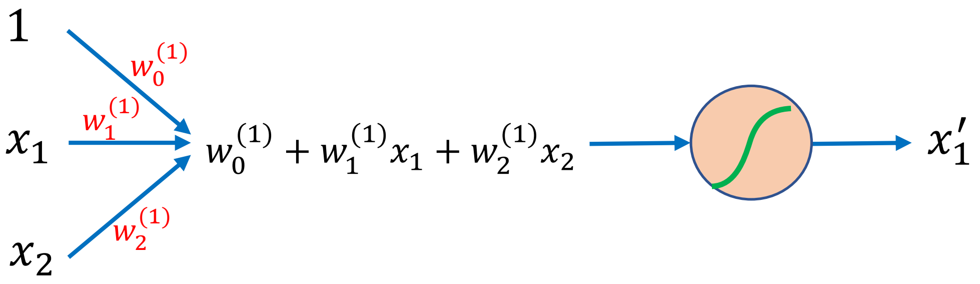


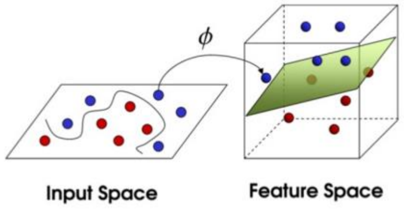
Courtesy of Dr. Hung-yi Lee

https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Limitation of Logistic Regression

- Cascading logistic regression models



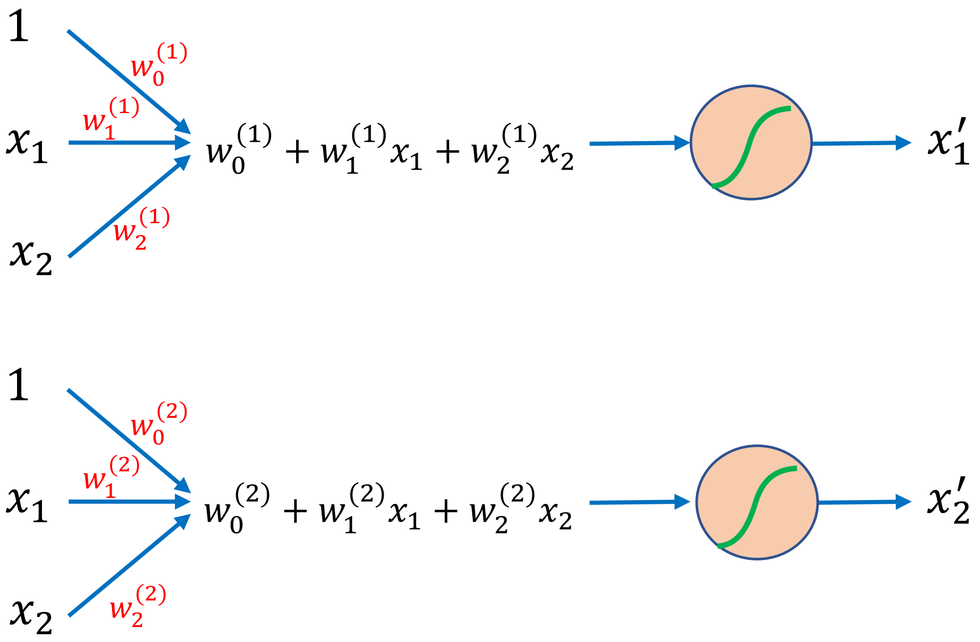


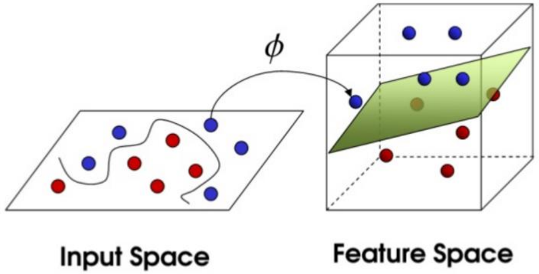
https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

- Cascading logistic regression models



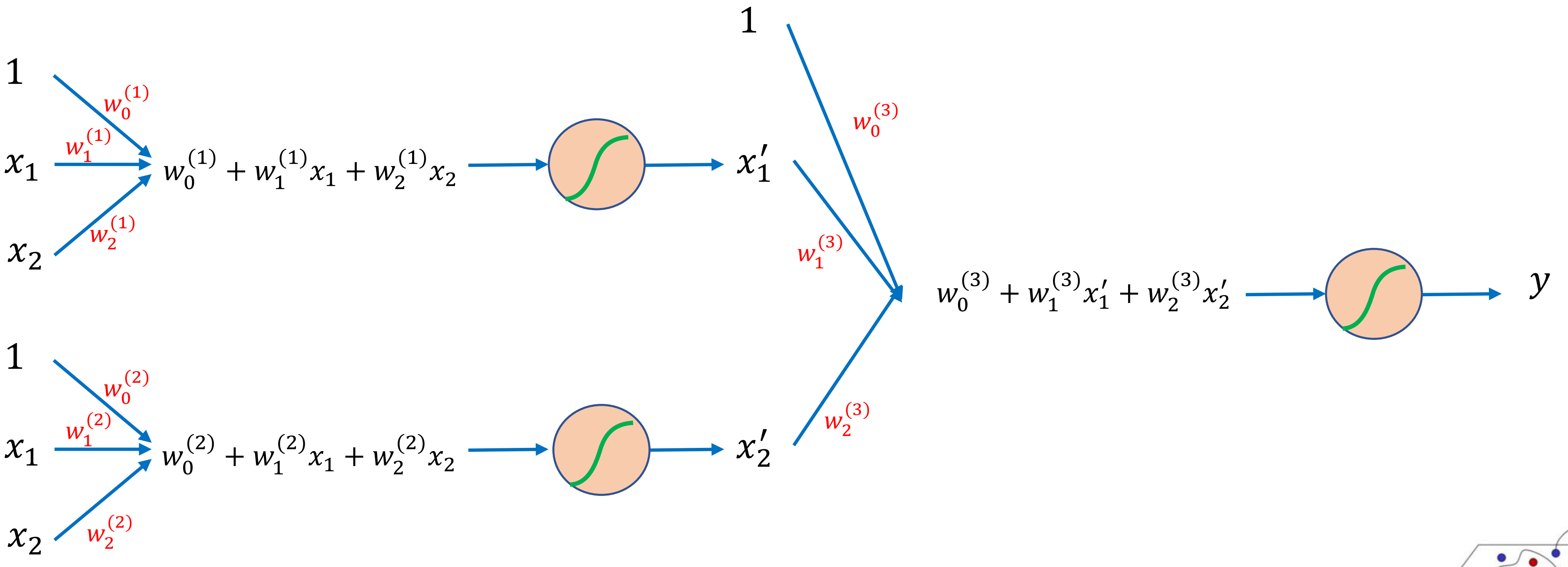


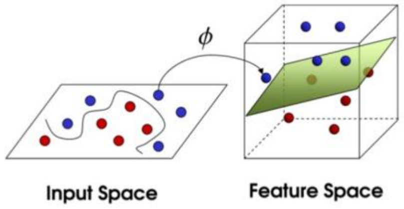
https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Courtesy of Dr. Hung-yi Lee

Limitation of Logistic Regression

- Cascading logistic regression models



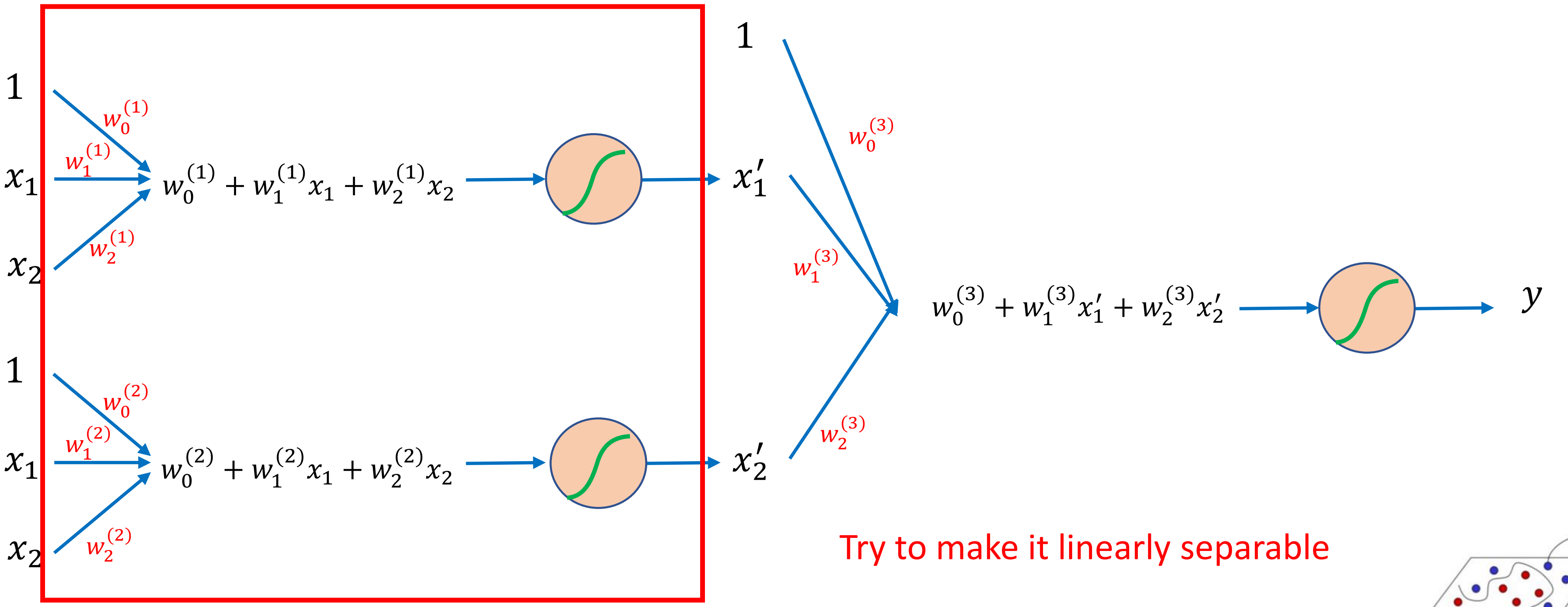


https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

Courtesy of Dr. Hung-yi Lee

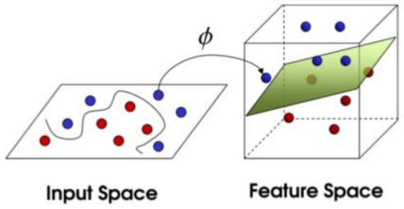
Limitation of Logistic Regression

- Cascading logistic regression models



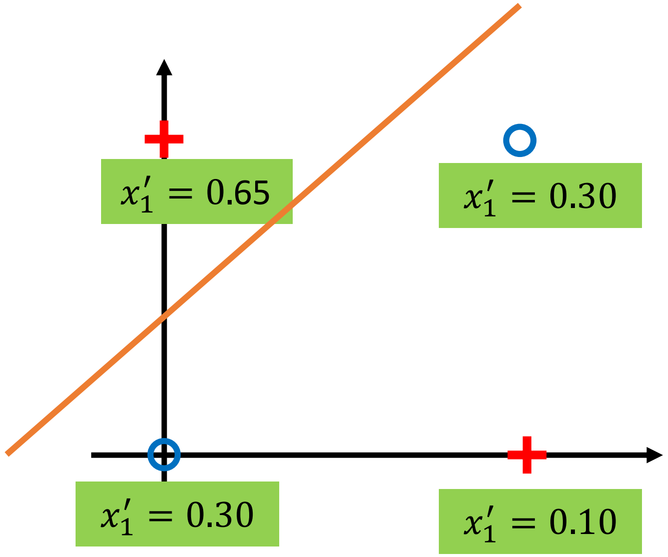
Feature Transformation

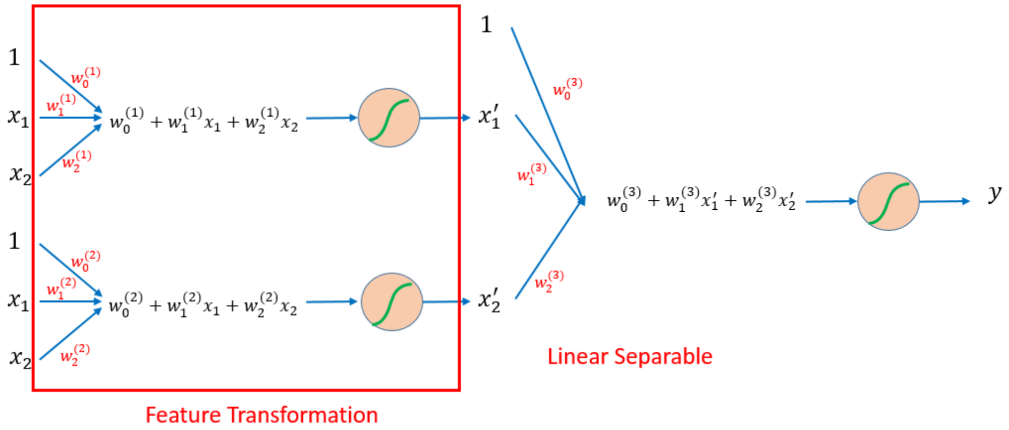
Try to make it linearly separable



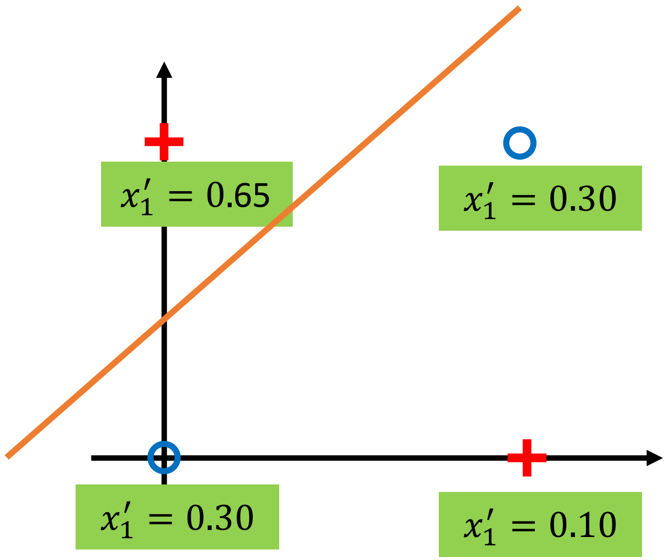
https://www.slideshare.net/cacois/machine-learningfor-moderndevelopers-35608763

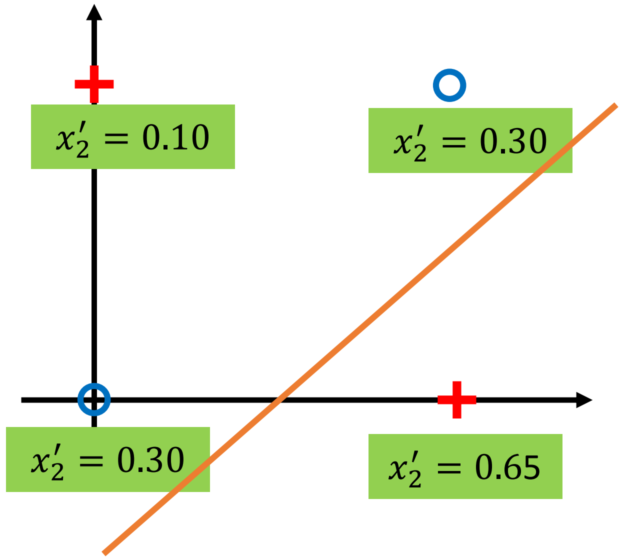
Courtesy of Dr. Hung-yi Lee

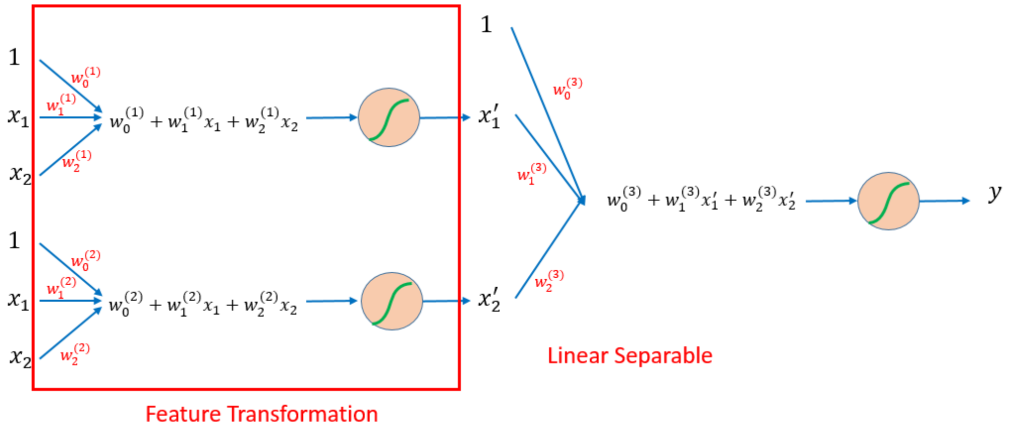




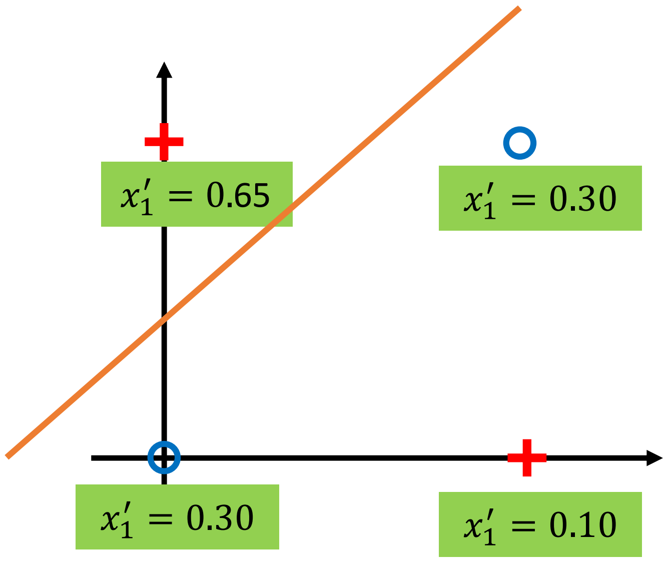
Courtesy of Dr. Hung-yi Lee

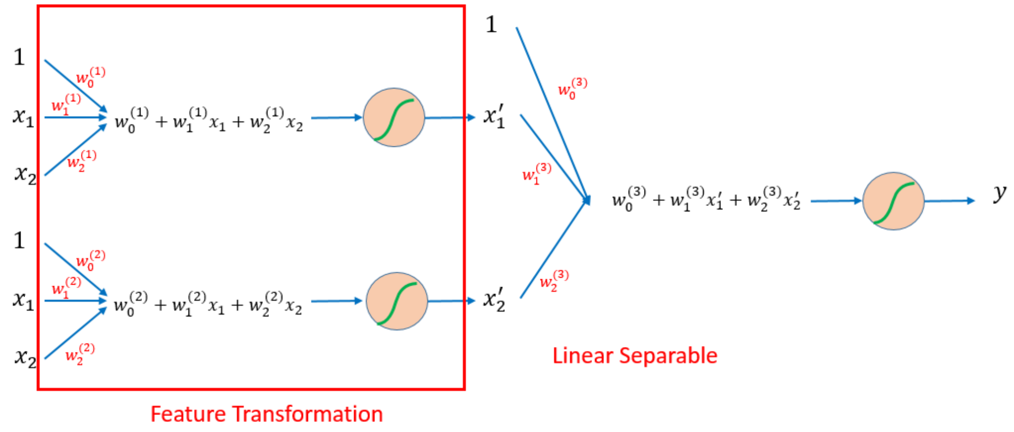


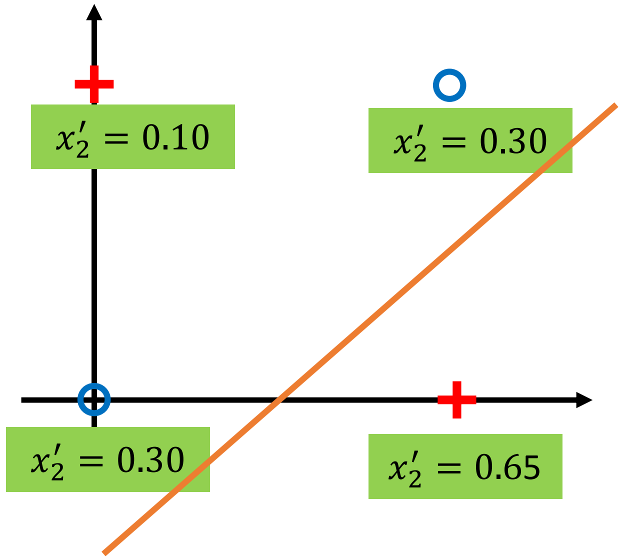


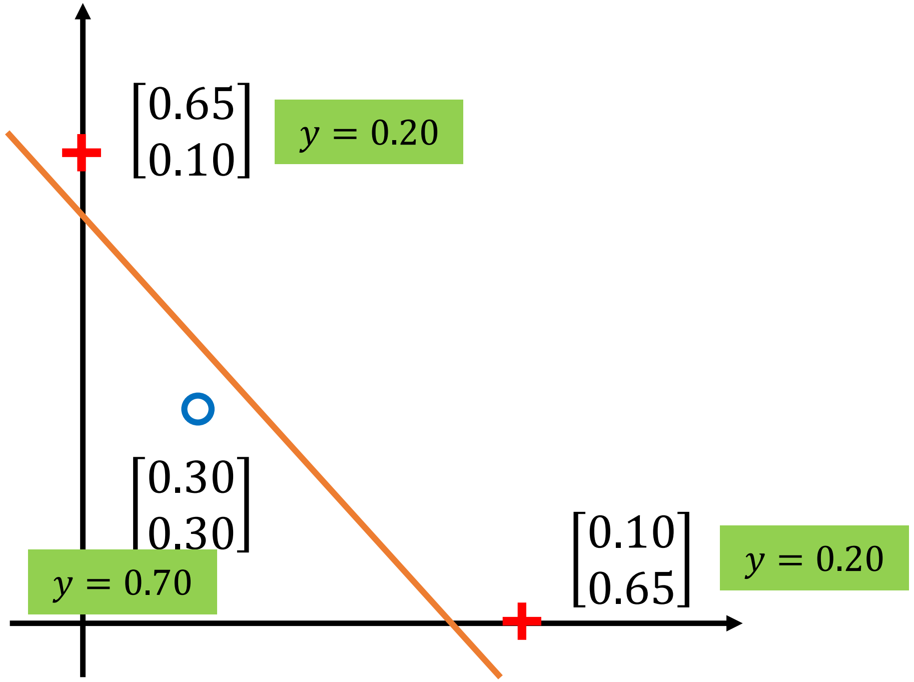


Courtesy of Dr. Hung-yi Lee









$$y = w_0^{(3)} + w_1^{(3)}x_1' + w_2^{(3)}x_2'$$

Courtesy of Dr. Hung-yi Lee

Metrics to Evaluate Classification

- Accuracy
  - The ratio of correct predictions to total predictions made.

Metrics to Evaluate Classification

- Accuracy
  - The ratio of correct predictions to total predictions made.

- Cases
  - True Positive: Correctly identified as relevant
  - True Negative: Correctly identified as not relevant
  - False Positive: Incorrectly labeled as relevant
  - False Negative: Incorrectly labeled as not relevant

Metrics to Evaluate Classification

- Accuracy
  - The ratio of correct predictions to total predictions made.

- Cases
  - True Positive: Correctly identified as relevant
  - True Negative: Correctly identified as not relevant
  - False Positive: Incorrectly labeled as relevant
  - False Negative: Incorrectly labeled as not relevant

		Predicted Class	
		1	0
Actual Class	1		
	0		

Metrics to Evaluate Classification

- Accuracy
  - The ratio of correct predictions to total predictions made.

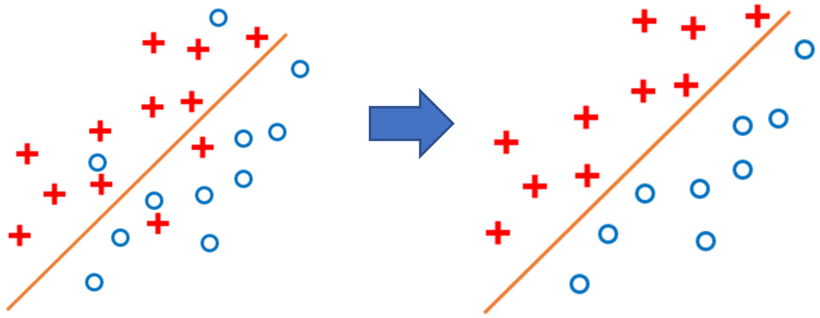
- Cases
  - True Positive: Correctly identified as relevant
  - True Negative: Correctly identified as not relevant
  - False Positive: Incorrectly labeled as relevant
  - False Negative: Incorrectly labeled as not relevant

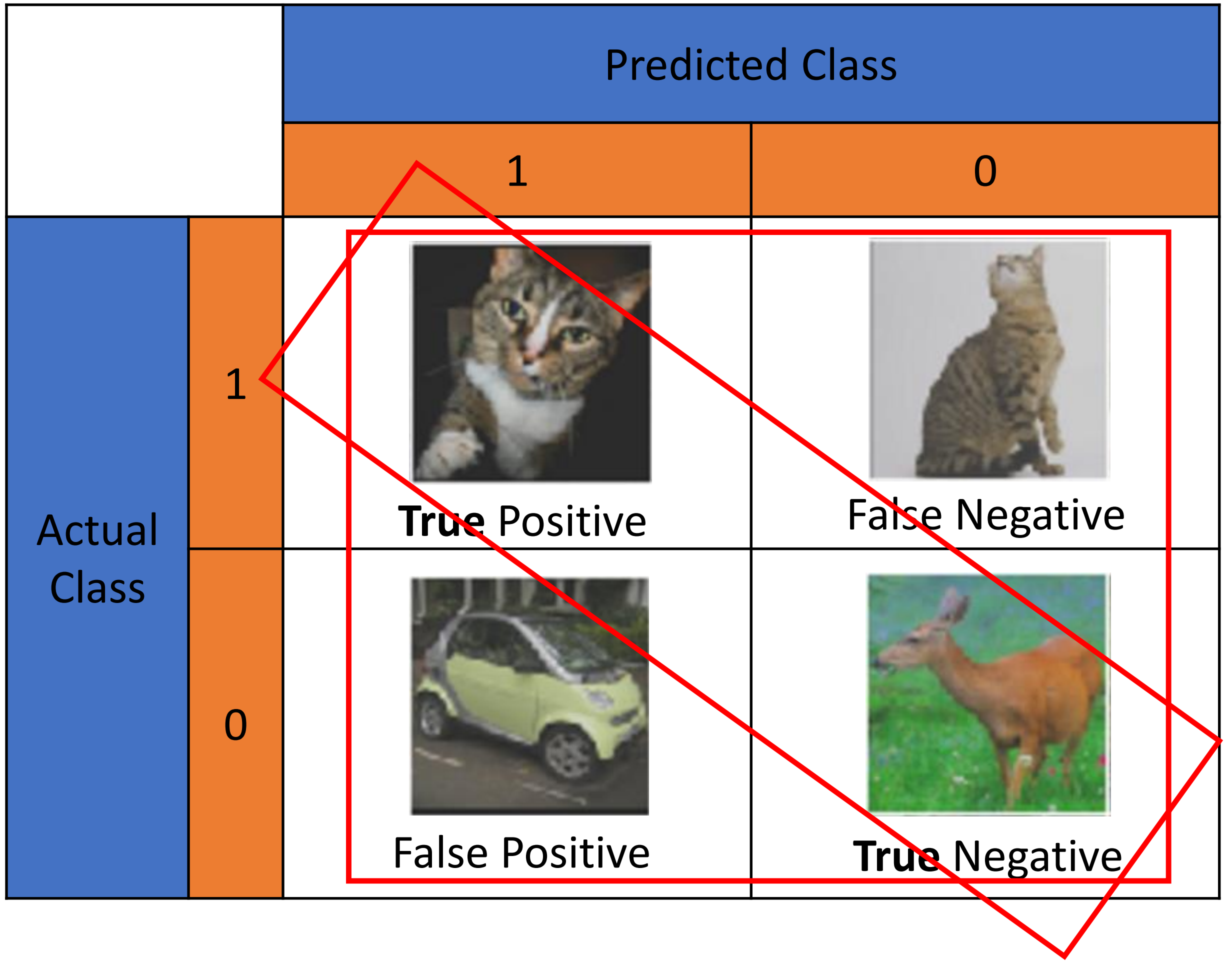
		Predicted Class	
		1	0
Actual Class	1	 **True** Positive	 False Negative
	0	 False Positive	 **True** Negative

Metrics to Evaluate Classification

- Accuracy
  - The ratio of correct predictions to total predictions made.

$$Accuracy = \frac{\#\ of\ True\ Positives + \#\ of\ True\ Negatives}{\#\ of\ samples}$$



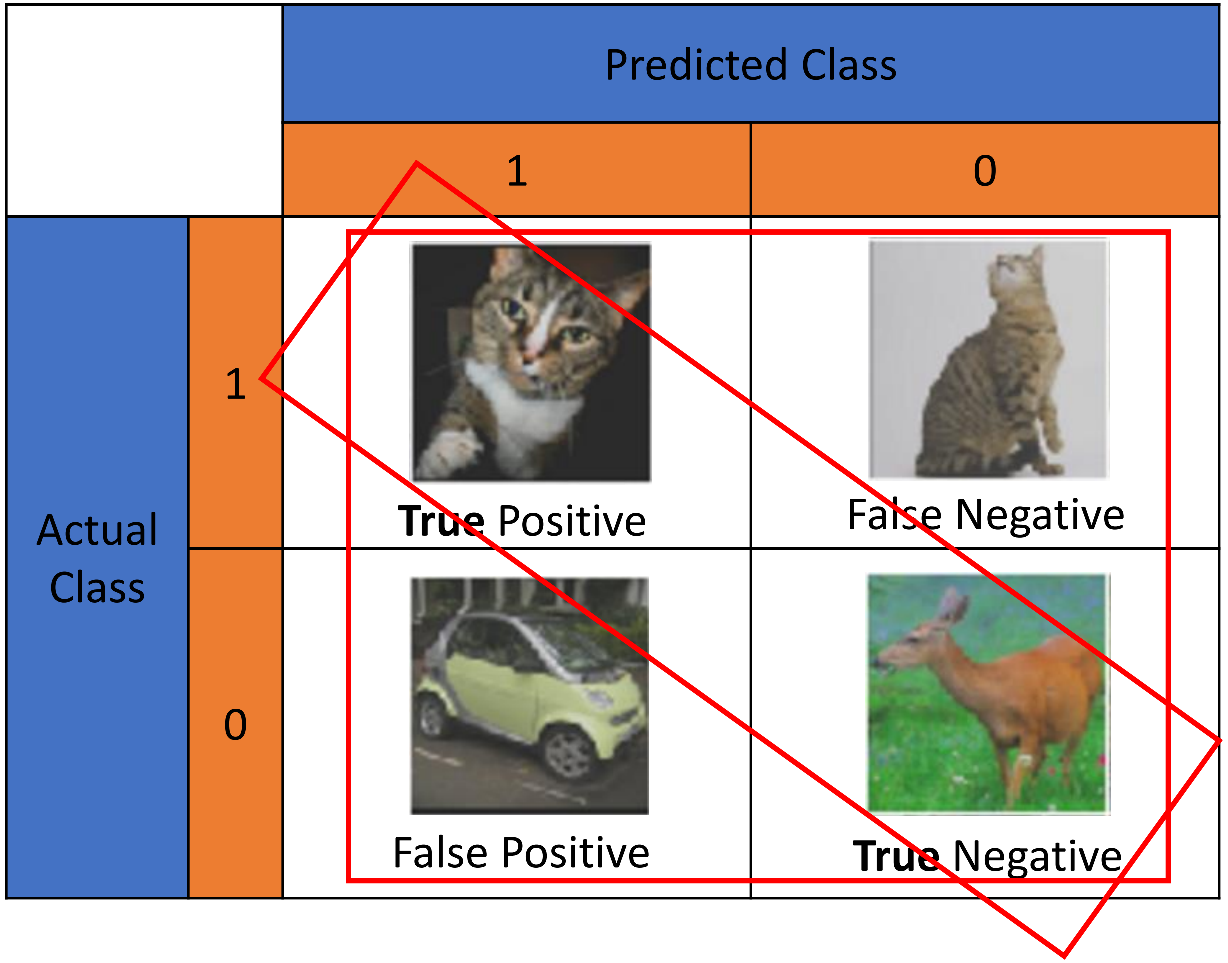


References: http://machinelearningmastery.com/confusion-matrix-machine-learning/

Metrics to Evaluate Classification

- Accuracy
  - The ratio of correct predictions to total predictions made.

$$Accuracy = \frac{\#\ of\ True\ Positives + \#\ of\ True\ Negatives}{\#\ of\ samples}$$



However, using classification accuracy only can be misleading,

- when you have an unequal number of observations in each class
- when you have more than two classes in your dataset.

References: http://machinelearningmastery.com/confusion-matrix-machine-learning/

Metrics to Evaluate Classification

- Suppose we have a classifier for identify the object in a image is cat or not.

If $\boldsymbol{y(x)} \geq \mathbf{0.5}$, we predict 1 (cat)
If $\boldsymbol{y(x)} < \mathbf{0.5}$, we predict 0 (not cat)

the accuracy is 99% on testing datasets (1000 images).

- Is this classifier good or not?

Metrics to Evaluate Classification

- Suppose we have a classifier for identify the object in a image is cat or not.

If $\boldsymbol{y(x)} \geq \mathbf{0.5}$, we predict 1 (cat)
If $\boldsymbol{y(x)} < \mathbf{0.5}$, we predict 0 (not cat)

the accuracy is 99% on testing datasets (1000 images).

- Is this classifier good or not?

- What do think if the 1000 images contain only 5 cat images?

Metrics to Evaluate Classification

- Suppose we have a classifier for identify the object in a image is cat or not.

If $\boldsymbol{y}(\boldsymbol{x}) \geq \mathbf{0.5}$, we predict 1 (cat)
If $\boldsymbol{y}(\boldsymbol{x}) < \mathbf{0.5}$, we predict 0 (not cat)

the accuracy is 99% on testing datasets (1000 images).

- Is this classifier good or not?

- What do think if the 1000 images contain only 5 cat images?

```
def classifier(img):
    return 0
```

The problem comes from an unequal number of observations in each class

Metrics to Evaluate Classification

- Precision, Recall, and Accuracy
  - Precision
    - Percentage of positive labels that are correct
    - Precision = (# true positives) / (# true positives + # false positives)
  - Recall
    - Percentage of positive examples that are correctly labeled
    - Recall = (# true positives) / (# true positives + # false negatives)
  - Accuracy
    - Percentage of correct labels
    - Accuracy = (# true positives + # true negatives) / (# of samples)

Reference: Nvidia - Deep Learning Teaching Kit

Metrics to Evaluate Classification

- Precision
  - Percentage of positive labels that are correct

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

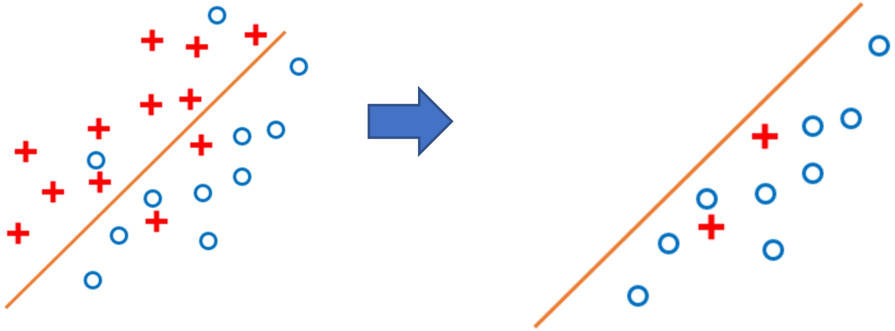
		Predicted Class	
		1	0
Actual Class	1	 **True** Positive	 False Negative
	0	 False Positive	 **True** Negative

References: http://machinelearningmastery.com/confusion-matrix-machine-learning/

Metrics to Evaluate Classification

- Precision
  - Percentage of positive labels that are correct

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$



		Predicted Class	
		1	0
Actual Class	1	 **True** Positive	 False Negative
	0	 False Positive	 **True** Negative

References: http://machinelearningmastery.com/confusion-matrix-machine-learning/

Metrics to Evaluate Classification

- Recall
  - Percentage of positive examples that are correctly labeled

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

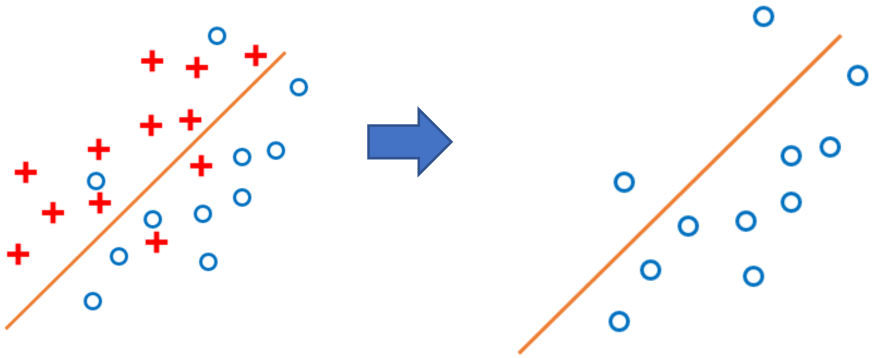
		Predicted Class	
		1	0
Actual Class	1	 **True** Positive	 False Negative
	0	 False Positive	 **True** Negative

References: http://machinelearningmastery.com/confusion-matrix-machine-learning/

Metrics to Evaluate Classification

- Recall
  - Percentage of positive examples that are correctly labeled

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$



		Predicted Class	
		1	0
Actual Class	1	 **True** Positive	 False Negative
	0	 False Positive	 **True** Negative

References: http://machinelearningmastery.com/confusion-matrix-machine-learning/

Metrics to Evaluate Classification

- Suppose we have a classifier for identify the object in a image is cat or not.

If $\boldsymbol{y(x)} \geq \mathbf{0.5}$, we predict 1 (cat)
If $\boldsymbol{y(x)} < \mathbf{0.5}$, we predict 0 (not cat)

- The testing datasets (1000 images contain only 5 cat images)

```
def classifier(img):
    return 0
```

		Predicted Class	
		1	0
Actual Class	1		
	0		

Metrics to Evaluate Classification

- Suppose we have a classifier for identify the object in a image is cat or not.

If $\boldsymbol{y}(\boldsymbol{x}) \geq \mathbf{0.5}$, we predict 1 (cat)
If $\boldsymbol{y}(\boldsymbol{x}) < \mathbf{0.5}$, we predict 0 (not cat)

- The testing datasets (1000 images contain only 5 cat images)

```
def classifier(img):
    return 0
```

		Predicted Class	
		1	0
Actual Class	1	0	5
	0	0	995

Metrics to Evaluate Classification

- Suppose we have a classifier for identify the object in a image is cat or not.

If $\boldsymbol{y(x)} \geq \mathbf{0.5}$, we predict 1 (cat)
If $\boldsymbol{y(x)} < \mathbf{0.5}$, we predict 0 (not cat)

- The testing datasets (1000 images contain only 5 cat images)

```
def classifier(img):
    return 0
```

Accuracy = 99.5%
Precision = ? (0/0)
Recall = 0 %

Confusion Matrix

		Predicted Class	
		1	0
Actual Class	1	0	5
	0	0	995

Metrics to Evaluate Classification

- F1 score is a measure of a classifier’s performance

$$F_1 = 2\frac{Precision \times Recall}{Precision + Recall}$$

- Measure precision and recall on validation sets and select a model that gives max F1 score.

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