Feature Engineering

Dr. Kalidas Y.,

Dept. of CSE.,

IIT Tirupati

In this lecture you will understand dimensionality transformation, feature engineering and pipeline of transformations

Some key terminology

- Given a k dimensional vector x,

it should be imagined as composed of k components $\begin{bmatrix} x_0 \\ \cdots \\ x_{k-1} \end{bmatrix}_{k\times 1}$

- It's i[th] component is denoted by x_i or $x[i]$
- L-j Norm of the vector is defined as $|x|_j = \left(|x_0^j| + \cdots + |x_{k-1}^j|\right)^{1/j}$
- Popular norms
  - L-1 norm
  - L-2 norm
- A matrix can be *flattened to a vector,* $vec(M_{k\times l}) = [\ M[0][0], \ldots, M[k-1][l-1]\ ]$
- Dot product of two vectors, $x \cdot y = x_0 y_0 + \cdots x_{k-1} y_{k-1}$
- Other usual operations as you must be familiar with

Fitting a Line Passing Through Origin

- $y = m\, x$
- $L(m) = \sum_{i=1}^{i=N} (y_i - m\, x_i)^2$
- $X = \begin{bmatrix} x_1 \\ \cdots \\ x_N \end{bmatrix}_{N\times 1}, Y = \begin{bmatrix} y_1 \\ \cdots \\ y_N \end{bmatrix}_{N\times 1}, W = [m\,]_{1\times 1}$
- $L([m]) = (XW - Y)^T (XW - Y)$
- $\nabla L = \left[\frac{\partial L}{\partial m}\right]$ //It's a function
- $W_{(new)} = W_{(old)} - \nabla L|_{W=W_{(old)}}$

Squared error type loss function

Fitting a Line – slope and intercept

- $y = m\,x + c$
- $L(m) = \sum_{i=1}^{i=N}(y_i - (m\,x_i + c))^2$
- $X = \begin{bmatrix} x_1 & 1 \\ & \ldots \\ x_N & 1 \end{bmatrix}_{N\times 2}, Y = \begin{bmatrix} y_1 \\ \ldots \\ y_N \end{bmatrix}_{N\times 1},$

  $W = \begin{bmatrix} m \\ c \end{bmatrix}_{2\times 1}$
- $L\left(\begin{bmatrix} m \\ c \end{bmatrix}\right) = (XW - Y)^T(XW - Y)$

- $\nabla L = \begin{bmatrix} \frac{\partial L}{\partial m} \\ \frac{\partial L}{\partial c} \end{bmatrix}$ //It's a function
- $W_{(new)} = W_{(old)} - \nabla L|_{W=W_{(old)}}$

Squared error type loss function

Fitting a Parabola?

- $y = a\,x^2 + b\,x + c$
- $L(a, b, c) = \sum_{i=1}^{i=N} \big(y_i - (a\,x_i^2 +$

Squared error type loss function

Fitting a Cubic curve?

- $y = a\,x^3 + b\,x^2 + c\,x + d$
- $L(m) = \sum_{i=1}^{i=N}\left(y_i - (a\,x_i^3 + b\,x_i^2 +\right.$

Squared error type loss function

Fitting a Degree-K polynomial?

- $y = a_k\, x^k + \cdots + a_0$
- $L(m) = \sum_{i=1}^{i=N} \left(y_i - \sum_{j=0}^{k} a_j\, x^j\right)^2$
- $X = \begin{bmatrix} x_1^k \ldots x_1^2 & x_1^1 & 1 \\ & \ldots & \\ x_N^k \ldots x_N^2 & x_N^1 & 1 \end{bmatrix}_{N \times (k+1)}$,
- $Y = \begin{bmatrix} y_1 \\ \ldots \\ y_N \end{bmatrix}_{N \times 1}$,
- $W = \begin{bmatrix} a_0 \\ \ldots \\ a_k \end{bmatrix}_{(k+1) \times 1}$
- $L(W) = (XW - Y)^T (XW - Y)$
- $\nabla L = \begin{bmatrix} \frac{\partial L}{\partial a_0} \\ \ldots \\ \frac{\partial L}{\partial a_k} \end{bmatrix}$ //It's a function
- $W_{(new)} = W_{(old)} - \nabla L|_{W = W_{(old)}}$

Squared error type loss function

31) key phrase…
“feature/dimensionality/input transformation”

An example of “feature transformation” of x i

$$x_i \vdash (x_i^0, x_i^1, \ldots, x_i^k)$$

Example… Polynomial transformation

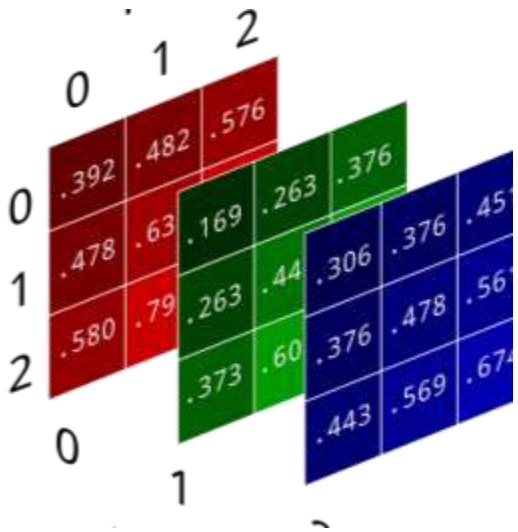
- Polynomial transformation
  - Quadratic features: (a,b,c) → $(1, a, b, c, a^2, b^2, c^2, ab, ac, bc)$
  - Cubic features:

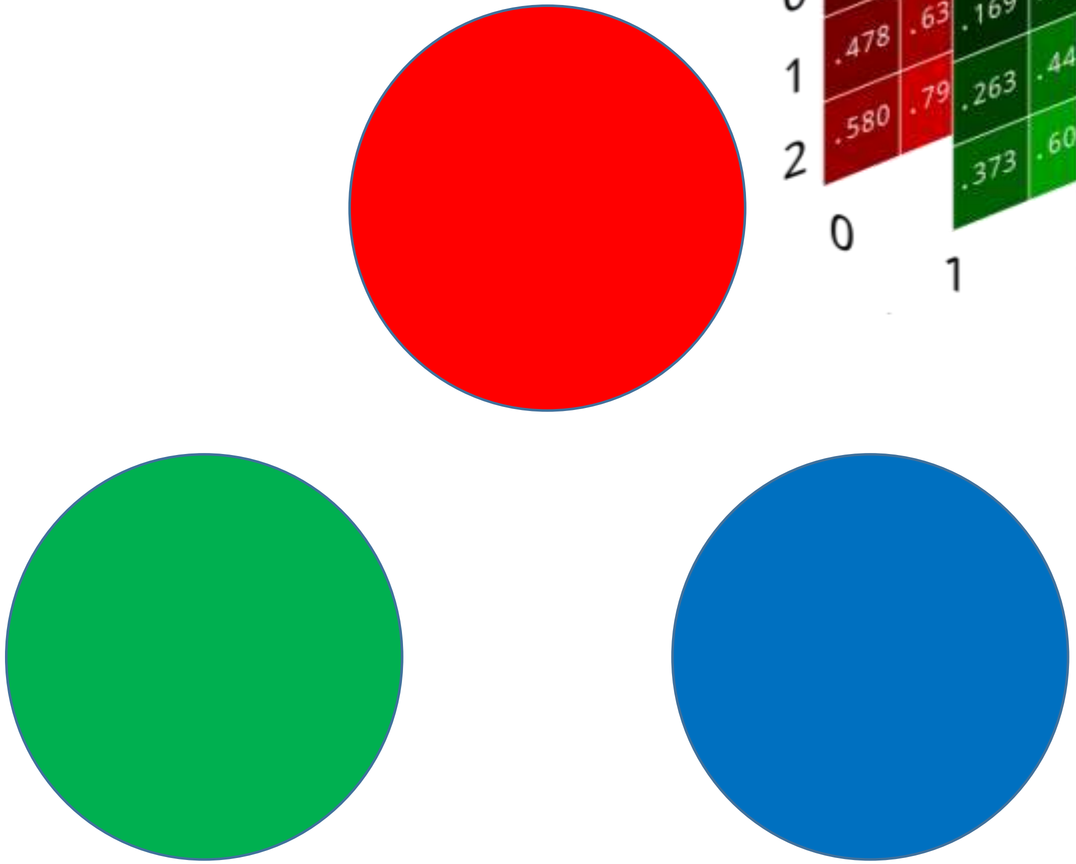
    (a,b,c) → $(1, a, b, c, a^2, b^2, c^2, ab, ac, bc, a^3, b^3, c^3, a^2b, a^2c, b^2a, b^2c, c^2a, c^2b, abc)$
  - ***Degree k features…***
- Scikit-learn
  - ***sklearn.preprocessing.PolynomialFeatures()***
  - ***Example, degree=3***
  - ***interaction_only = False (or True)***
  - ***include_bias = True (or False)***

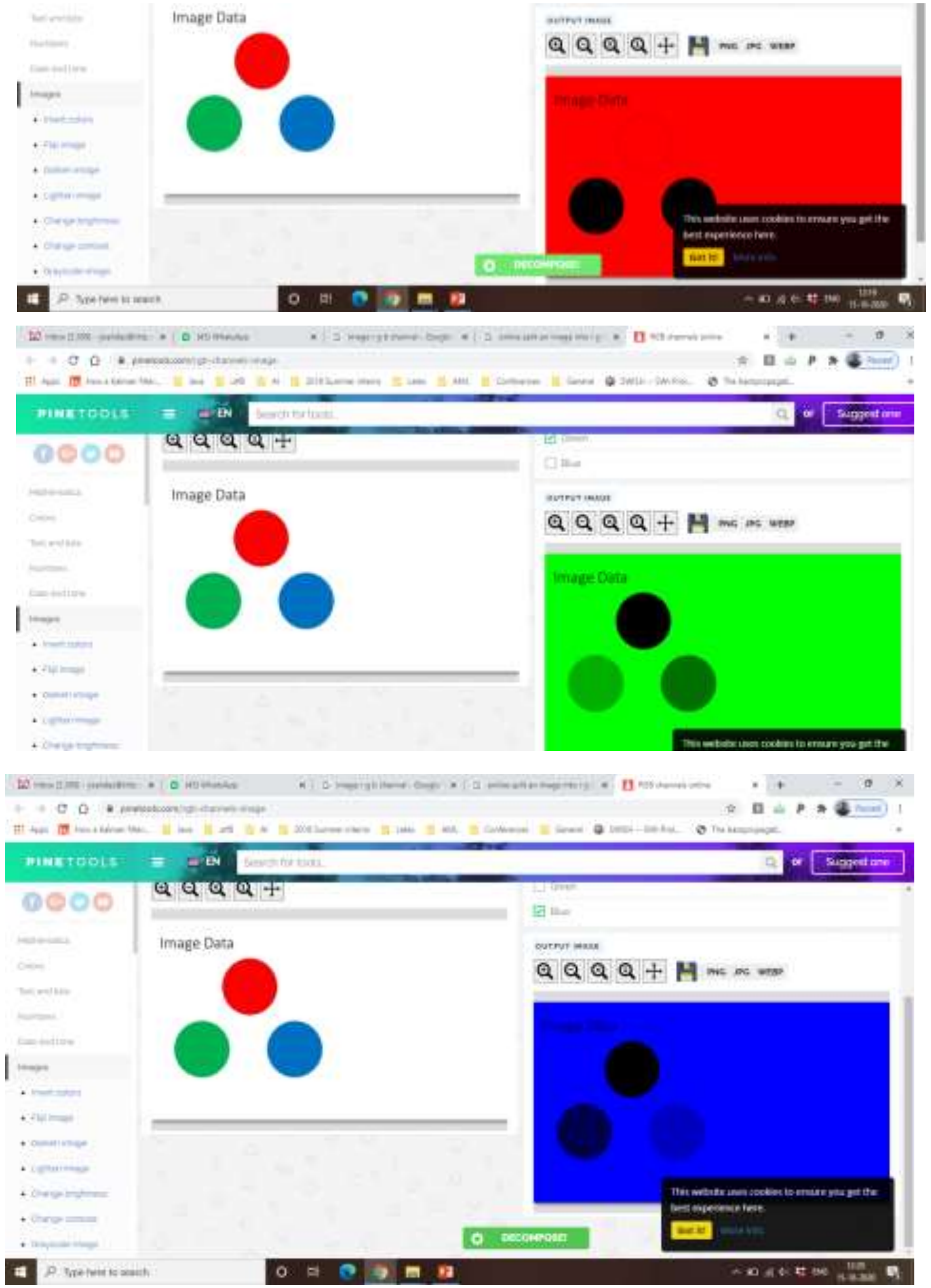
32) key phrase… “feature engineering”

- Designing X matrix
- using feature transformation

Image Data

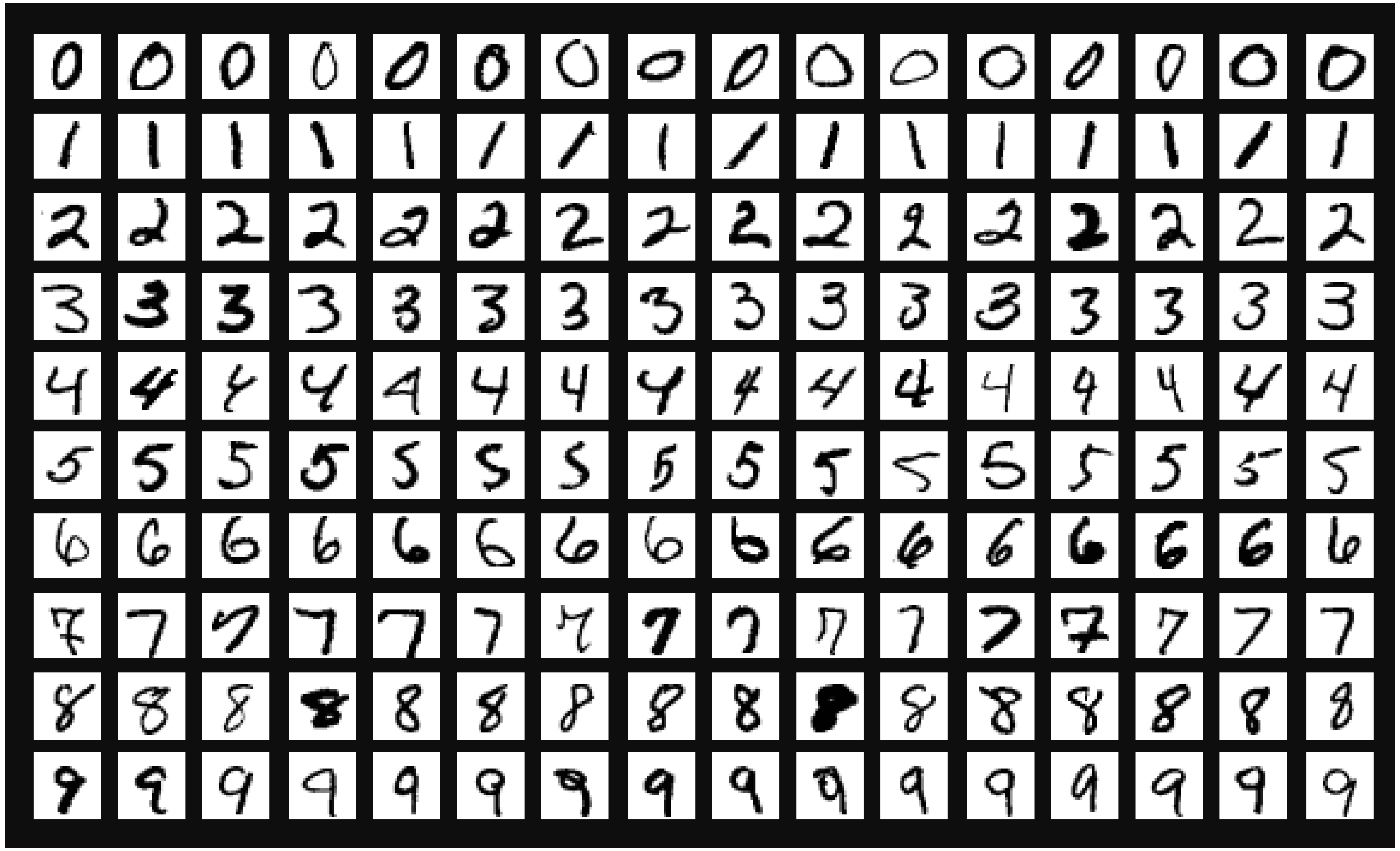






1. *Three channels – red, blue and green*
2. *Each channel is a matrix of intensities*
3. *How do you vectorize? (easy..!?)*

DIGITS



X0	X1						X7
X8	…	…	…	…	…	…	X15
…							…
…							…
X56	X57						X63

- Gray scale
- Each digit is on 8 x 8 pixel matrix
- Total 64 pixels
- An array of 64 numbers
- You can take some part as xi

  For example, x0 to x47 (48 pixels)
- Remaining part as yi

  For example, x48 to x63 (16 pixels)

Text Features

Capitalization, small letters, capital letters punctuation etc.

- Data set - Corpus
  - "I am going to school"
  - "A teacher is talking about National movement"
  - "We had machine learning examination today"
  - "Ramu is explaining how to see through a binoculars"
  - "Zavid is playing on a smart phone"

Pre-processing

- all small letters
- Stemming
- Lemmatization
- Named Entity Recognition
- Several several several others… https://www.nltk.org/
- Good news – *READY TO USE LIBRARIES ARE THERE*

- Vocabulary 1 - dictionary

["i", "am", "go", "to", "school", "a", "teacher", "is", "talk", "about", "national", "movement", "we", "had", "machine", "learn", "examination", "today", "person", "explain", "how", "to", "see", "through", "binoculars", "play", "on", "smart", "phone"]

- Vocabulary 2 – dictionary

["i", "am", "go", "to", "school", "a", "teacher", "is", "talk", "about", "national", "movement", "we", "had", "machine learning", "examination", "today", "ramu", "zavid", "explain", "how", "to", "see", "through", "binoculars", "play", "on", "smart", "phone"]

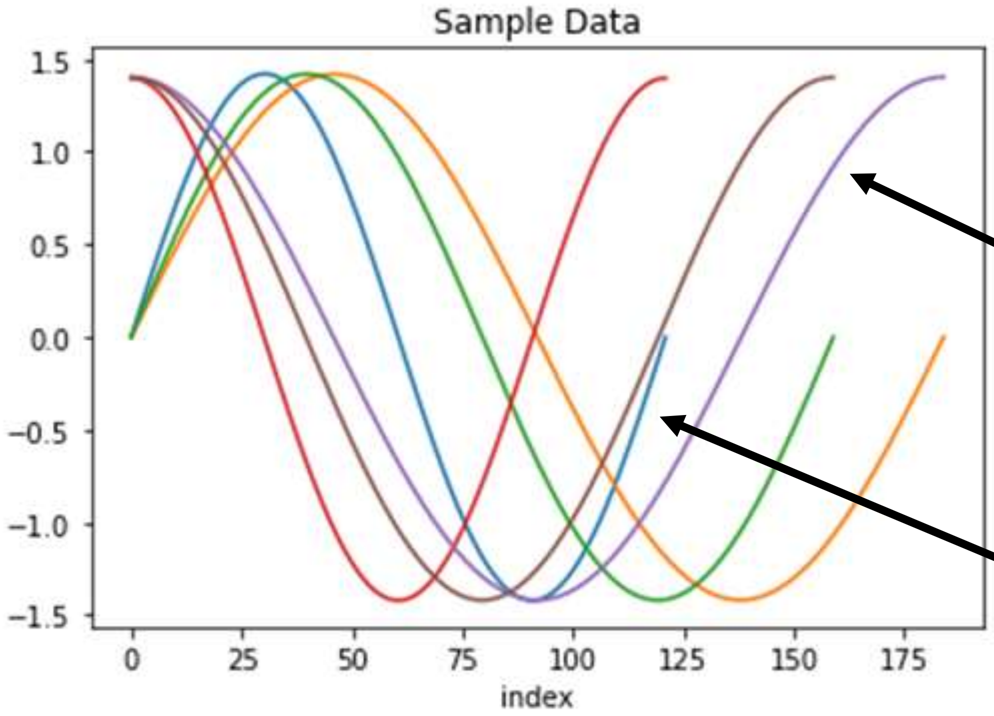
"I am going to school" → [1, 1, 1, 1, 1, 0, 0, 0, … 0, 0]

"A teaching is talking about National movement" → [0,0,0,0,0,1,1,1,… …0 0,0]

xi vector

Challenge…

Time series data → Vector



longer wave

shorter wave

Challenge..

If you directly take raw amplitudes…

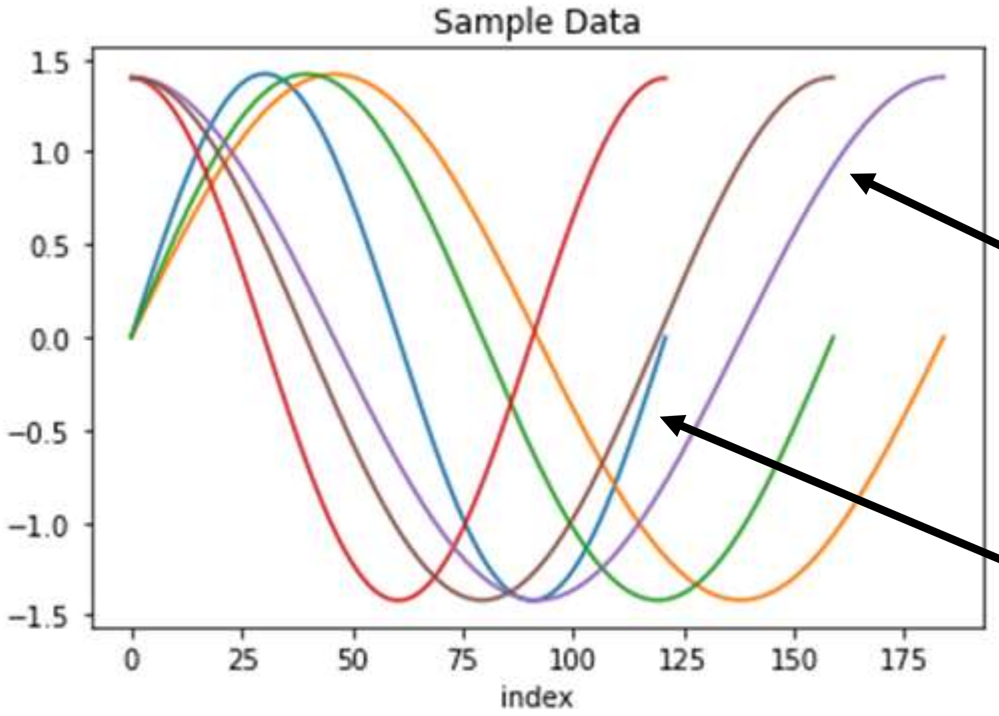
xi for longer wave will have more dimensions
xi for shorter wave will have less dimensions

→ You cannot chose w vector appropriately

→ Means you cannot create

Challenge… → Solution!

Time series data → Vector



xi vector

Challenge…

If you directly take raw amplitudes…

xi for longer wave will have more dimensions
xi for shorter wave will have less dimensions

→ You cannot chose w vector appropriately

→ Means you cannot create

longer wave

shorter wave

Solution
Take mean, median, mode
Take maximum amplitude
Take maximum in 1st half, 2nd half etc.
Other ***Length agnostic features***

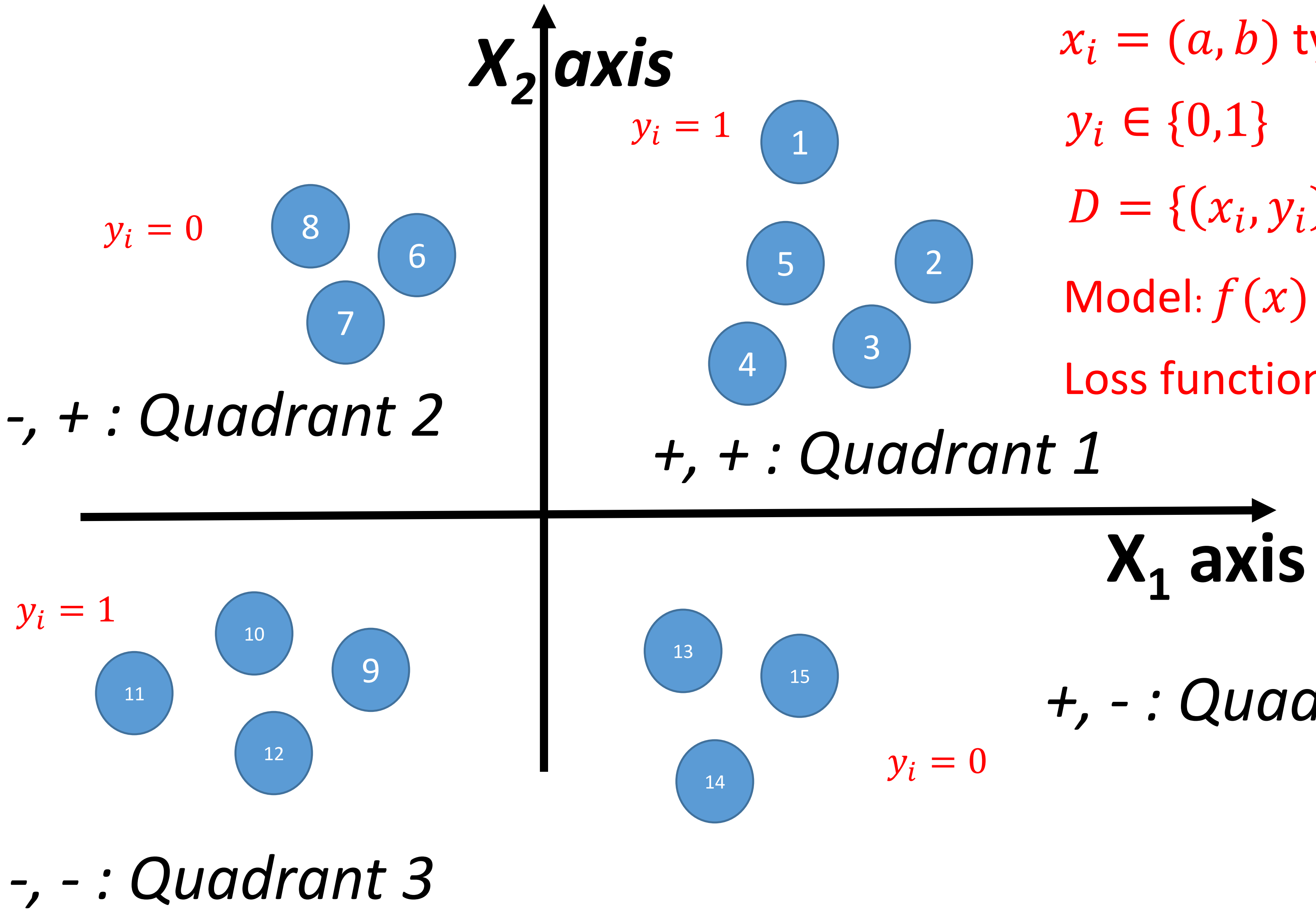
$x_i = (a, b)$ type of 2D point

$y_i \in \{0,1\}$

$D = \{(x_i, y_i)\}_{i \in [1 \ldots N]}, N = 15$

Model: $f(x) = x^T w$

Loss function: $L(y, f(x)) = (f(x) - y)^2$



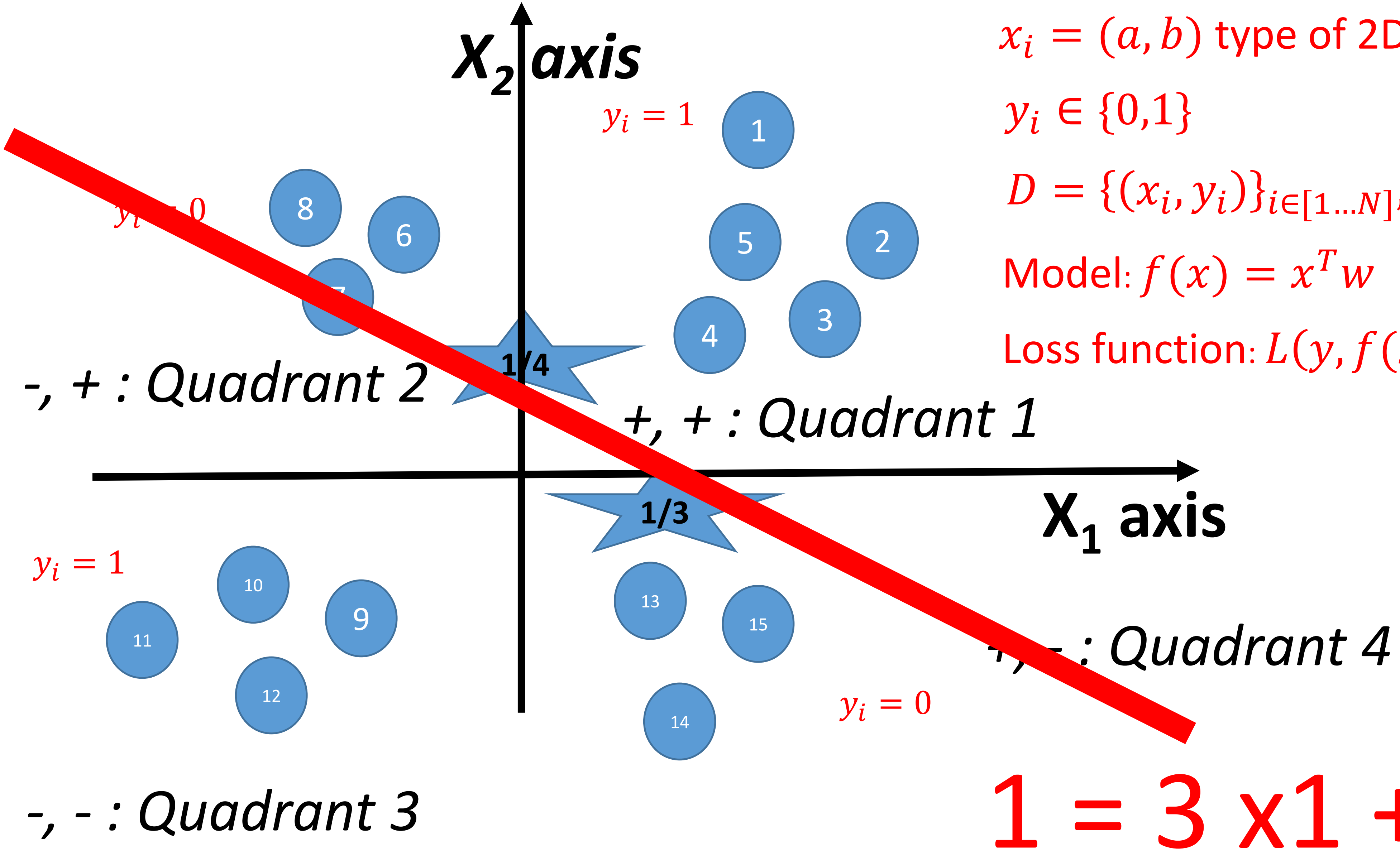
$x_i = (a, b)$ type of 2D point

$y_i \in \{0,1\}$

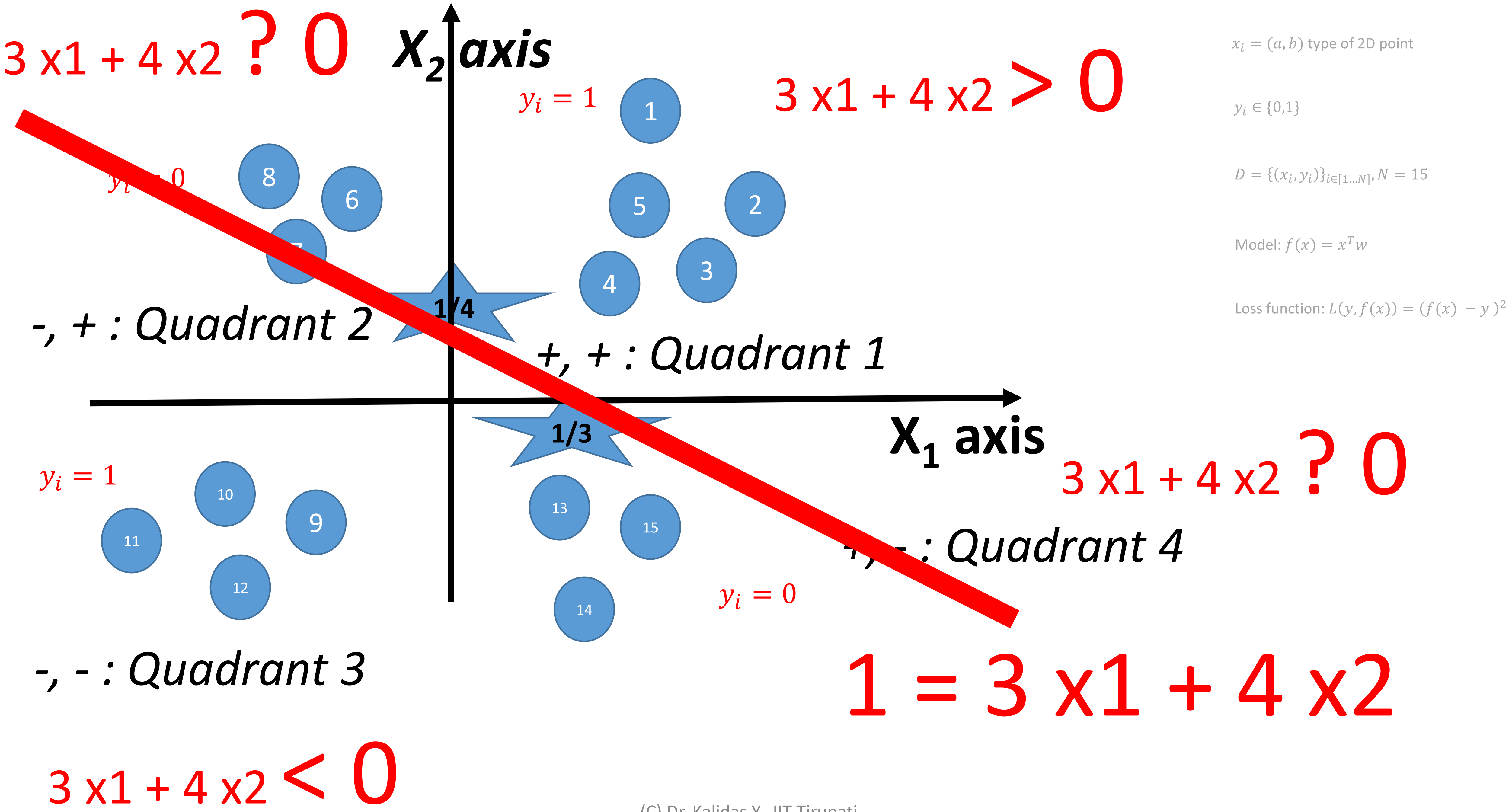
$D = \{(x_i, y_i)\}_{i\in[1\ldots N]}, N = 15$

Model: $f(x) = x^T w$

Loss function: $L(y, f(x)) = (f(x) - y)^2$



1 = 3 x1 + 4 x2



3 x1 + 4 x2 ? 0
X_2 axis
$y_i = 1$
3 x1 + 4 x2 > 0
$x_i = (a, b)$ type of 2D point
$y_i \in \{0,1\}$
$D = \{(x_i, y_i)\}_{i \in [1 \ldots N]}, N = 15$
Model: $f(x) = x^T w$
Loss function: $L(y, f(x)) = (f(x) - y)^2$
1/4
-, + : Quadrant 2
+, + : Quadrant 1
X_1 axis
1/3
3 x1 + 4 x2 ? 0
$y_i = 1$
: Quadrant 4
$y_i = 0$
-, - : Quadrant 3
1 = 3 x1 + 4 x2
3 x1 + 4 x2 < 0

See the beauty of *Feature Engineering*...

1. xi *is of type* (x[0],x[1])
2. yi *is of type* a single number 0 or 1
3. Data set is of type {(xi,yi)} i over 1 to 15 points
4. Model *is of type* f(x) = x dot w
   - w = (w[0], w[1])
   - f(xi) = w[0] * xi[0] + w[1] * xi[1]
5. Loss function *is of type* L(yi, f(xi)) = $(yi – f(xi))^2$

Where to engineer features???

See the beauty of *Feature Engineering*…

1. xi *is of type* (x[0],x[1]) **(x[0], x[1], <u>*1 if x[0]*x[1] > 0 else -1*</u>)**
2. yi *is of type* a single number 0 or 1
3. Data set is of type {(xi,yi)} i over 1 to 15 points
4. Model *is of type* f(x) = x dot w
   - w = (w[0], w[1])
   - f(xi) = w[0] * xi[0] + w[1] * xi[1]
5. Loss function *is of type* L(yi, f(xi)) = (yi – f(xi))^2

engineer a feature
which +1 in Quadrants 1 and 3
which is -1 in Quadrants 2 and 4

See the beauty of *Feature Engineering*… *xi' – "x i dash"*

Symbol to indicate feature transformation

1. xi *is of type* (x[0],x[1]) ⊢ (x[0], x[1], ***1 if x[0]*x[1] > 0 else -1***)
2. yi *is of type* a single number 0 or 1
3. Data set is of type {(xi,yi)} i over 1 to 15 points
4. Model *is of type* f(x) = x dot w
   - w = (w[0], w[1])
   - f(xi) = w[0] * xi[0] + w[1] * xi[1]
5. Loss function *is of type* L(yi, f(xi)) = $(yi – f(xi))^2$

engineer a feature
which +1 in Quadrants 1 and 3
which is -1 in Quadrants 2 and 4

See the beauty of *Feature Engineering*…

1. xi *is of type* (x[0],x[1]) $\vdash$ $(1\ if\ x[0] * x[1] > 0\ else\ 0) = xi'$
2. yi *is of type* a single number 0 or 1
3. Data set is of type {(xi,yi)} i over 1 to 15 points
4. Model *is of type* f(x’) = x’ dot w
   - w = (w[0], ~~w[1]~~)
   - f(xi’) = w[0] * xi’[0] ~~+ w[1] * xi[1]~~
5. Loss function *is of type* L(yi, f(xi’)) = (yi – f(xi’))^2

Where to engineer features???

xi’

is the engineered feature vector

33) key phrase… "feature reduction" *[will see more later]*

- Transform k dimensional vector to lower dimensional vector
- Example
  - Consider a gray image 1000x1000 pixels
  - Input = 10,00,000 (in our words, 10 lakh dimensions)
  - Transform it into 2 dimensional point!
- *Very Easy To Do! than you might have thought!!*

34) key phrase… "PCA transformation"

- More about this later on… in the *unsupervised learning* classes

33) key phrase… "pipeline of transformations"

- xi
- Transform xi to xi' using transformation 1
- Transform xi' to xi'' using transformation 2
- … and so on…
- Transform using transformation n

- **Bundle up all transformations** into a pipeline

```
def pipeline (x) :
  t1 = transformation1( x )
  t2 = transformation2( t1 )
  t3 = transformation3( t2 )
  t4 = transformation4( t3 )
 return t4
```

Challenges…

34) key phrase… “homogenous features”

Example, Image pixels

- All pixels have same meaning
- They capture intensity
- All those tiny devices are all

created using similar processes

Same type of features

homogenous features

They are comparatively easier to handle!

most of the deep neural networks require “homogenous features”

35) key phrase… “non-homogenous features”

Bike quality assessment

- Distance travelled – kilometres
- Year of purchase – date type
- Model type – text
- Previous repairs – description
- Any accidents – yes/no

Different types of features

non-homogenous features

They are *very tricky* to handle!

choice of them they directly affect performance and decide what model types to be used

36) key phrase… "feature correlation"

- It corresponds to relationships between features
- Some features may be derived versions of others
- *This is a problem in case of regression based methods (including deep networks)*
- *In case of tree based methods… it does not matter!*

37) key phrase… “curse of dimensionality”

- When there are several thousands of features
- Typically discussed in the context of text features
- *This is THE DIFFERENCE between “Human understanding vs machine programming”*
  - Humans need more dimensions, machines need less dimensions
  - For example, a good lecture, “touches upon related concepts”, you will understand well
  - When two people meet, they try to find common interests
  - All commercial advertisements, present a context and then the product
  - When conveying a point, you build up the context
- To get an intuition behind this statement:
  - For example, when dimensionality is high, distance between (1,1,….,1) all 1’s to *any point is almost same*
  - It relates to depends on ***all distances being almost same***, with very minute difference in 5th or 6th digits after decimal point
  - Repercussion – Data becomes *sensitive to location of origin* and translation affects results or models