Gentlest Intro to Tensorflow (Part 3)

Khor Soon Hin, @neth_6, re:Culture

In collaboration with Sam & Edmund

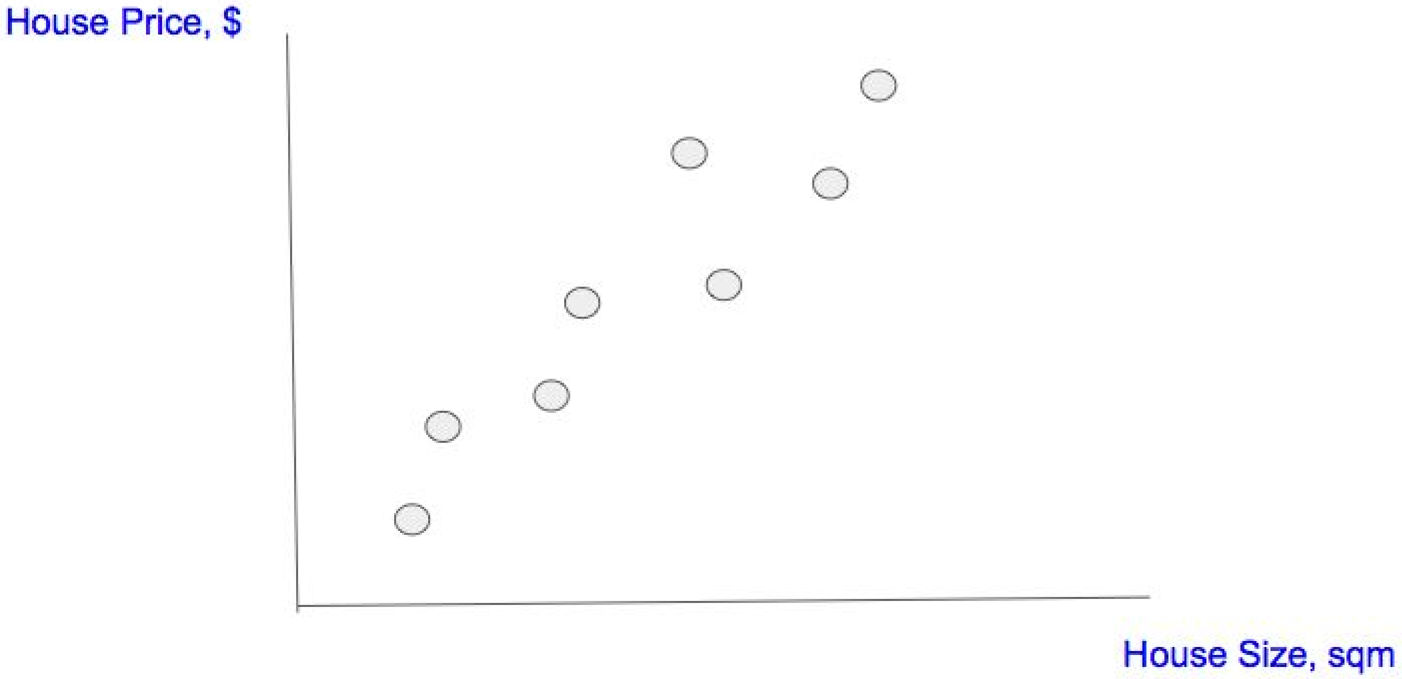
AheadConsulting

Overview

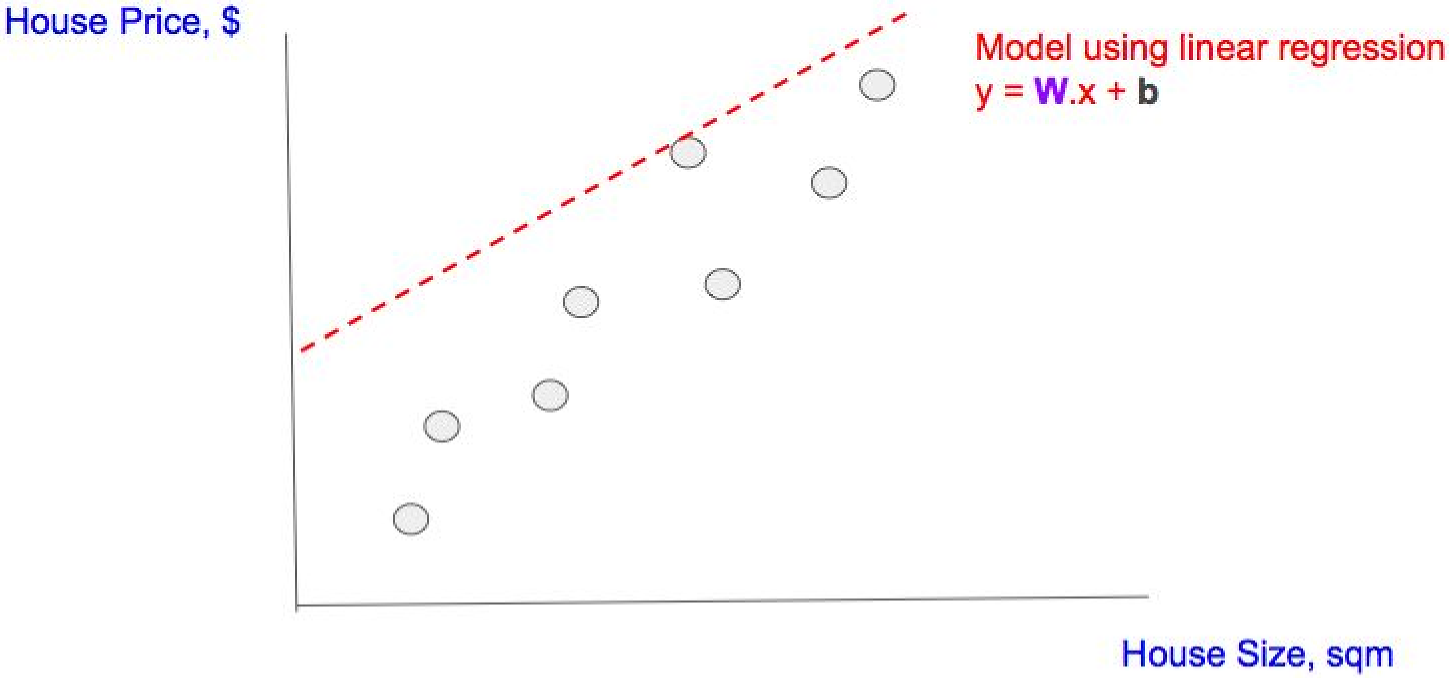
- Multi-feature Linear Regression
- Logistic Regression
  - Multi-class prediction
  - Cross-entropy
  - Softmax
- Tensorflow Cheatsheet #1

Review: Predict from Single Feature with Linear Regression

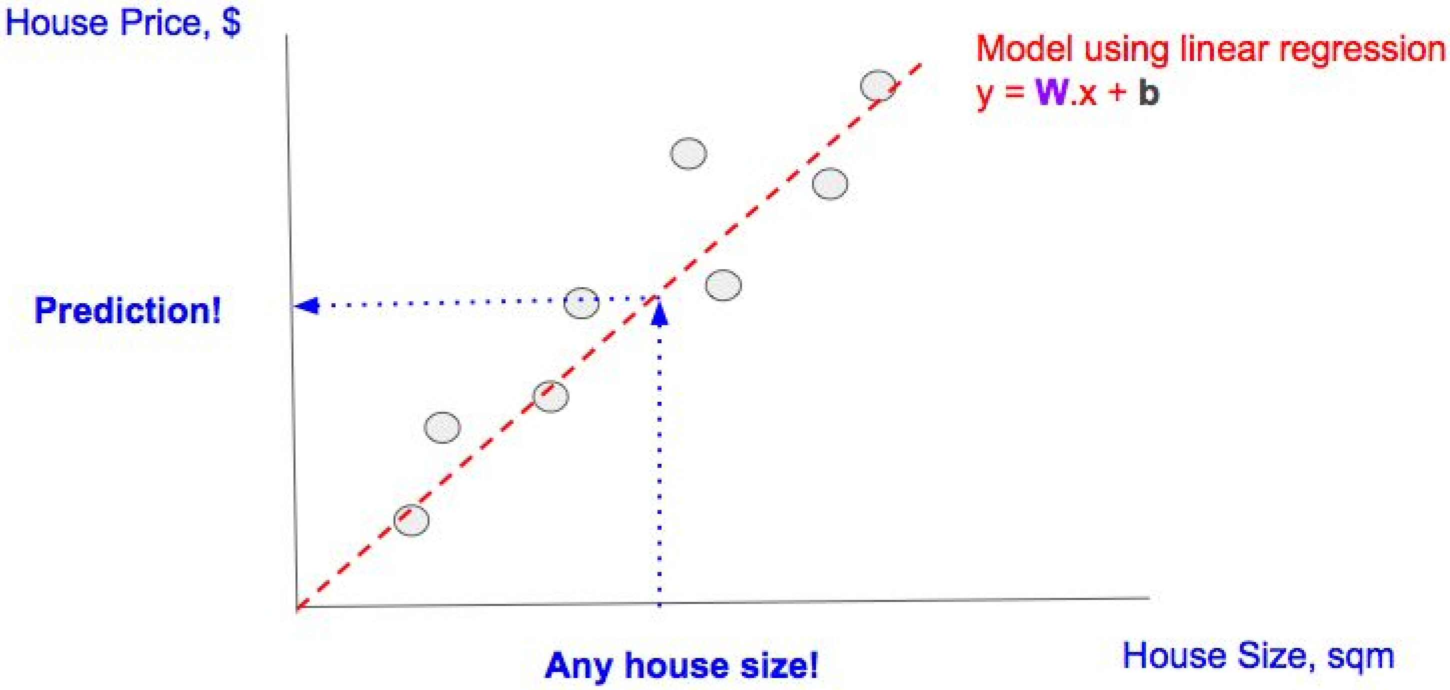
Quick Review: Predict from Single Feature (House Size)



Quick Review: Use Linear Regression

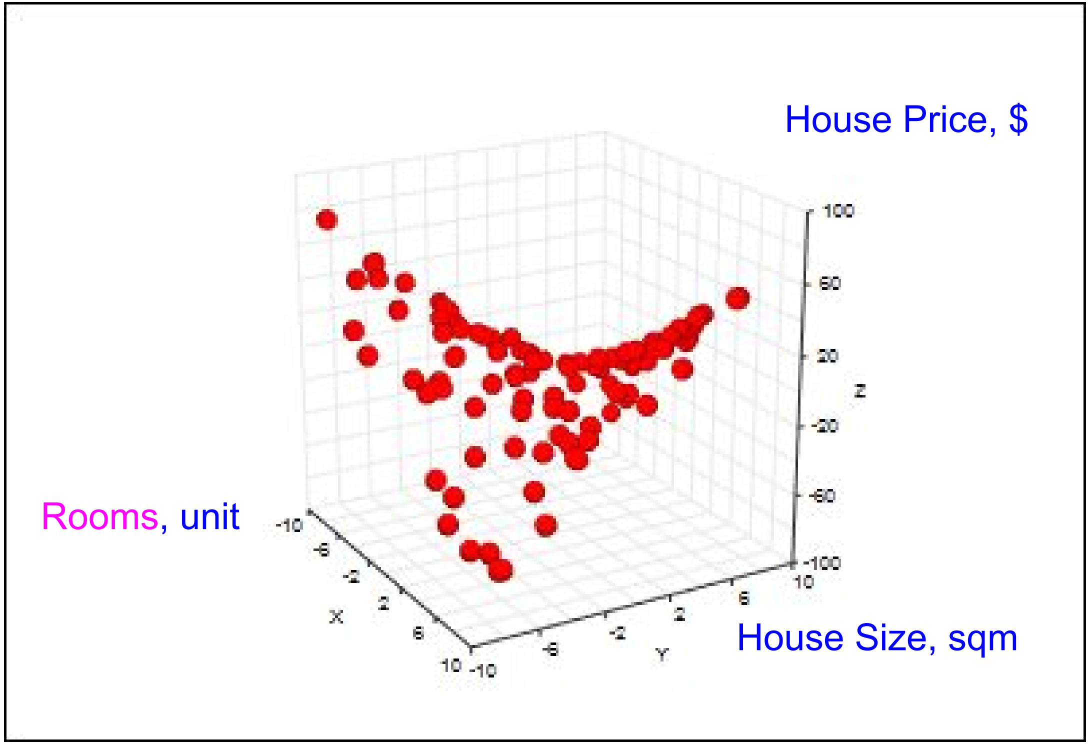


Quick Review: Predict using Linear Regression



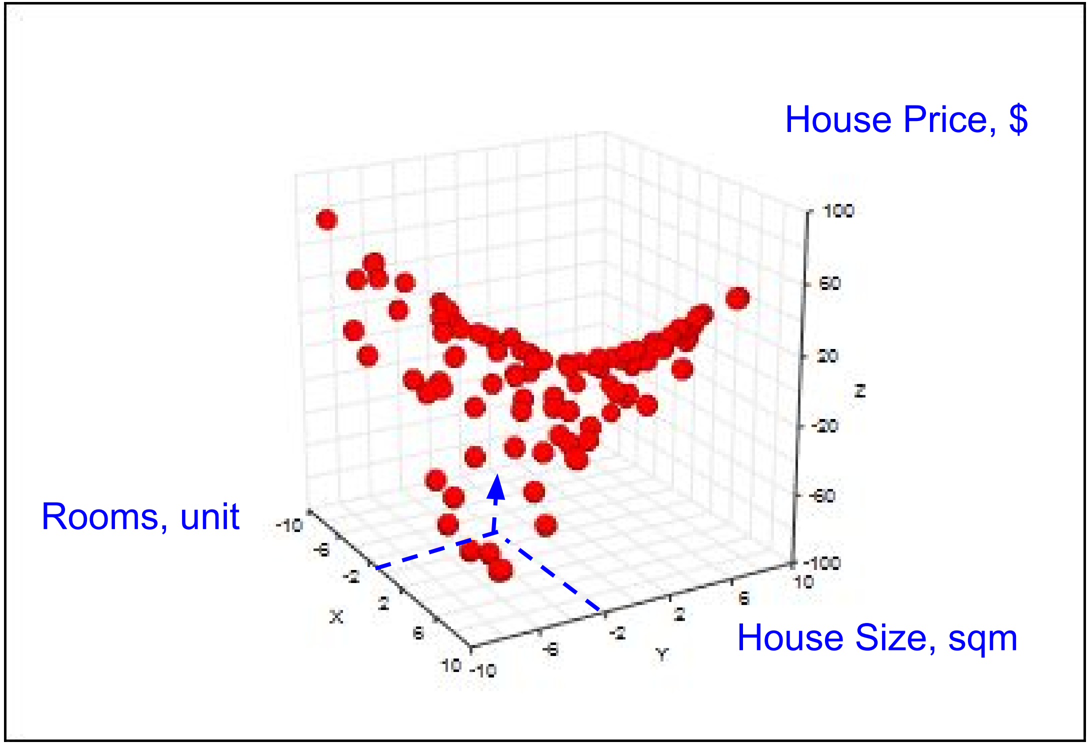
Linear Regression: Predict from Two (or More) Features

Two Features: House Size, Rooms



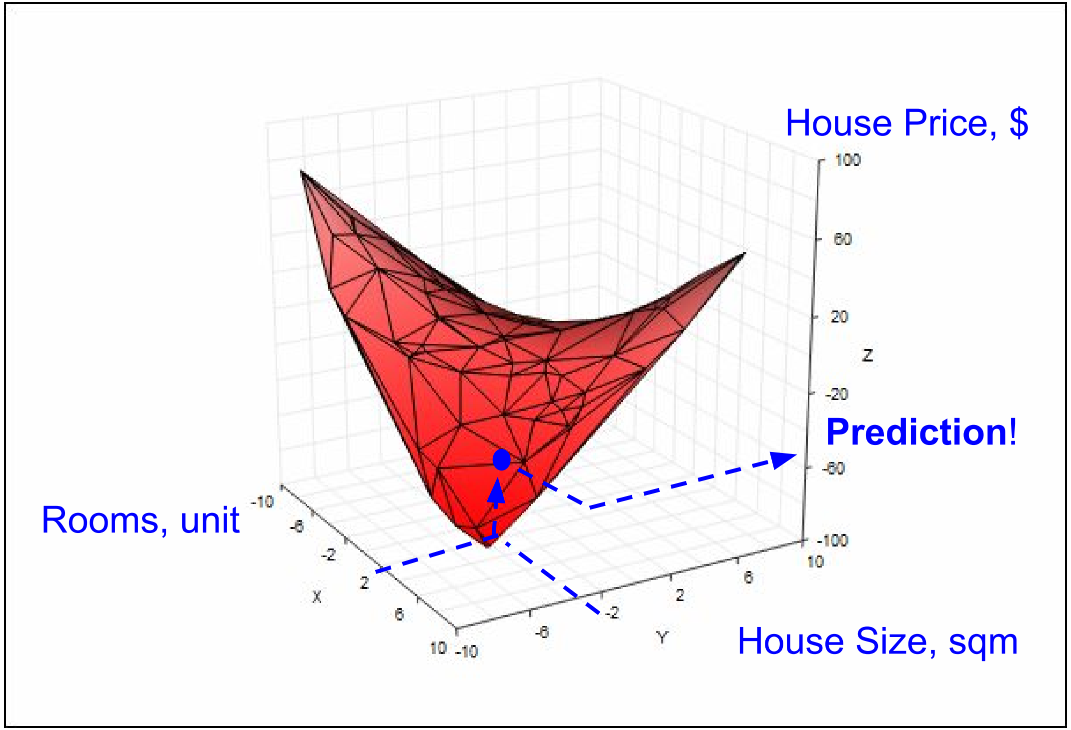
Source: teraplot.com

Same Issue: Predict for Values without Datapoint



Source: teraplot.com

Same Solution: Find Best-Fit



Source: teraplot.com

Review: Tensorflow Code

Tensorflow Code

```
# Model linear regression y = Wx + b
x = tf.placeholder(tf.float32, [None, 1])
W = tf.Variable(tf.zeros([1,1]))
b = tf.Variable(tf.zeros([1]))
product = tf.matmul(x,W)
y = product + b
y_ = tf.placeholder(tf.float32, [None, 1])

# Cost function 1/n * sum((y_-y)**2)
cost = tf.reduce_mean(tf.square(y_-y))

# Training using Gradient Descent to minimize cost
train_step = tf.train.GradientDescentOptimizer(0.0000001).minimize(cost)
```

Multi-feature: Change in Model & Cost Function

Model

1 Feature

y = W.x + b

y: House price prediction

x: House size

Goal: Find scalars **W,b**

Model

1 Feature

$y = W.x + b$

y: House price prediction

x: House size

Goal: Find scalars **W,b**

2 Features

$y = W.x + W2.x2 + b$

y: House price prediction

x: House size

x2: Rooms

Goal: Find scalars **W, W2, b**

Tensorflow Graph

1 Feature

```
y = tf.matmul(x, W) + b
```

```
W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])
```

```
x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

Tensorflow Graph

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

2 Features

```
y = matmul(x, W) + matmul(x2, W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

Tensorflow Graph: Train

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

```
Train: feed = { x: … , y_: … }
```

2 Features

```
y = matmul(x, W) + matmul(x2, W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

```
Train: feed = { x: … , x2: ... y_: … }
```

Tensorflow Graph: Scalability Issue

Model gets messy!

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

```
Train: feed = { x: … , y_: … }
```

2 Features

```
y = tf.matmul(x, W) + tf.matmul(x2, W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

```
Train: feed = { x: … , x2: ... y_: … }
```

3 Features

```
y = tf.matmul(x, W) + tf.matmul(x2, W2) + tf.matmul(x3, W3) +b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
W3 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
x3 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

```
Train: feed = { x: … , x2: ... , x3: … , y_: … }
```

Data Representation

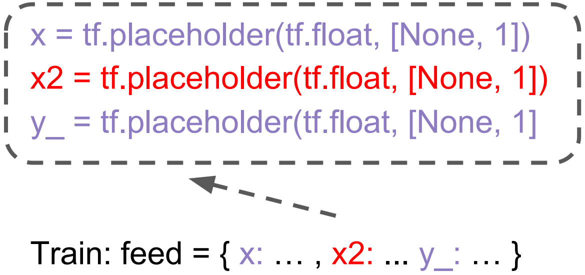
	Feature values		Actual outcome
House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

Lots of Data Manipulation

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

2 Features

```
y = tf.matmul(x, W) + tf.matmul(x2,
W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])
```



```
x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

Train: feed = { x: … , x2: ... y_: … }

Lots of Data Manipulation 2

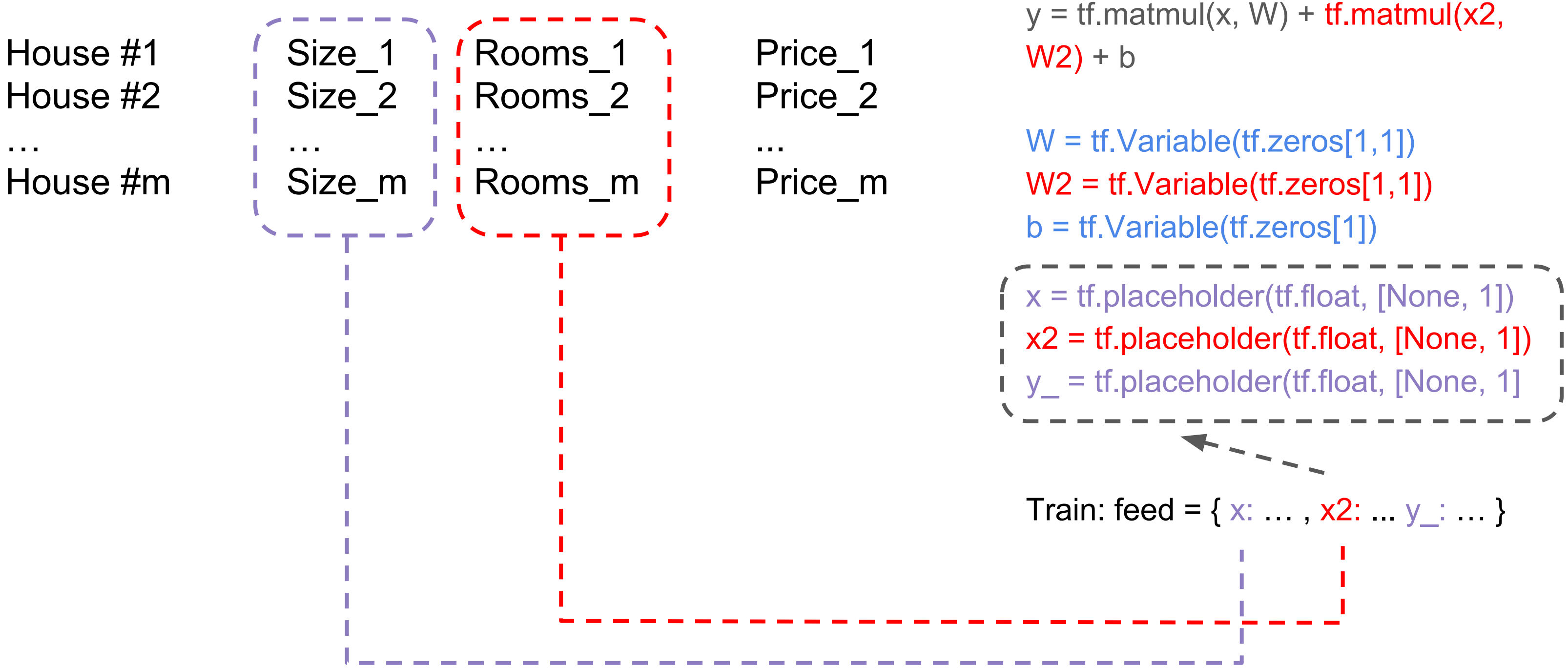
House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

2 Features

```
y = tf.matmul(x, W) + tf.matmul(x2,
W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]

Train: feed = { x: … , x2: ... y_: … }
```

Lots of Data Manipulation 3

2 Features



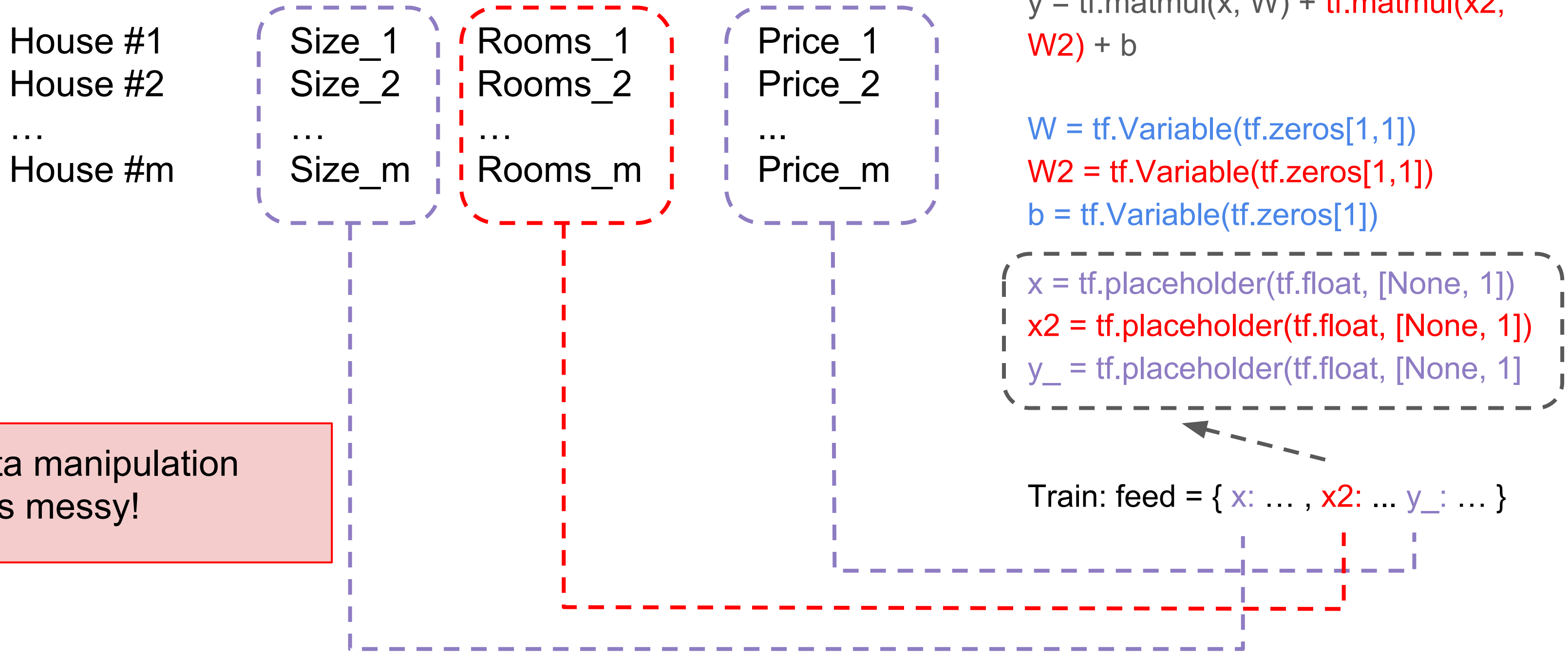
House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

```
y = tf.matmul(x, W) + tf.matmul(x2,
W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]

Train: feed = { x: … , x2: ... y_: … }
```

Lots of Data Manipulation 4

2 Features

```
y = tf.matmul(x, W) + tf.matmul(x2, W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```



House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

Train: feed = { x: … , x2: ... y_: … }

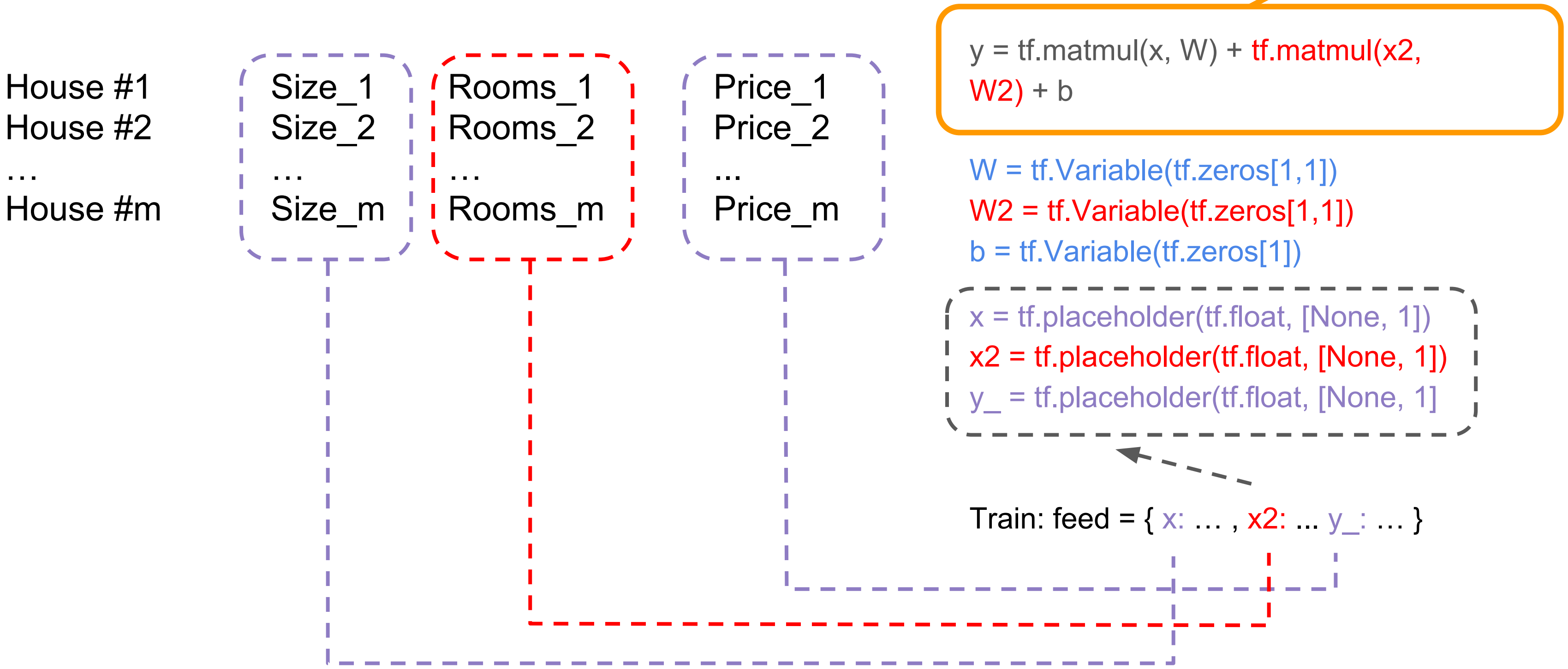
Data manipulation gets messy!

Lots of Data Manipulation 4

Find better way

y = W.x + W2.x2 + b

2 Features



House #1
House #2
…
House #m

Size_1
Size_2
…
Size_m

Rooms_1
Rooms_2
…
Rooms_m

Price_1
Price_2
...
Price_m

```
y = tf.matmul(x, W) + tf.matmul(x2,
W2) + b
```

```
W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])
```

```
x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

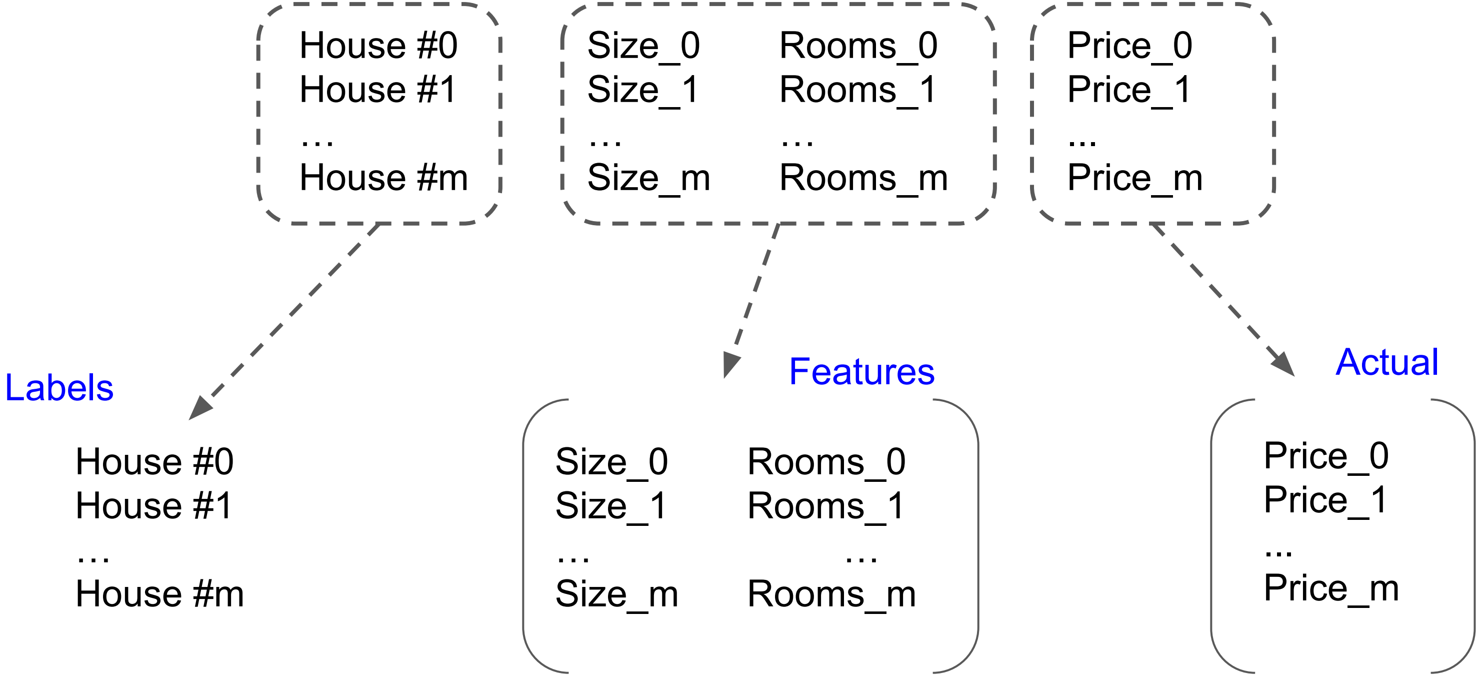
Train: feed = { x: … , x2: ... y_: … }

Matrix: Cleaning Up Representations

Matrix Representation

House #0	Size_0	Rooms_0	Price_0
House #1	Size_1	Rooms_1	Price_1
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

Matrix Representation



House #0
House #1
…
House #m

Size_0 Rooms_0
Size_1 Rooms_1
… …
Size_m Rooms_m

Price_0
Price_1
...
Price_m

Labels

House #0
House #1
…
House #m

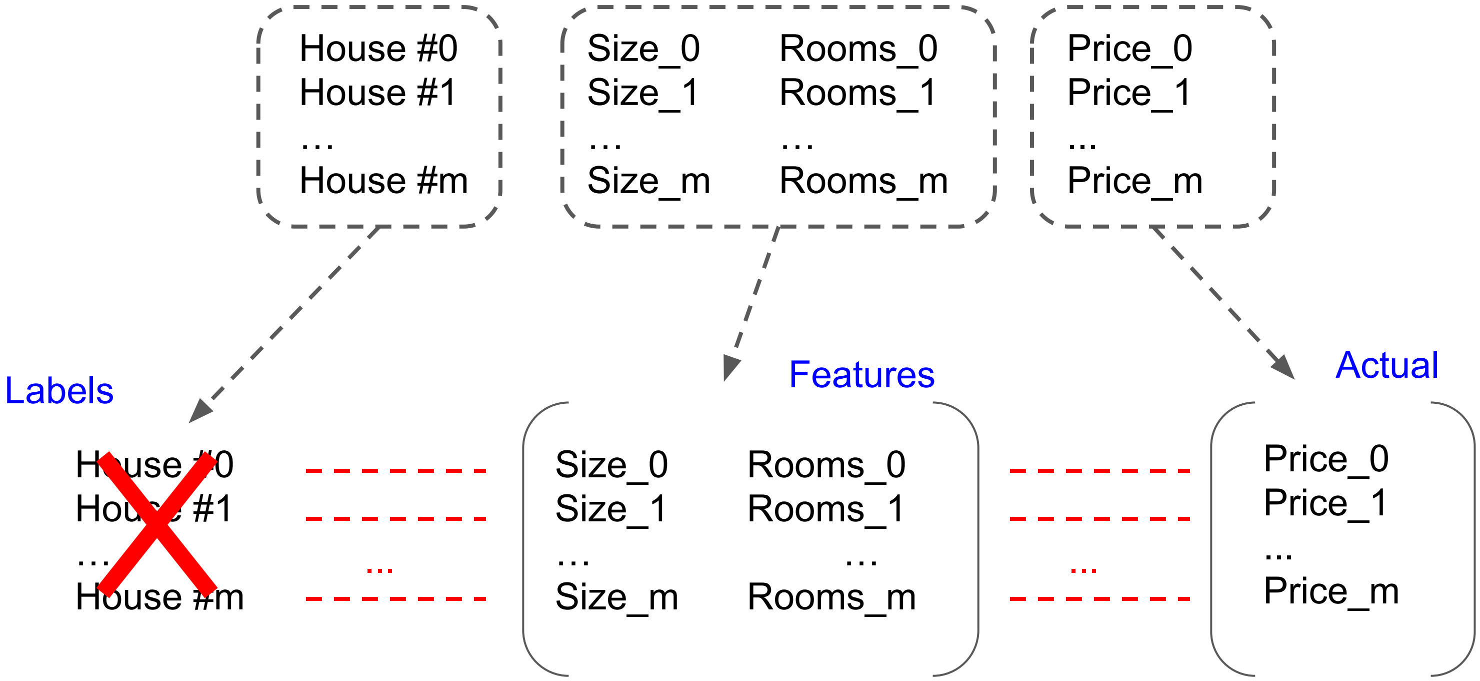
Features

Size_0 Rooms_0
Size_1 Rooms_1
… …
Size_m Rooms_m

Actual

Price_0
Price_1
...
Price_m

Matrix Representation



House #0
House #1
…
House #m

Size_0	Rooms_0
Size_1	Rooms_1
…	…
Size_m	Rooms_m

Price_0
Price_1
...
Price_m

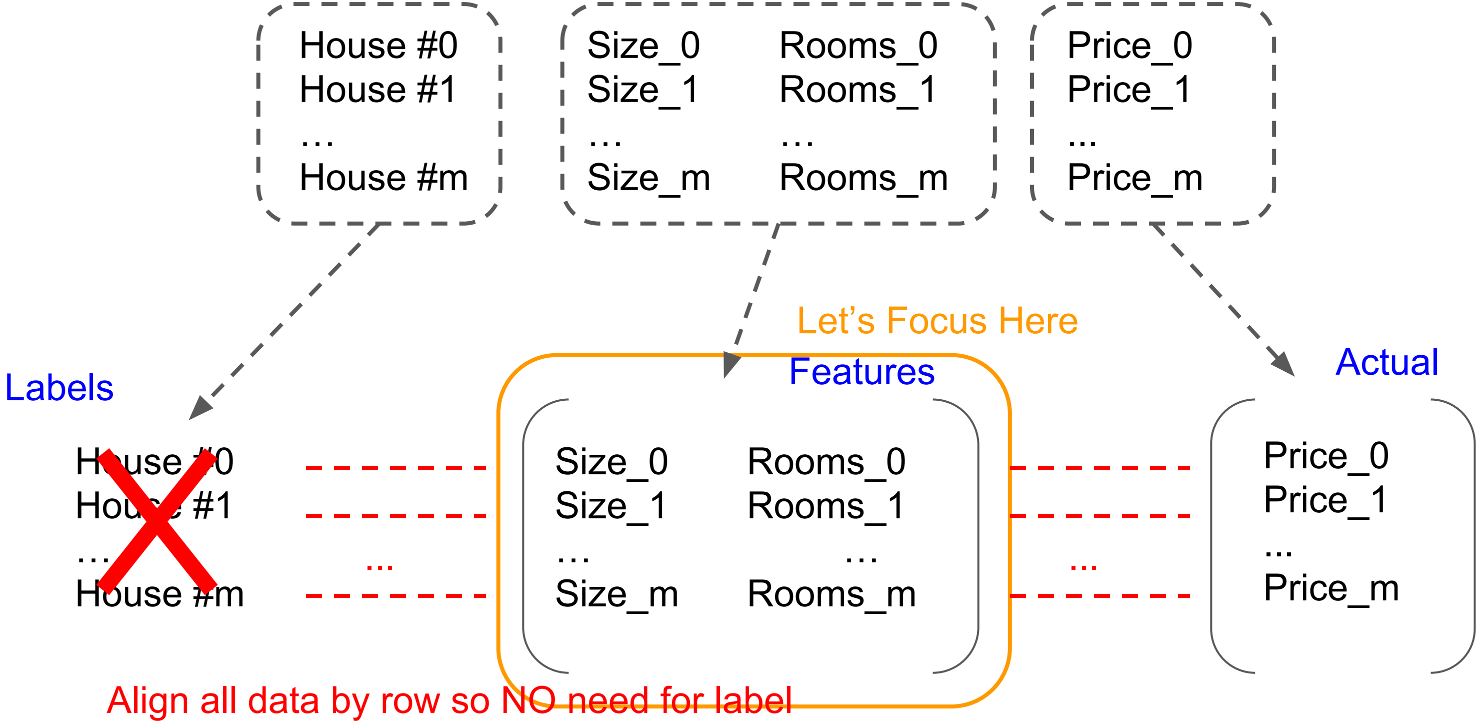
~~House #0~~
~~House #1~~
~~…~~
~~House #m~~

Size_0	Rooms_0
Size_1	Rooms_1
…	…
Size_m	Rooms_m

Price_0
Price_1
...
Price_m

Align all data by row so NO need for label

Matrix Representation



House #0
House #1
…
House #m

Size_0 Rooms_0
Size_1 Rooms_1
… …
Size_m Rooms_m

Price_0
Price_1
...
Price_m

Let's Focus Here

Labels

House #0
House #1
…
House #m

...

Features

Size_0 Rooms_0
Size_1 Rooms_1
… …
Size_m Rooms_m

...

Actual

Price_0
Price_1
...
Price_m

Align all data by row so NO need for label

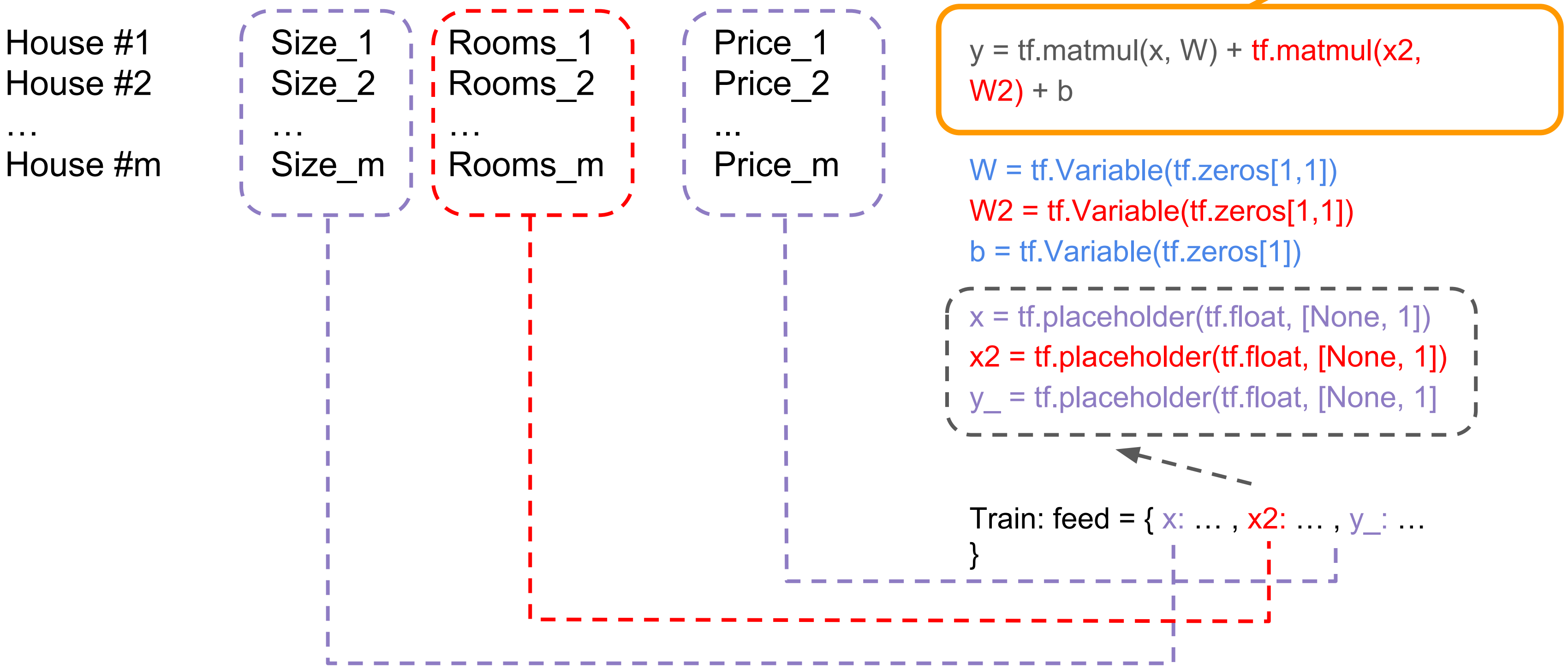
Matrix: Cleaning Up Models

Better Model Equation

Find better way

$y = W.x + W2.x2 + b$

2 Features



House #1
House #2
…
House #m

Size_1
Size_2
…
Size_m

Rooms_1
Rooms_2
…
Rooms_m

Price_1
Price_2
...
Price_m

```
y = tf.matmul(x, W) + tf.matmul(x2, W2) + b
```

```
W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])
```

```
x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

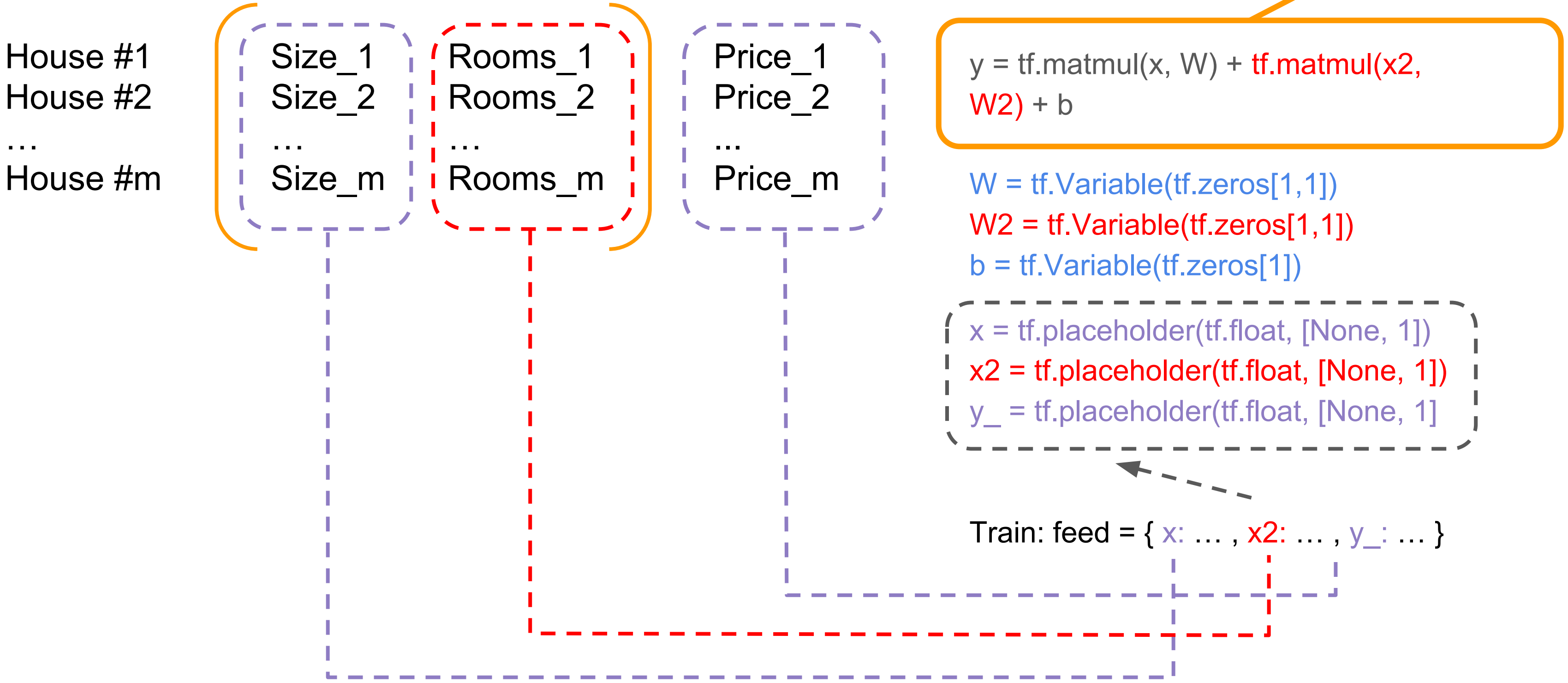
```
Train: feed = { x: … , x2: … , y_: …
}
```

Find better way

y = W.x + W2.x2 + b

Better Model Equation

2 Features



House #1
House #2
…
House #m

Size_1
Size_2
…
Size_m

Rooms_1
Rooms_2
…
Rooms_m

Price_1
Price_2
...
Price_m

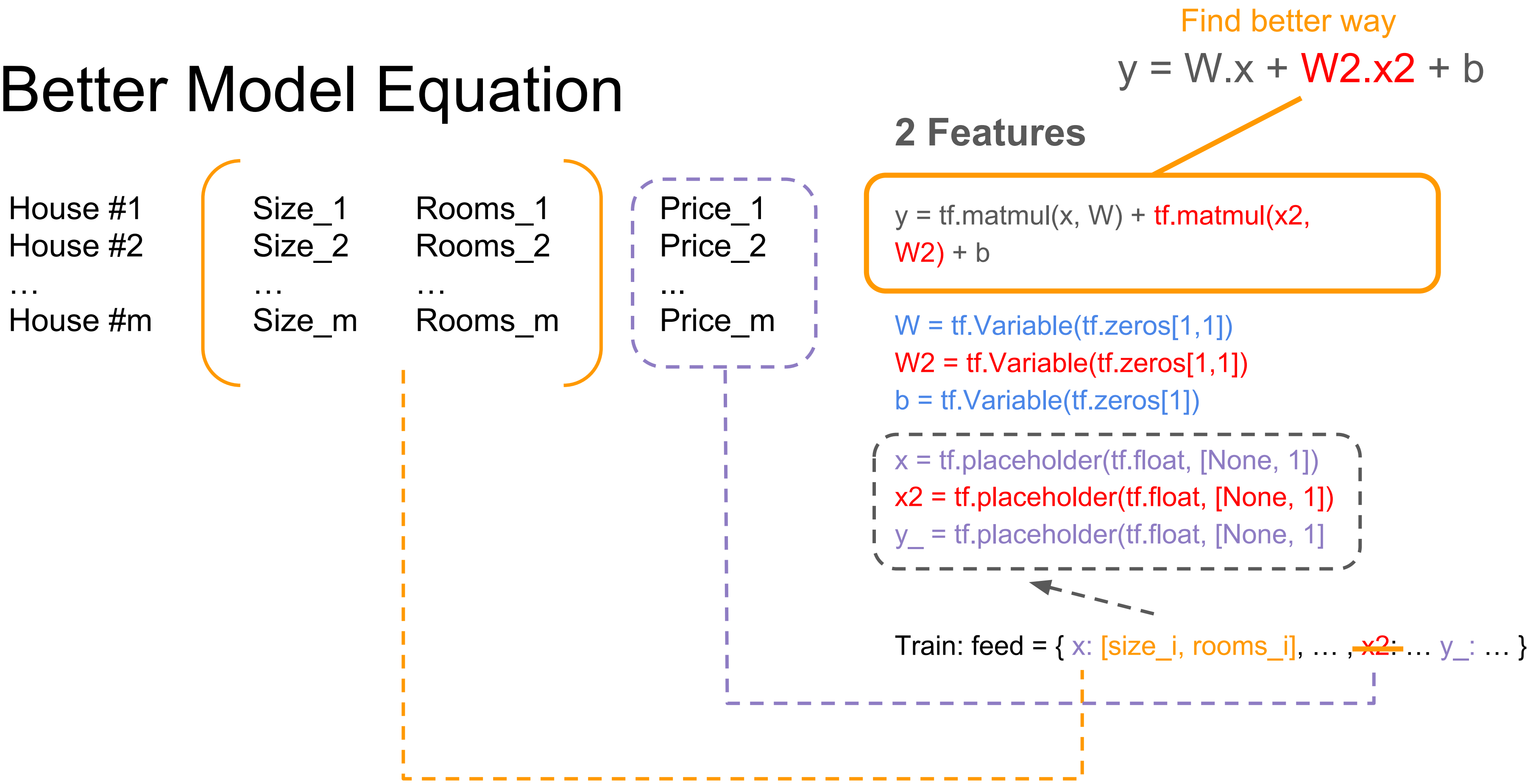
```
y = tf.matmul(x, W) + tf.matmul(x2, W2) + b
```

```
W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])
```

```
x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

Train: feed = { x: … , x2: … , y_: … }

Better Model Equation



Find better way

y = W.x + W2.x2 + b

2 Features

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

```
y = tf.matmul(x, W) + tf.matmul(x2,
W2) + b
```

```
W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])
```

```
x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1]
```

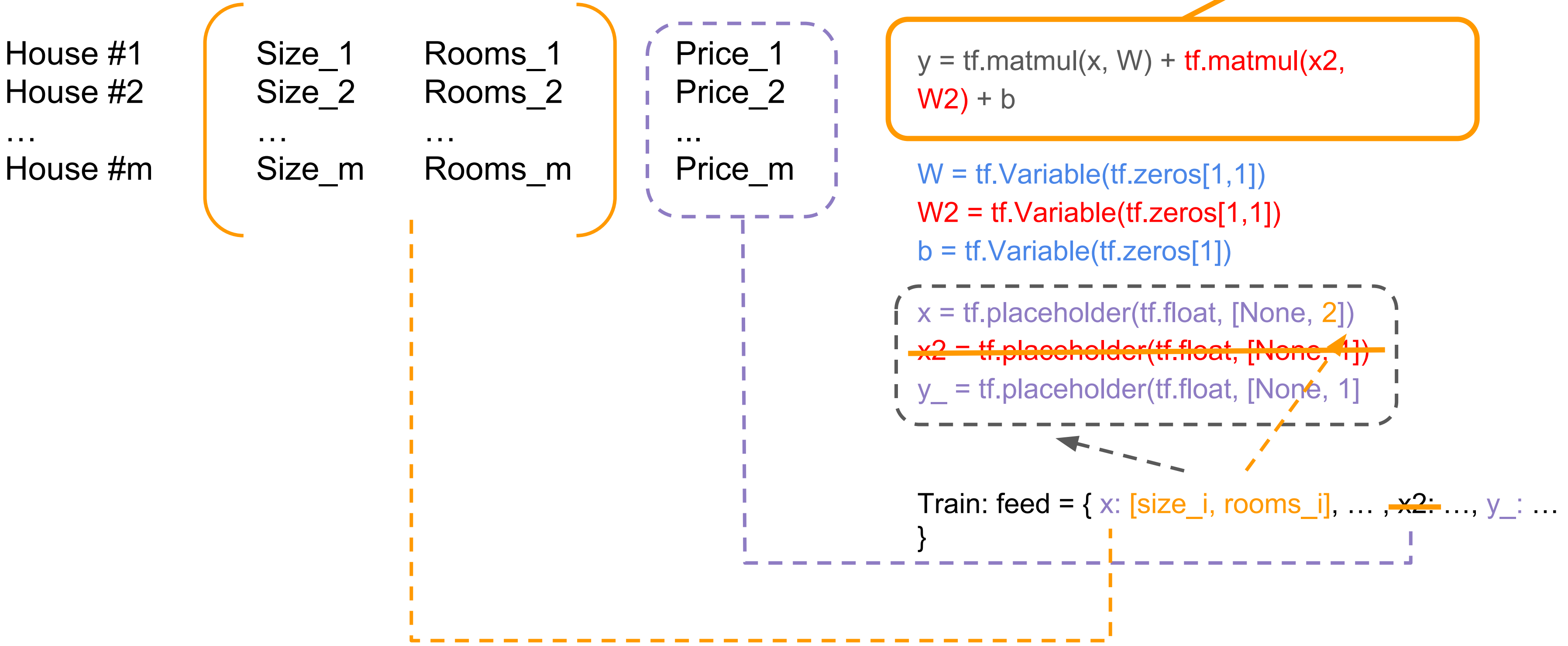
Train: feed = { x: [size_i, rooms_i], … , ~~x2:~~ … y_: … }

Better Model Equation

Find better way

$y = W.x + W2.x2 + b$

2 Features



House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

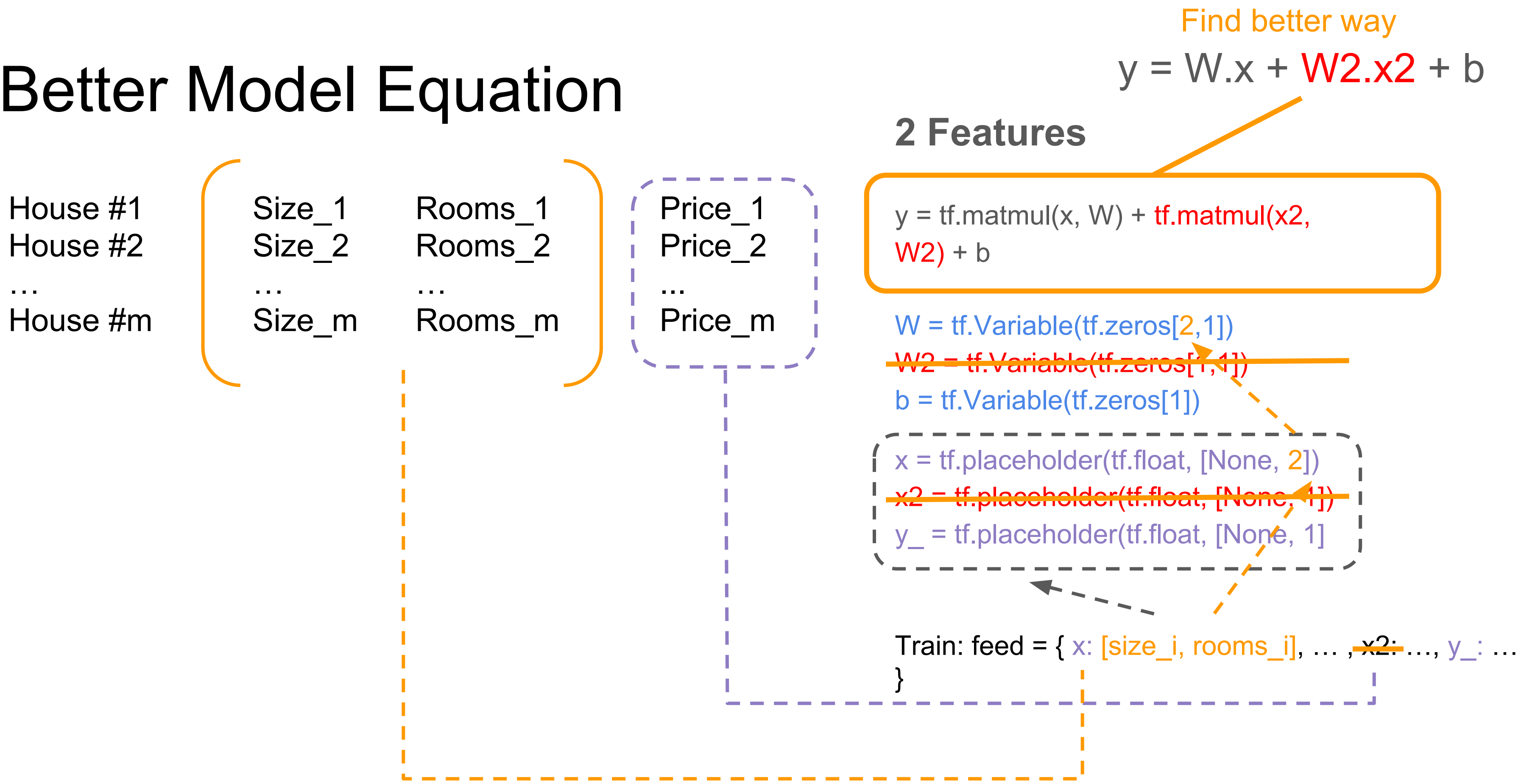
y = tf.matmul(x, W) + tf.matmul(x2, W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
~~x2 = tf.placeholder(tf.float, [None, 1])~~
y_ = tf.placeholder(tf.float, [None, 1]

Train: feed = { x: [size_i, rooms_i], … , ~~x2:~~ …, y_: …
}

Better Model Equation



Find better way

$y = W.x + W2.x2 + b$

2 Features

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

y = tf.matmul(x, W) + tf.matmul(x2, W2) + b

W = tf.Variable(tf.zeros[2,1])

~~W2 = tf.Variable(tf.zeros[1,1])~~

b = tf.Variable(tf.zeros[1])

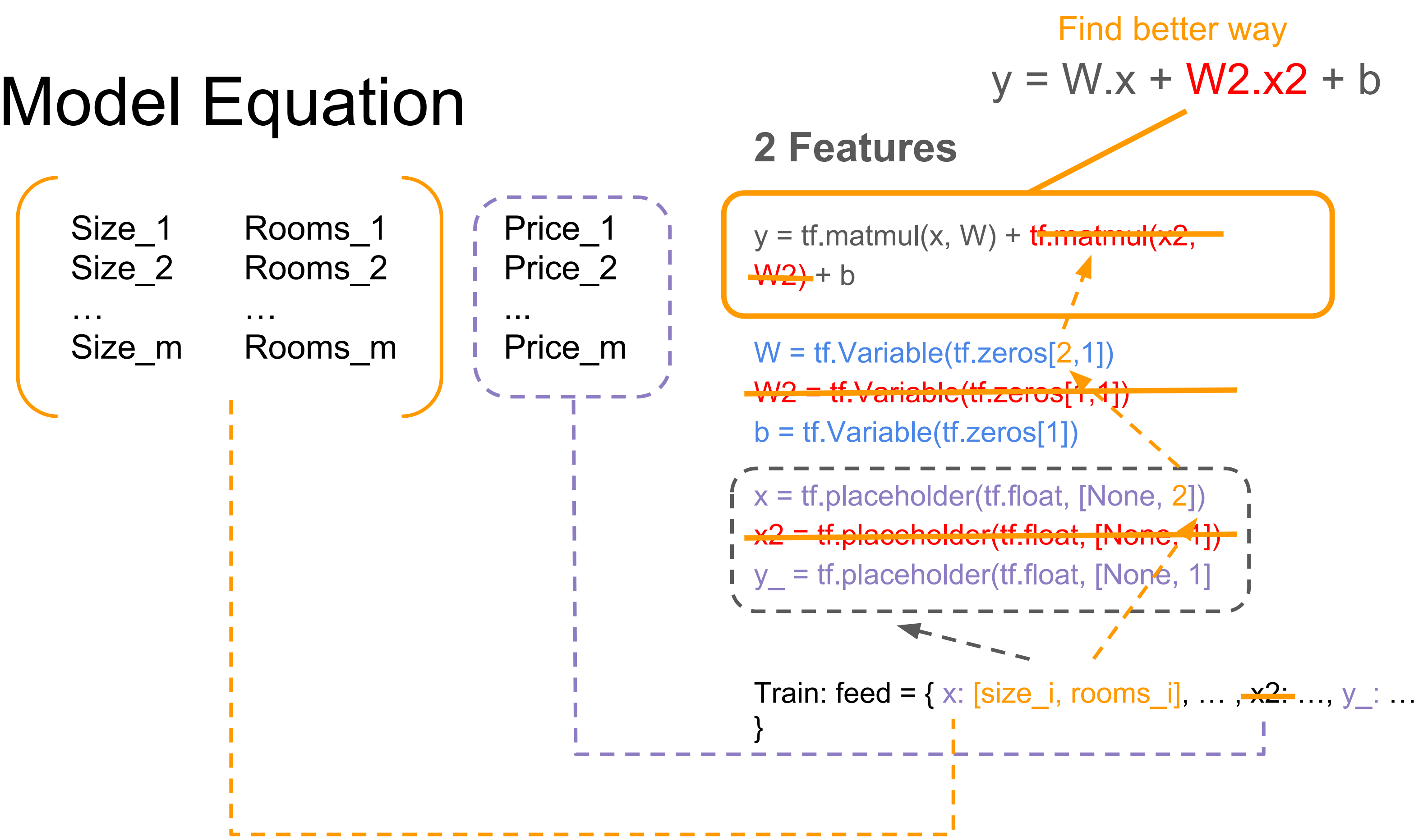
x = tf.placeholder(tf.float, [None, 2])

~~x2 = tf.placeholder(tf.float, [None, 1])~~

y_ = tf.placeholder(tf.float, [None, 1]

Train: feed = { x: [size_i, rooms_i], … , ~~x2:~~ …, y_: …
}

Better Model Equation



Find better way

y = W.x + W2.x2 + b

2 Features

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

y = tf.matmul(x, W) + ~~tf.matmul(x2, W2)~~ + b

W = tf.Variable(tf.zeros[2,1])

~~W2 = tf.Variable(tf.zeros[1,1])~~

b = tf.Variable(tf.zeros[1])

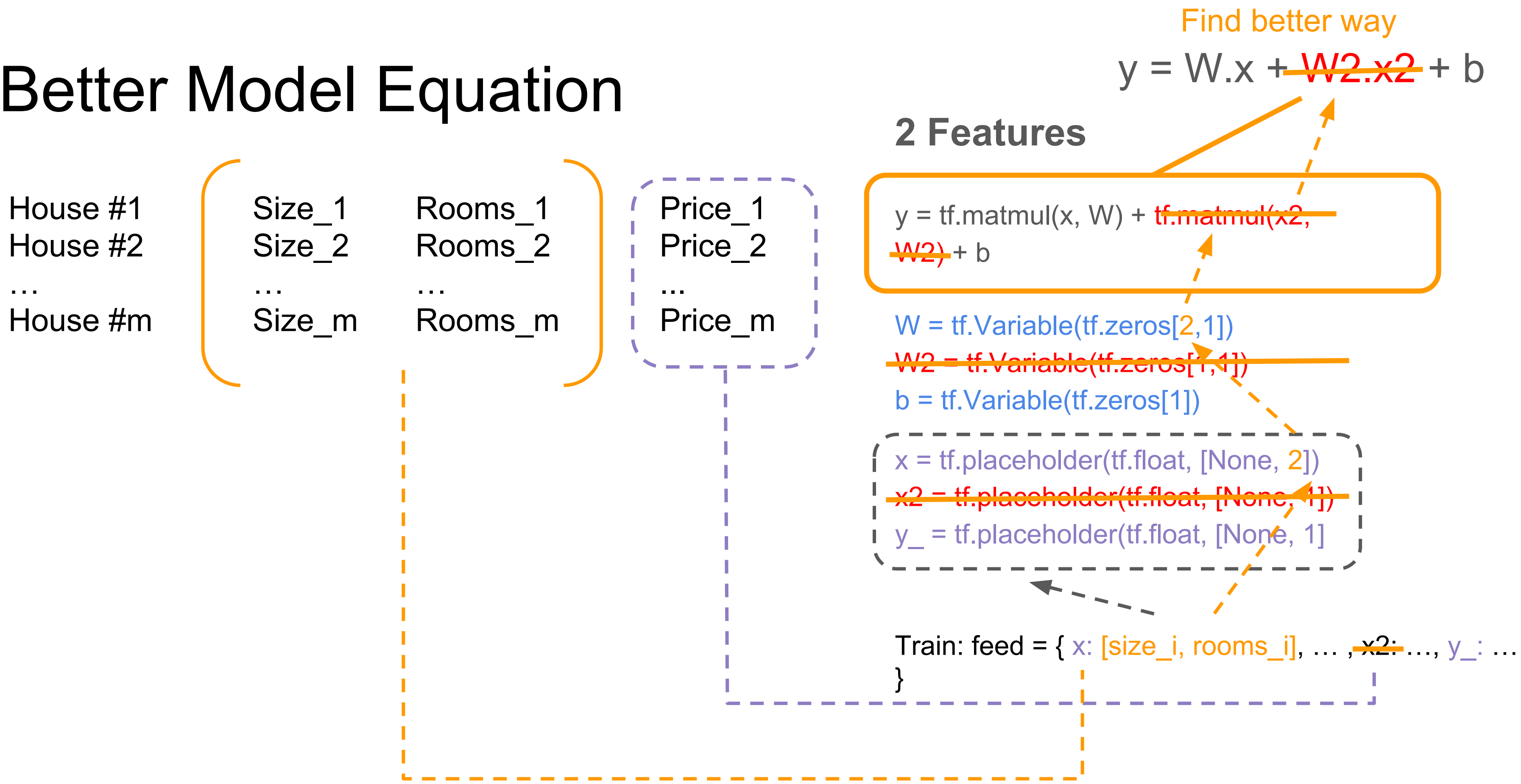
x = tf.placeholder(tf.float, [None, 2])

~~x2 = tf.placeholder(tf.float, [None, 1])~~

y_ = tf.placeholder(tf.float, [None, 1]

Train: feed = { x: [size_i, rooms_i], … ,~~x2:~~ …, y_: …
}

Better Model Equation



Find better way

y = W.x + ~~W2.x2~~ + b

2 Features

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

y = tf.matmul(x, W) + ~~tf.matmul(x2, W2)~~ + b

W = tf.Variable(tf.zeros[2,1])

~~W2 = tf.Variable(tf.zeros[1,1])~~

b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])

~~x2 = tf.placeholder(tf.float, [None, 1])~~

y_ = tf.placeholder(tf.float, [None, 1]

Train: feed = { x: [size_i, rooms_i], … , ~~x2:~~ …, y_: …
}

Tensorflow Graph (Messy)

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

2 Features

```
y = matmul(x, W) + matmul(x2, W2) + b

W = tf.Variable(tf.zeros[1,1])
W2 = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
x2 = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

Tensorflow Graph (Clean)

1 Feature

y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])

2 Features

y = matmul(x, W) + ~~matmul(x2, W2)~~ + b

W = tf.Variable(tf.zeros[2,1])
~~W2 = tf.Variable(tf.zeros[1,1])~~
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
~~x2 = tf.placeholder(tf.float, [None, 1])~~
y_ = tf.placeholder(tf.float, [None, 1])

Tensorflow Graph (Clean and Formatted)

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

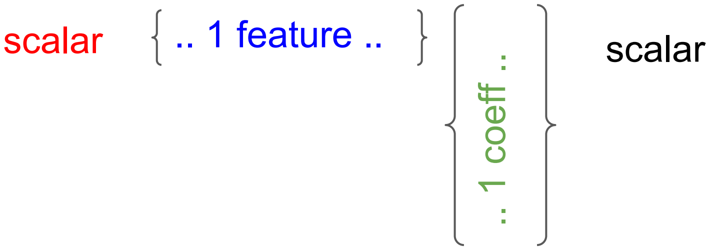
2 Features

```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 1])
```

Tensorflow Graph (Illustration)

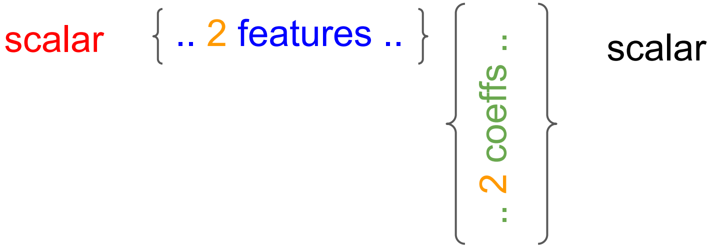
1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```



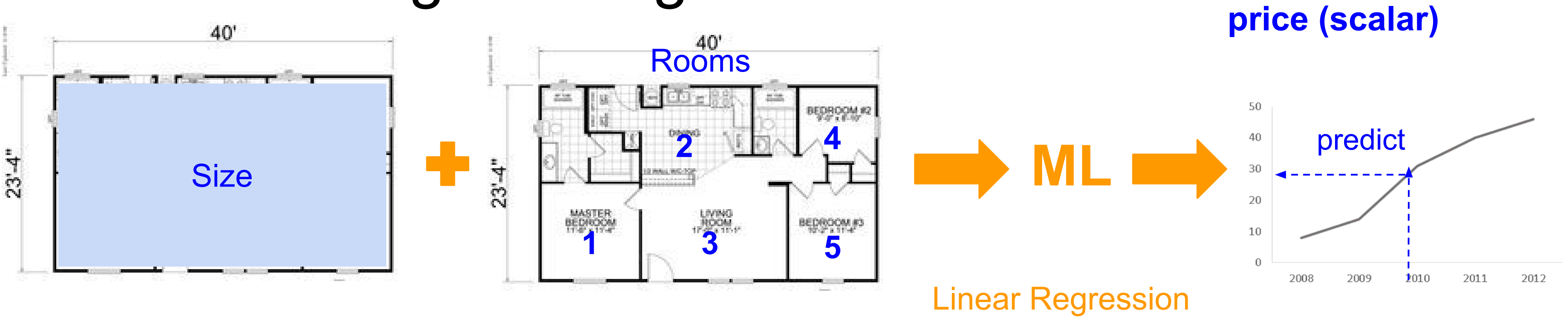
2 Features

```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 1])
```



Logistic Regression

Linear vs. Logistic Regression



Size

+

Rooms

1 2 3 4 5

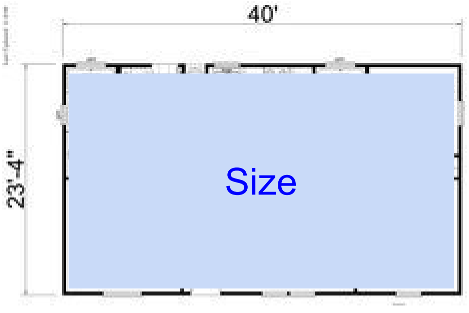
ML

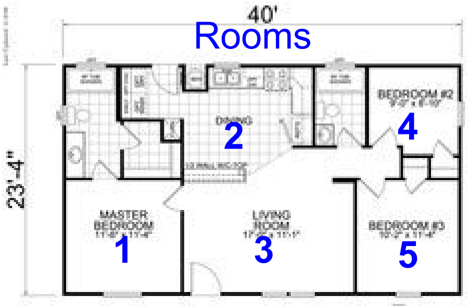
price (scalar)

predict

Linear Regression

Linear vs. Logistic Regression

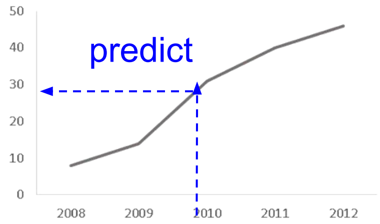




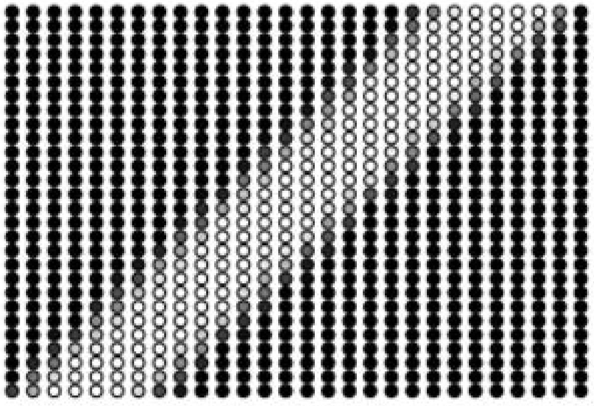
ML

Linear Regression

price (scalar)



Image



ML

Logistic Regression

number (discrete classes)

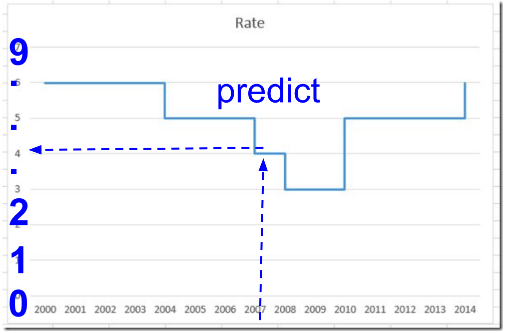
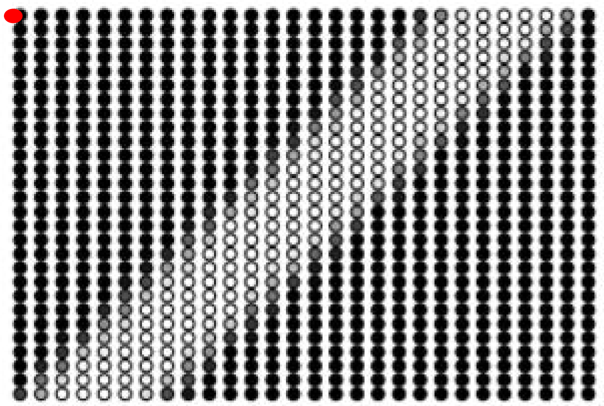


Image Features

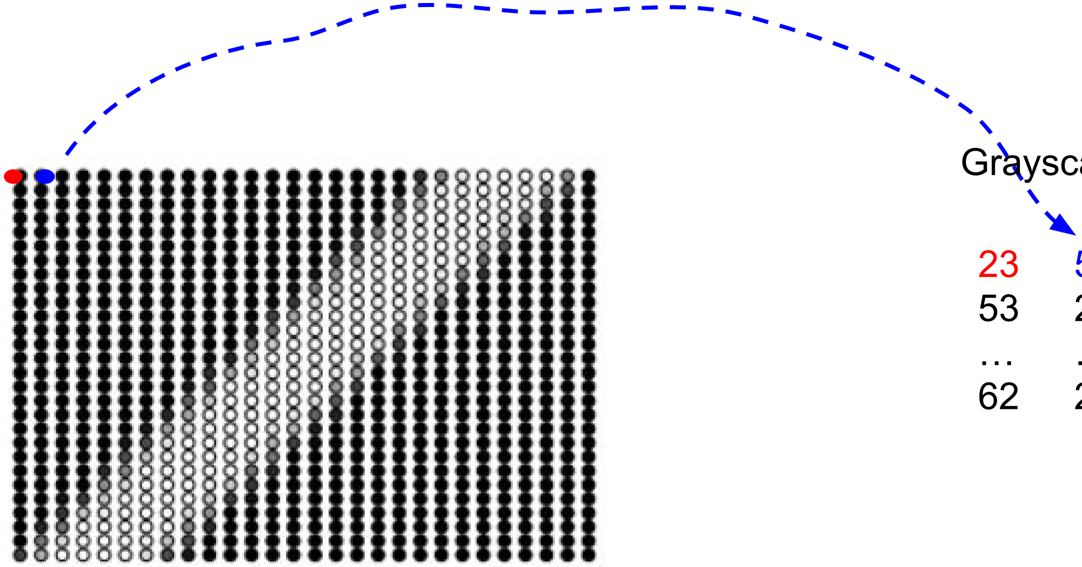
Image Features 1



Grayscale value of each pixel

23	53	…	33
53	20	…	88
…	…	…	....
62	2	…	193

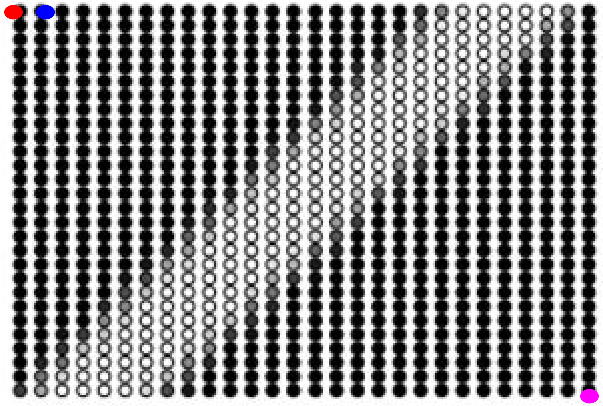
Image Features 2



Grayscale value of each pixel

23	53	…	33
53	20	…	88
…	…	…	....
62	2	…	193

Image Features 3



Grayscale value of each pixel

23	53	…	33
53	20	…	88
…	…	…	....
62	2	…	193

Logistic Regression: Change in Models

Model

Linear Regression

$y = W.x + b$

y: House price (scalar) prediction

x: [House size, Rooms]

Goal: Find scalars **W,b**

Logistic Regression

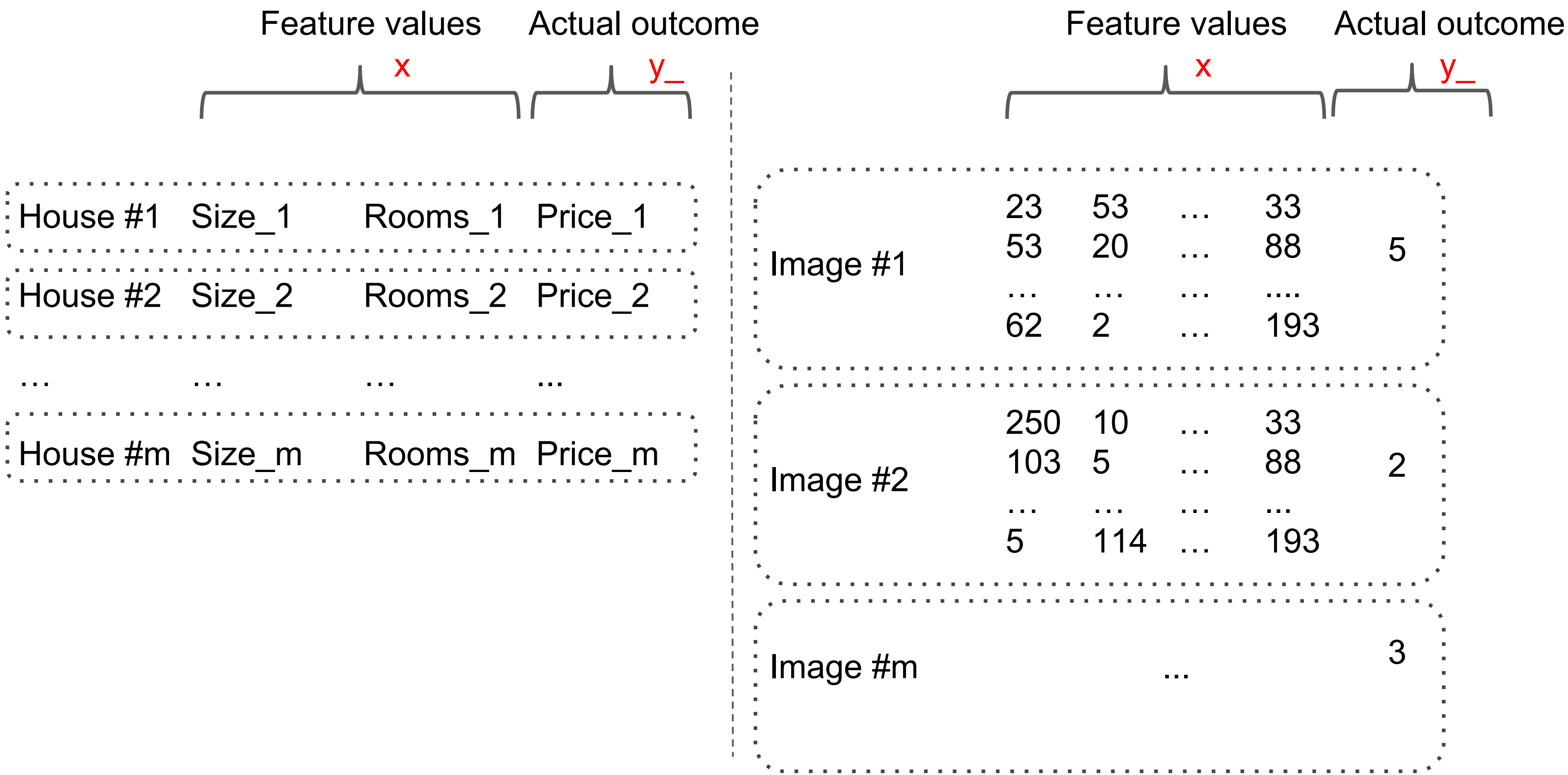
$y = W.x + b$

y: Discrete class [0,1,...9] prediction

x: [2-Dim pixel grayscale colors]

Goal: Find scalars **W, b**

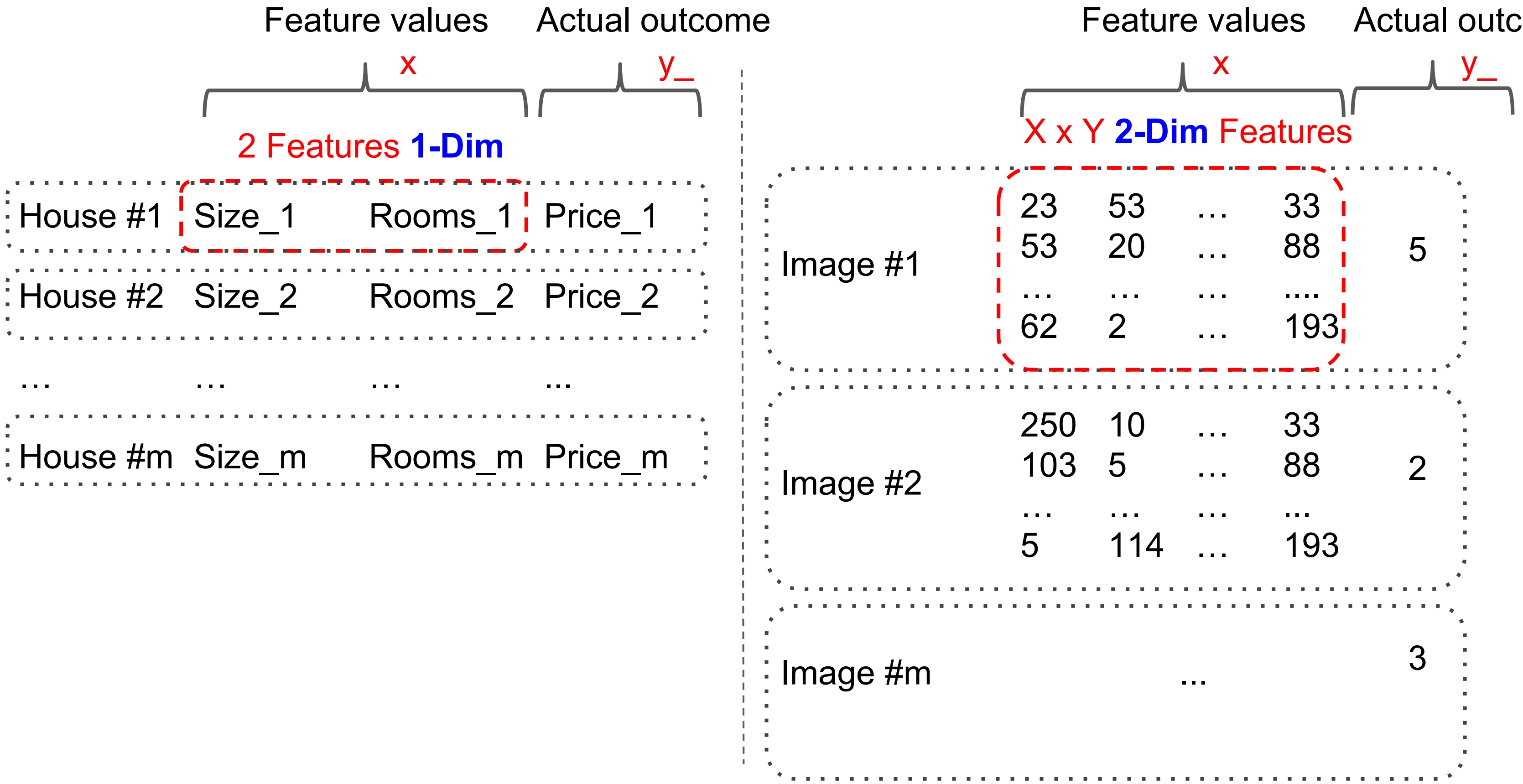
Data Representation Comparison



	Feature values x		Actual outcome y_
House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

	Feature values x	Actual outcome y_
Image #1	23 53 … 33 53 20 … 88 … … … .... 62 2 … 193	5
Image #2	250 10 … 33 103 5 … 88 … … … ... 5 114 … 193	2
Image #m	...	3

Data Representation Comparison



Feature values x

Actual outcome y_

2 Features **1-Dim**

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

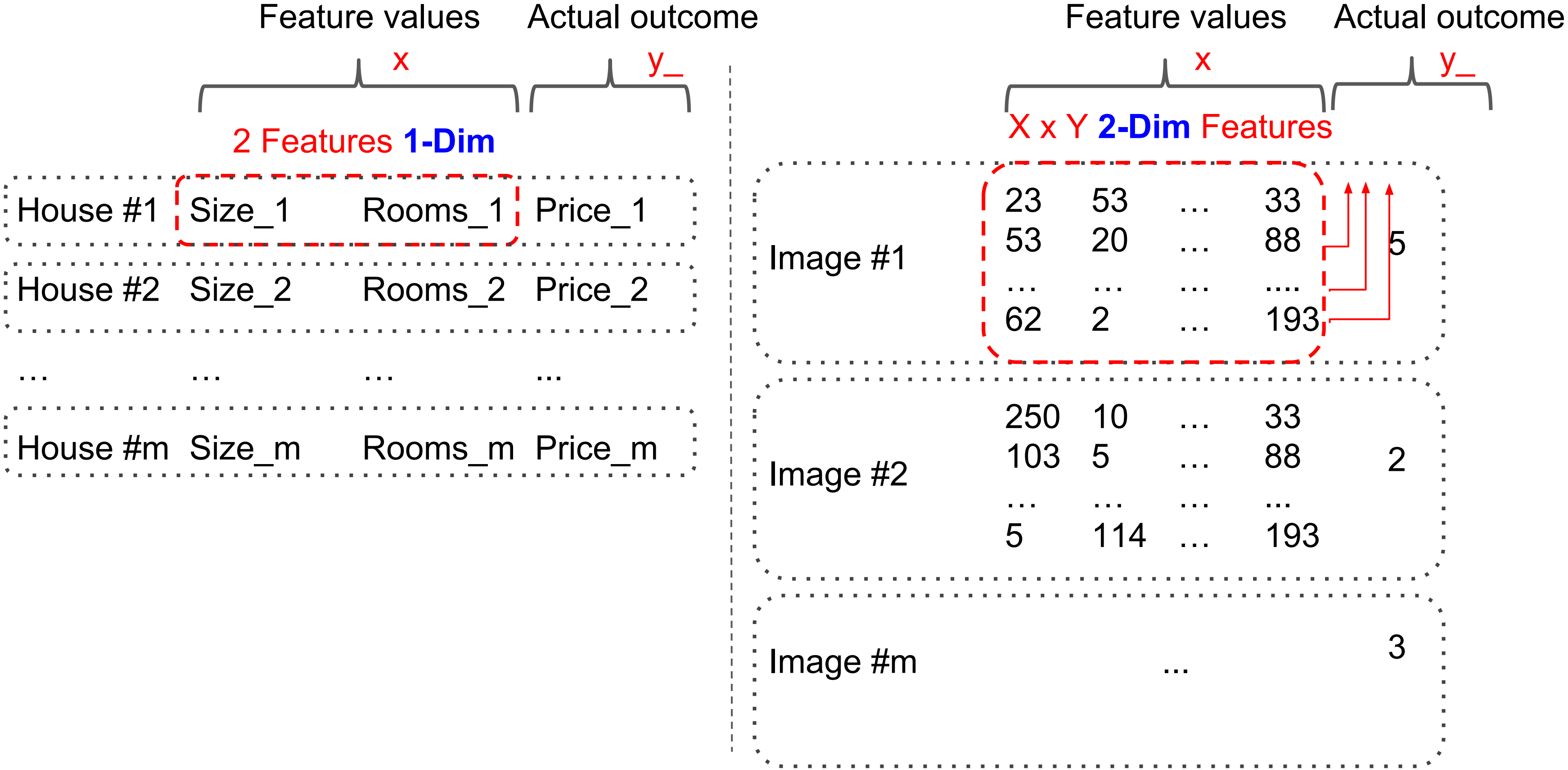
Feature values x

Actual outcome y_

X x Y **2-Dim** Features

Image #1	23	53	…	33	5
	53	20	…	88	
	…	…	…	....	
	62	2	…	193	
Image #2	250	10	…	33	2
	103	5	…	88	
	…	…	…	...	
	5	114	…	193	
Image #m			...		3

Data Representation Comparison



Feature values — x

Actual outcome — y_

2 Features 1-Dim

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

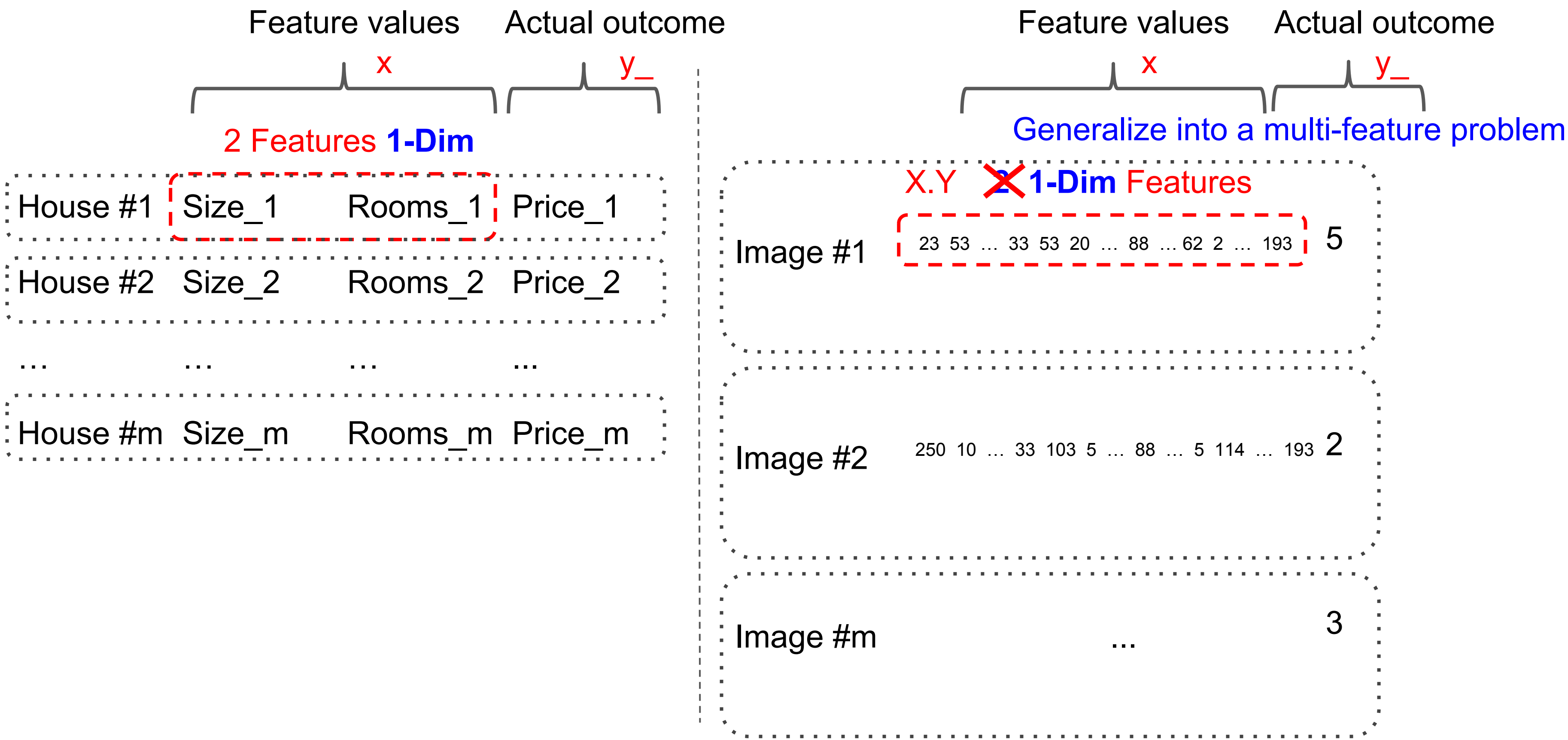
Feature values — x

Actual outcome — y_

X x Y 2-Dim Features

Image #1	23 53 … 33 53 20 … 88 … … … .... 62 2 … 193	5
Image #2	250 10 … 33 103 5 … 88 … … … ... 5 114 … 193	2
Image #m	...	3

Data Representation Comparison



Feature values x | Actual outcome y_

2 Features **1-Dim**

House #1	Size_1	Rooms_1	Price_1
House #2	Size_2	Rooms_2	Price_2
…	…	…	...
House #m	Size_m	Rooms_m	Price_m

Feature values x | Actual outcome y_

Generalize into a multi-feature problem

X.Y ~~2~~ **1-Dim** Features

Image #1	23 53 … 33 53 20 … 88 … 62 2 … 193	5
Image #2	250 10 … 33 103 5 … 88 … 5 114 … 193	2
Image #m	...	3

Model

Linear Regression

$y = W.x + b$

y: House price (scalar) prediction

x: [House size, Rooms]

Goal: Find scalars **W,b**

Logistic Regression

$y = W.x + b$

y: Discrete class [0,1,...9] prediction

~~x: [2-Dim pixel grayscale colors]~~

x: [Pixel 1, Pixel 2, …, Pixel X.Y]

Goal: Find scalars **W, b**

Model

Linear Regression

$y = W.x + b$

y: House price (scalar) prediction

x: [House size, Rooms]

Goal: Find scalars **W,b**

This needs change as well!

Logistic Regression

$y = W.x + b$

y: Discrete class [0,1,...9] prediction

~~x: [2-Dim pixel grayscale colors]~~

x: [Pixel 1, Pixel 2, …, Pixel X.Y]

Goal: Find scalars **W, b**

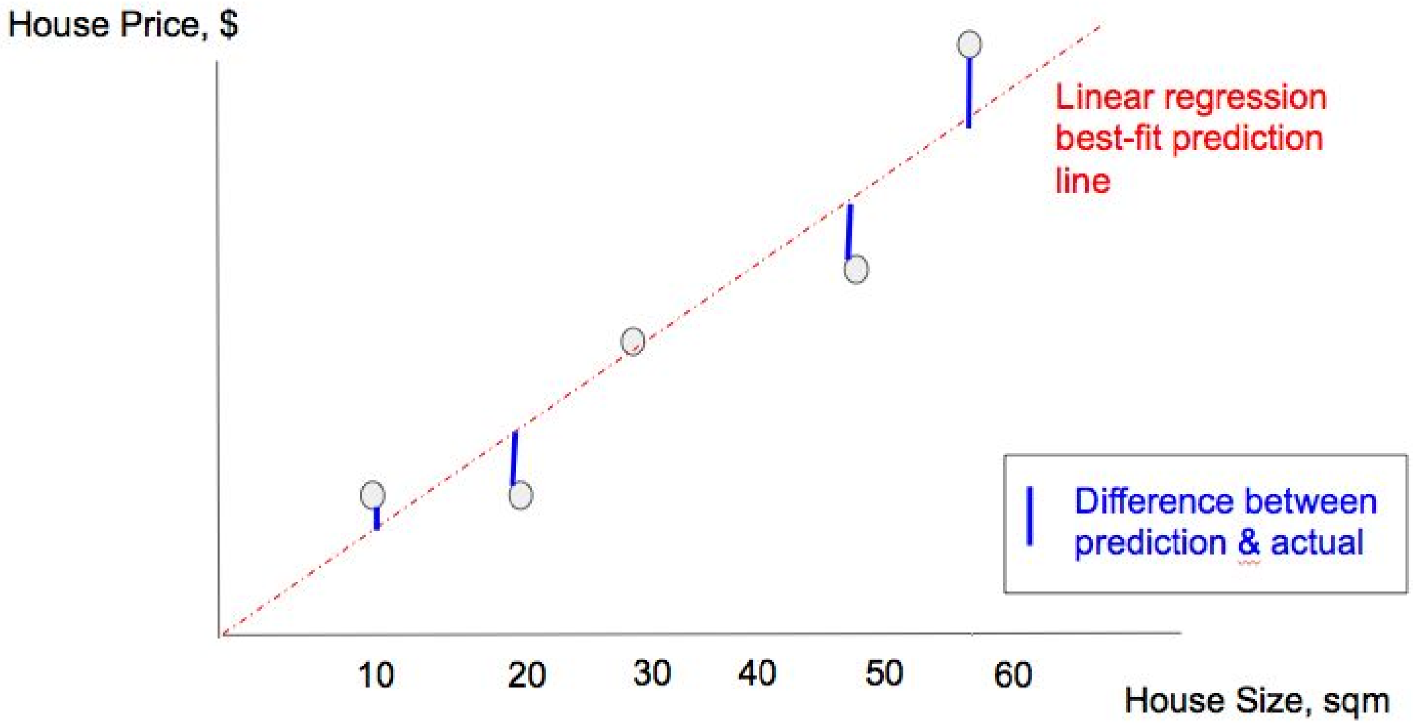
Why Can’t ‘y’ be left as scalar of 0-9?

HINT: Beside the model, when doing ML we need this function!

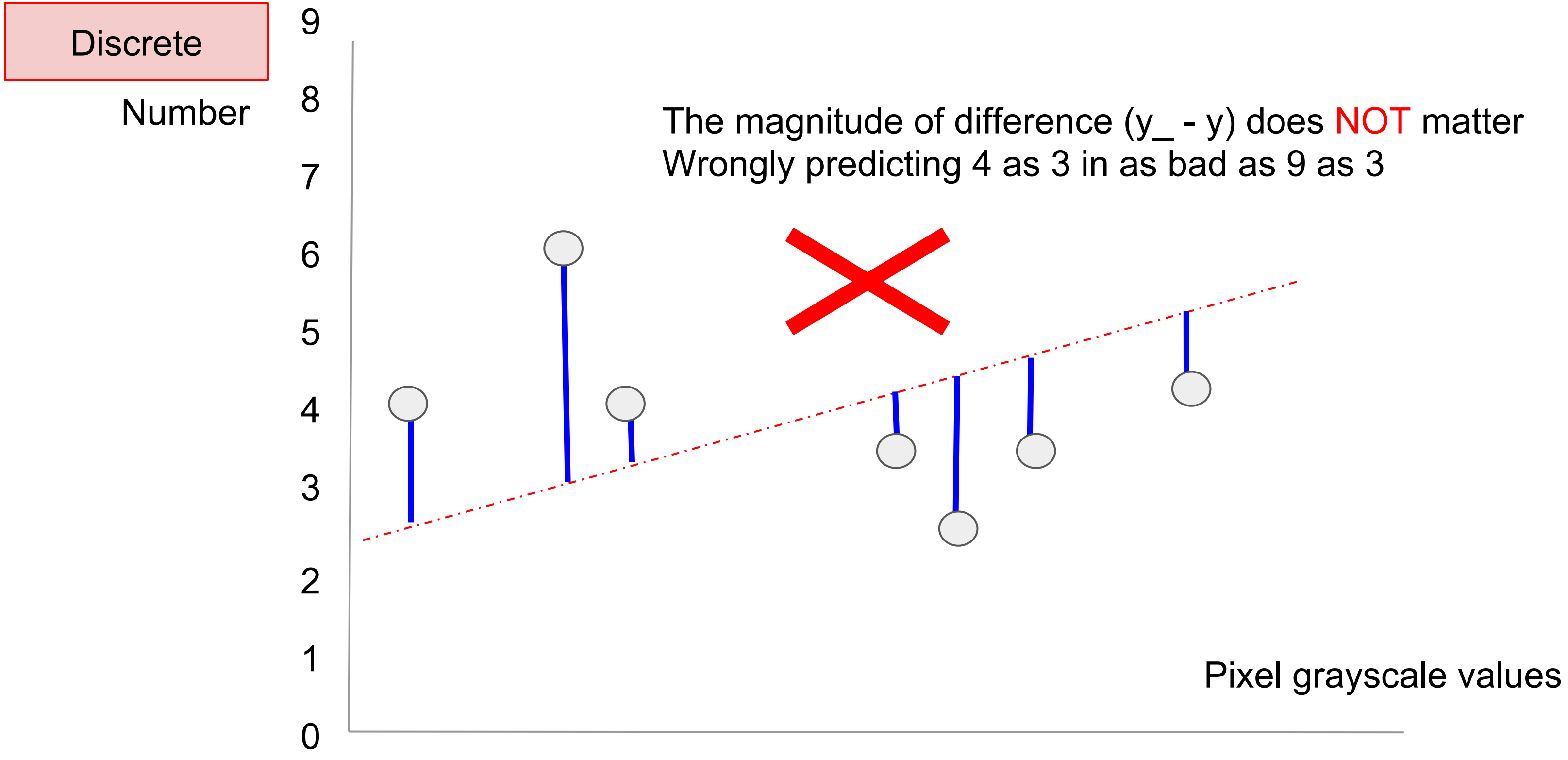
Logistic Regression: Change in Cost Function

Linear Regression: Cost

Scalar

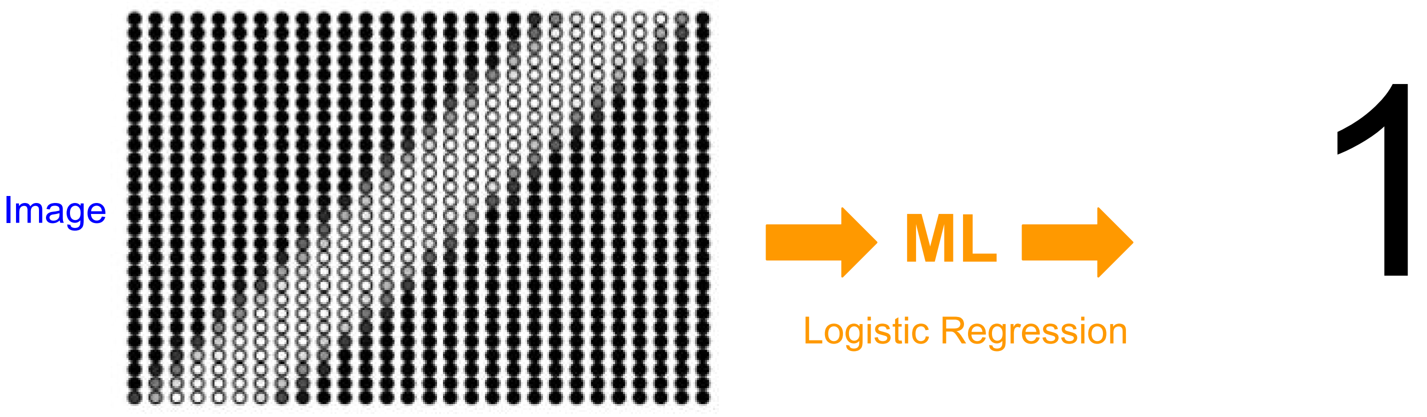


Linear Regression: Cost, NOT



The magnitude of difference (y_ - y) does NOT matter
Wrongly predicting 4 as 3 in as bad as 9 as 3

Logistic Regression: Prediction



Image

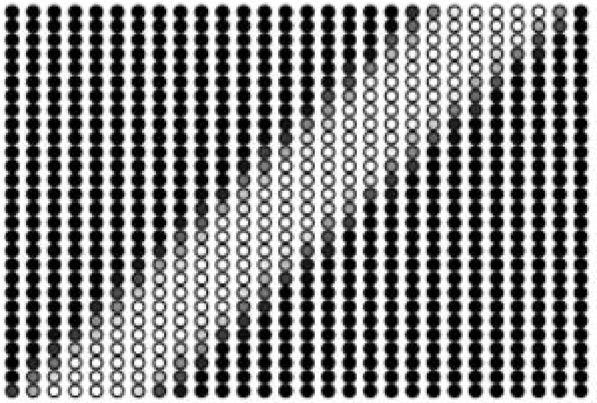
ML

Logistic Regression

1

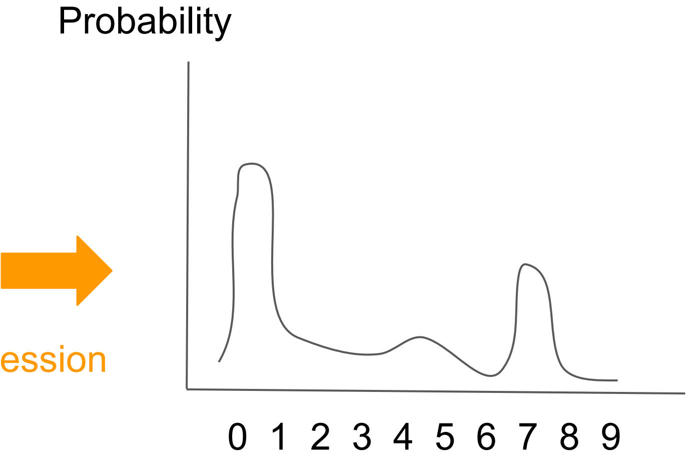
Logistic Regression: Prediction

Image

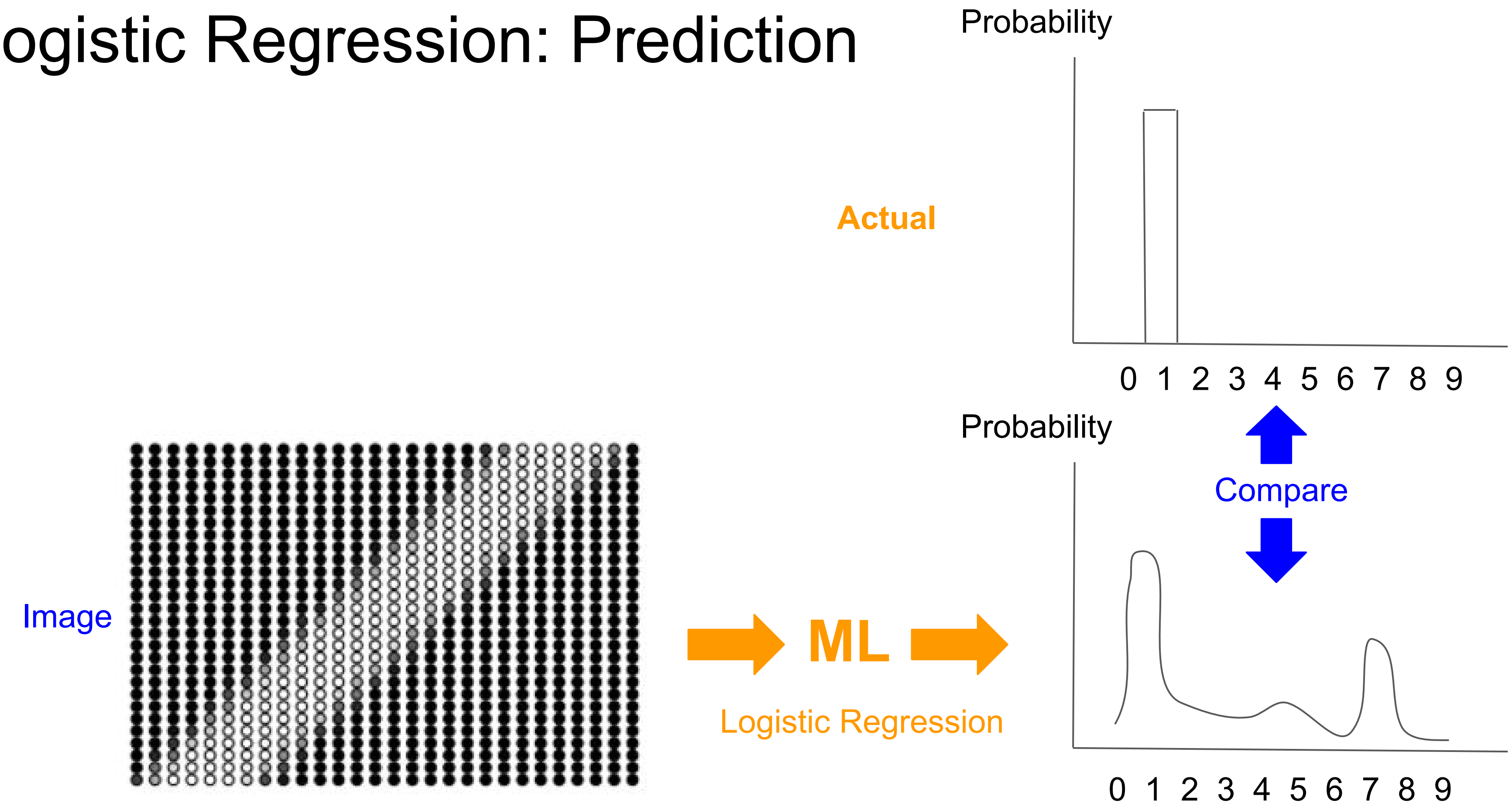


ML

Logistic Regression



Logistic Regression: Prediction



Actual

Image

ML

Logistic Regression

Compare

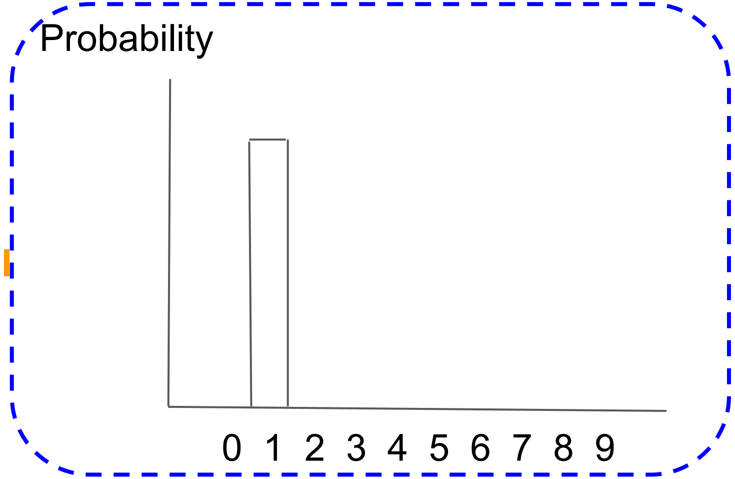
Cross-Entropy: Cost Function 1

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

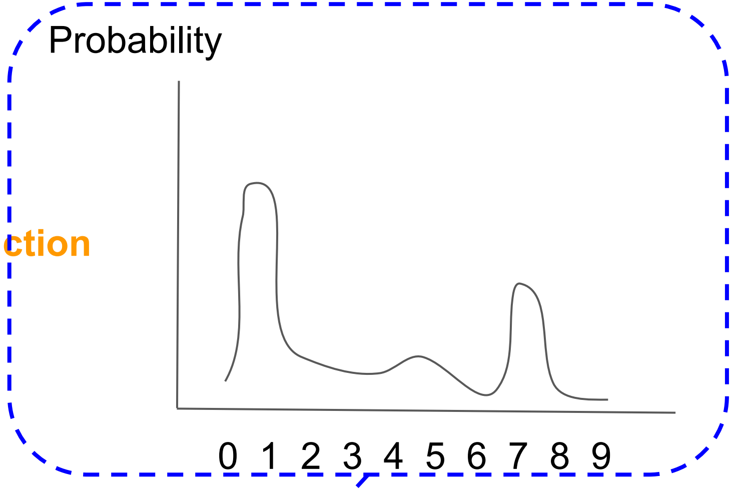
```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

Cross-Entropy: Cost Function 2

Actual



Prediction



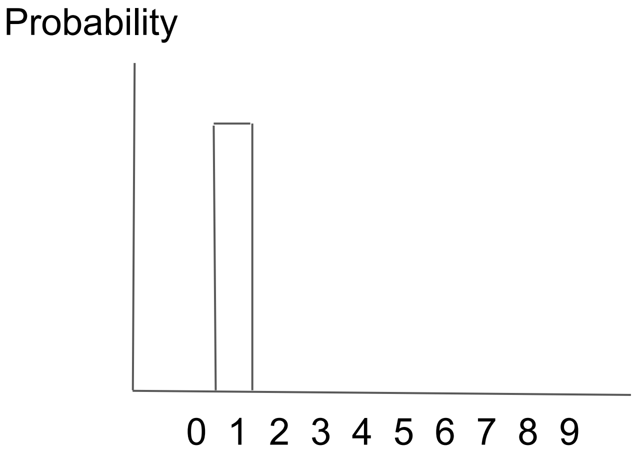
$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

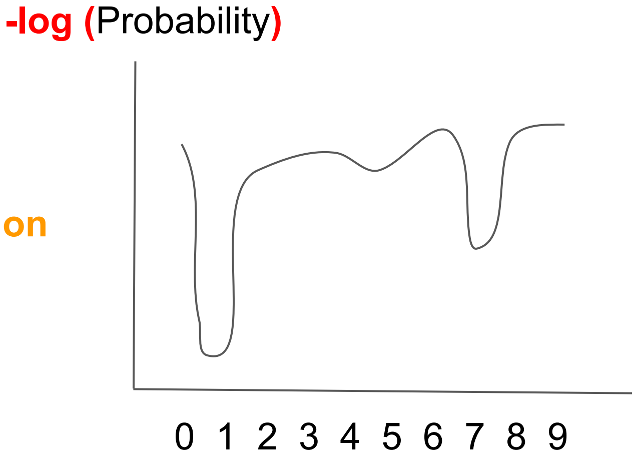
Cross-Entropy: Cost Function 3

Almost inverse because Probability < 1

Actual



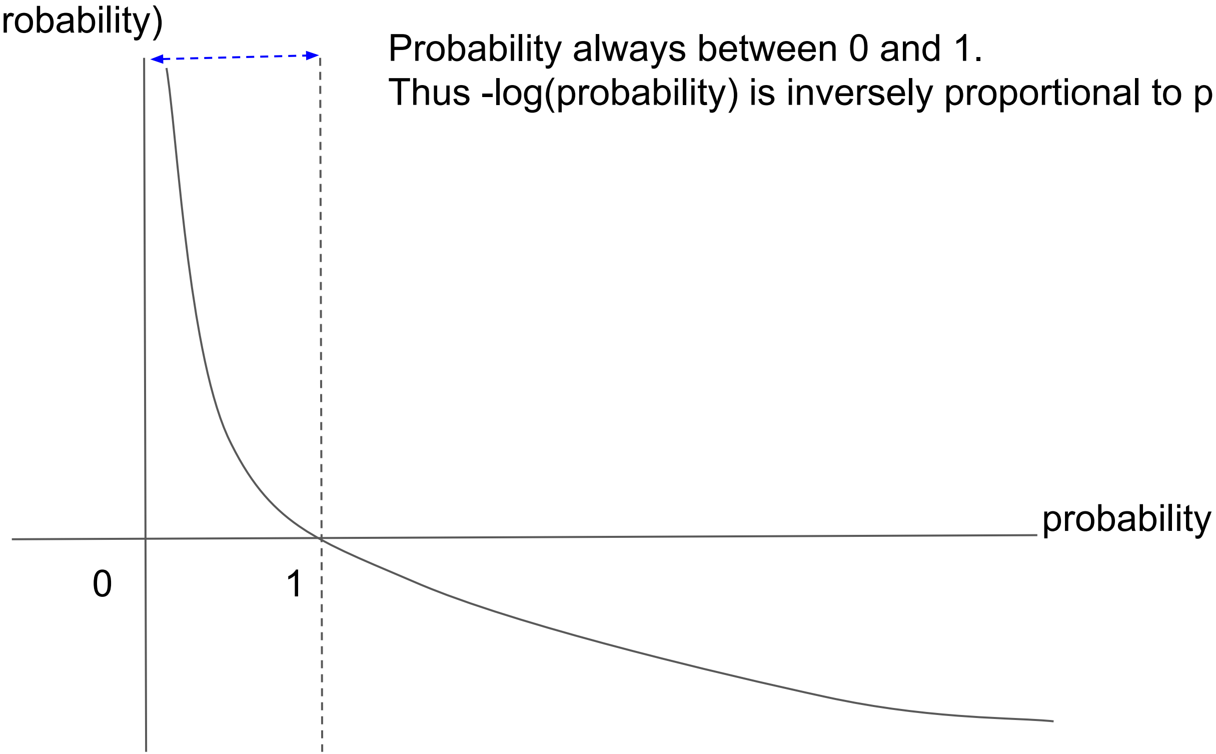
Prediction



$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

cross_entropy = -tf.reduce_sum(y_*tf.log(y))

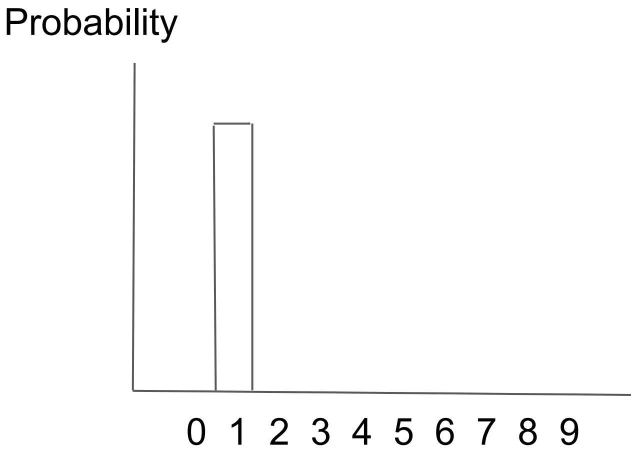
Graph: -log(probability)



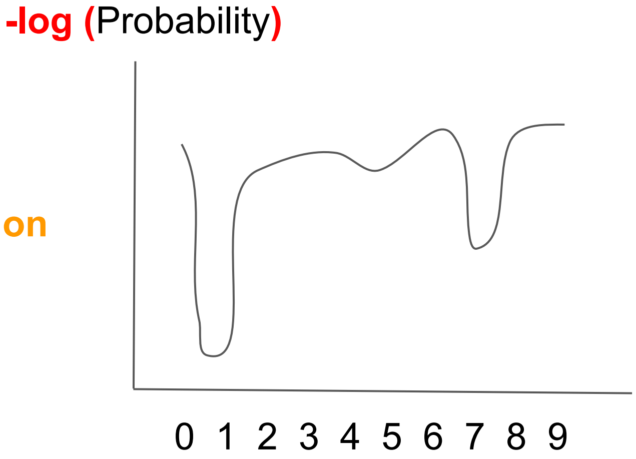
Probability always between 0 and 1.
Thus -log(probability) is inversely proportional to probability

Cross-Entropy: Cost Function x x xx

Actual



Prediction

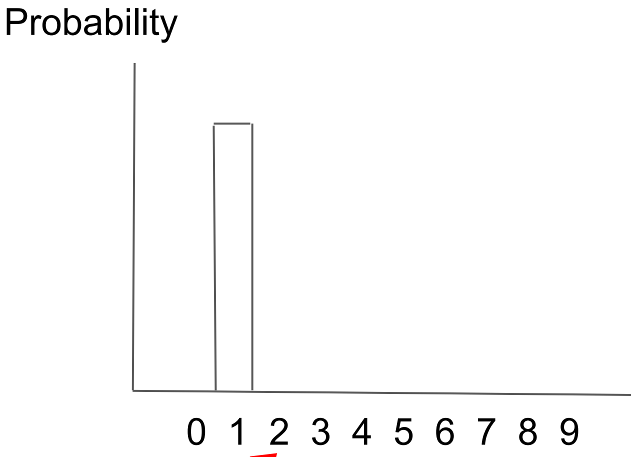


$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

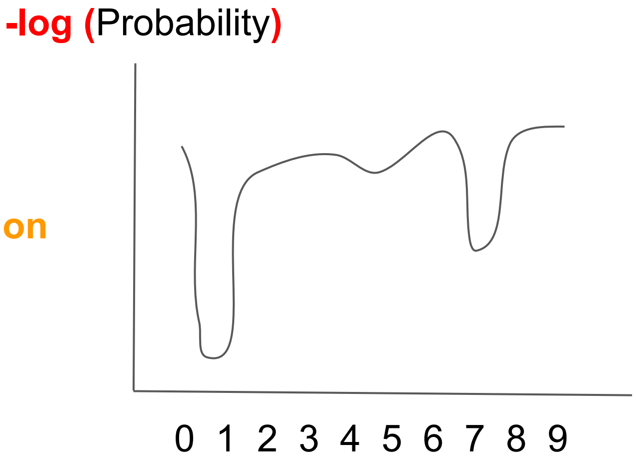
```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

Cross-Entropy: Cost Function

Actual



Prediction

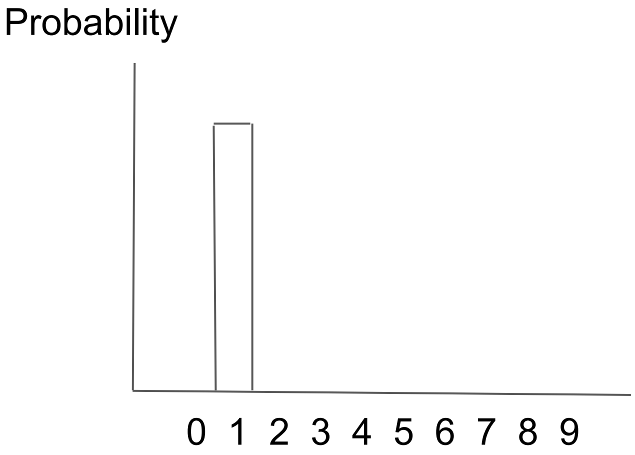


$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

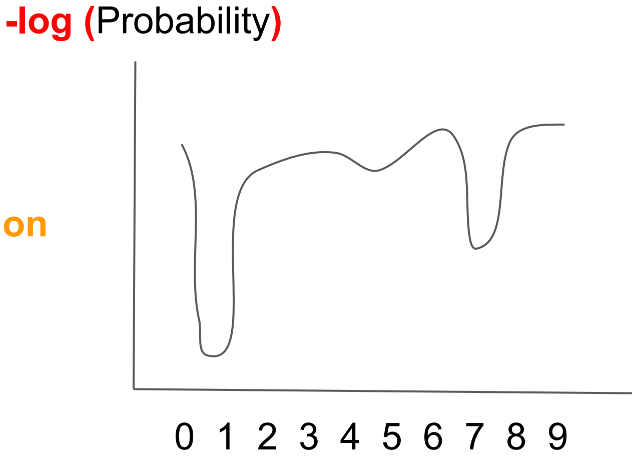
cross_entropy = -tf.reduce_sum(y_*tf.log(y))

Cross-Entropy: Cost Function

Actual



Prediction



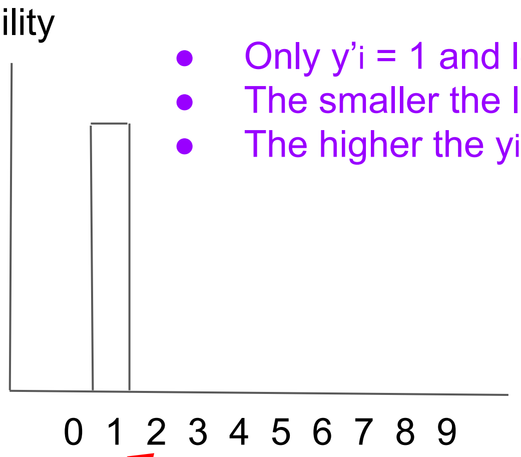
$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

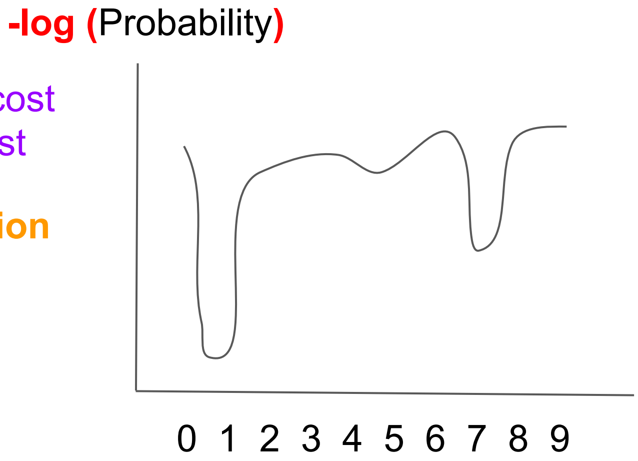
Cross-Entropy: Cost Function

- Only $y'_i = 1$ and $\log(y_i)$ matters
- The smaller the $\log(y_i)$ the lower cost
- The higher the y_i the lower the cost

Actual



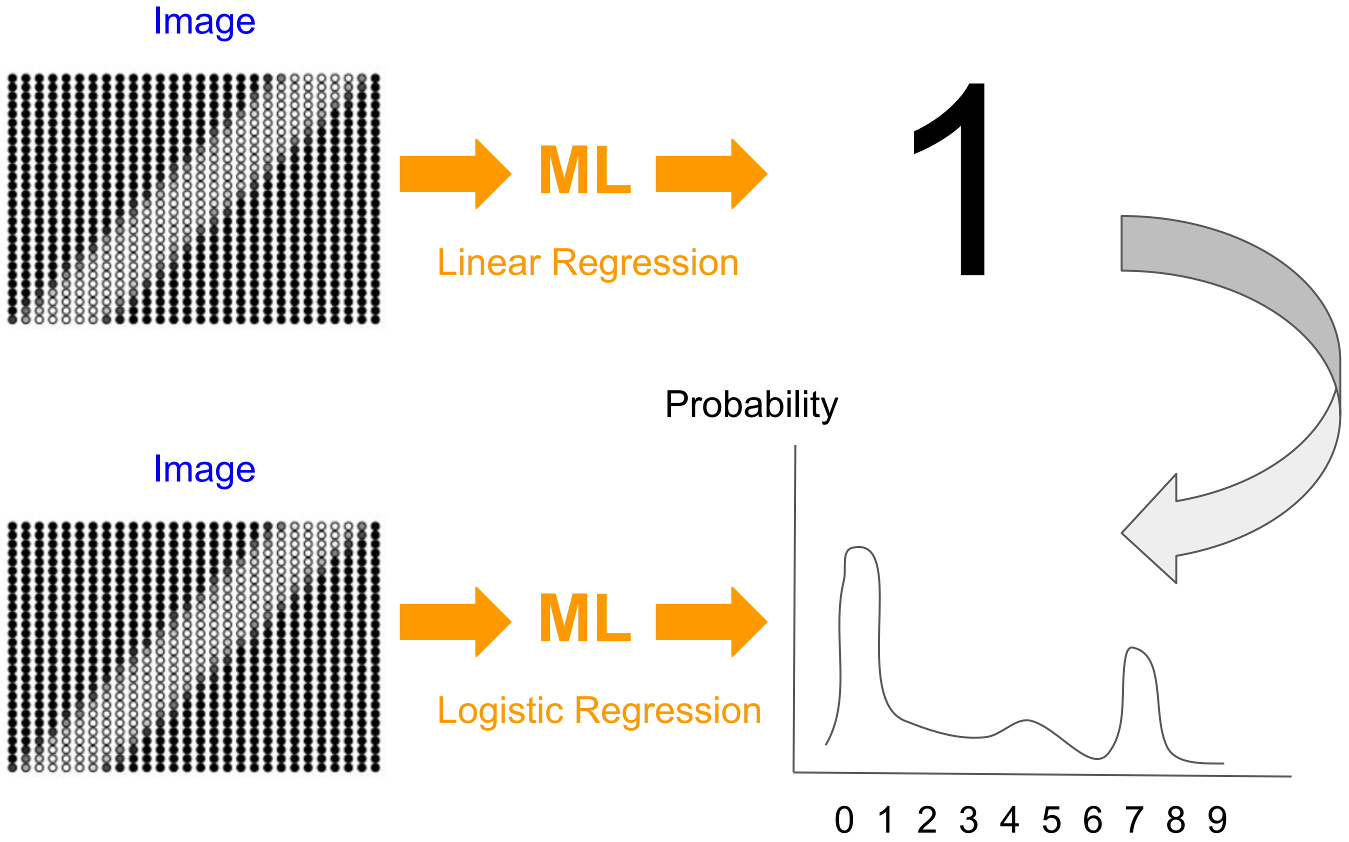
Prediction



$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

Logistic Regression: Prediction



Image

ML

Linear Regression

1

Image

ML

Logistic Regression

Tensorflow Graph (Review)

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

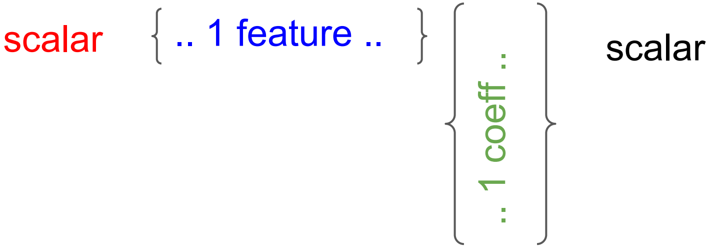
2 Features

```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 1])
```

Tensorflow Graph (Review 2)

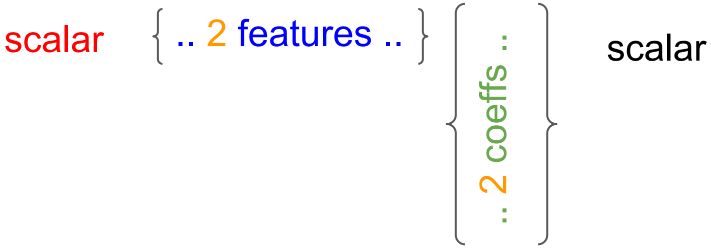
1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```



2 Features

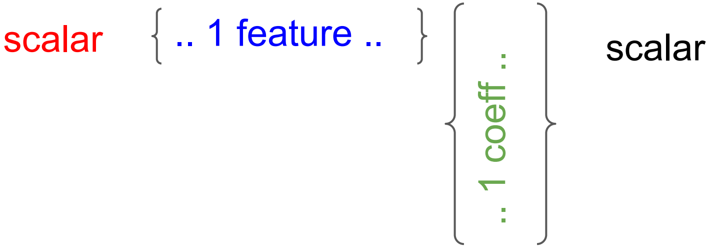
```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 1])
```



Tensorflow Graph (Review 2)

1 Feature

```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```



scalar

{ .. 1 feature .. }

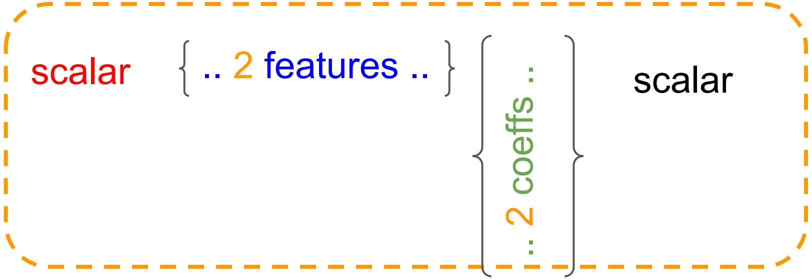
{ .. 1 coeff .. }

scalar

2 Features

```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 1])
```

Apply multi-feature linear regression



scalar

{ .. 2 features .. }

{ .. 2 coeffs .. }

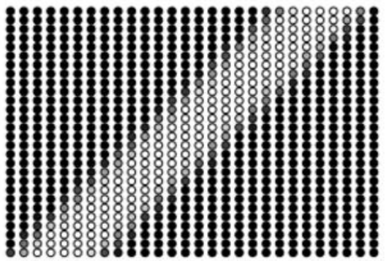
scalar

Logistic Regression: Prediction

y = tf.matmul(x,W) + b

scalar { .. n features .. } { .. n coeffs .. } scalar

Image

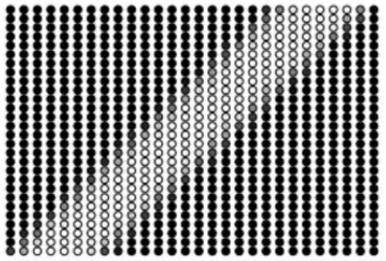


ML

Linear Regression

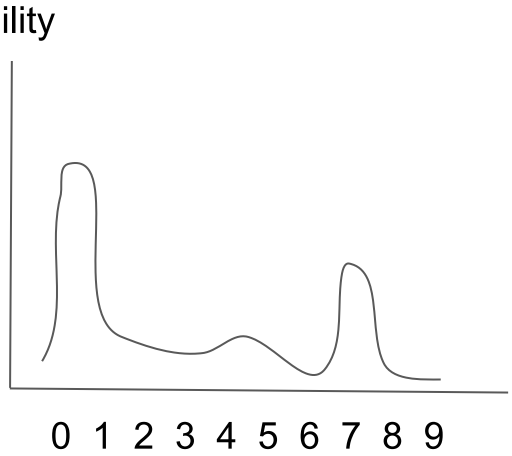
1

Image



ML

Logistic Regression



Logistic Regression: Prediction

y = tf.matmul(x,W) + b

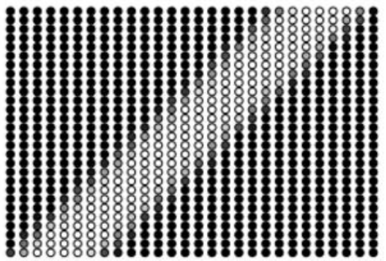
scalar

{ .. n features .. }

{ .. n coeffs .. }

scalar

Image

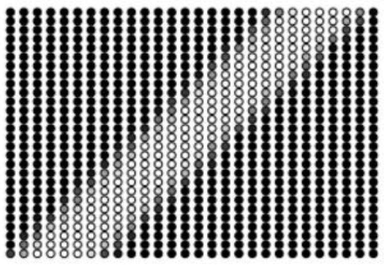


ML

Linear Regression

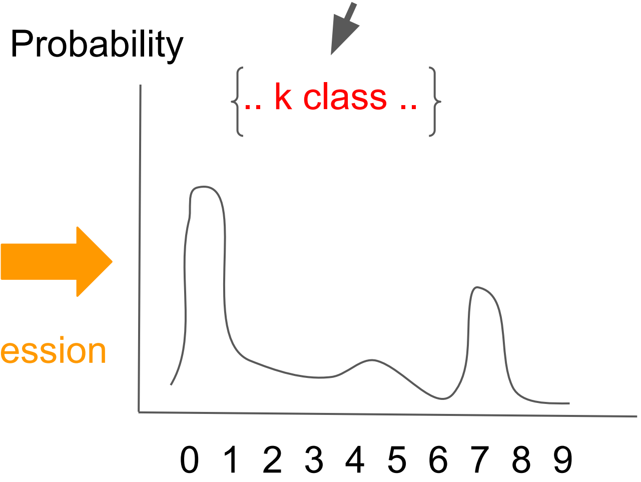
1

Image



ML

Logistic Regression



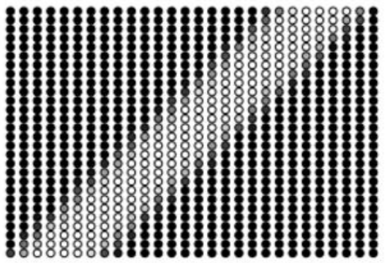
{.. k class .. }

Logistic Regression: Prediction

y = tf.matmul(x,W) + b

scalar   { .. n features .. }   { .. n coeffs .. }   scalar

Image



ML

Linear Regression

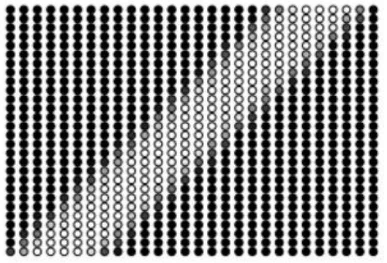
1

y = tf.matmul(x,W) + b

{ .. k class .. }   { .. n features .. }   { .. k class .. }   { .. k class .. }

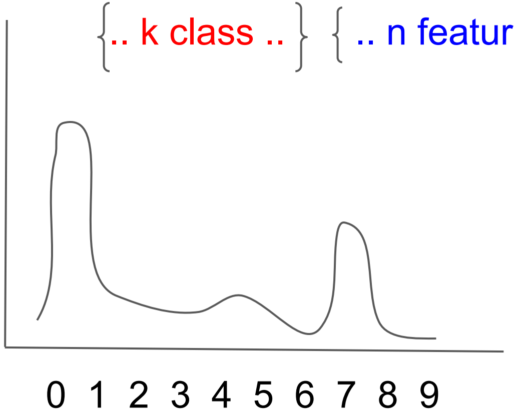
{ .. n coeffs .. }

Image



ML

Logistic Regression



Tensorflow Graph: Basic, Multi-feature, Multi-class

1 Feature

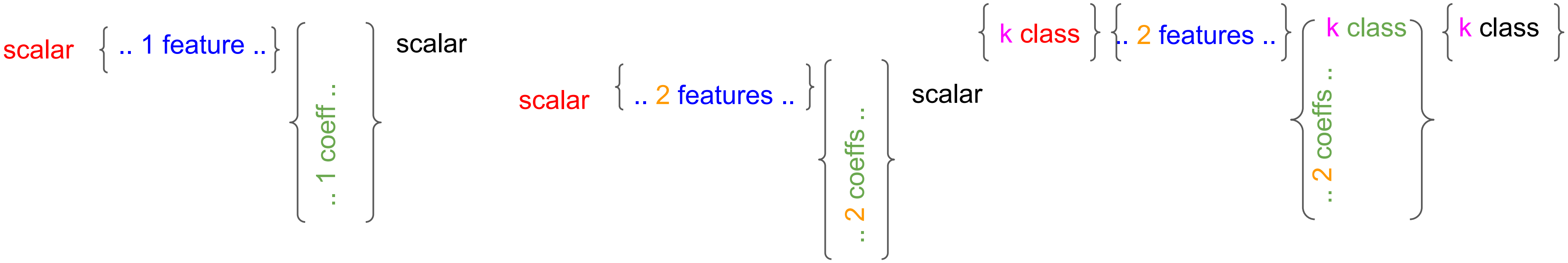
```
y = tf.matmul(x, W) + b

W = tf.Variable(tf.zeros[1,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 1])
y_ = tf.placeholder(tf.float, [None, 1])
```

2 Features

```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,1])
b = tf.Variable(tf.zeros[1])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 1])
```

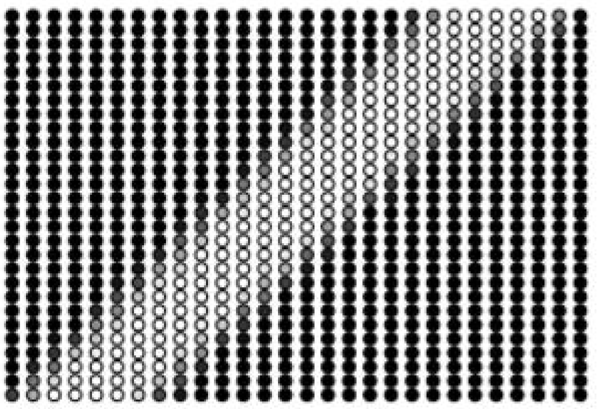
2 Features, 10 Classes

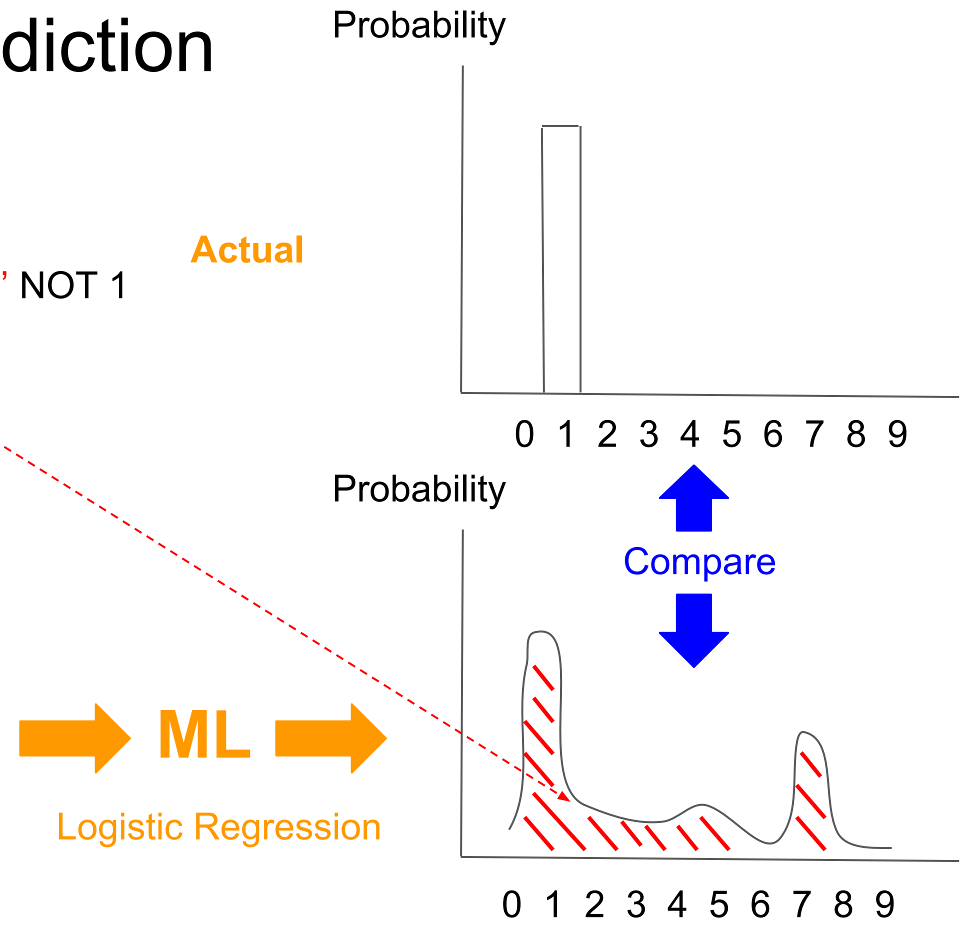
```
y = matmul(x, W) + b

W = tf.Variable(tf.zeros[2,10])
b = tf.Variable(tf.zeros[10])

x = tf.placeholder(tf.float, [None, 2])
y_ = tf.placeholder(tf.float, [None, 10])
```



Logistic Regression: Prediction

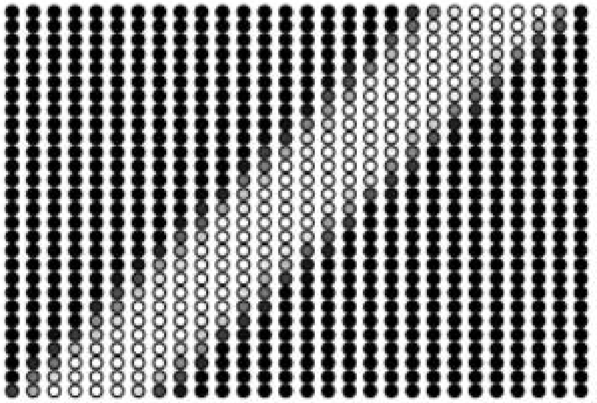
Great but….sum of all prediction ‘probability’ NOT 1





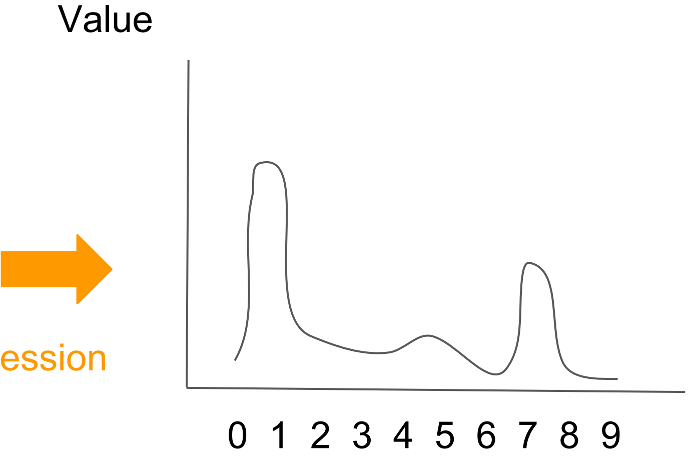
$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Image

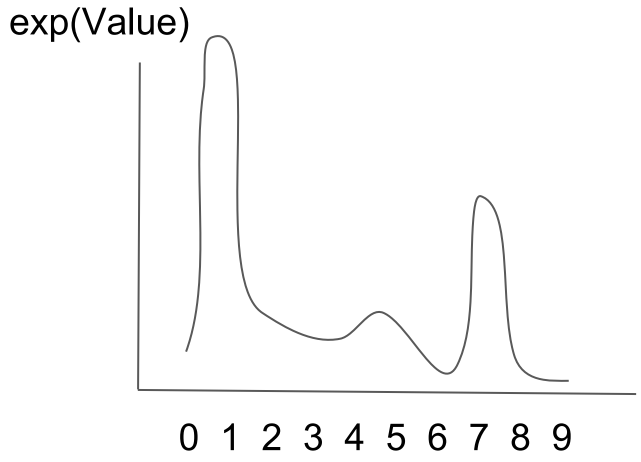


ML

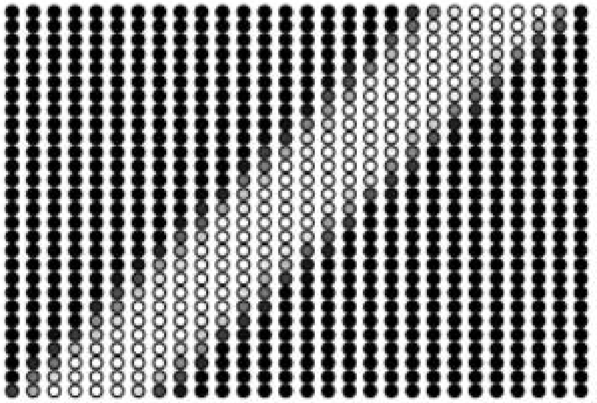
Logistic Regression



$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

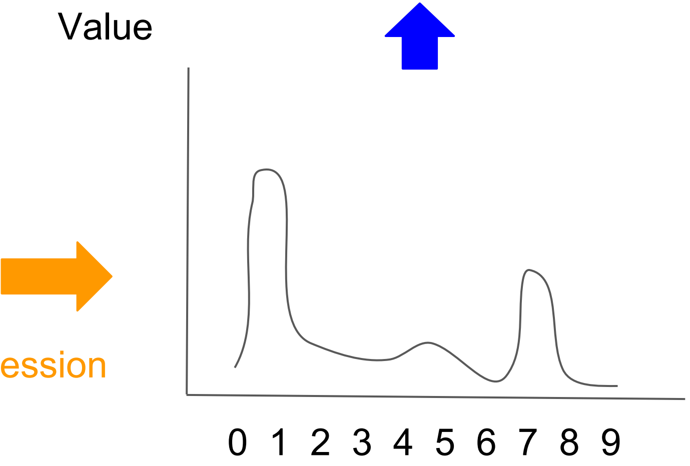


Image



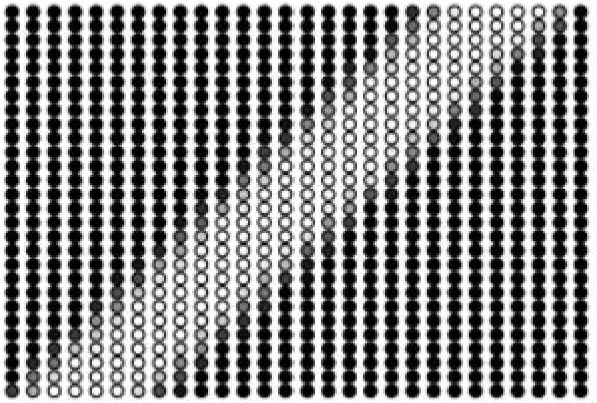
ML

Logistic Regression



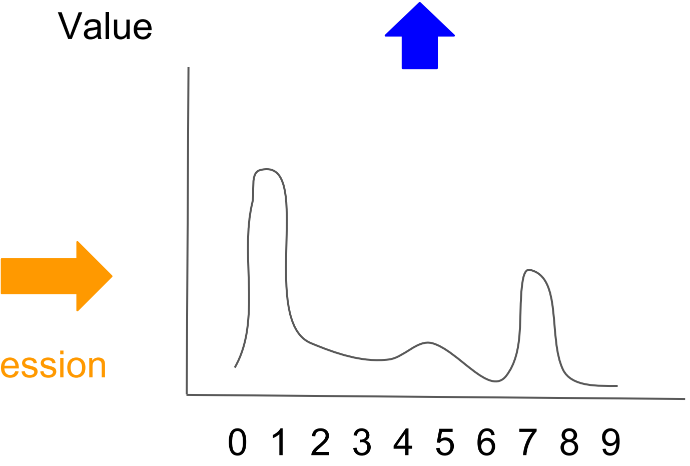
$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

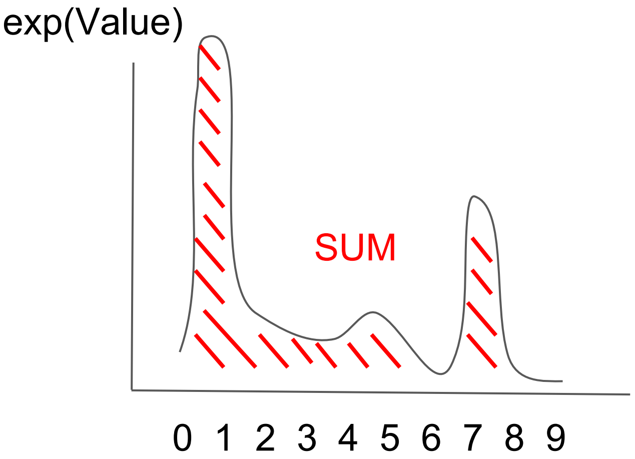
Image



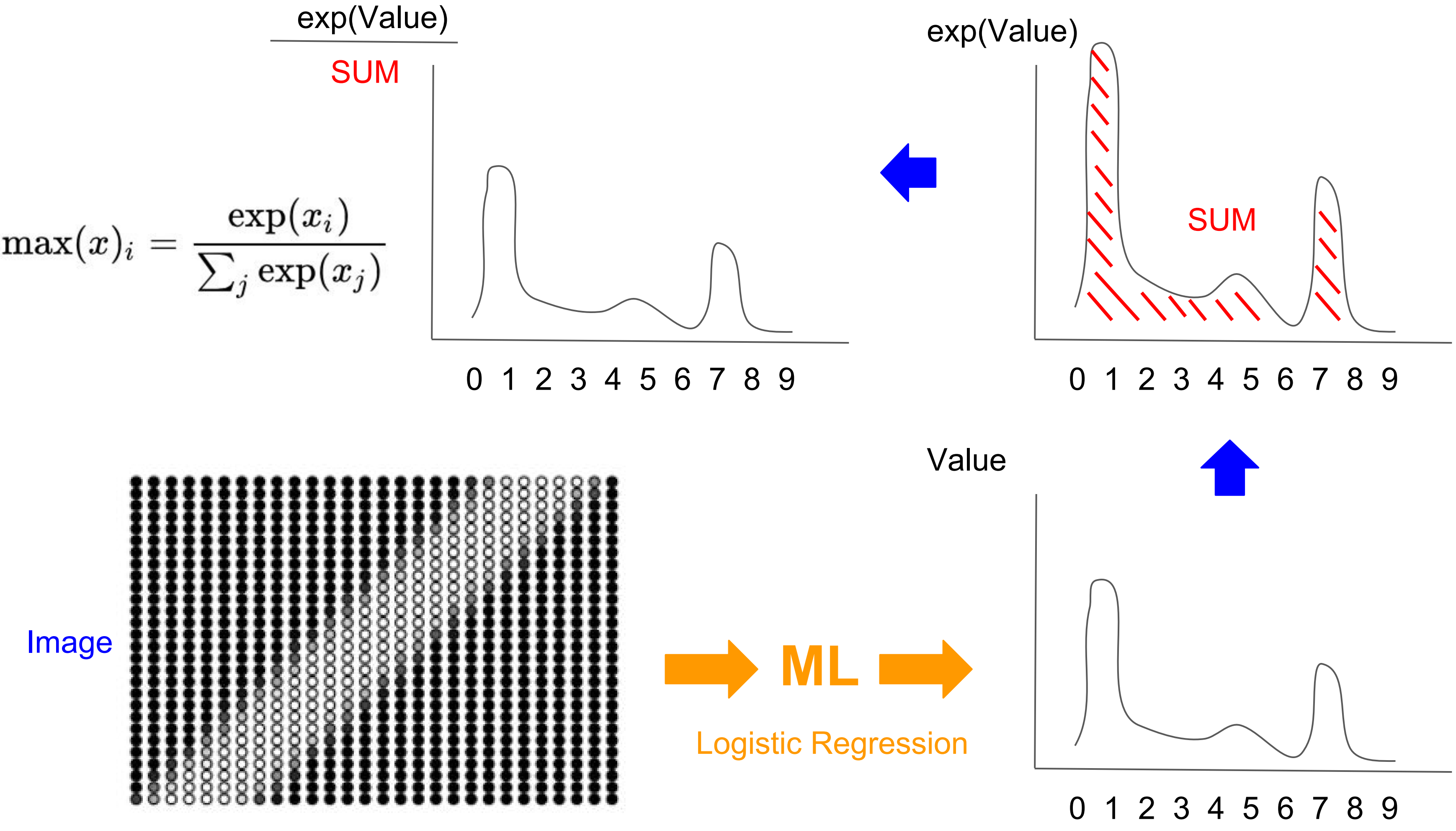
ML

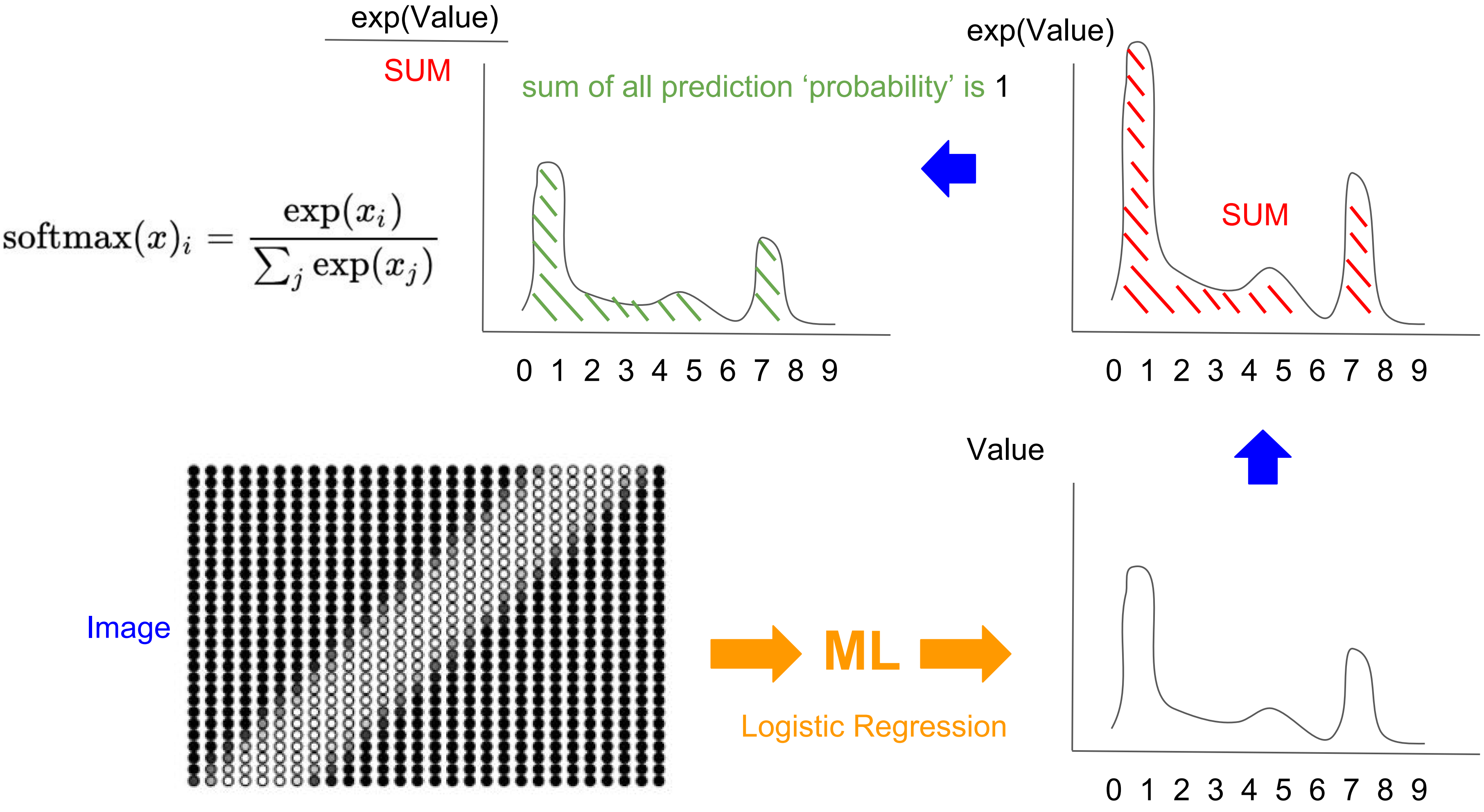
Logistic Regression





$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$





$$\frac{\text{exp(Value)}}{\text{SUM}}$$

sum of all prediction ‘probability’ is 1

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

exp(Value)

SUM

Value

Image

ML

Logistic Regression

Before softmax

```
y = tf.matmul(x, W) + b
```

With softmax

~~y = tf.matmul(x, W) + b~~

$$\mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

y = tf.nn.softmax(tf.matmul(x, W) + b)

Summary

- Cheat sheet: Single feature, Multi-feature, Multi-class
- Logistic regression:
    - Multi-class prediction: Ensure that prediction is one of a discrete set of values
    - Cross-entropy: Measure difference between multi-class prediction and actual
    - Softmax: Ensure the multi-class prediction probability is a valid distribution (sum = 1)

Congrats!

You can now understand Google’s Tensorflow Beginner’s Tutorial

(https://www.tensorflow.org/versions/r0.7/tutorials/mnist/beginners/index.html)

References

- Perform ML with TF using multi-feature linear regression (the wrong way)
  - https://github.com/nethsix/gentle_tensorflow/blob/master/code/linear_regression_multi_feature_using_mini_batch_with_tensorboard.py
- Perform ML with TF using multi-feature linear regression
  - https://github.com/nethsix/gentle_tensorflow/blob/master/code/linear_regression_multi_feature_using_mini_batch_without_matrix_with_tensorboard.py
- Tensorflow official tutorial for character recognition
  - https://www.tensorflow.org/versions/r0.7/tutorials/mnist/beginners/index.html
- Colah’s excellent explanation of cross-entropy
  - http://colah.github.io/posts/2015-09-Visual-Information/