Building Logistic Regression Models Using TensorFlow

Vitthal Srinivasan

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Given causes, predict probability of effects - that's logistic regression

Linear regression and logistic regression are similar, yet quite different

Logistic regression can be used for categorical y-variables

Logistic regression in TensorFlow differs from linear regression in two ways

- Softmax as the activation function
- cross-entropy as the cost function

Two Approaches to Deadlines

Start 5 minutes before deadline

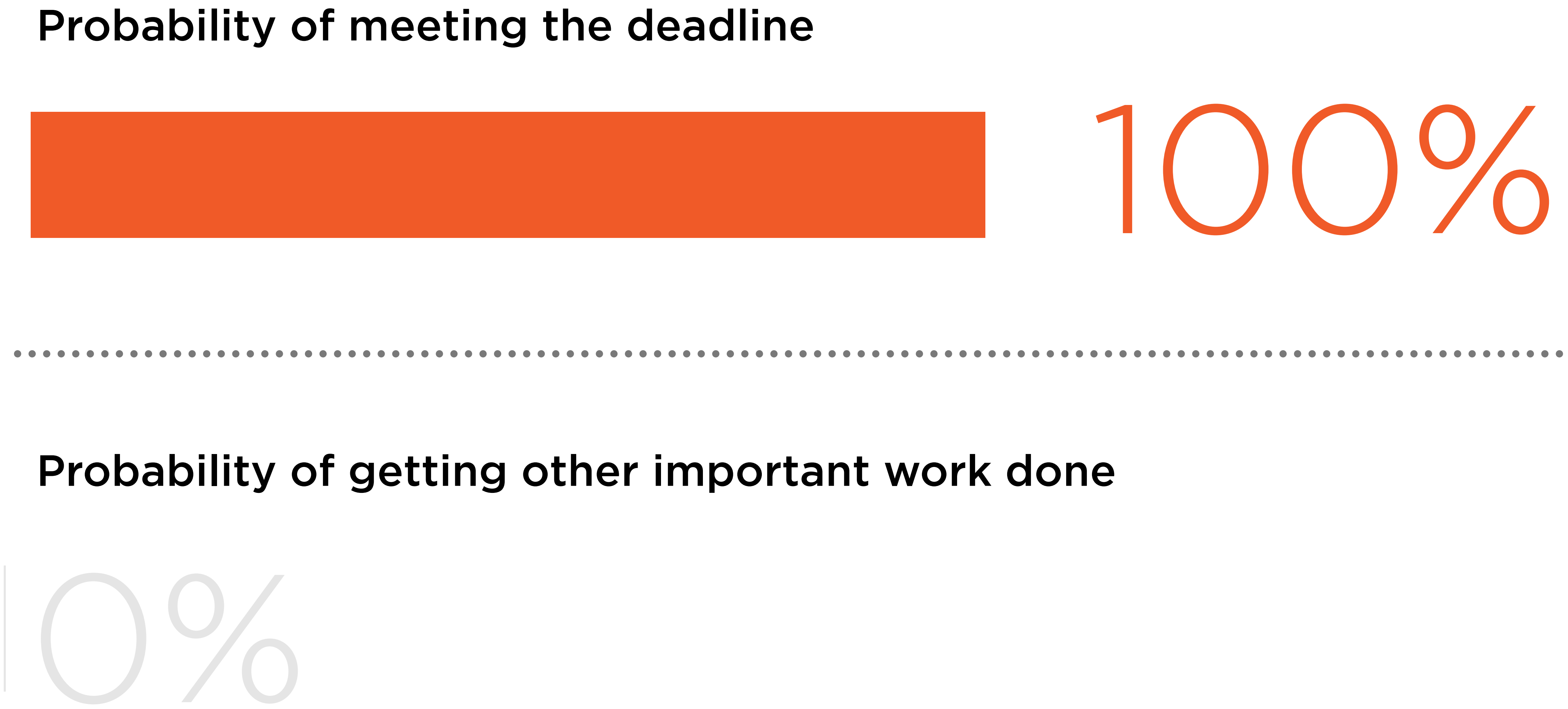
Good luck with that

Start 1 year before deadline

Maybe overkill

Neither approach is optimal

Starting a Year in Advance



Probability of meeting the deadline

100%

Probability of getting other important work done

0%

Starting Five Minutes in Advance

Probability of meeting the deadline

0%

Probability of getting other important work done

100%

The Goldilocks Solution

Work fast

Start very late and hope for the best

Work smart

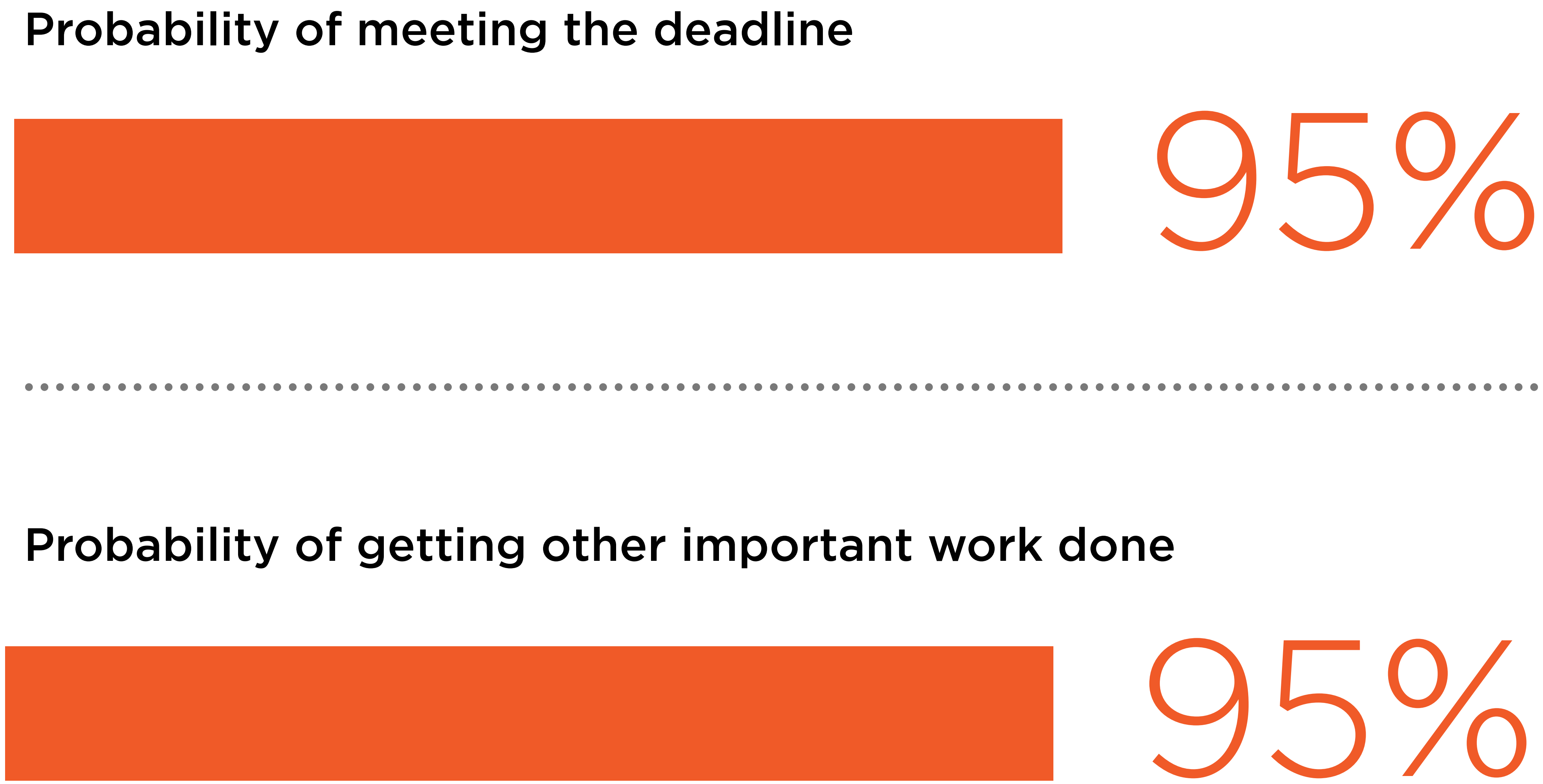
Start as late as possible to be sure to make it

Work hard

Start very early and do little else

As usual, the middle path is best

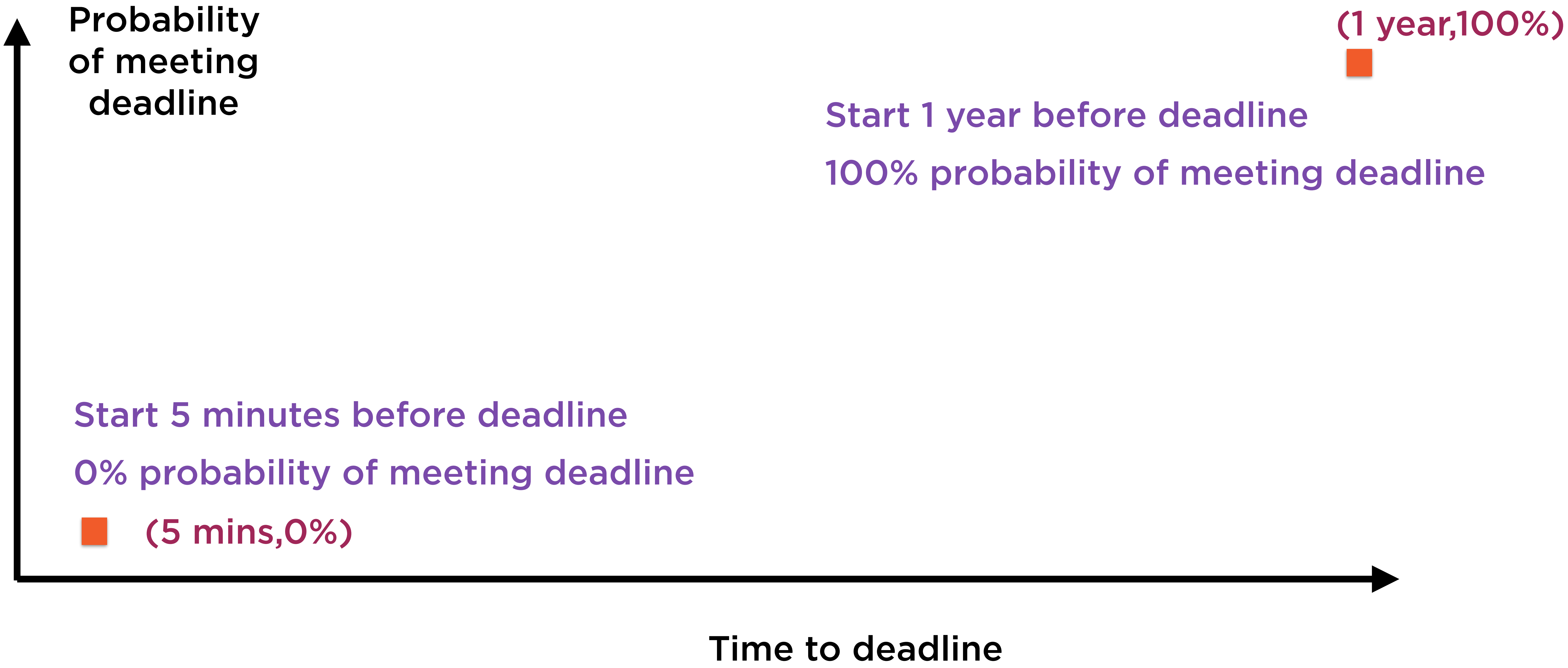
Working Smart



Probability of meeting the deadline

Probability of getting other important work done

Working Hard, Fast, Smart



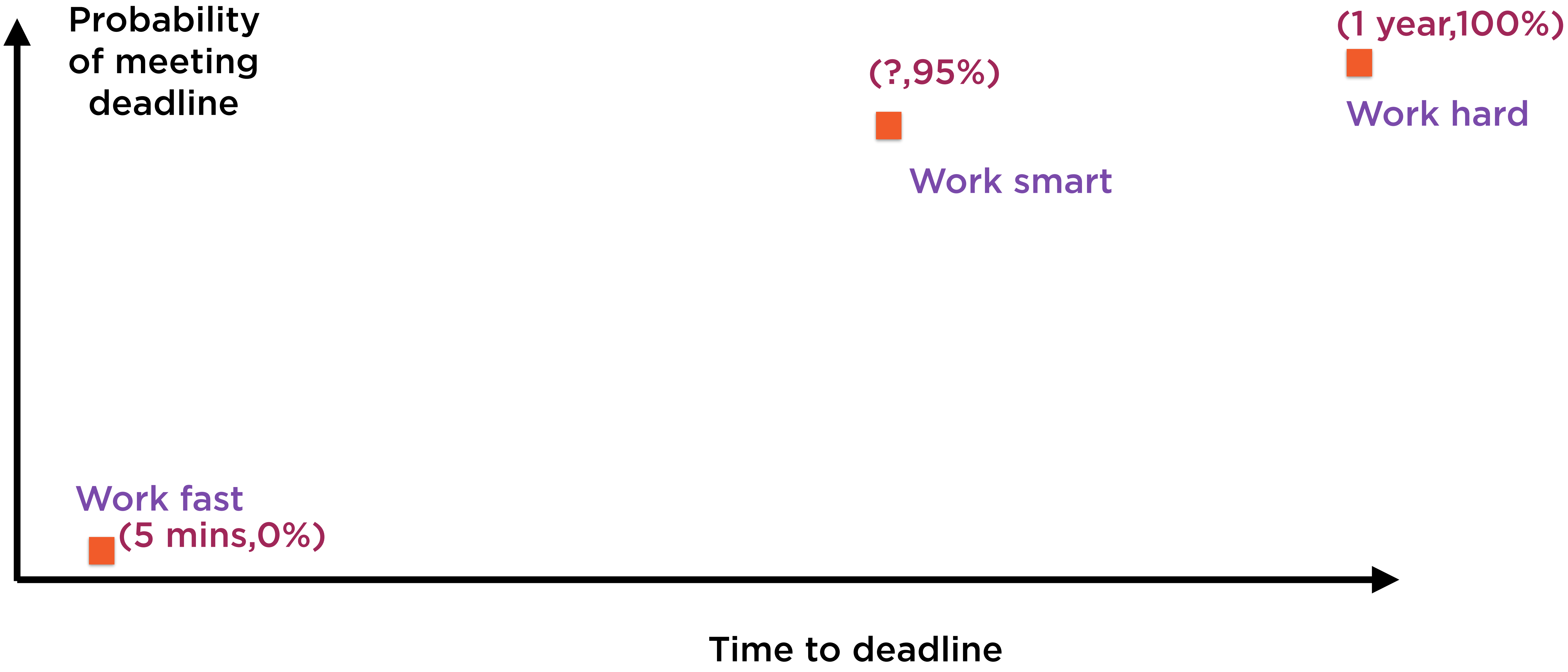
Start 1 year before deadline

100% probability of meeting deadline

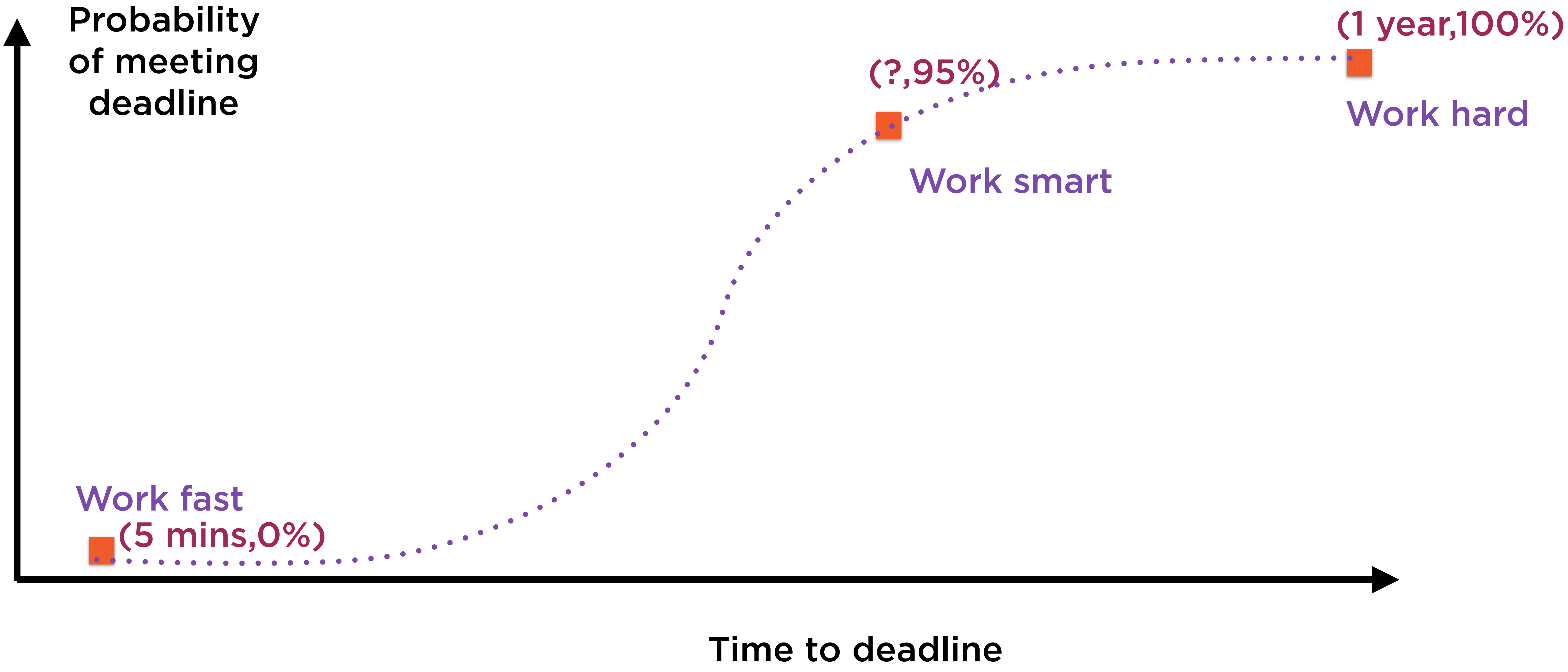
Start 5 minutes before deadline

0% probability of meeting deadline

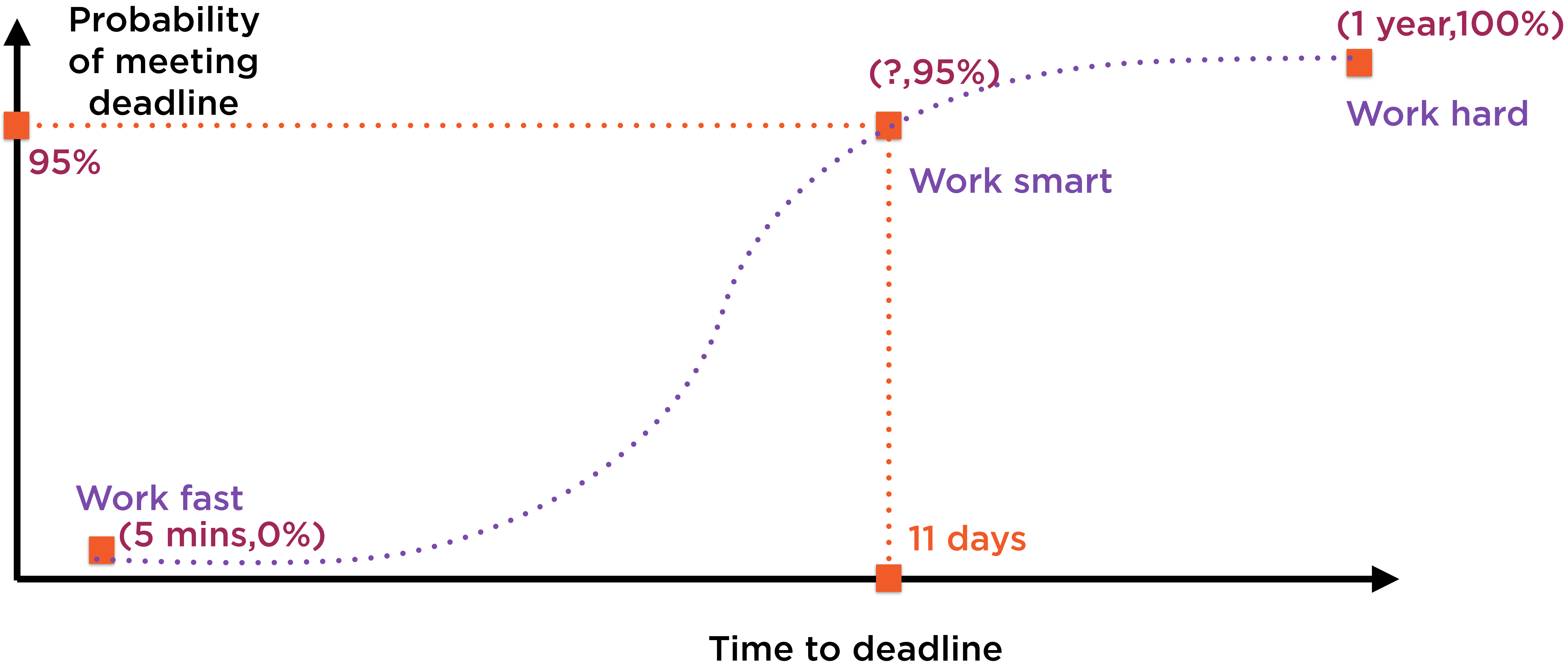
Working Hard, Fast, Smart



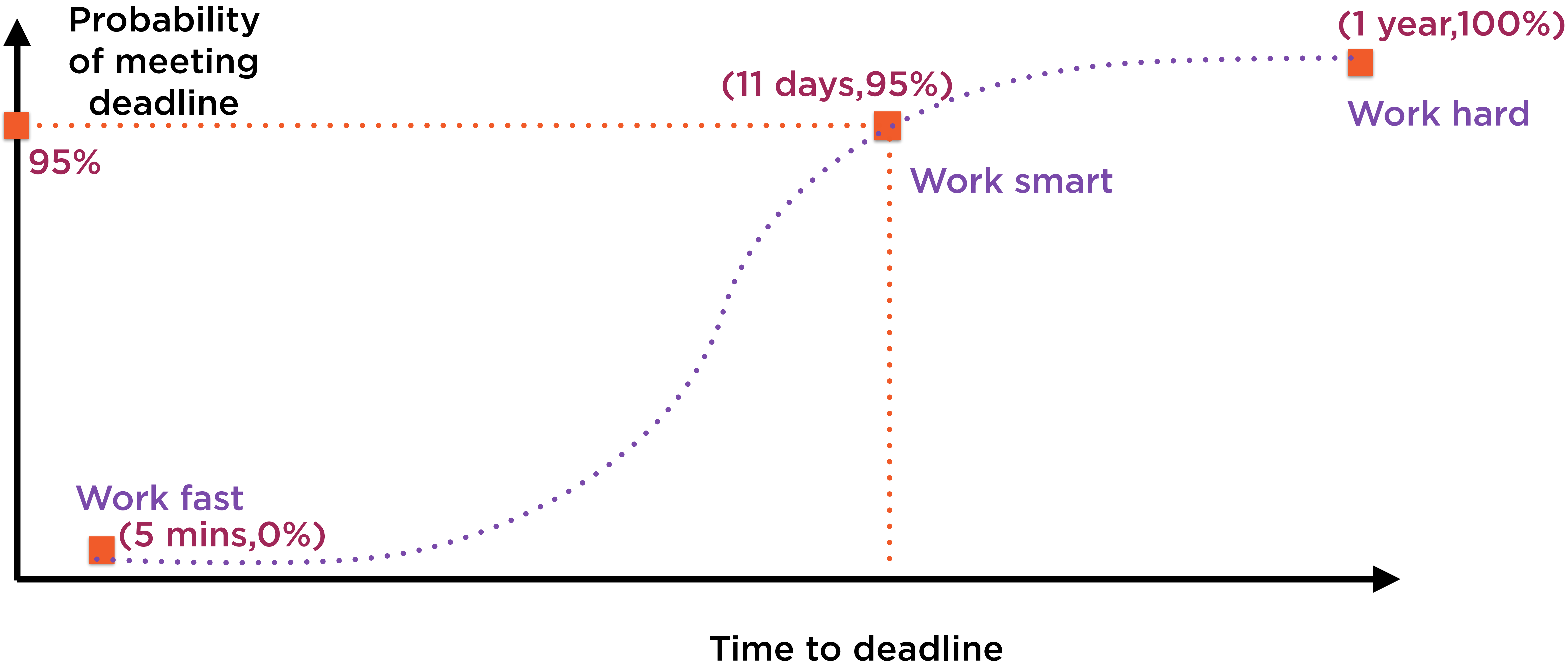
Working Hard, Fast, Smart



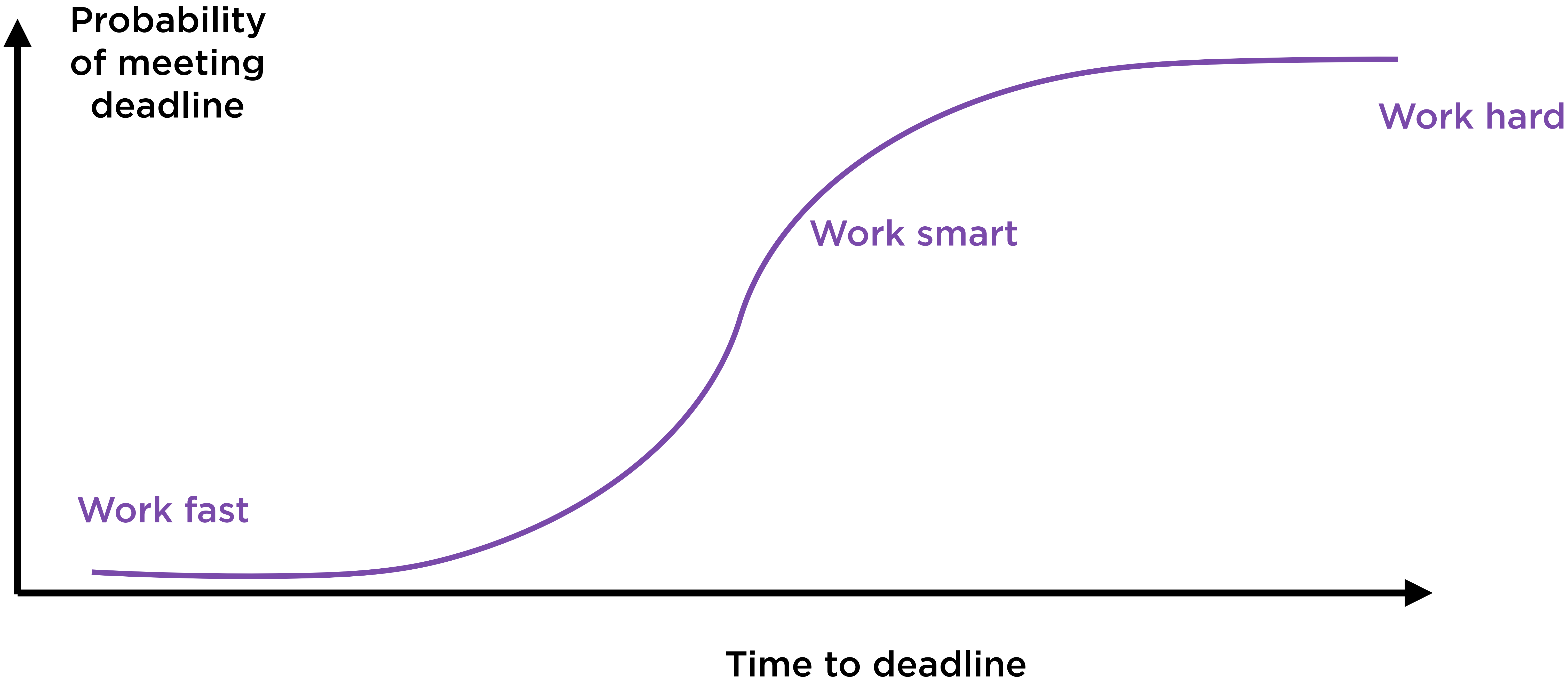
Working Hard, Fast, Smart



Working Hard, Fast, Smart

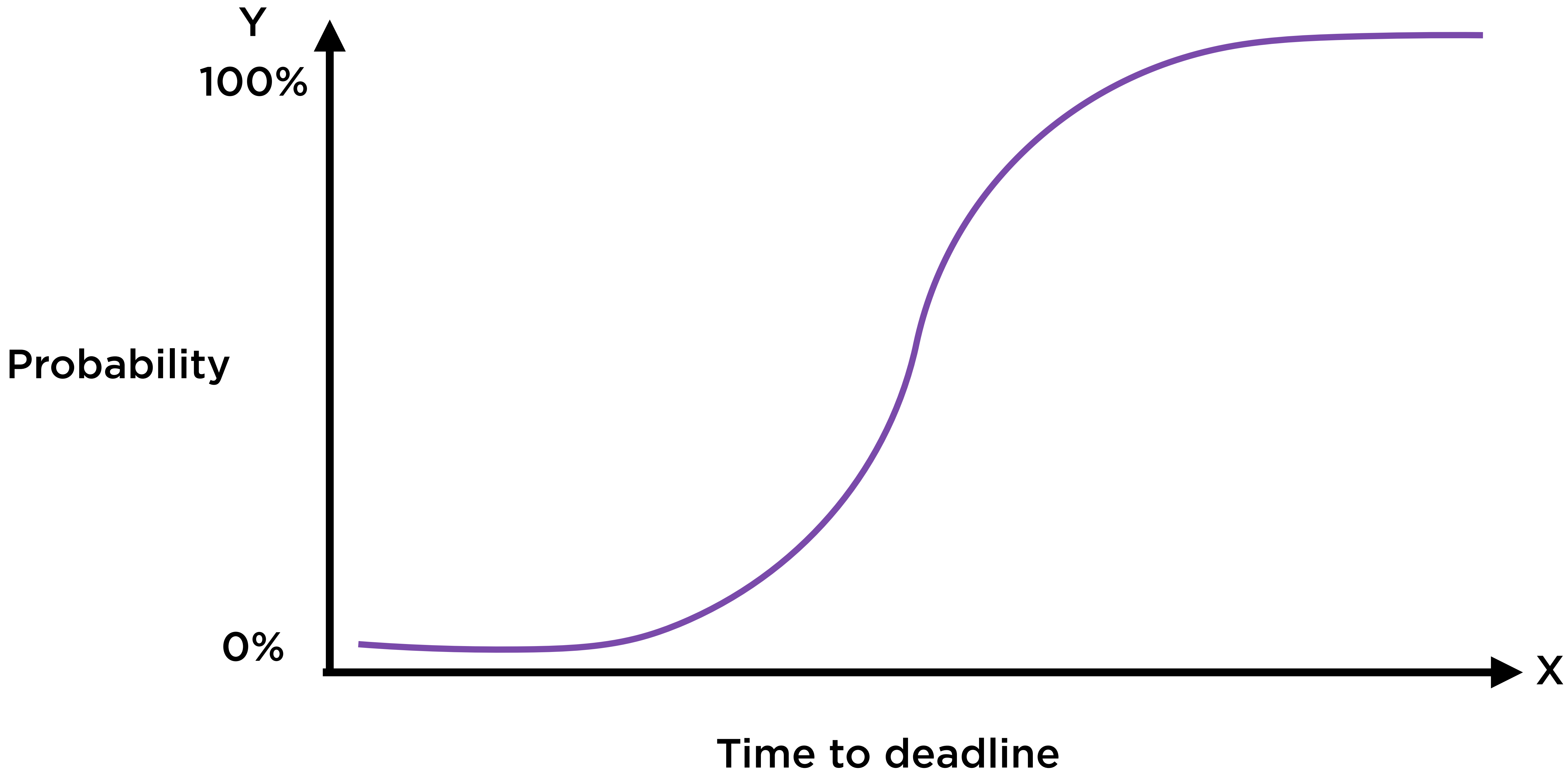


Working Hard, Fast, Smart

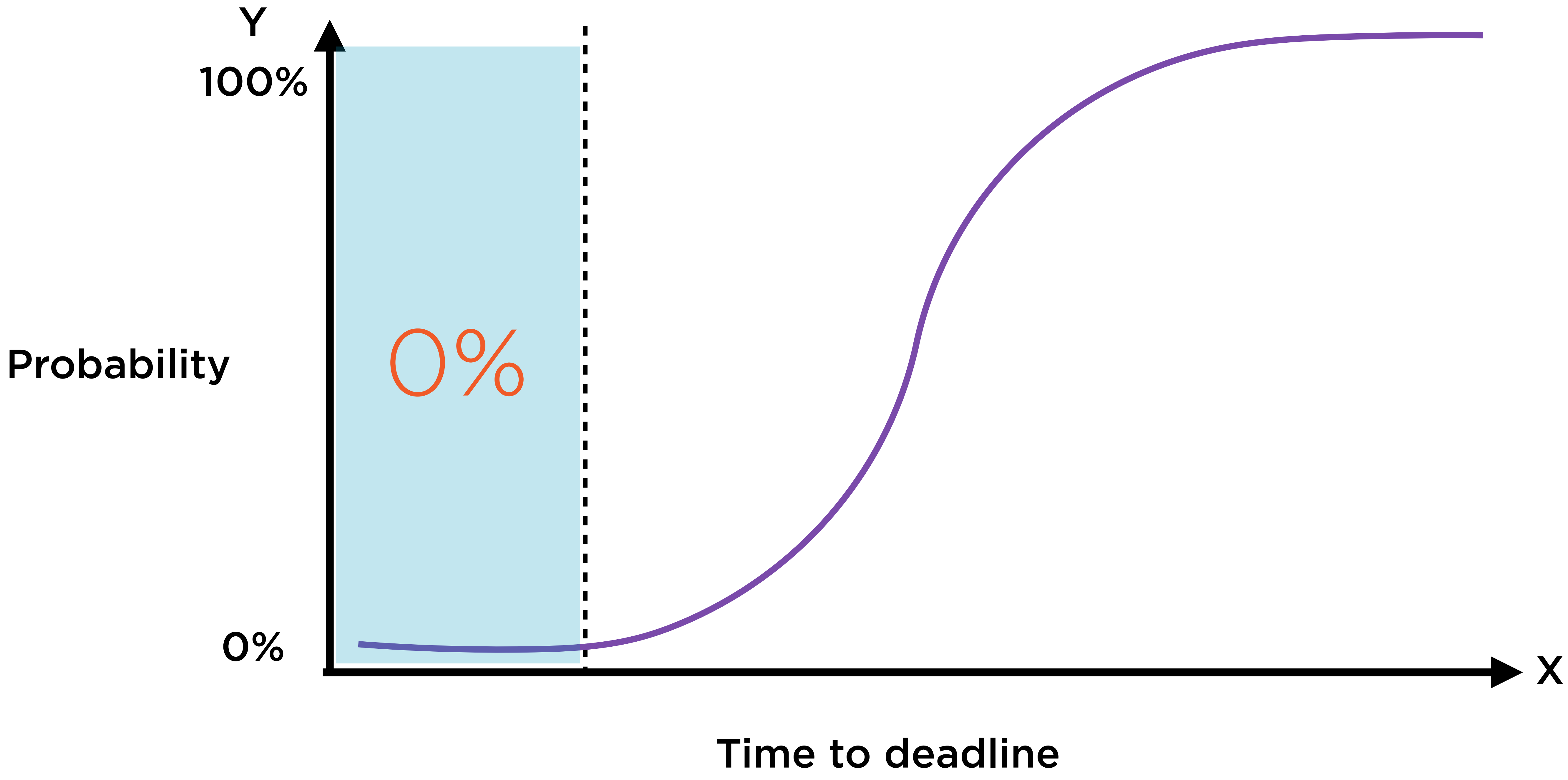


Logistic Regression helps find how probabilities are changed by actions

Working Smart with Logistic Regression

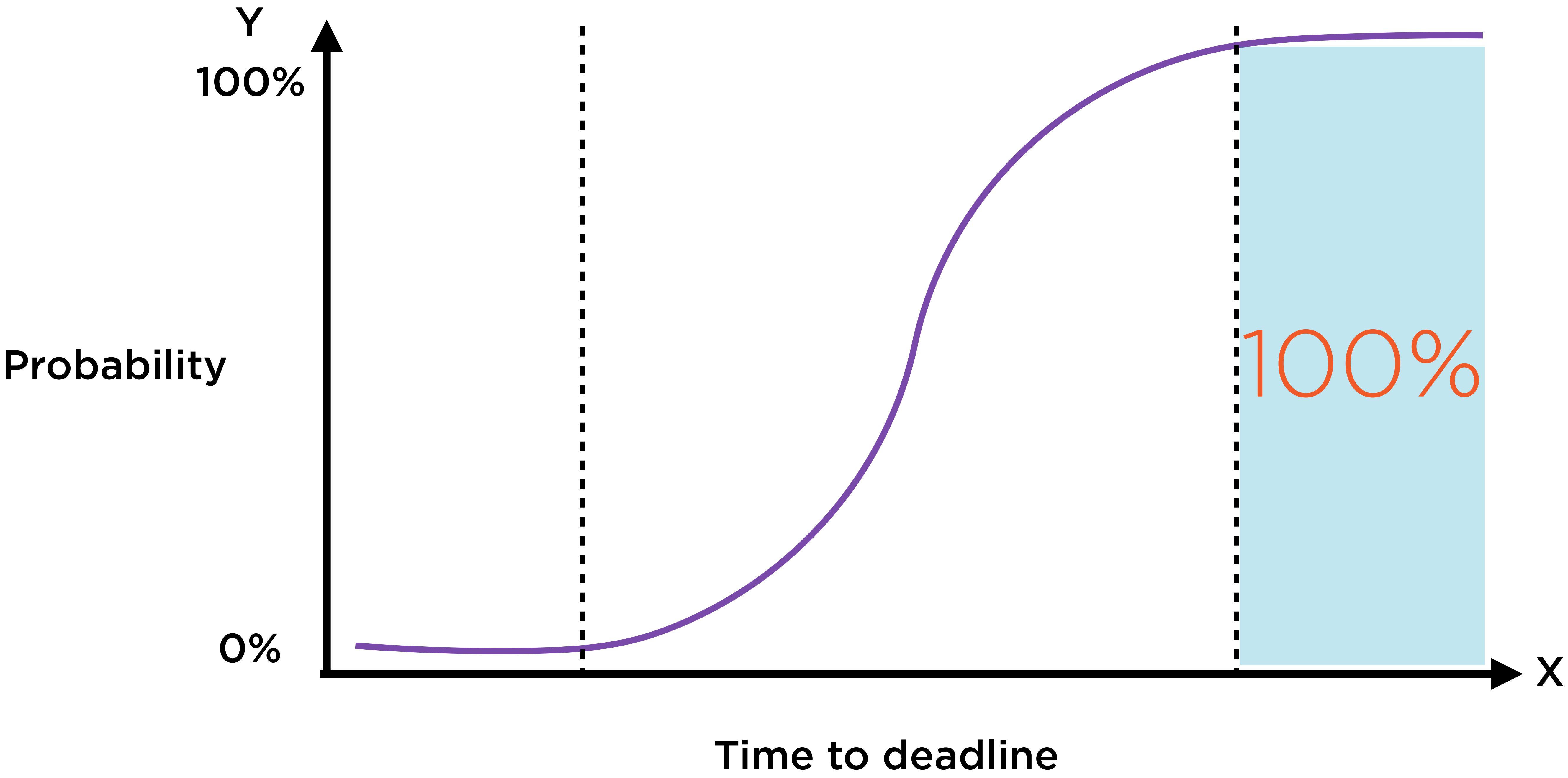


Working Smart with Logistic Regression



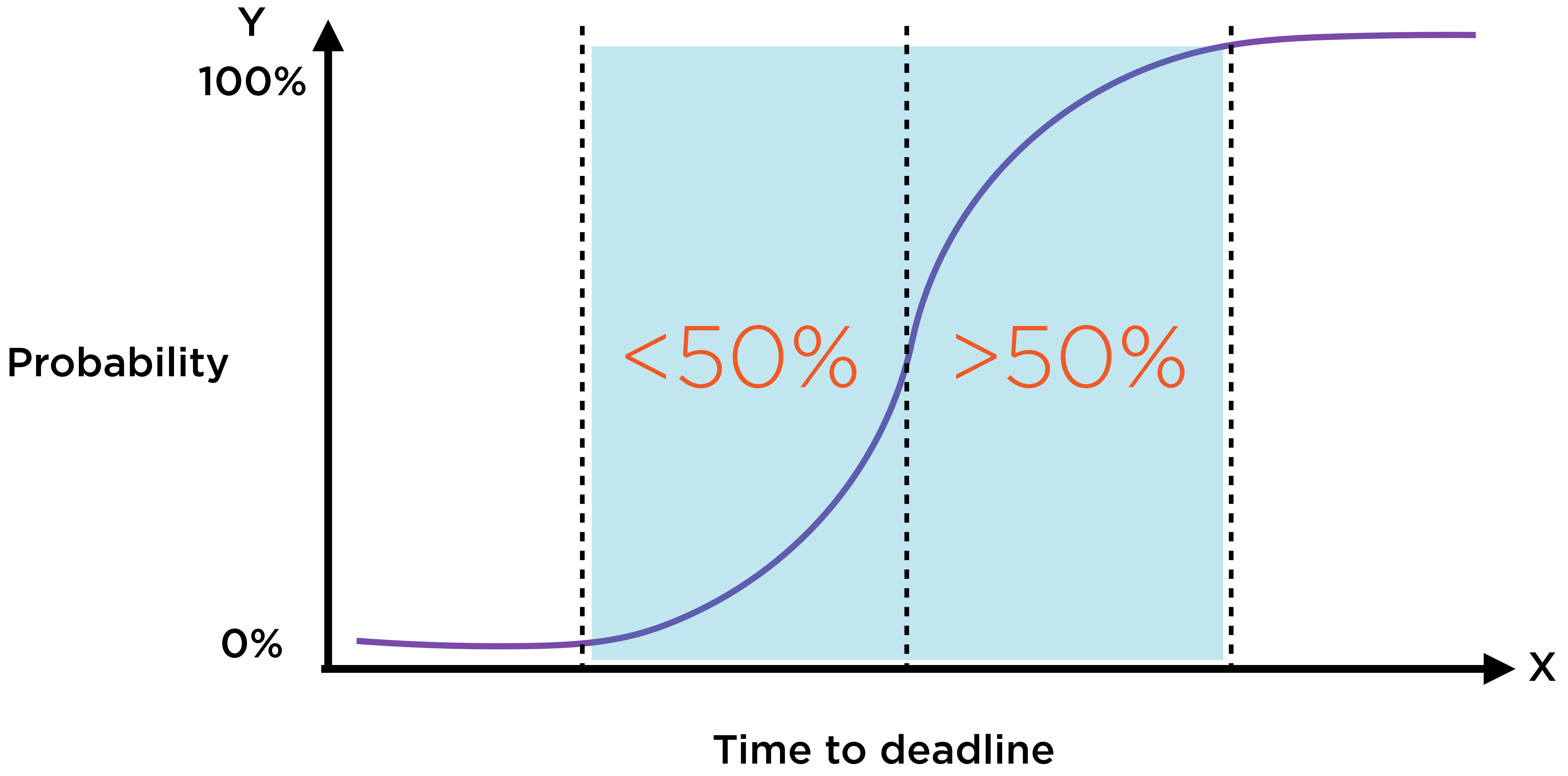
Start too late, and you’ll definitely miss

Working Smart with Logistic Regression

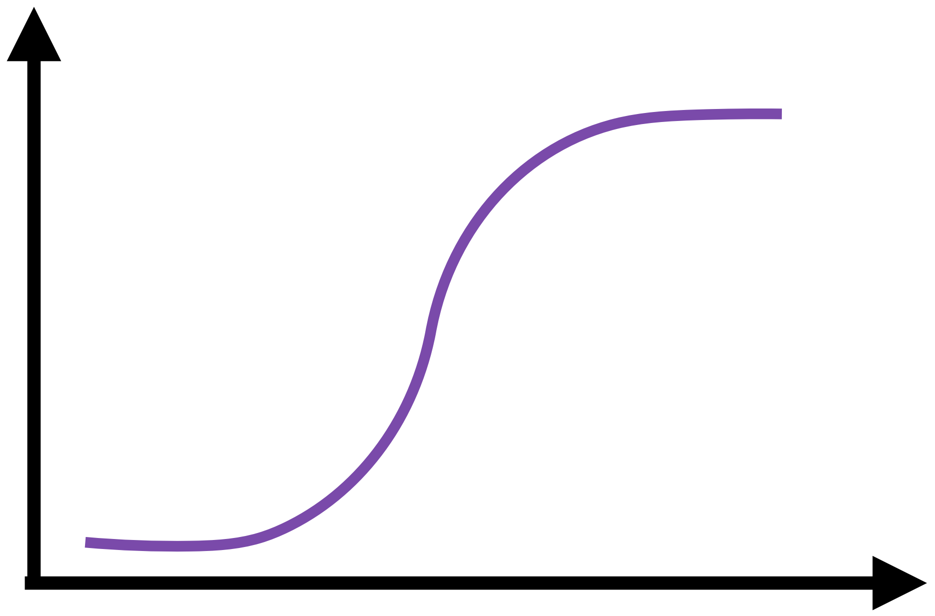


Start too early, and you'll definitely make it

Working Smart with Logistic Regression



Working smart is knowing when to start



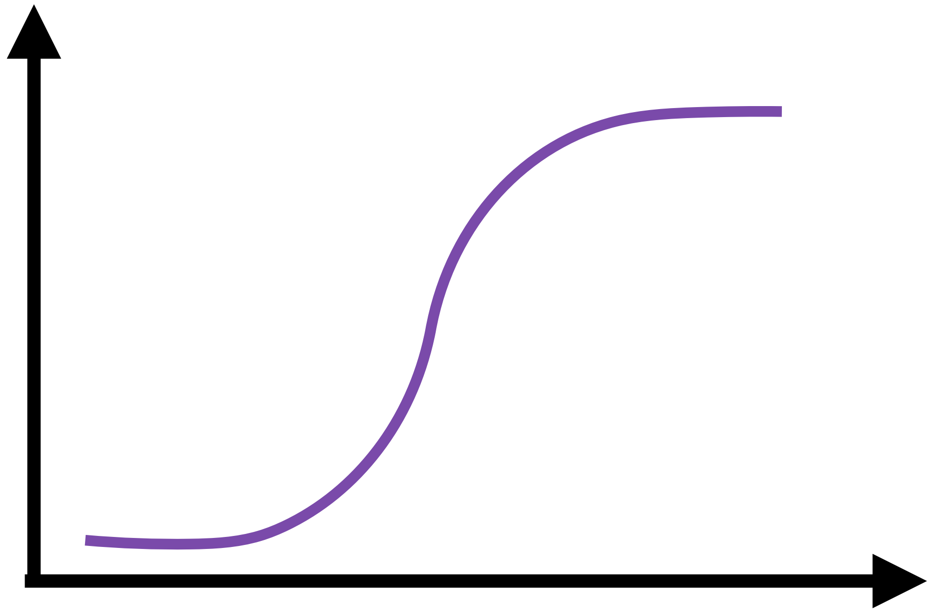
Y-axis: probability of meeting deadline

X-axis: time to deadline

Meeting or missing deadline is binary

Probability curve flattens at ends

- floor of 0
- ceiling of 1



y: hit or miss? (0 or 1?)

x: start time before deadline

p(y) : probability of y = 1

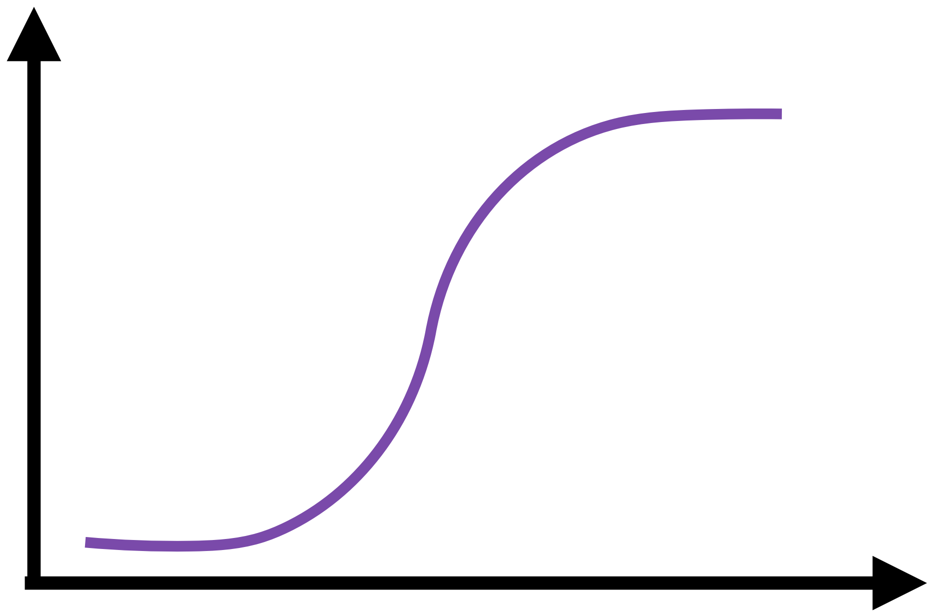
$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

Logistic regression involves finding the “best fit” such curve

- A is the intercept

- B is the regression coefficient

(e is the constant 2.71828)



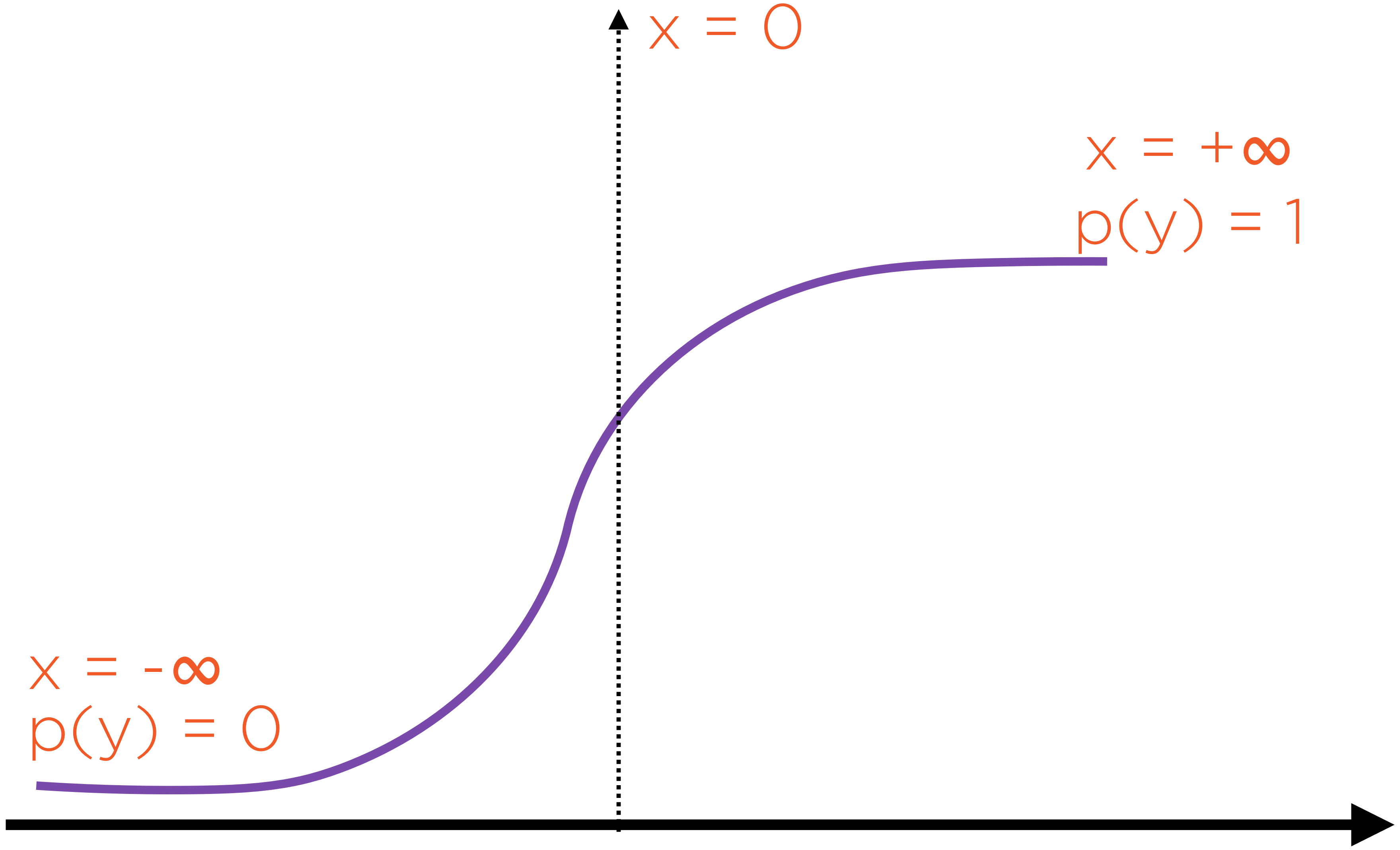
S-curves are widely studied, well understood

$$y = \frac{1}{1 + e^{-(A+Bx)}}$$

Logistic regression uses S-curve to estimate probabilities

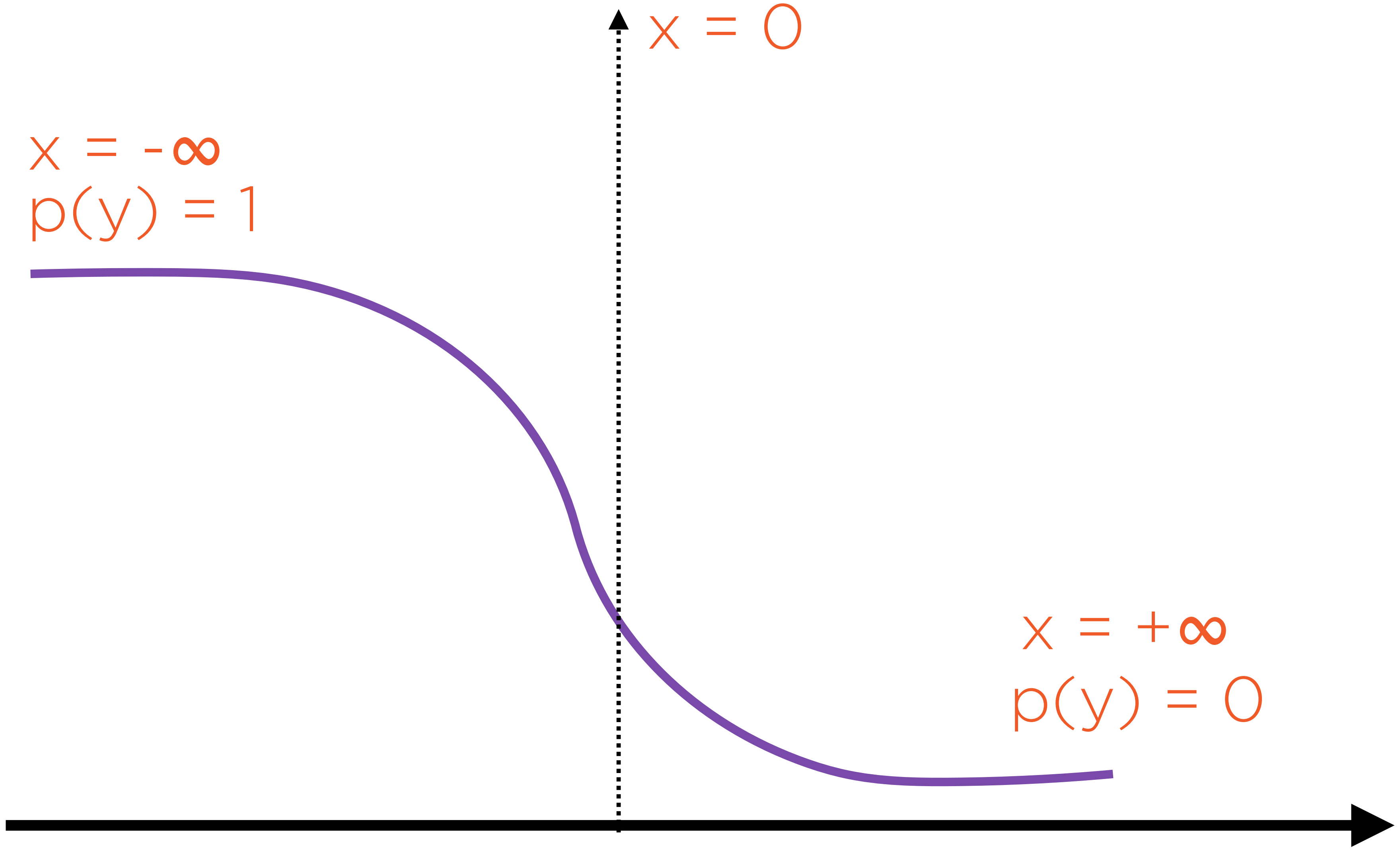
$$p(y) = \frac{1}{1 + e^{-(A+Bx)}}$$

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$



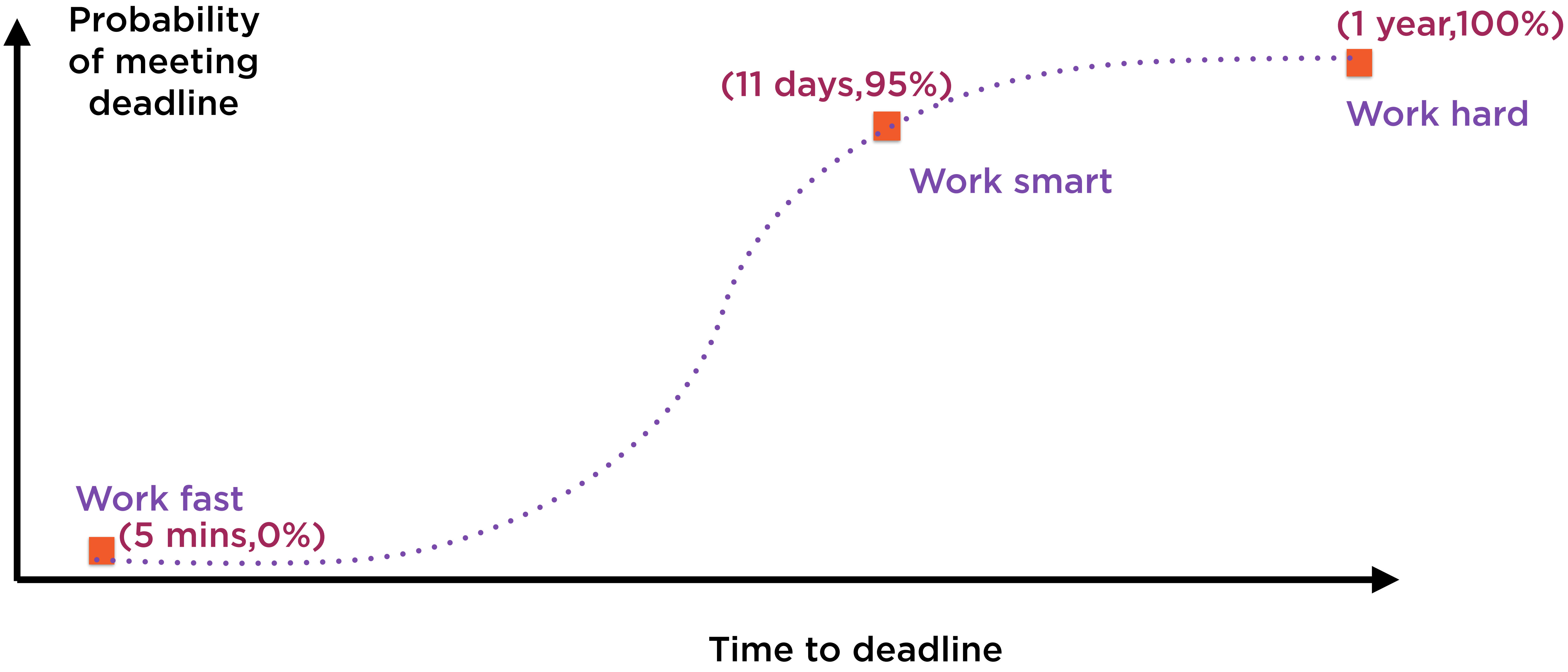
If A and B are **positive**

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

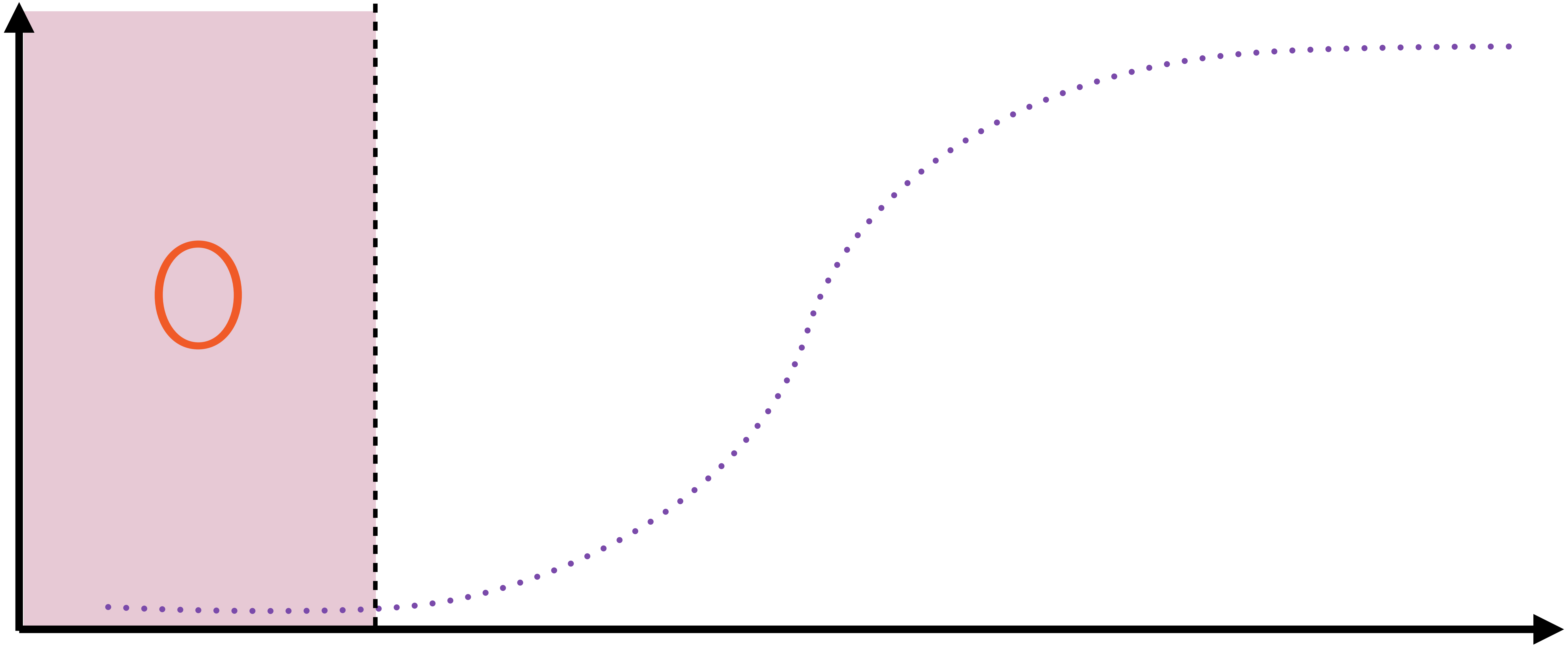


If A and B are **negative**

Working Hard, Fast, Smart

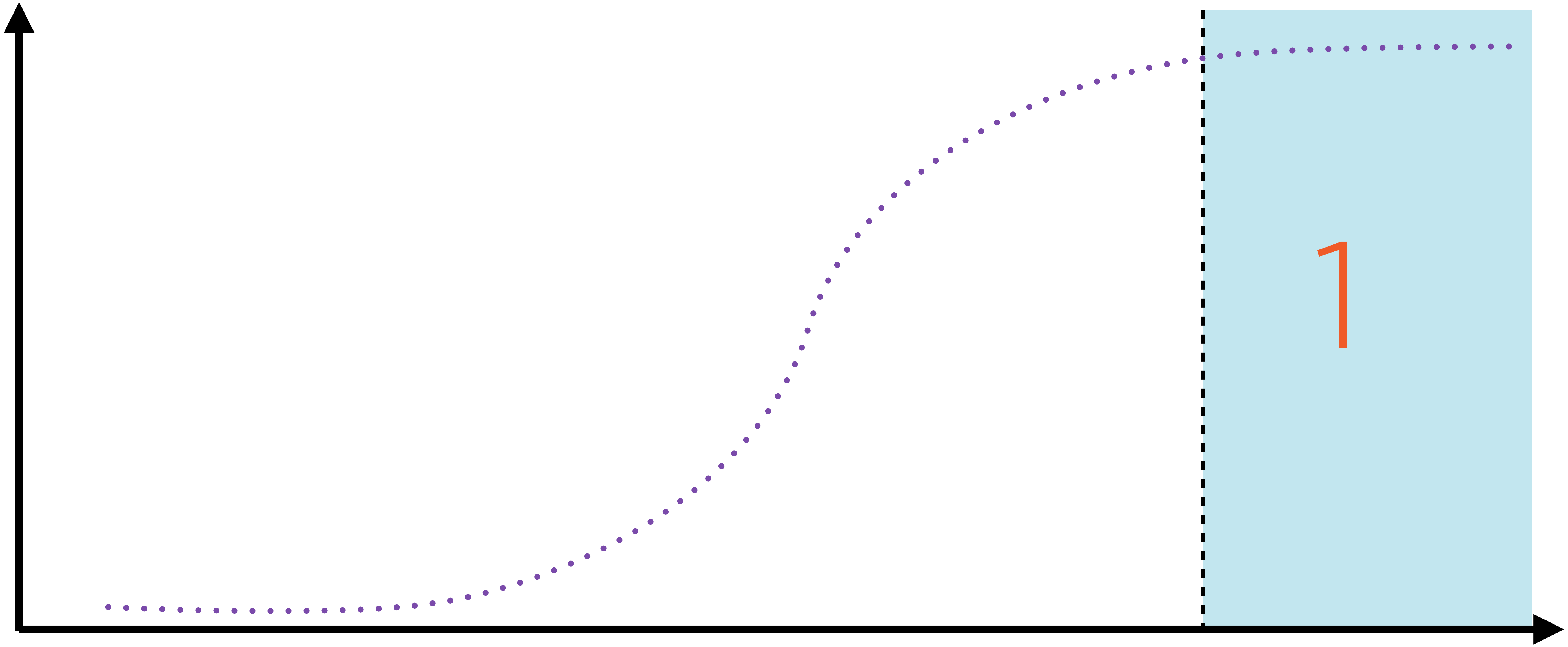


Working Hard, Fast, Smart



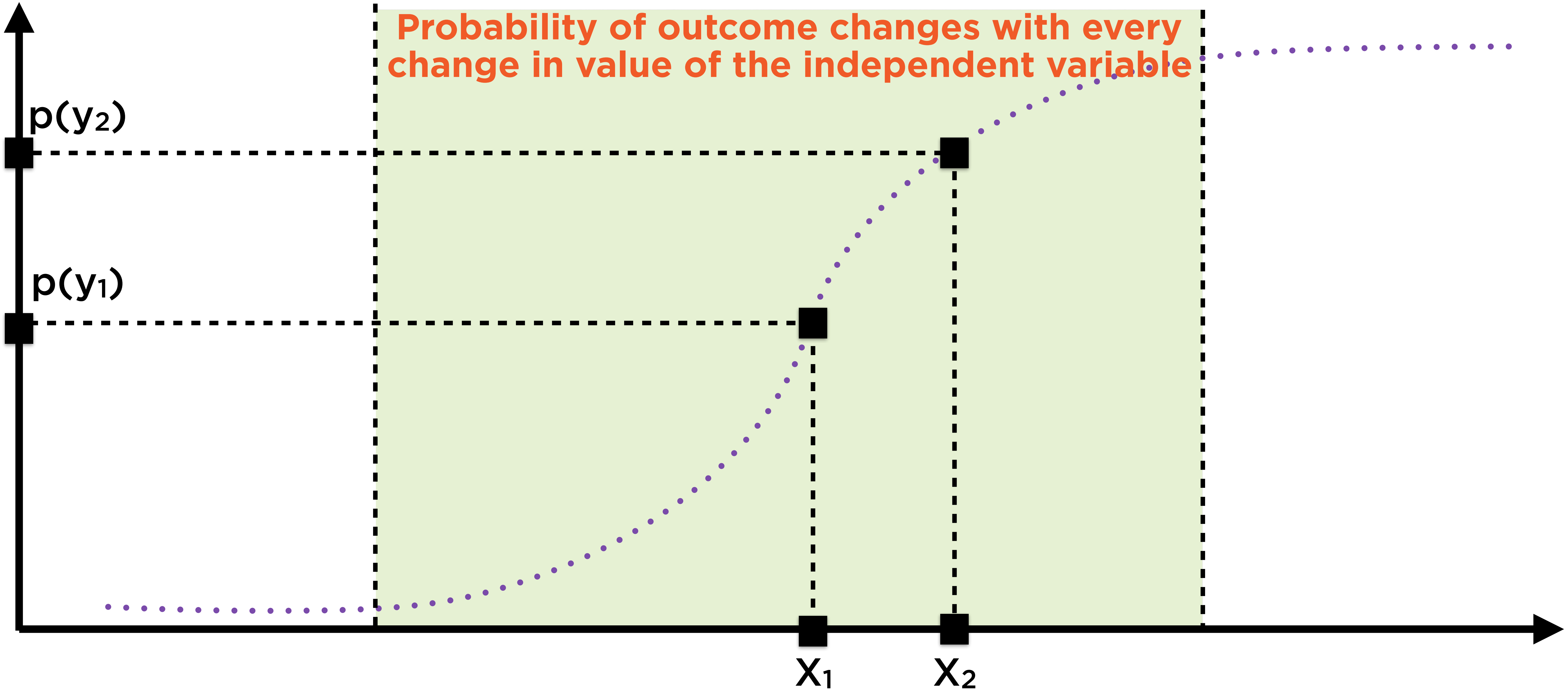
Minimum value of $p(y_i)$

Working Hard, Fast, Smart



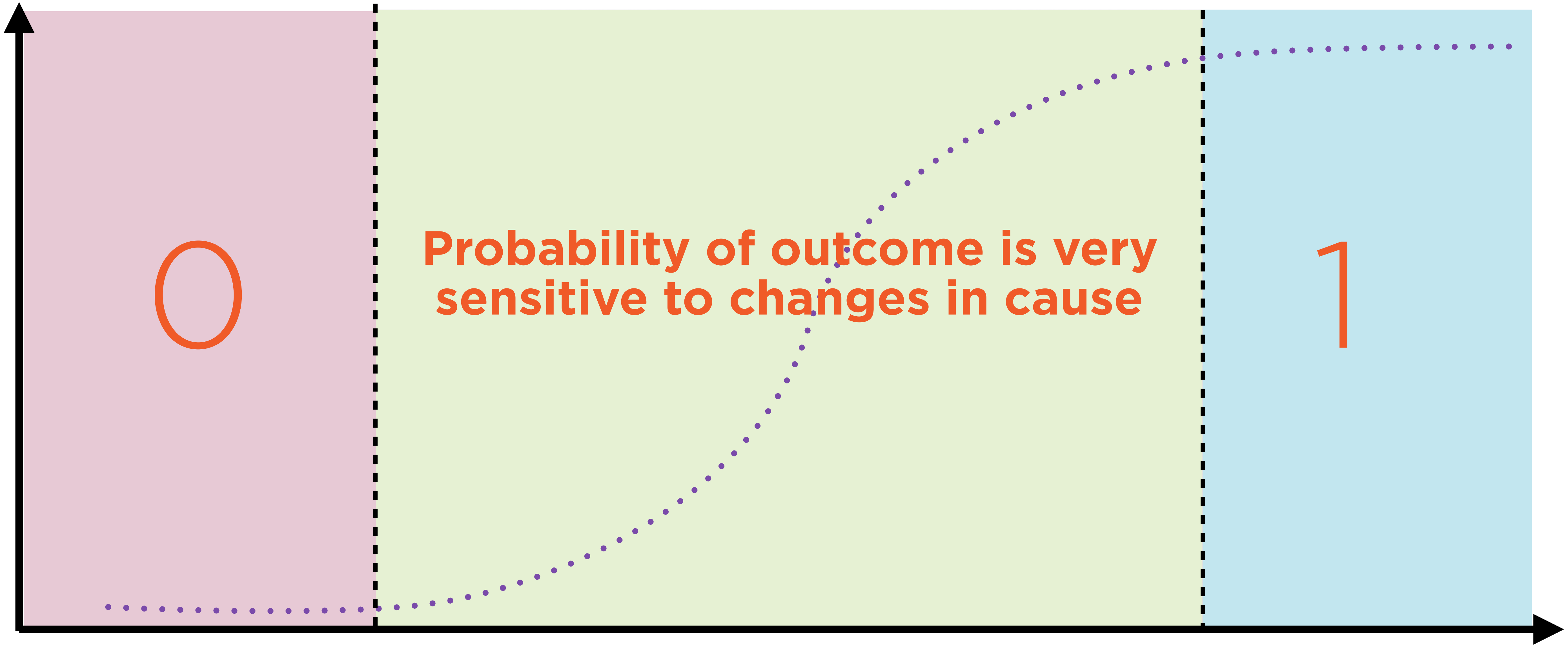
Maximum value of $p(y_i)$

Working Hard, Fast, Smart


Probability of outcome changes with every change in value of the independent variable
p(y₂)
p(y₁)
x₁
x₂

Between maximum and minimum values of $p(y_i)$

Logistic Regression



$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

Categorical and Continuous Variables

Continuous

Can take an infinite set of values
(height, weight, income...)

Categorical

Can take a finite set of values
(male/female, day of week...)

Categorical variables that can take just two values are called binary variables

Logistic Regression helps estimate how **probabilities** of **categorical variables** are influenced by **causes**

Logistic Regression in Classification

Whales: Fish or Mammals

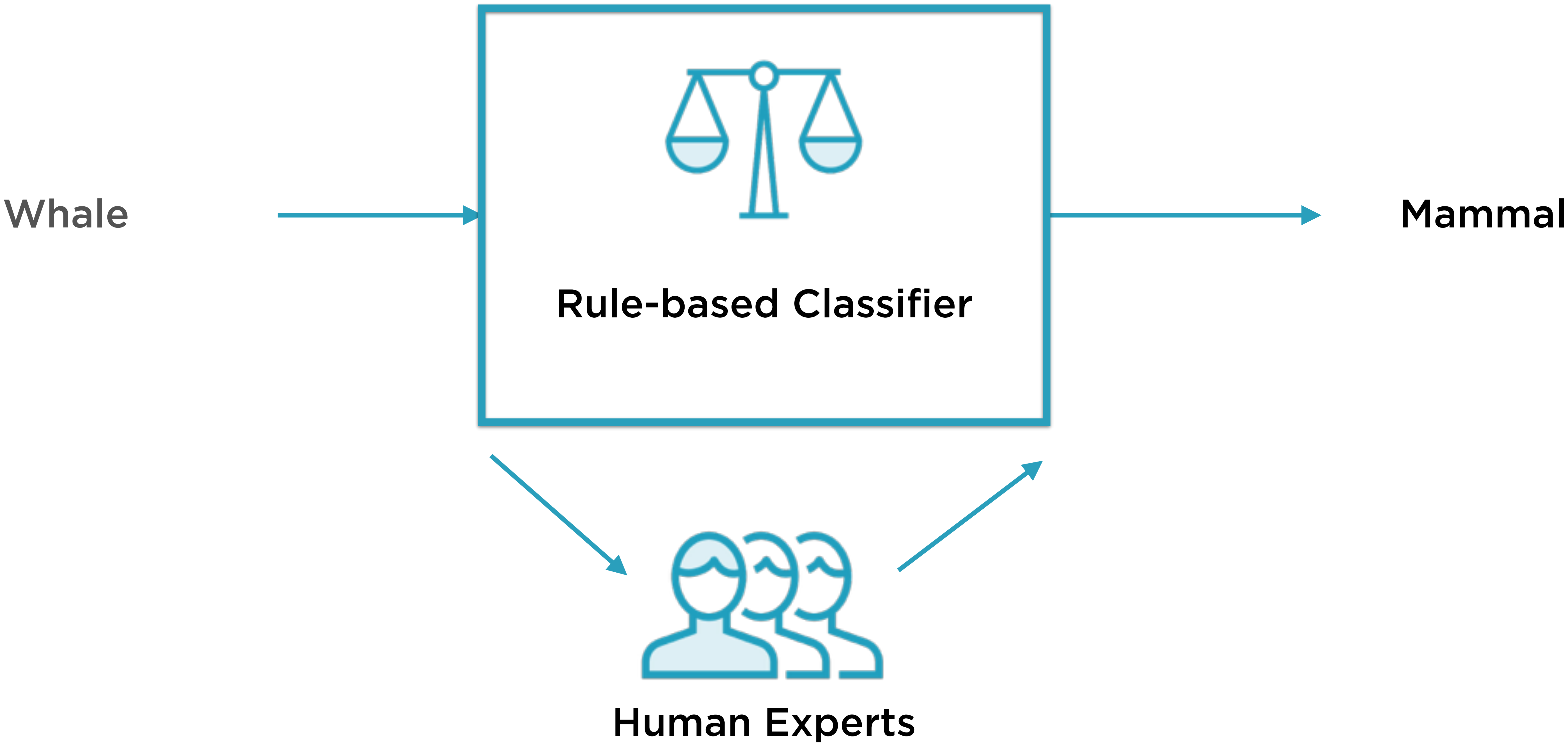
Mammal

Member of the infraorder *Cetacea*

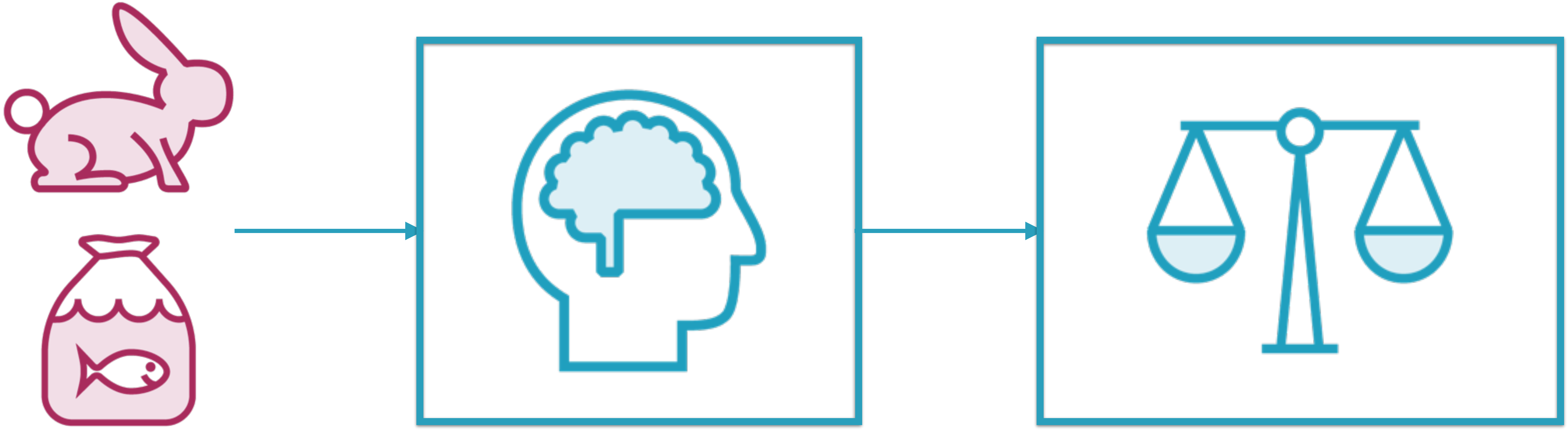
Fish

Looks like a fish, swims like a fish, moves like a fish

Rule-based Binary Classifier



ML-based Binary Classifier

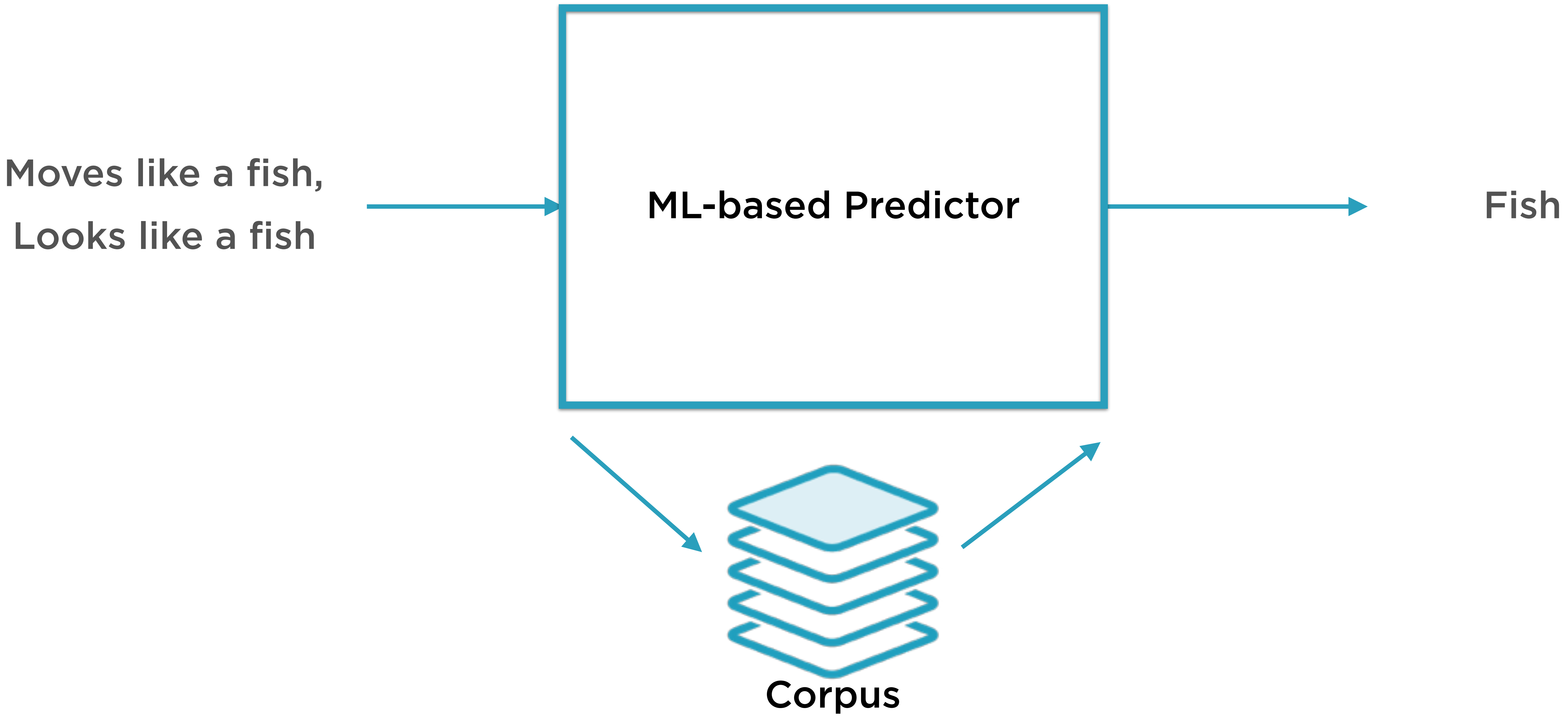


Corpus

Classification Algorithm

ML-based Classifier

ML-based Binary Classifier



Moves like a fish,

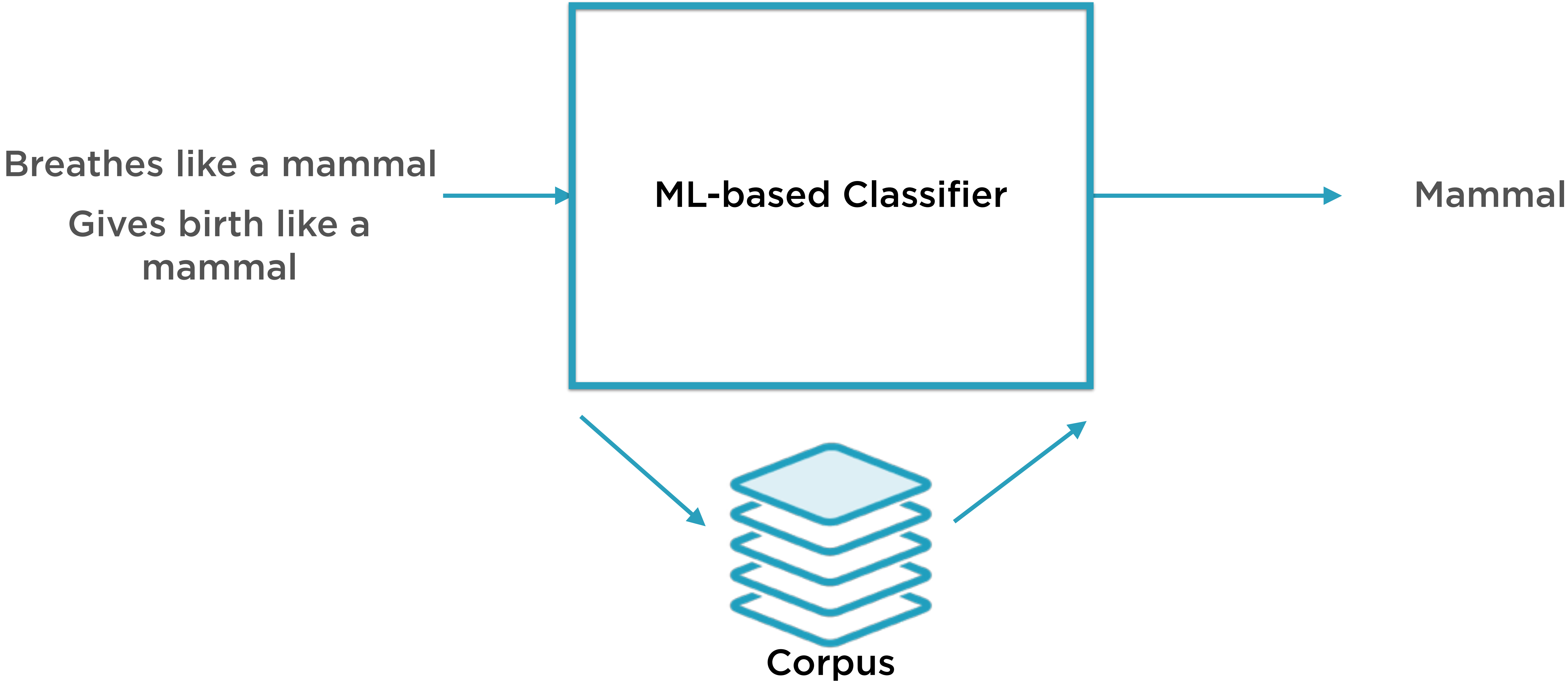
Looks like a fish

ML-based Predictor

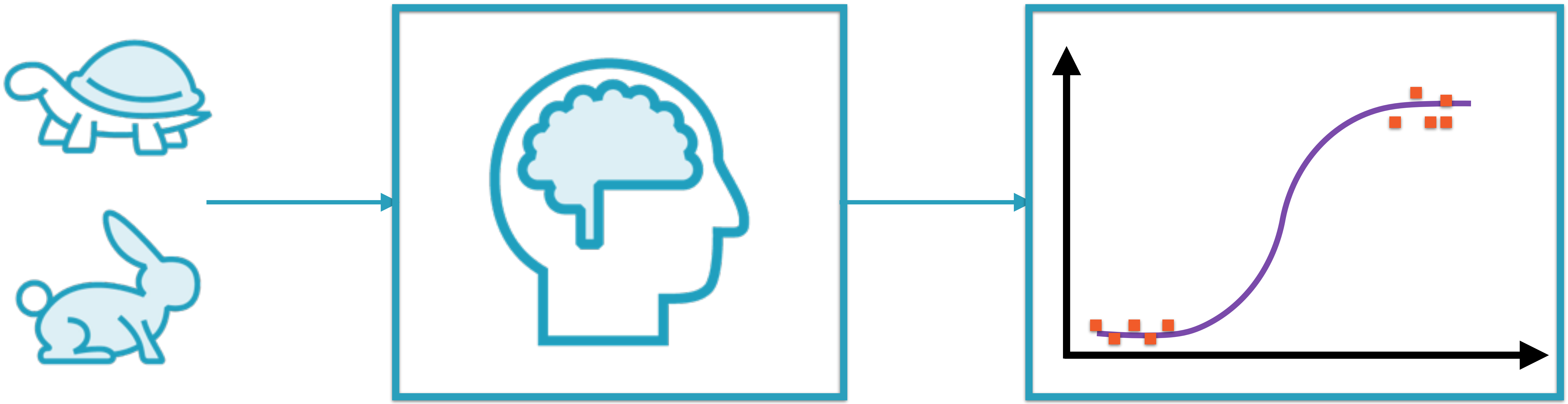
Fish

Corpus

ML-based Binary Classifier



ML-based Predictor



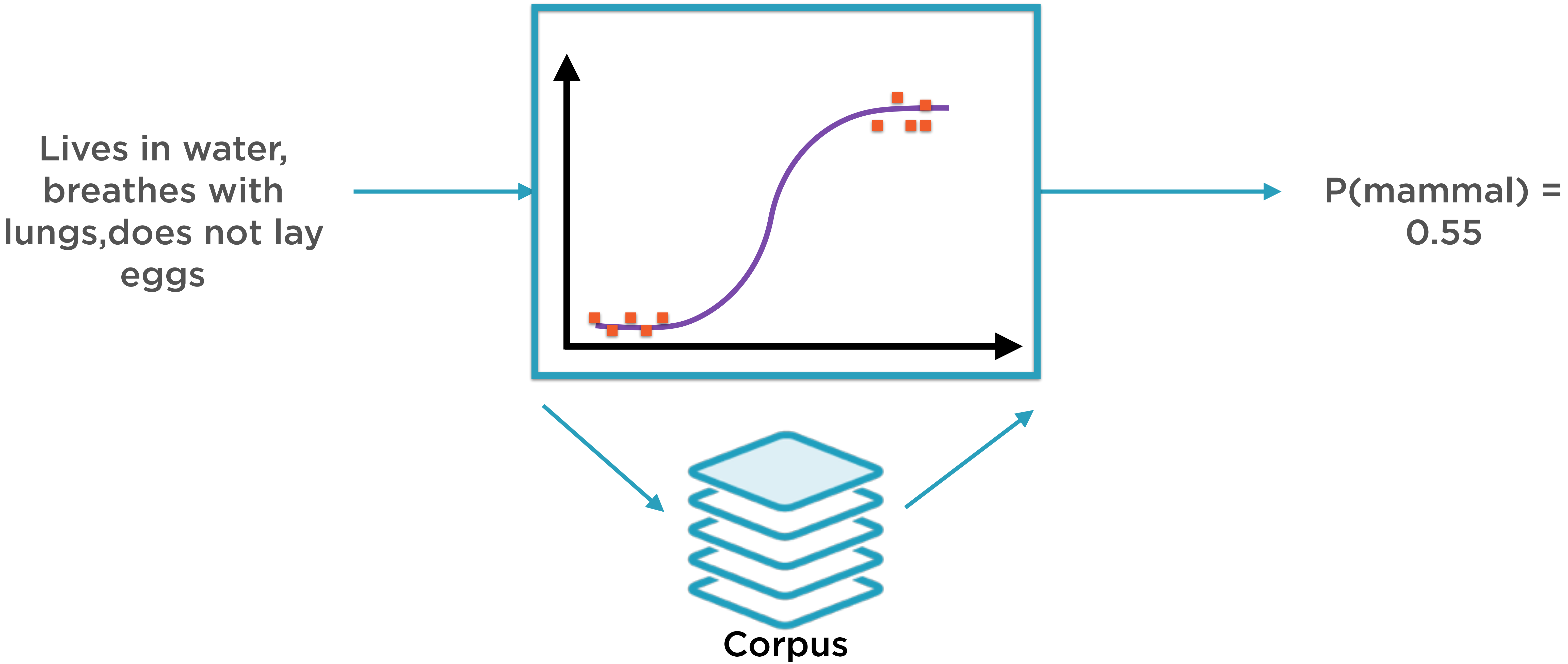
Corpus

Logistic Regression

ML-based Predictor

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

ML-based Predictor

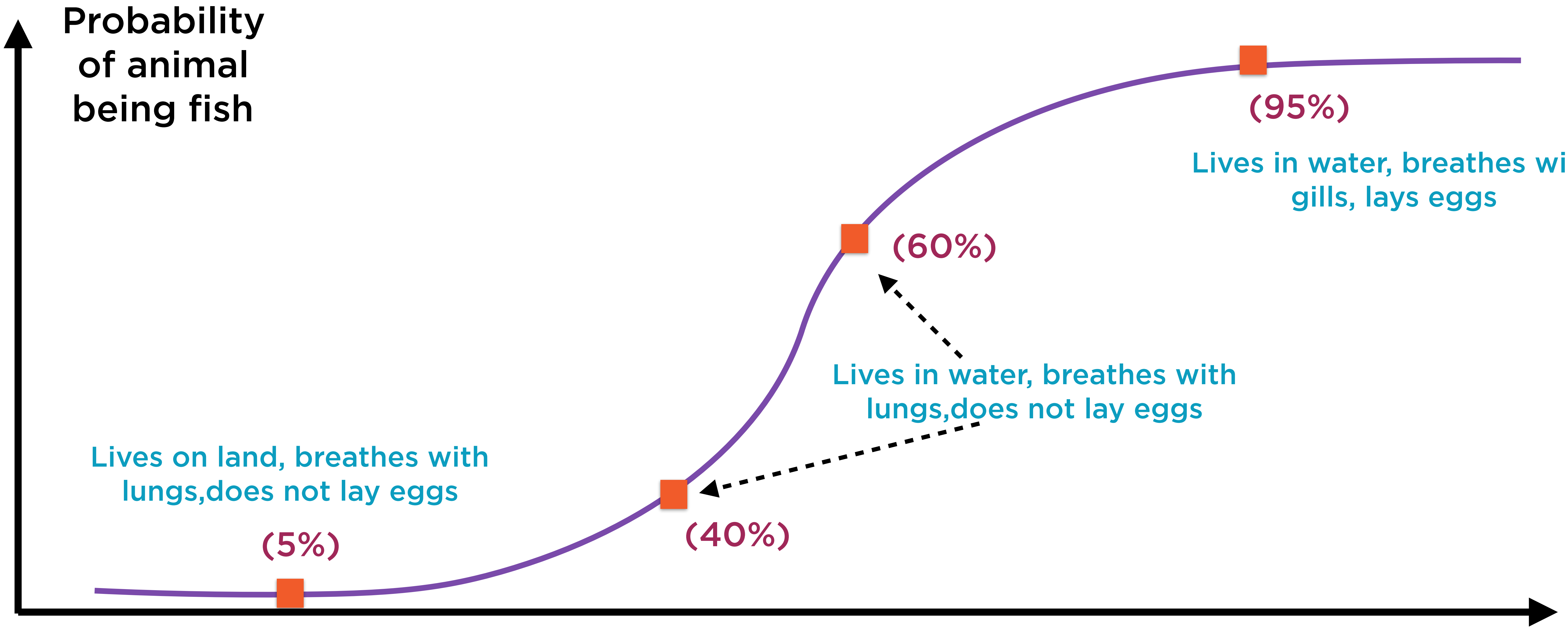


Lives in water, breathes with lungs,does not lay eggs

P(mammal) = 0.55

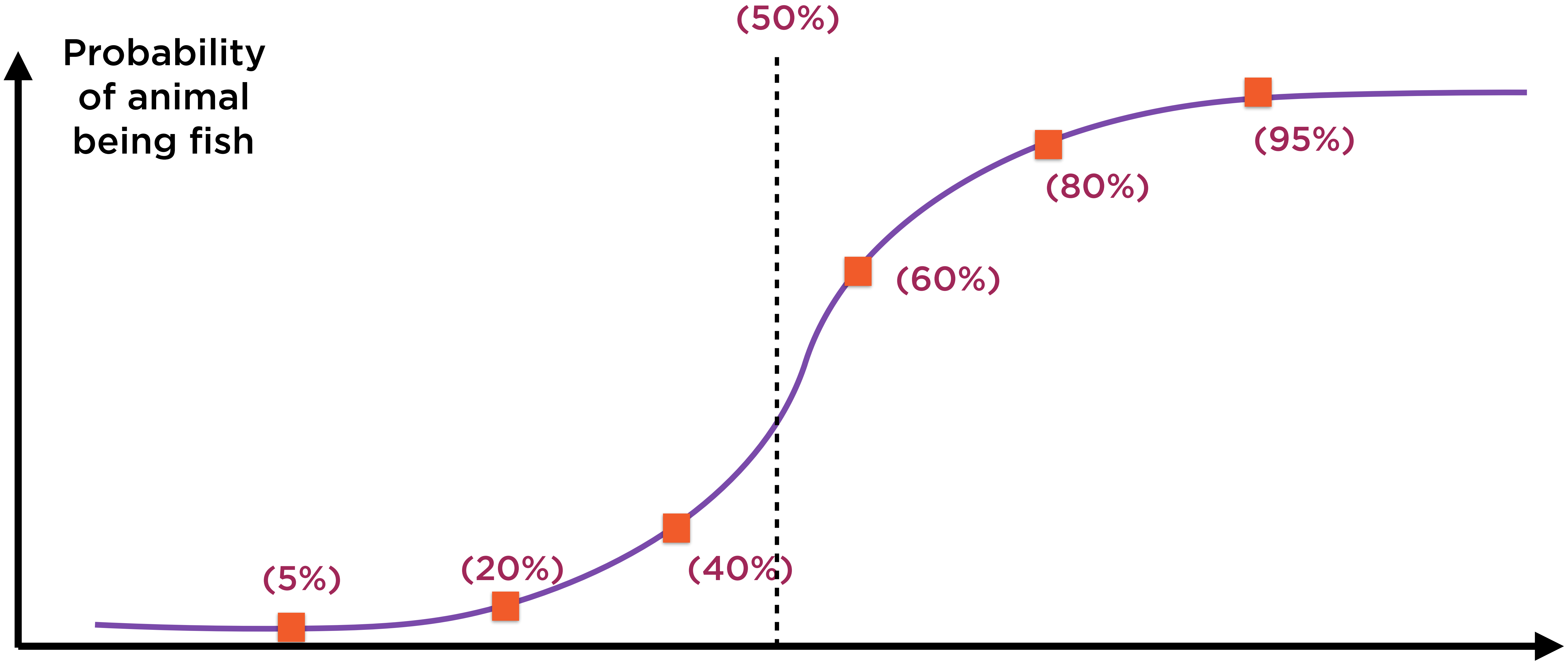
Corpus

Applying Logistic Regression



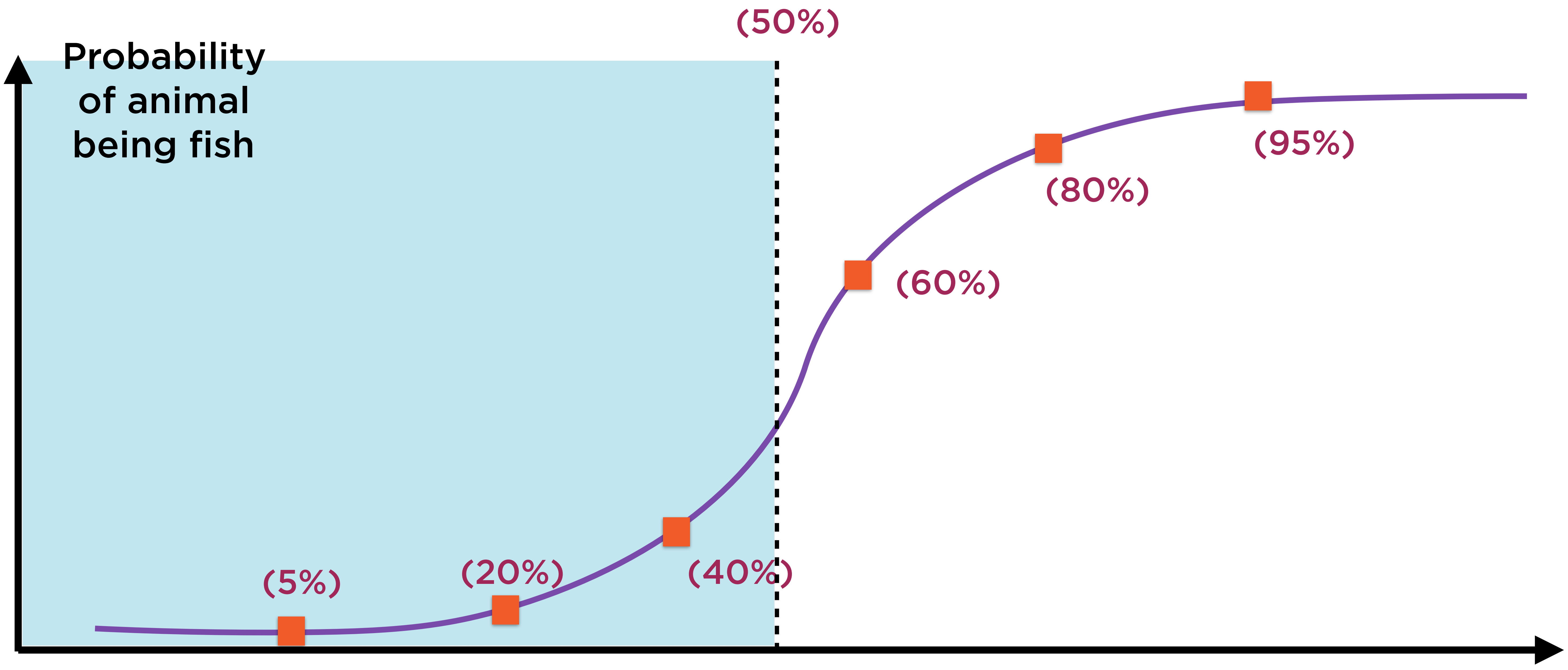
Whales: Fish or Mammals?

Applying Logistic Regression



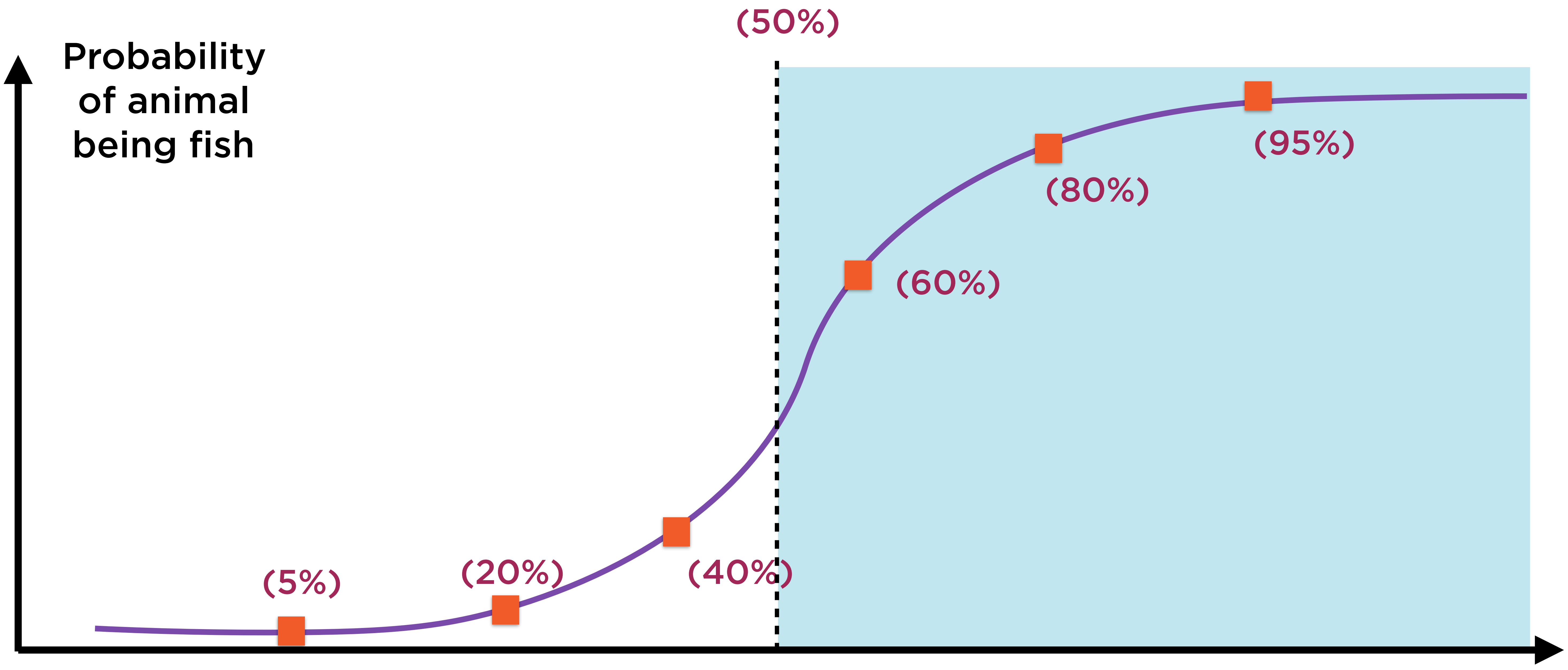
Rule of 50%

Applying Logistic Regression



If probability < 50%, it’s a mammal

Applying Logistic Regression



If probability > 50%, it's a fish

Applying Logistic Regression

Mammal

Fish

Probability of whales being fish < 50%

Applying Logistic Regression

Mammal

Fish ✓

Probability of whales being fish > 50%

Logistic Regression and Linear Regression

X Causes Y

Cause

Independent variable

Effect

Dependent variable

X Causes Y

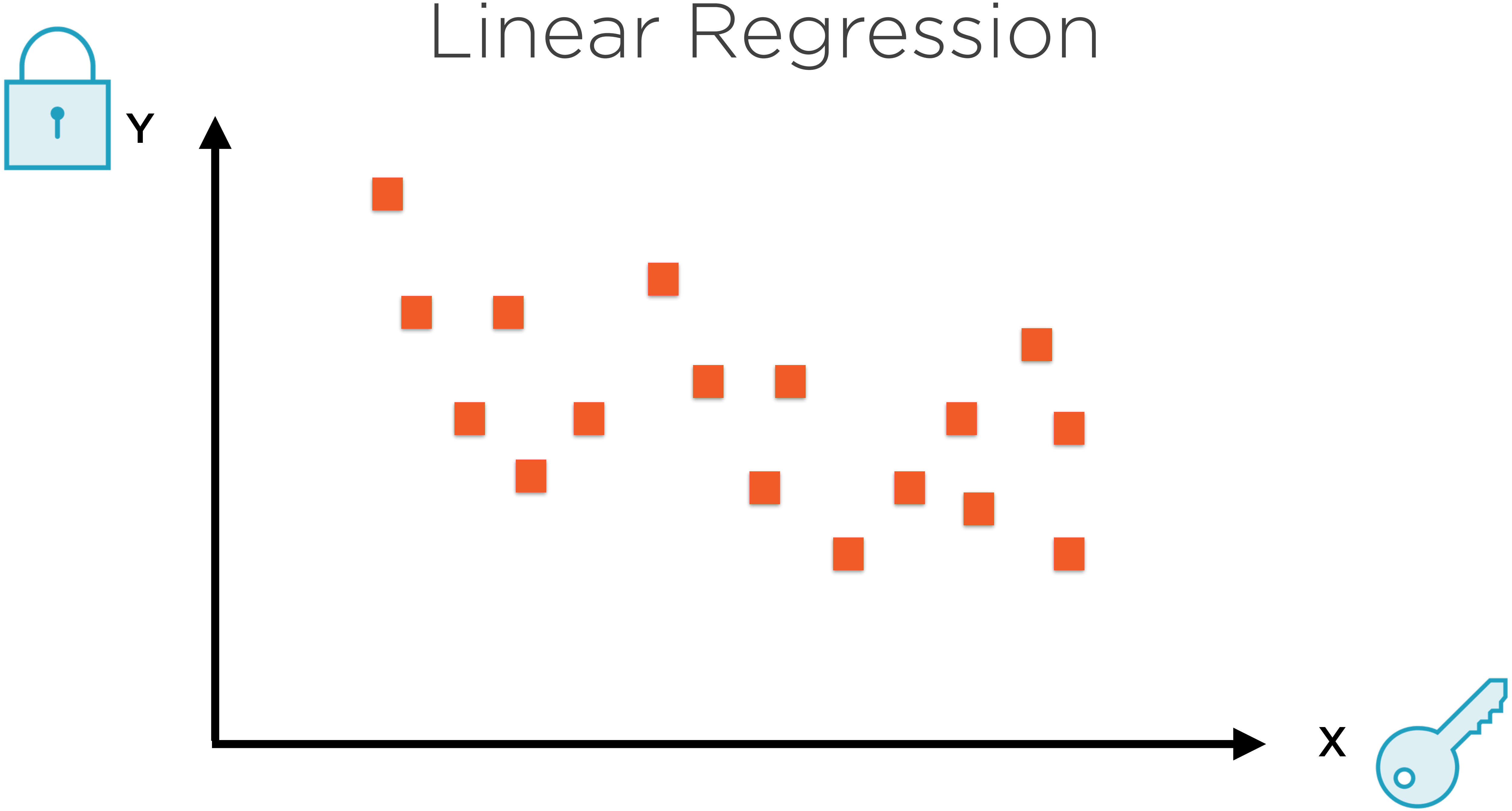
Cause

Explanatory variable

Effect

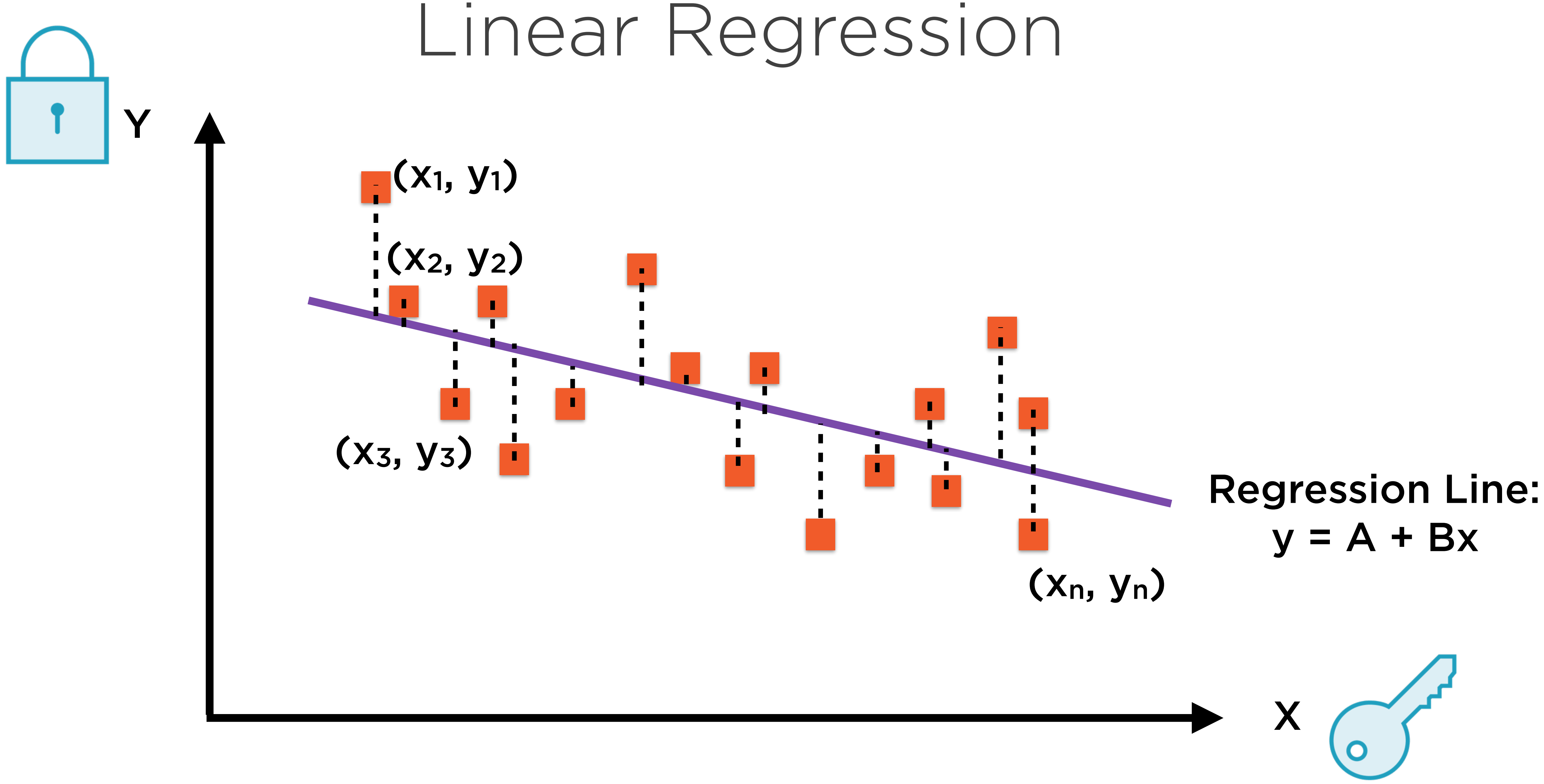
Dependent variable

Linear Regression



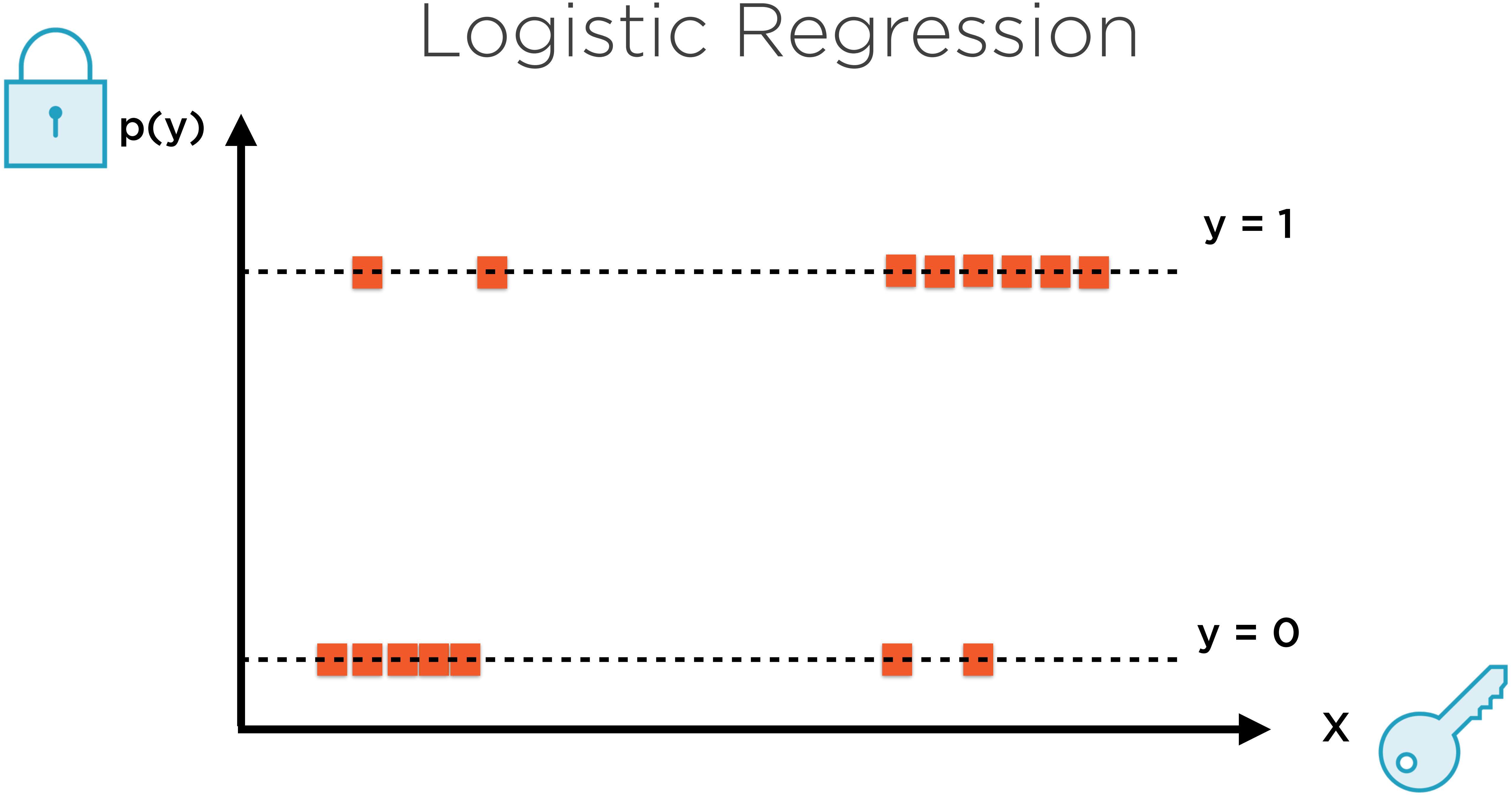
Represent all n points as
(x_i, y_i), where i = 1 to n

Linear Regression



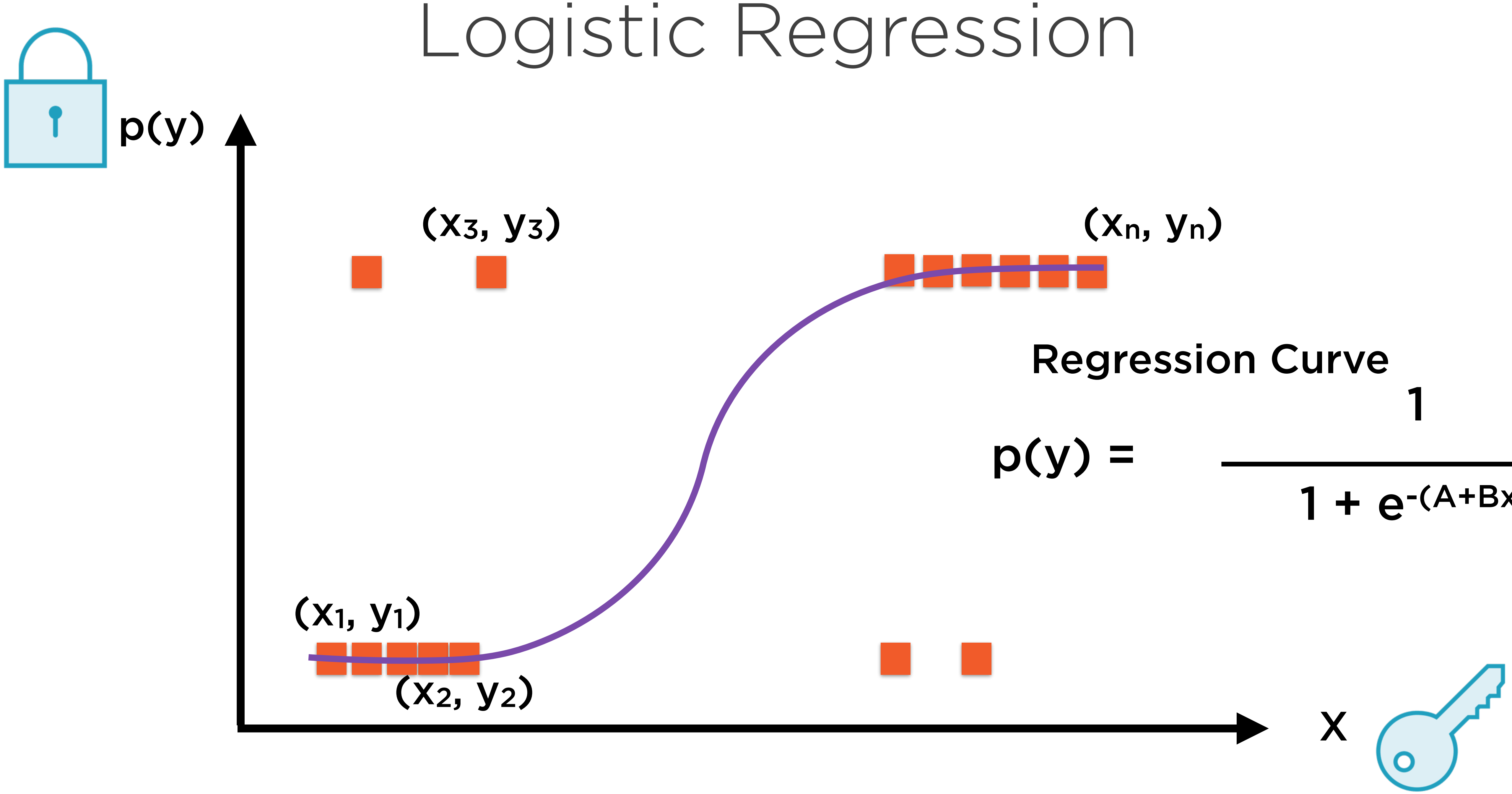
Represent all n points as (x_i, y_i), where $i = 1$ to n

Logistic Regression



Represent all n points as (x_i, y_i), where i = 1 to n

Logistic Regression



Regression Curve

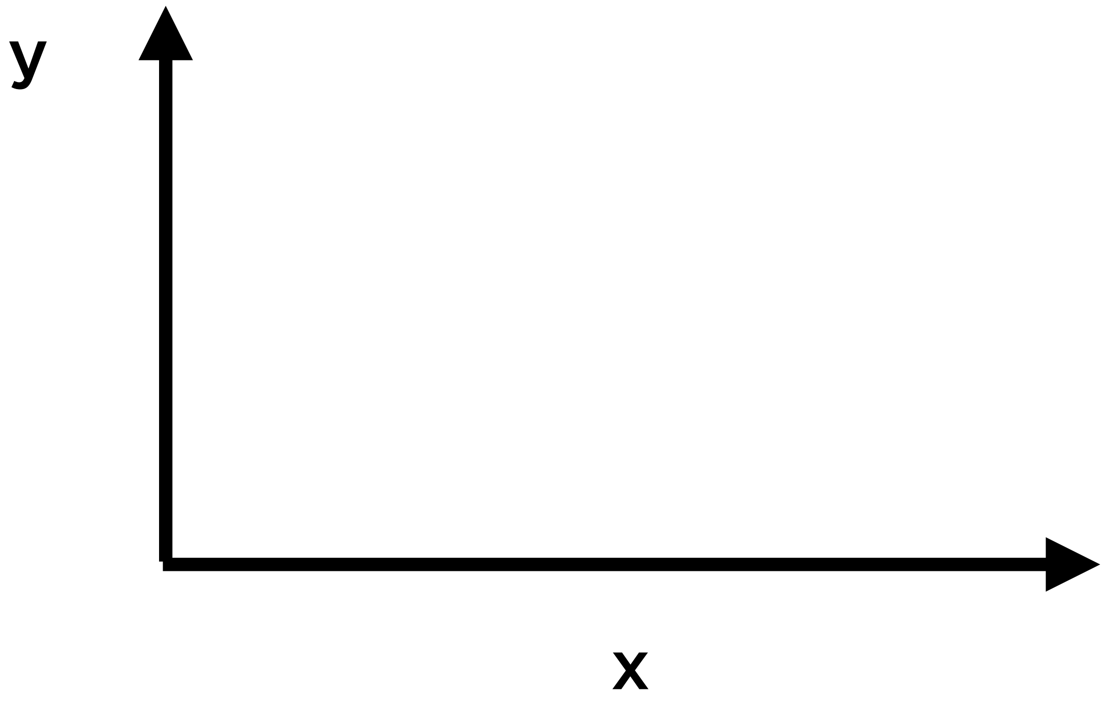
$$p(y) = \frac{1}{1 + e^{-(A+Bx)}}$$

Represent all n points as (x_i, y_i), where i = 1 to n

Similar, yet Different

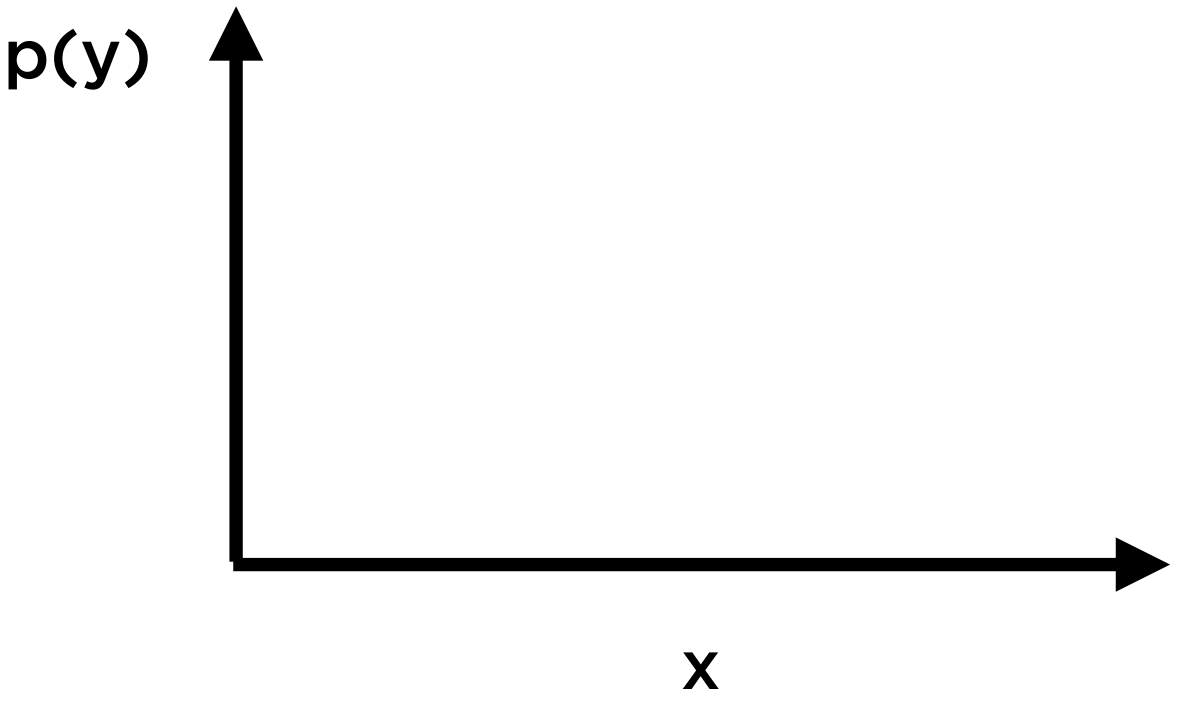
Linear Regression

Given causes, predict effect



Logistic Regression

Given causes, predict probability of effect



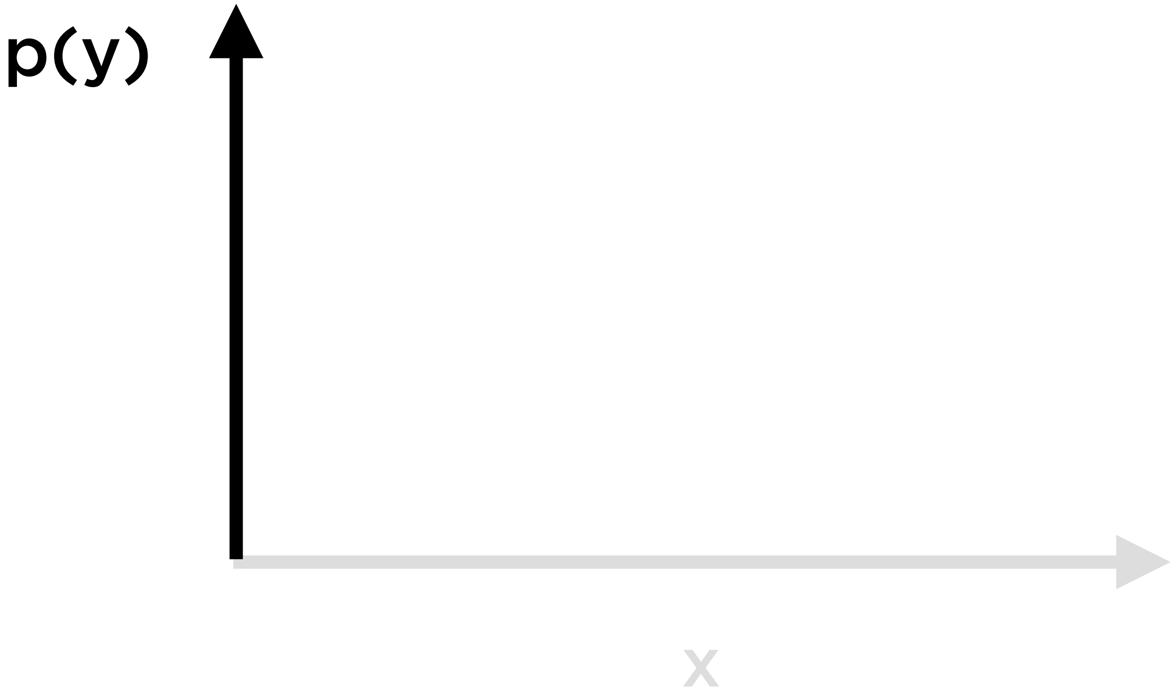
Similar, yet Different

Linear Regression

Effect variable (y) must be continuous

Logistic Regression

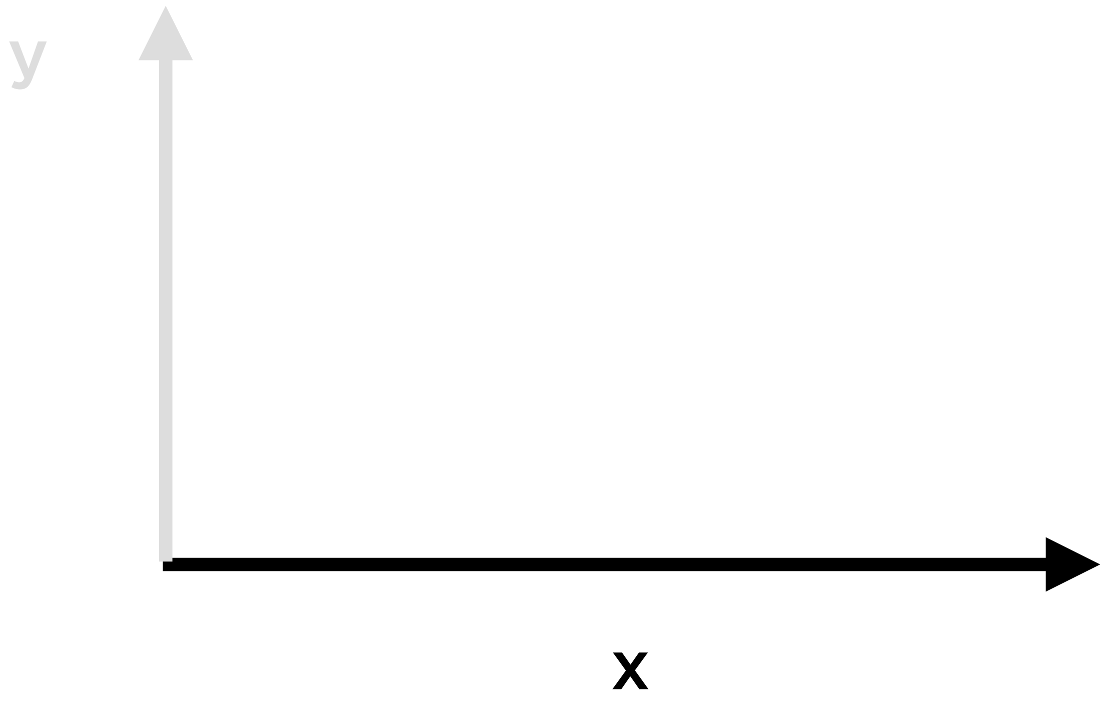
Effect variable (y) must be categorical



Similar, yet Different

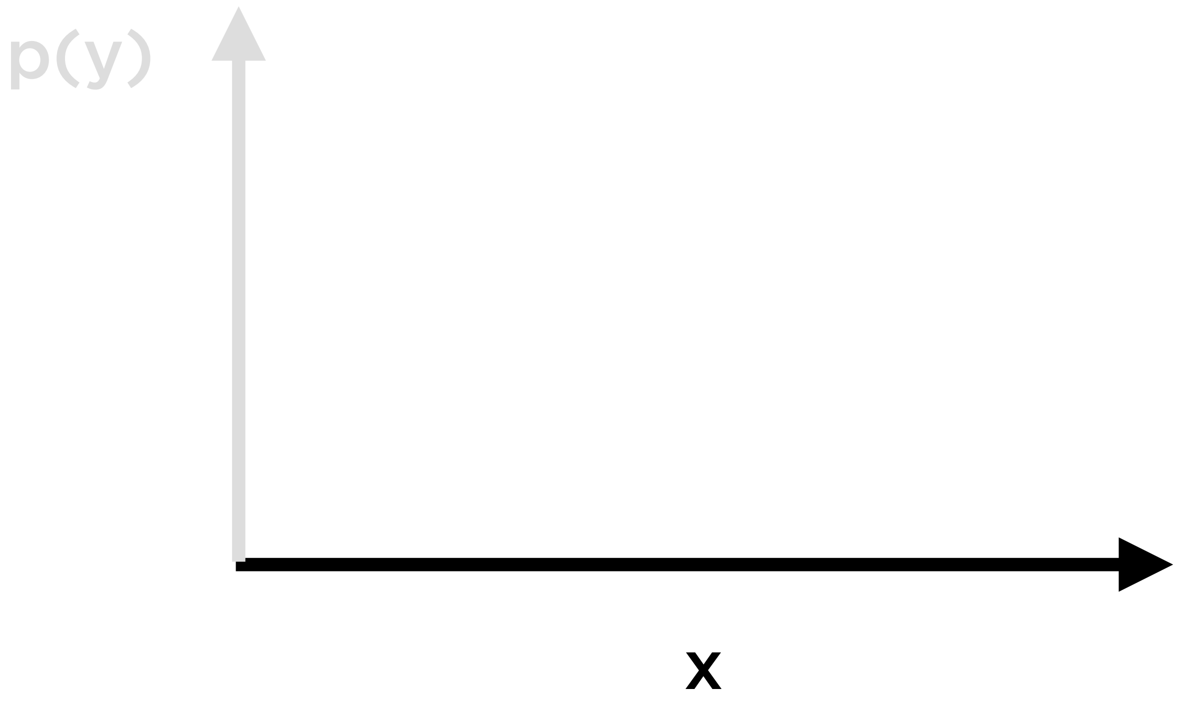
Linear Regression

Cause variables (x) can be continuous or categorical



Logistic Regression

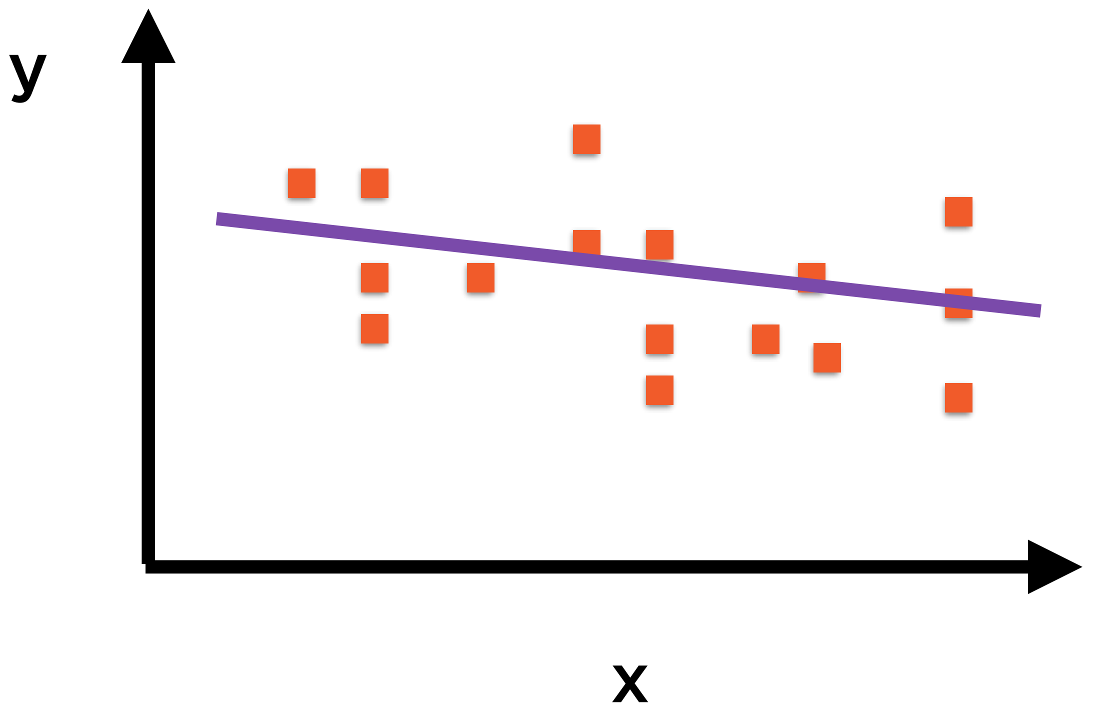
Cause variables (x) can be continuous or categorical



Similar, yet Different

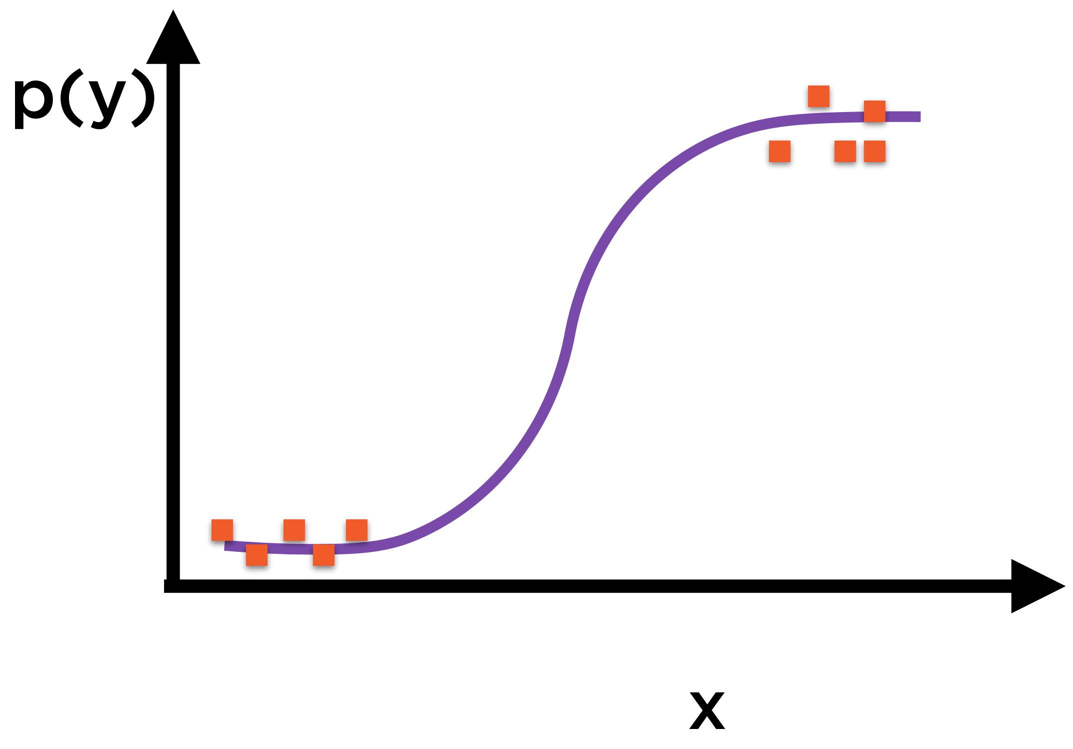
Linear Regression

Connect the dots with a straight line



Logistic Regression

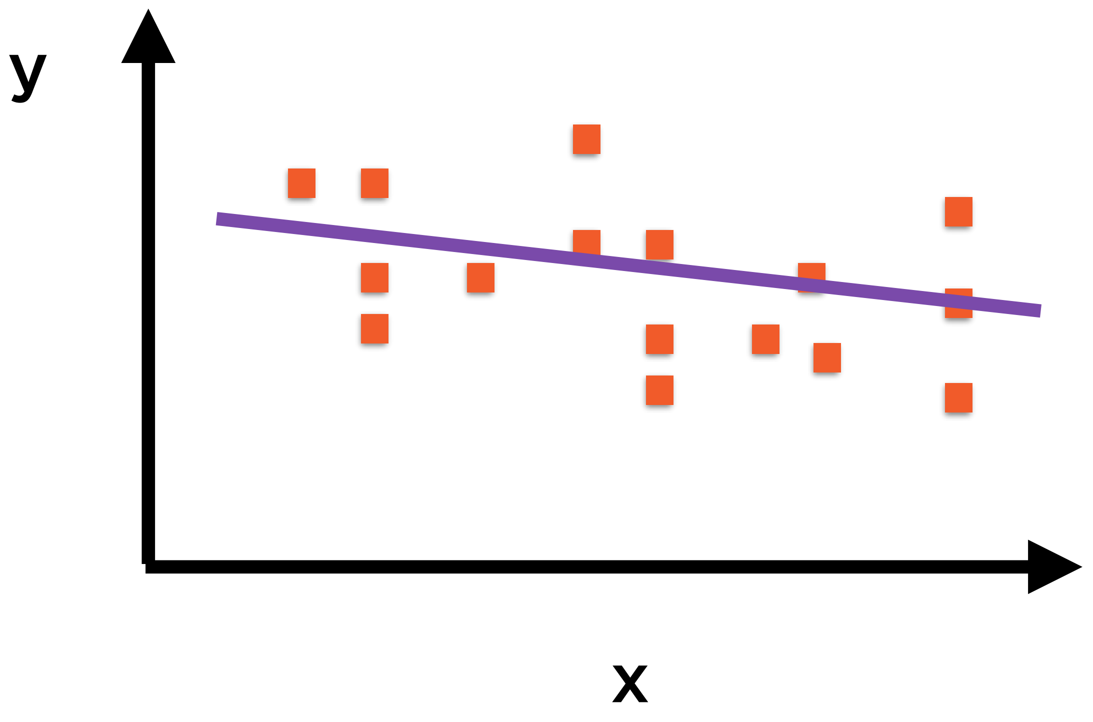
Connect the dots with an S-curve



Similar, yet Different

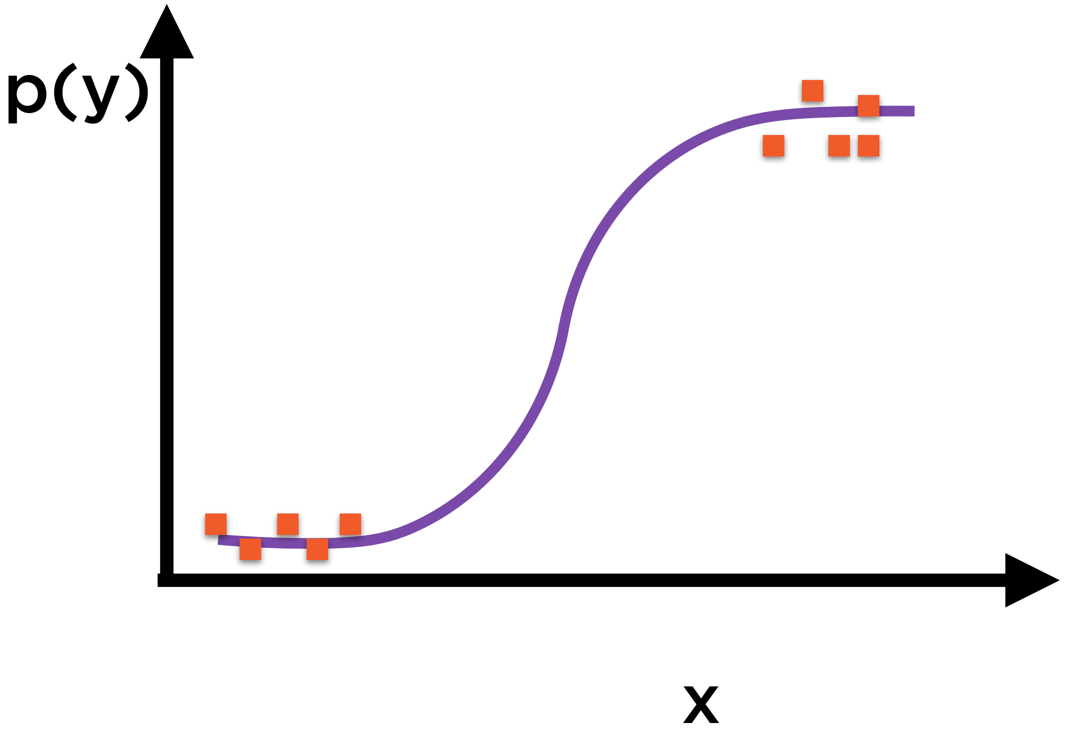
Linear Regression

$$y_i = A + Bx_i$$



Logistic Regression

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$



Similar, yet Different

Linear Regression

$$y_i = A + Bx_i$$

Objective of regression is to find A, B that “best fit” the data

Logistic Regression

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

Objective of regression is to find A, B that “best fit” the data

Similar, yet Different

Linear Regression

$$y_i = A + Bx_i$$

Relationship is already linear (by assumption)

Logistic Regression

$$\ln\left(\frac{p(y_i)}{1 - p(y_i)}\right) = A + Bx_i$$

Relationship can be made linear (by log transformation)

Similar, yet Different

Linear Regression

$$y_i = A + Bx_i$$

Solve regression problem using cookie-cutter solvers

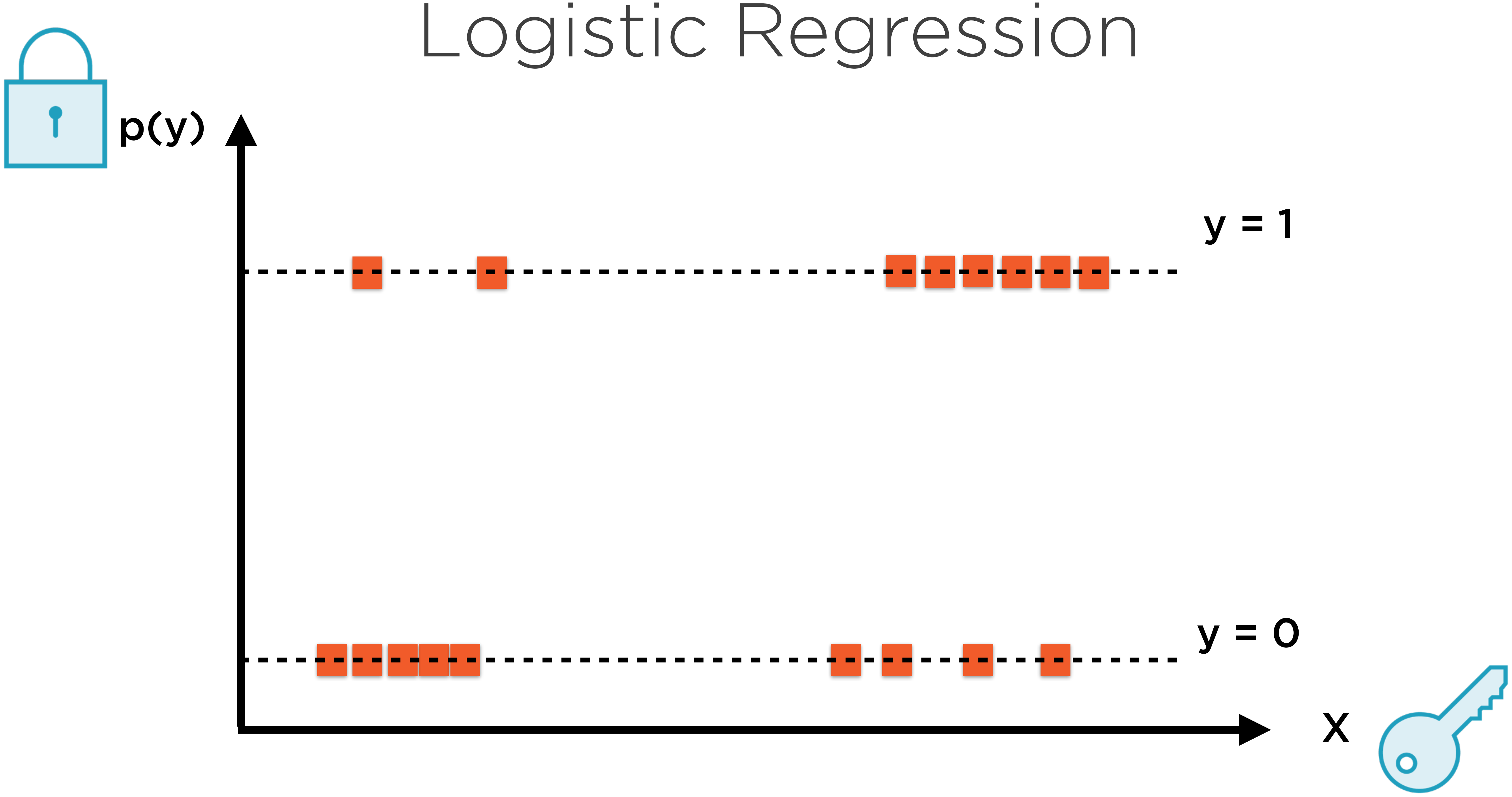
Logistic Regression

$$\text{logit}(p) = A + Bx_i$$

$$\text{logit}(p) = \ln\left(\frac{p}{1 - p}\right)$$

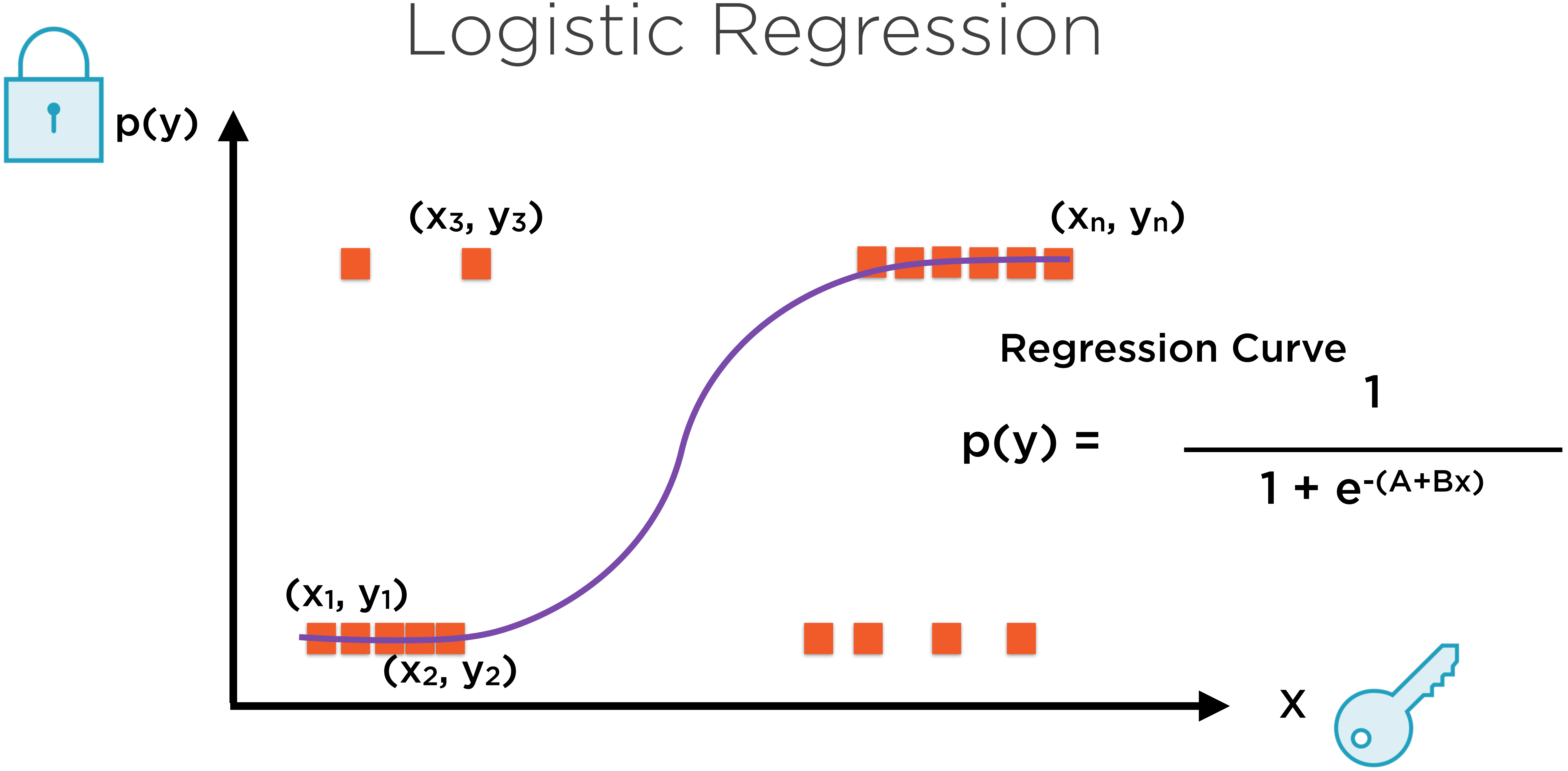
Solve regression problem using cookie-cutter solvers

Logistic Regression



Represent all n points as (x_i,y_i), where i = 1 to n

Logistic Regression



Regression Curve

$$p(y) = \frac{1}{1 + e^{-(A+Bx)}}$$

Represent all n points as (x_i,y_i), where i = 1 to n

Linear Regression

$$y = A + Bx$$

$$\begin{aligned}
y_1 &= A + Bx_1 \\
y_2 &= A + Bx_2 \\
y_3 &= A + Bx_3 \\
\ldots & \qquad \ldots \\
y_n &= A + Bx_n
\end{aligned}$$

Logistic Regression

$$p(y) = \frac{1}{1 + e^{-(A+Bx)}}$$

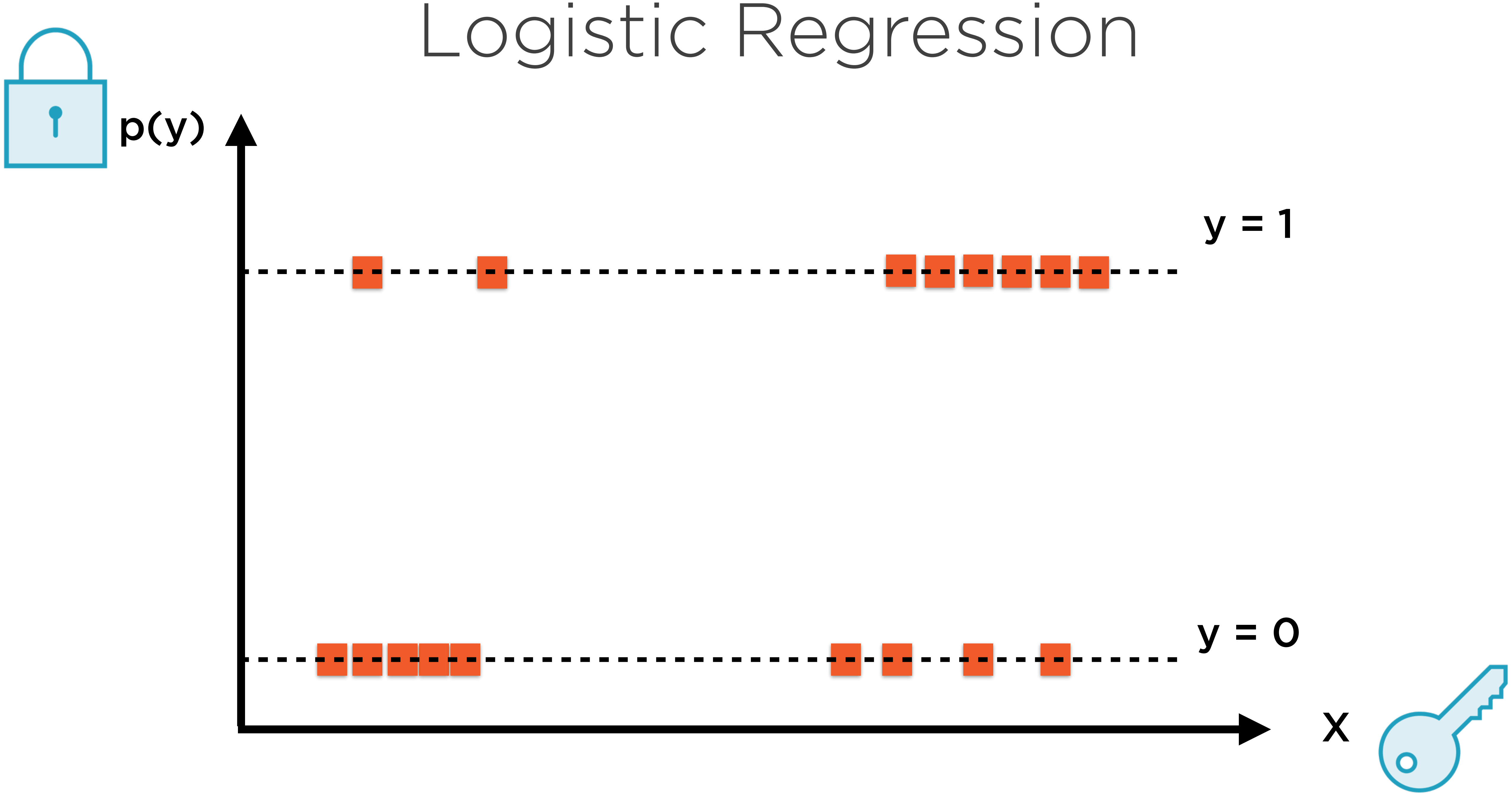
$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

$$p(y_1) = \frac{1}{1 + e^{-(A+Bx_1)}}$$

...

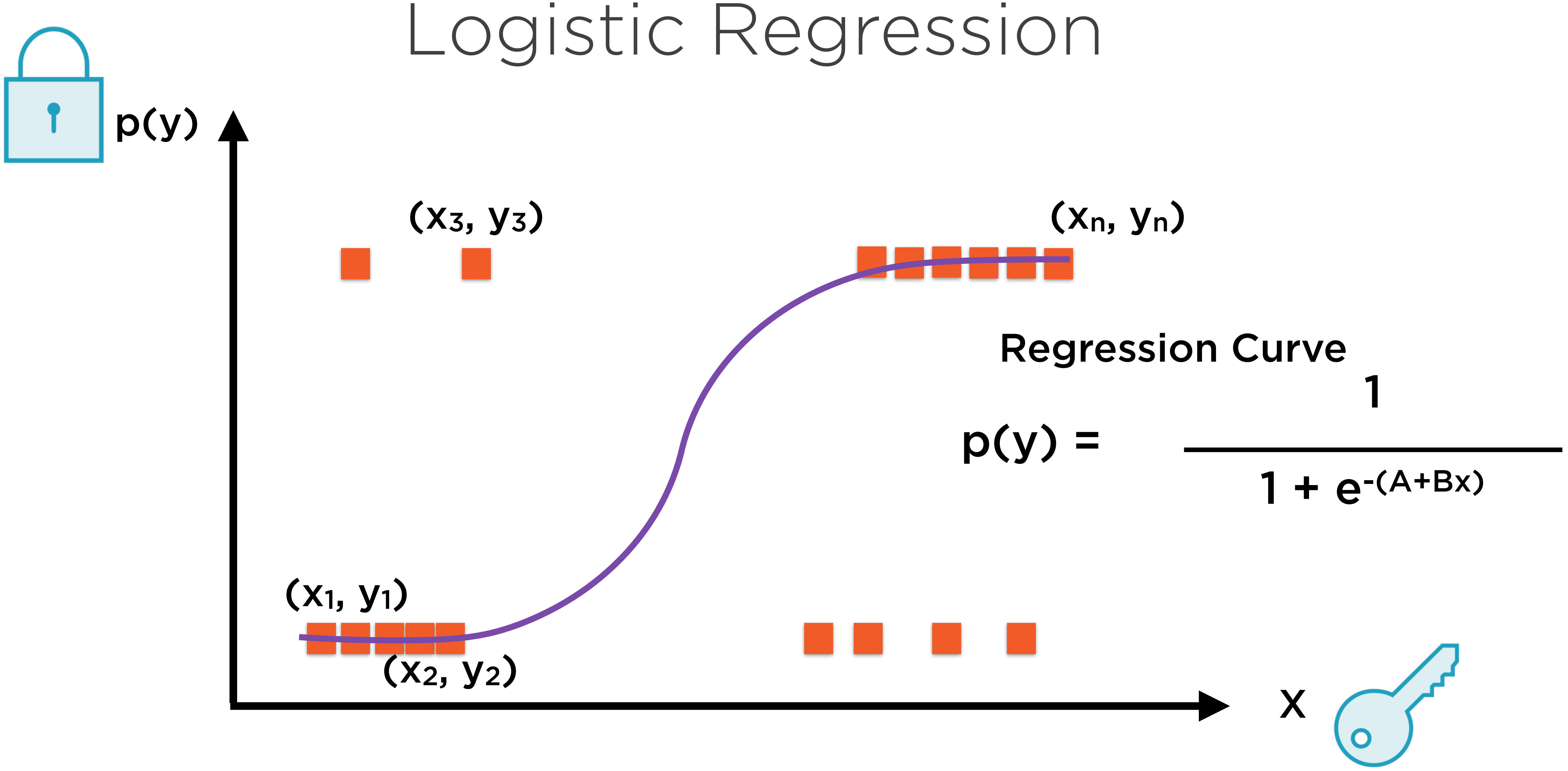
$$p(y_n) = \frac{1}{1 + e^{-(A+Bx_n)}}$$

Logistic Regression



Represent all n points as (x_i, y_i), where i = 1 to n

Logistic Regression



$$p(y) = \frac{1}{1 + e^{-(A+Bx)}}$$

Represent all n points as
(x_i,y_i), where i = 1 to n

Logistic Regression

Regression Equation:

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

Solve for A and B that “best fit” the data

Odds from Probabilities

$$\mathrm{Odds}(p) = \frac{p}{1-p}$$

Odds of an Event

$$p = \frac{1}{1 + e^{-(A+Bx)}}$$

$$p = \frac{e^{A + Bx}}{1 + e^{A + Bx}}$$

$$1 - p = 1 - \frac{e^{A + Bx}}{1 + e^{A + Bx}}$$

Odds of an Event

$$1 - p = 1 - \frac{e^{A + Bx}}{1 + e^{A + Bx}}$$

$$1 - p = \frac{1 + e^{A + Bx} - e^{A + Bx}}{1 + e^{A + Bx}}$$

$$1 - p = \frac{1}{1 + e^{A + Bx}}$$

Odds of an Event

$$p = \frac{e^{A + Bx}}{1 + e^{A + Bx}}$$

$$1 - p = \frac{1}{1 + e^{A + Bx}}$$

$$\mathrm{Odds}(p) = \frac{p}{1 - p} = e^{A + Bx}$$

Logit Is Linear

$$\text{Odds}(p) = \frac{p}{1 - p} = e^{A + Bx}$$

$$\text{logit}(p) = A + Bx$$

In(Odds(p)) is called the logit function

Logit Is Linear

$$\ln \text{Odds}(p) = \ln(p) - \ln(1-p)$$

$$p = \frac{1}{1 + e^{-(A+Bx)}}$$

$$\text{logit}(p) = \ln \text{Odds}(p) = A + Bx$$

This is a linear function!

Logistic Regression can be solved via **linear regression on the logit function** (log of the odds function)

Logistic Regression in TensorFlow

Logistic Regression

Cause

Changes in S&P 500

Effect

Changes in price of Google Stock

Logistic Regression

y = Returns on Google stock

(GOOG)

x = Returns on S&P 500

(S&P500)

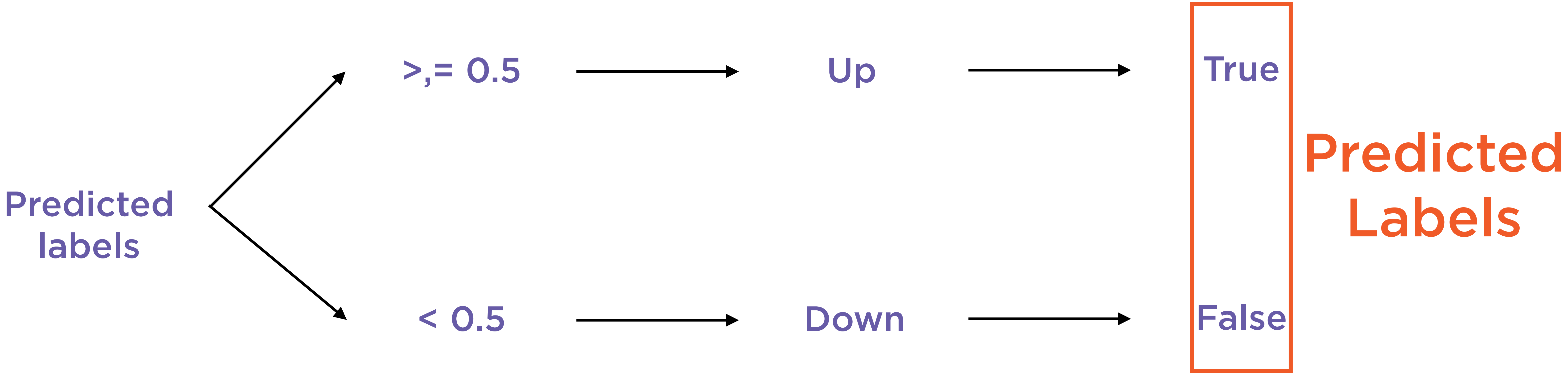
Logistic Regression

$$p(y_i) = \frac{1}{1 + e^{-(A+Bx_i)}}$$

P(y) = Probability of Google going up in the current month i

x = Returns on S&P 500 for current month

Logistic Regression



Predicted labels

>,= 0.5

< 0.5

Up

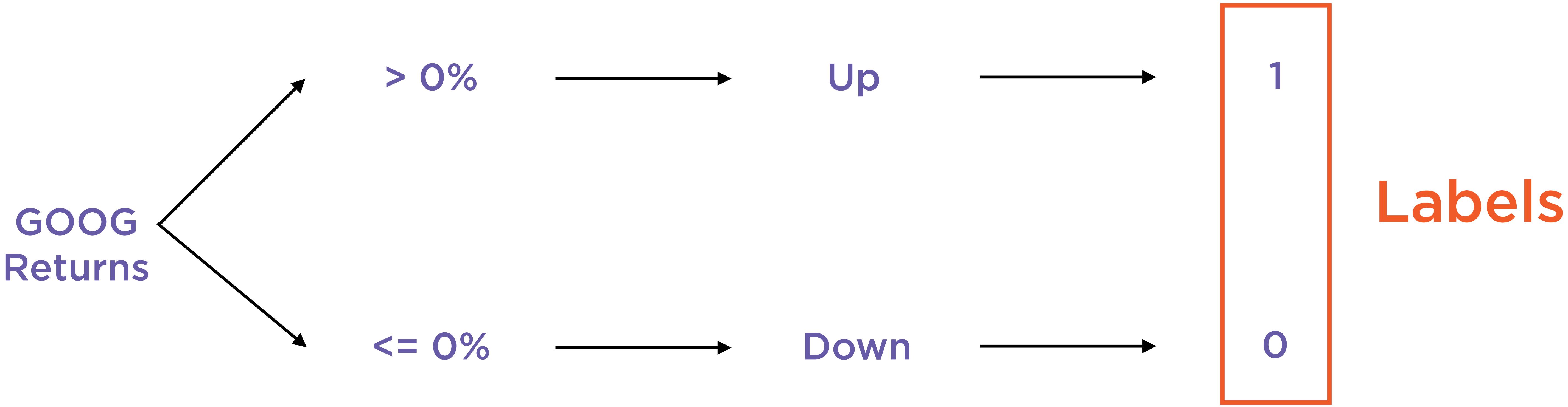
Down

True

False

Predicted Labels

Set up the Problem



GOOG Returns

> 0%

Up

1

<= 0%

Down

0

Labels

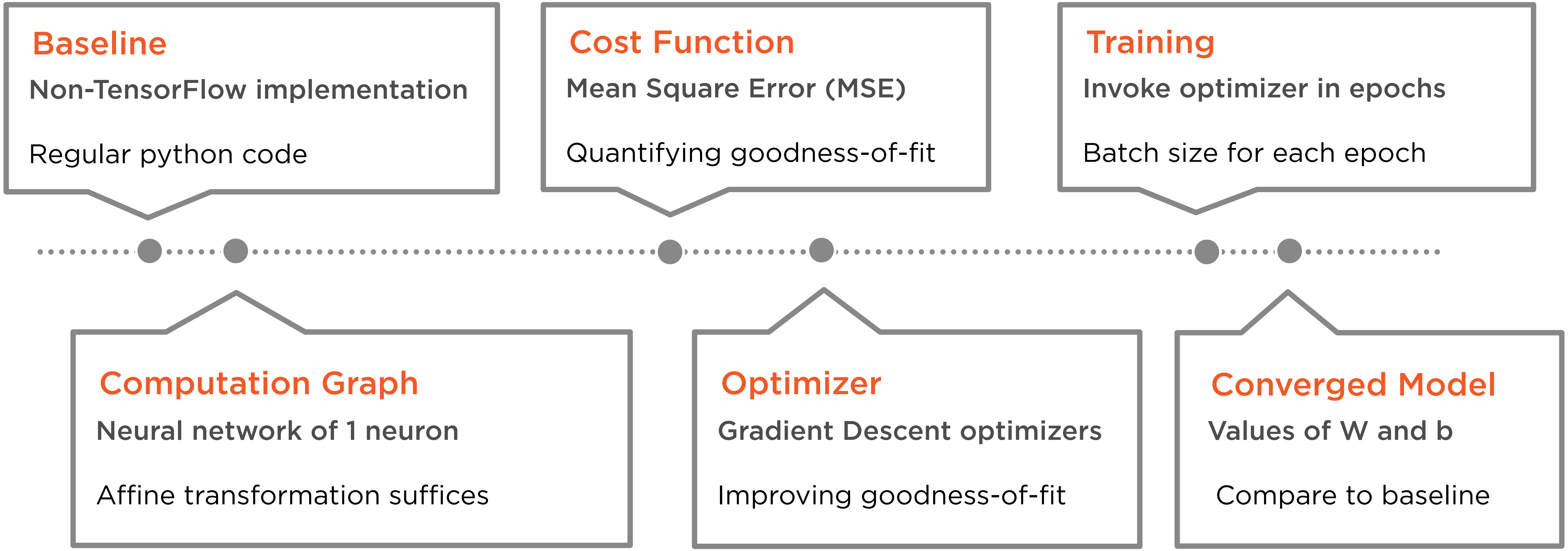
Label GOOG returns as binary (1,0)

Prediction Accuracy

DATE	ACTUAL	PREDICTED
2005-01-01	NA	NA
2005-02-01	0	1
2005-03-01	0	0
2017-01-01	1	1
2017-02-01	1	1

Compare GOOG’s actual labels vs. predicted labels

Linear Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Computation Graph

Neural network of 1 neuron

Affine transformation suffices

Cost Function

Mean Square Error (MSE)

Quantifying goodness-of-fit

Optimizer

Gradient Descent optimizers

Improving goodness-of-fit

Training

Invoke optimizer in epochs

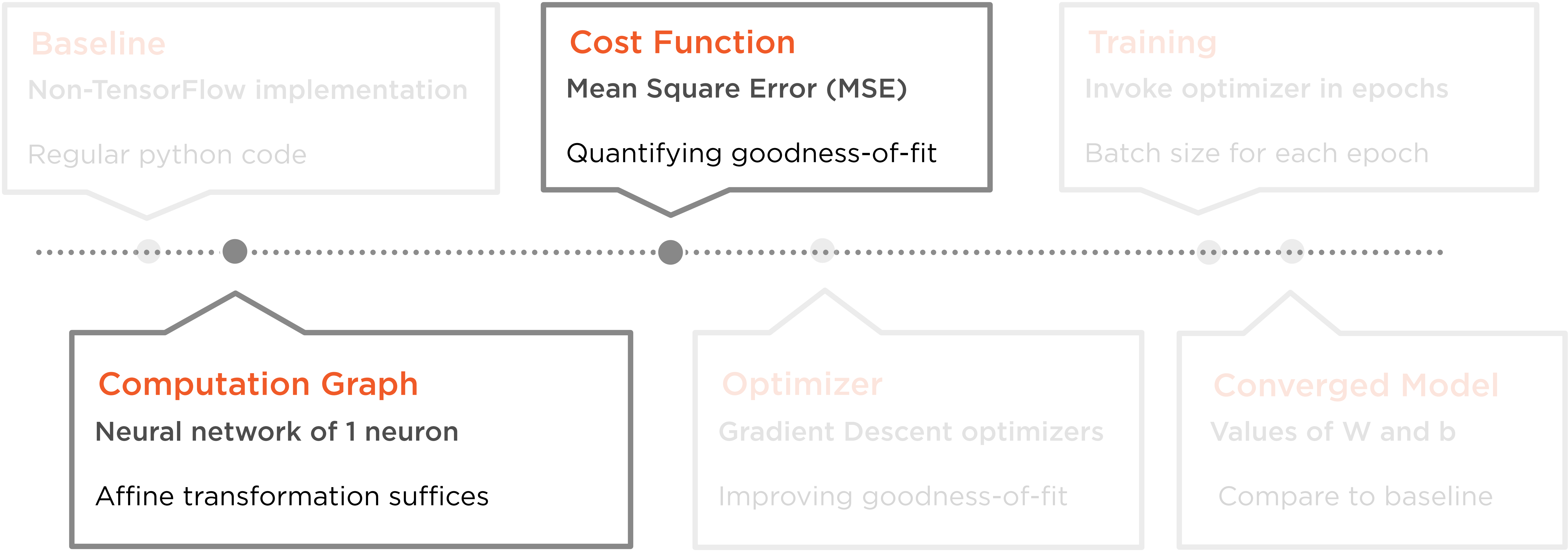
Batch size for each epoch

Converged Model

Values of W and b

Compare to baseline

Linear Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Computation Graph

Neural network of 1 neuron

Affine transformation suffices

Cost Function

Mean Square Error (MSE)

Quantifying goodness-of-fit

Optimizer

Gradient Descent optimizers

Improving goodness-of-fit

Training

Invoke optimizer in epochs

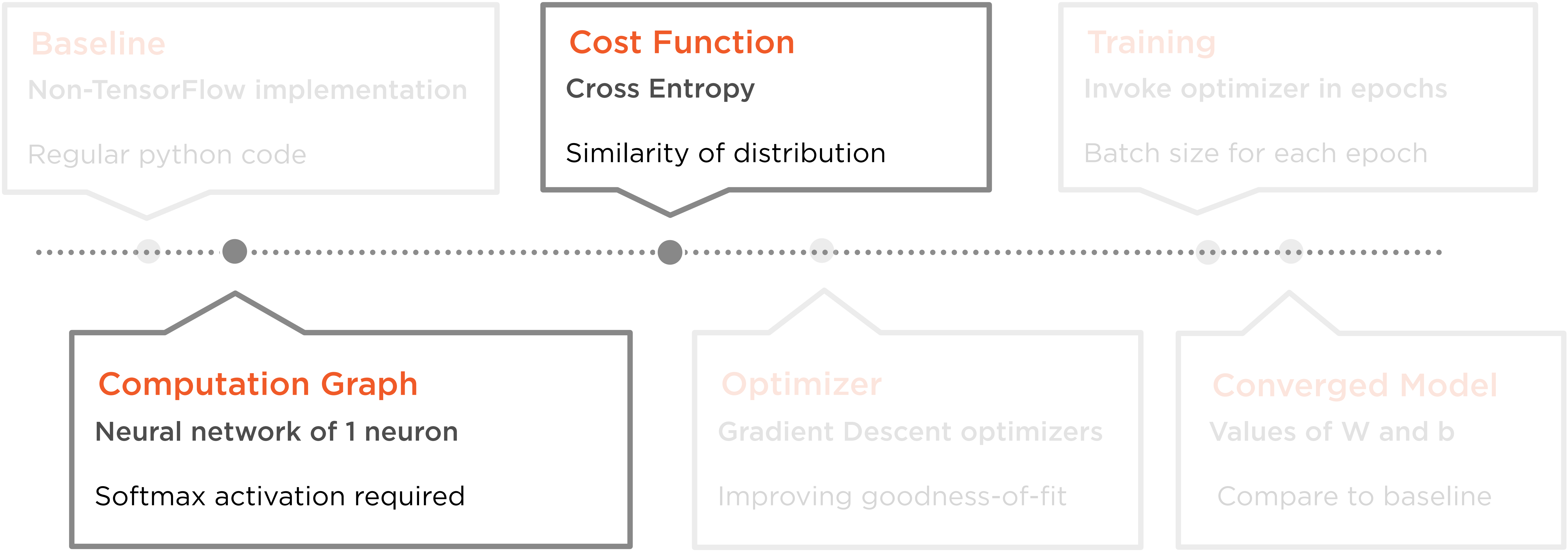
Batch size for each epoch

Converged Model

Values of W and b

Compare to baseline

Logistic Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Computation Graph

Neural network of 1 neuron

Softmax activation required

Cost Function

Cross Entropy

Similarity of distribution

Optimizer

Gradient Descent optimizers

Improving goodness-of-fit

Training

Invoke optimizer in epochs

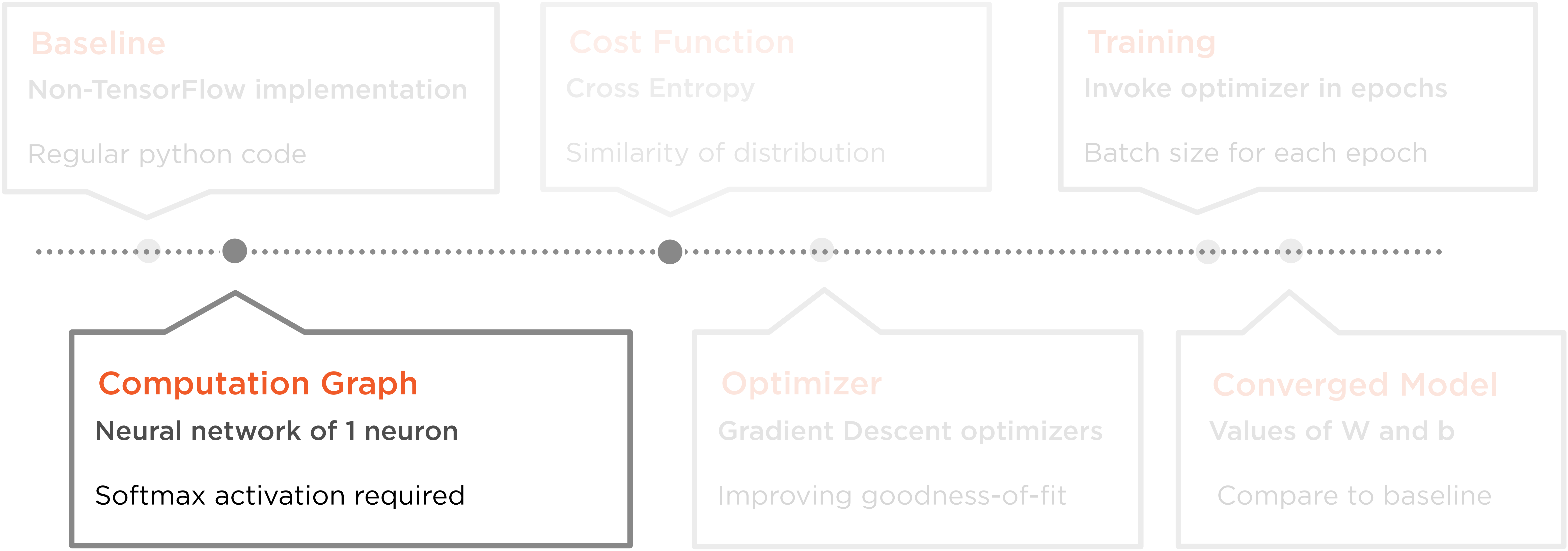
Batch size for each epoch

Converged Model

Values of W and b

Compare to baseline

Logistic Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Cost Function

Cross Entropy

Similarity of distribution

Training

Invoke optimizer in epochs

Batch size for each epoch

Computation Graph

Neural network of 1 neuron

Softmax activation required

Optimizer

Gradient Descent optimizers

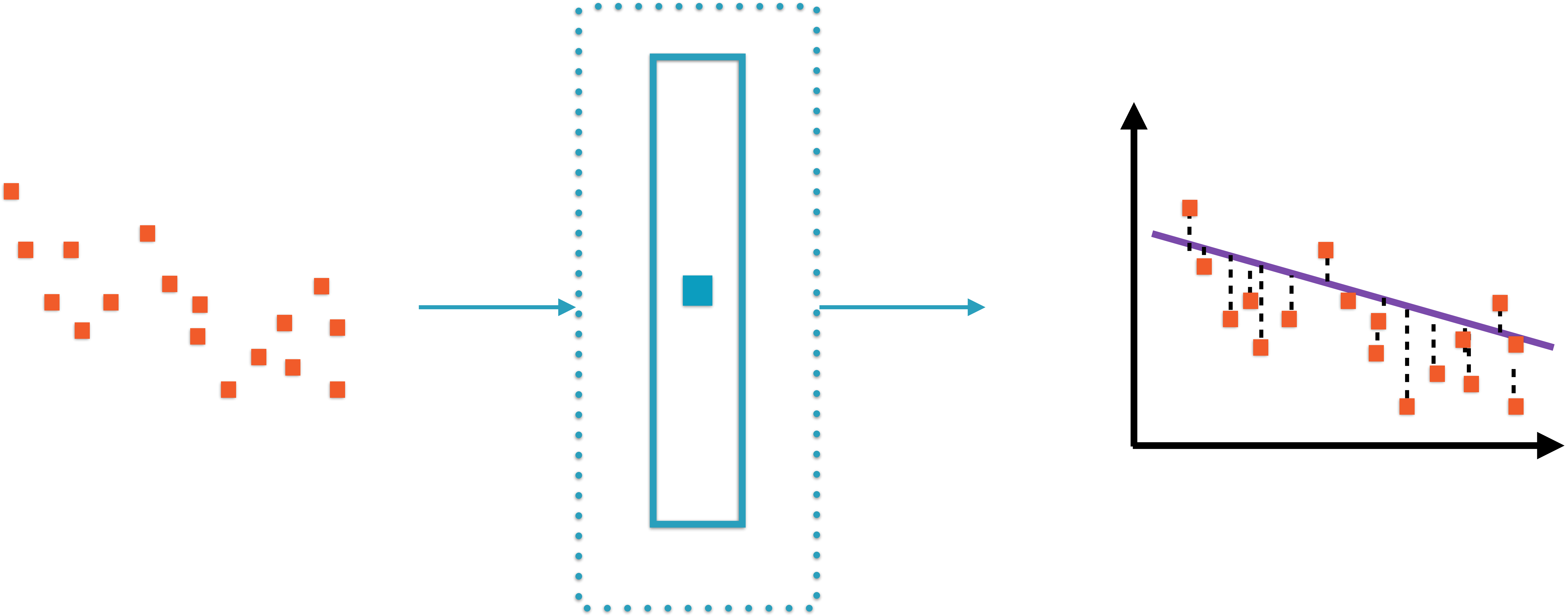
Improving goodness-of-fit

Converged Model

Values of W and b

Compare to baseline

Linear Regression with One Neuron

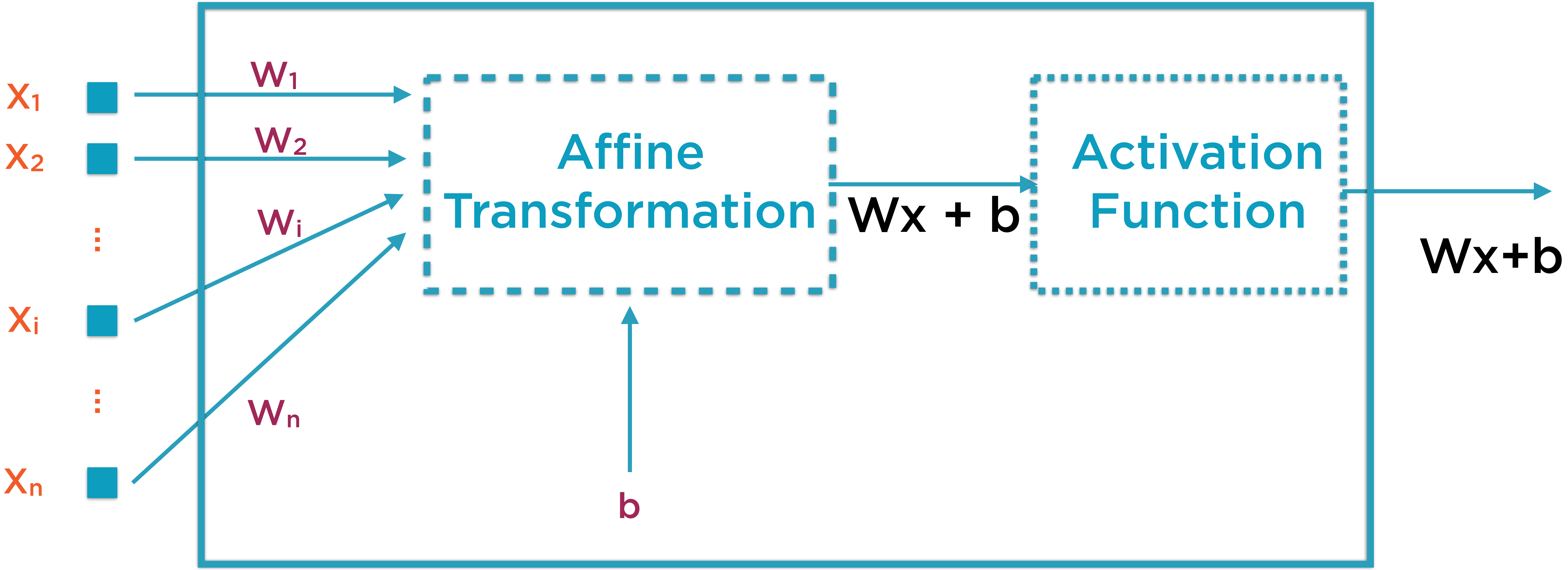


Set of Points

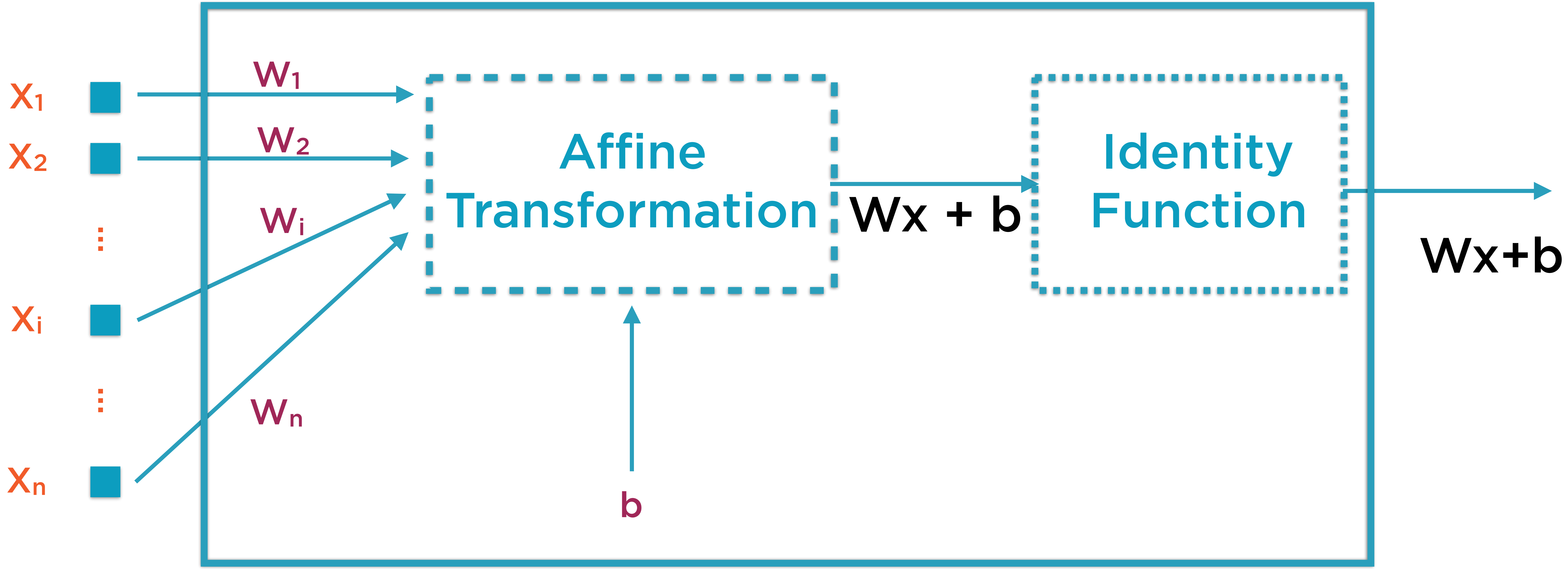
Single Neuron

Regression Line

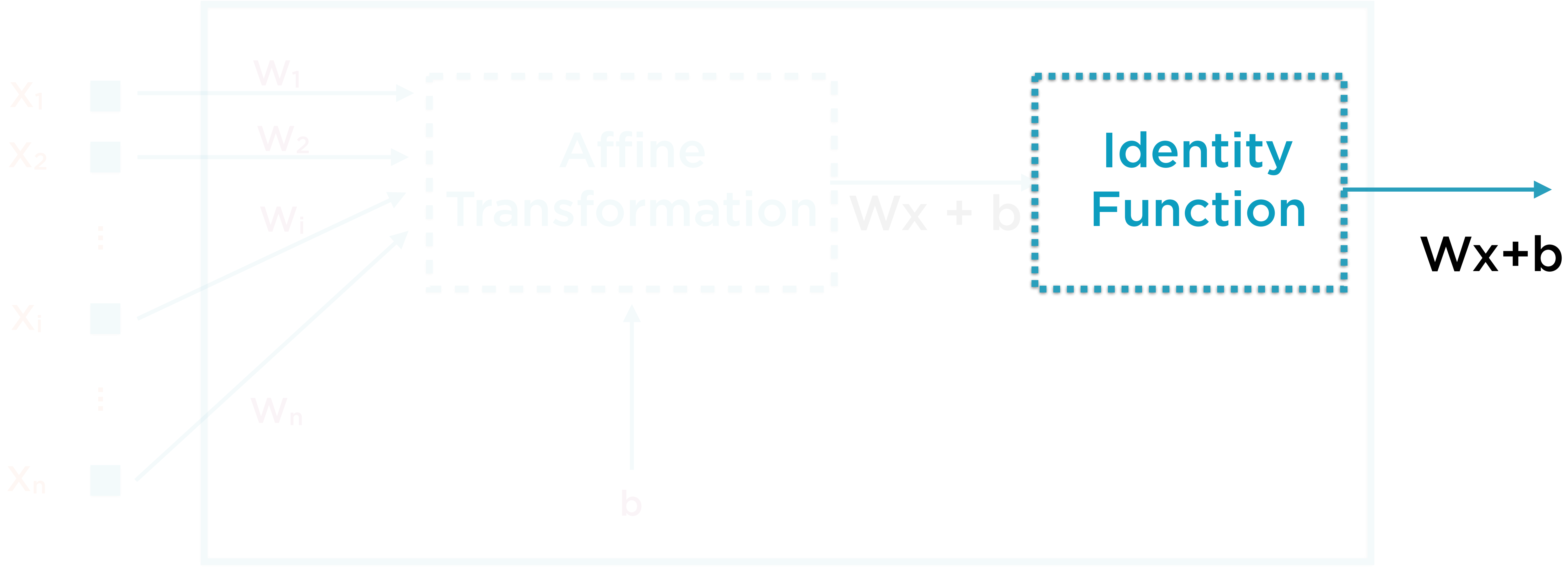
Linear Regression with One Neuron



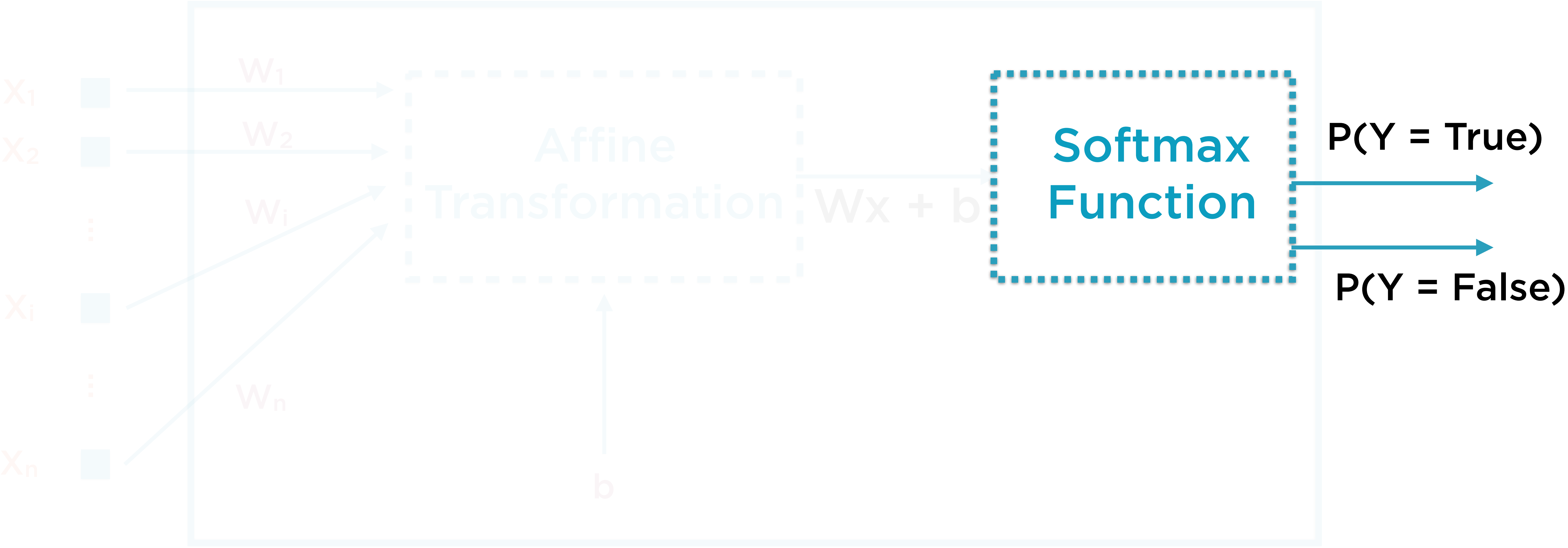
Linear Regression with One Neuron



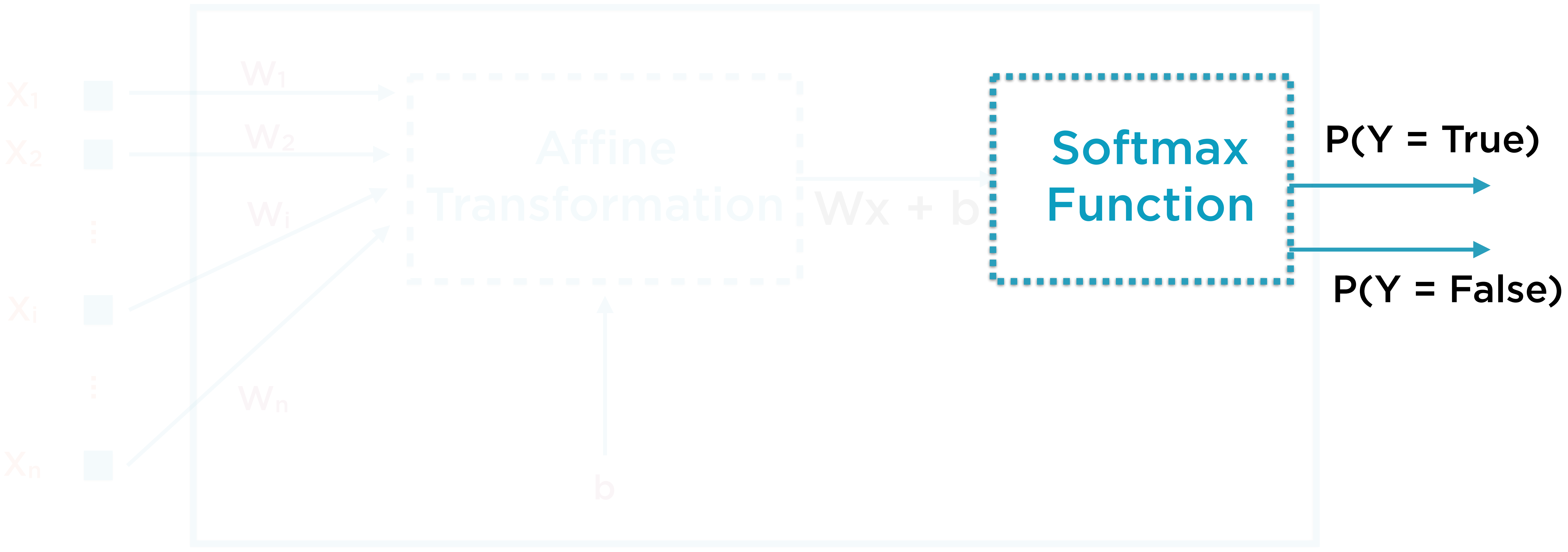
Linear Regression with One Neuron



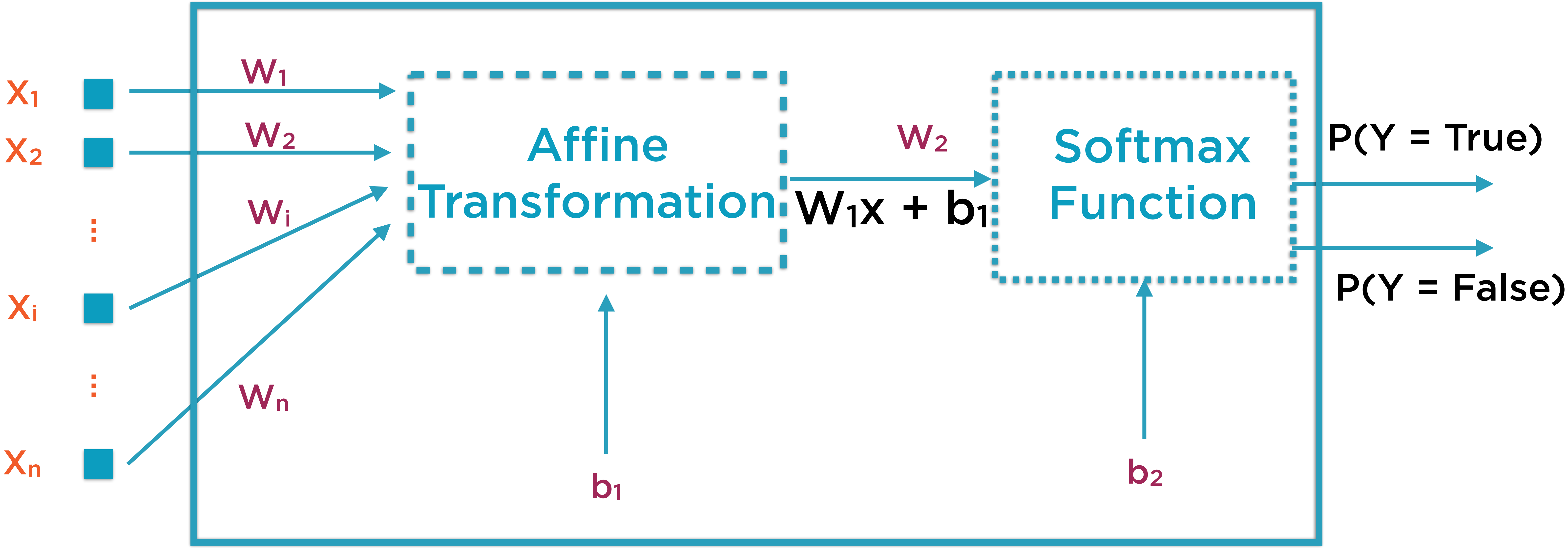
Logistic Regression with One Neuron



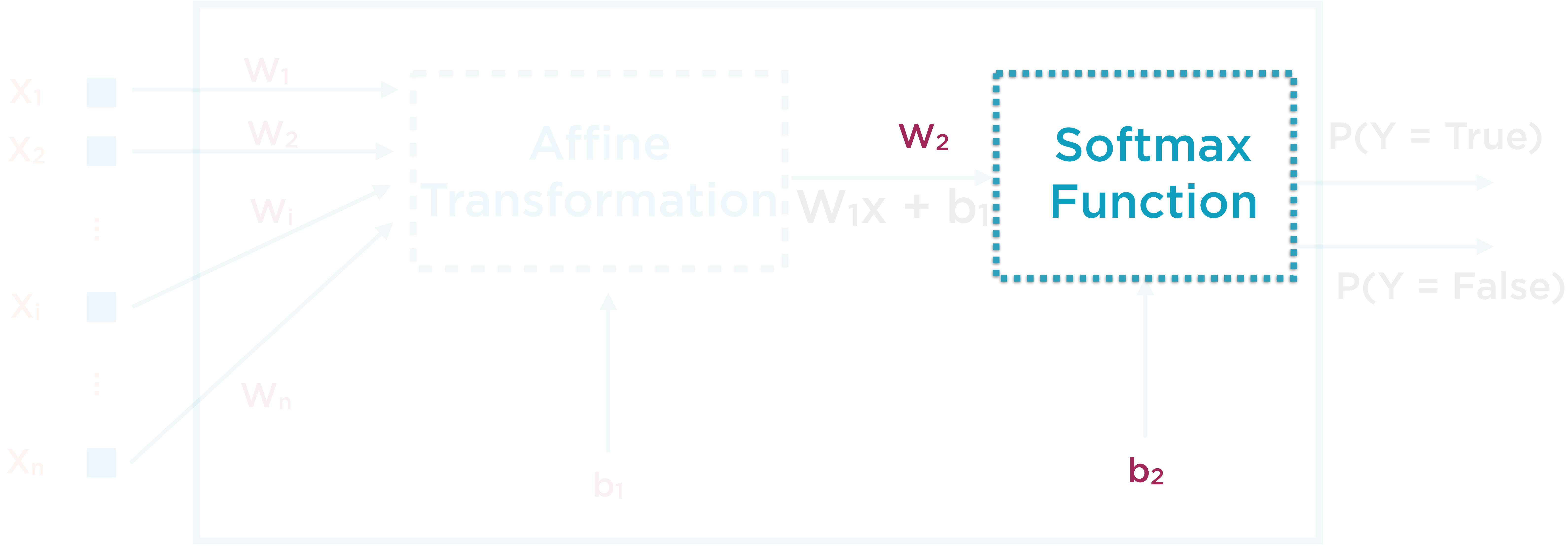
Logistic Regression with One Neuron



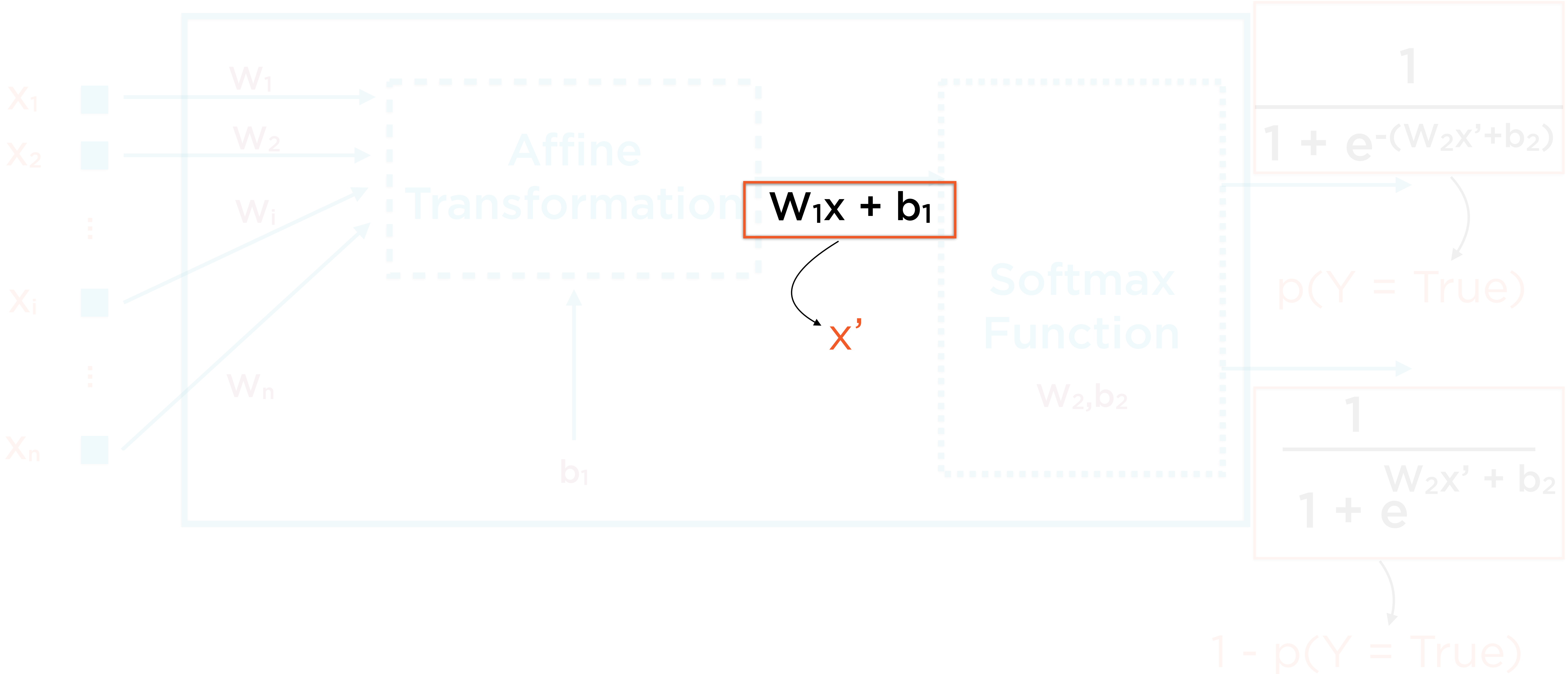
Logistic Regression with One Neuron



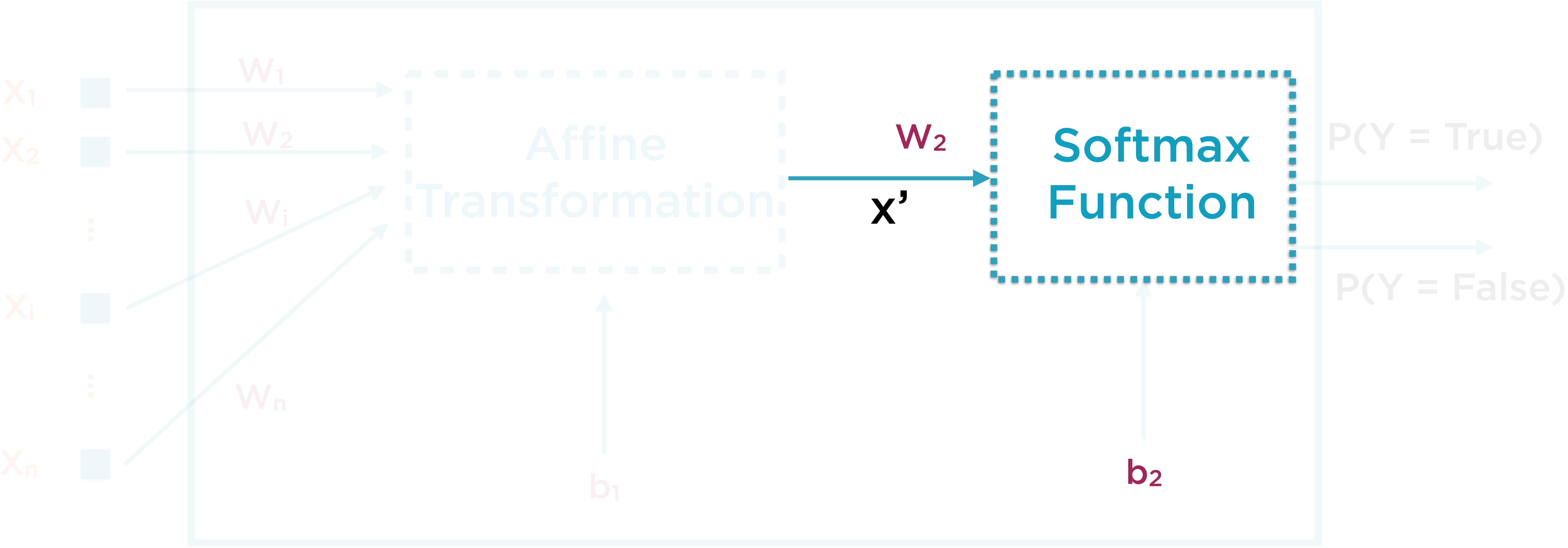
Logistic Regression with One Neuron



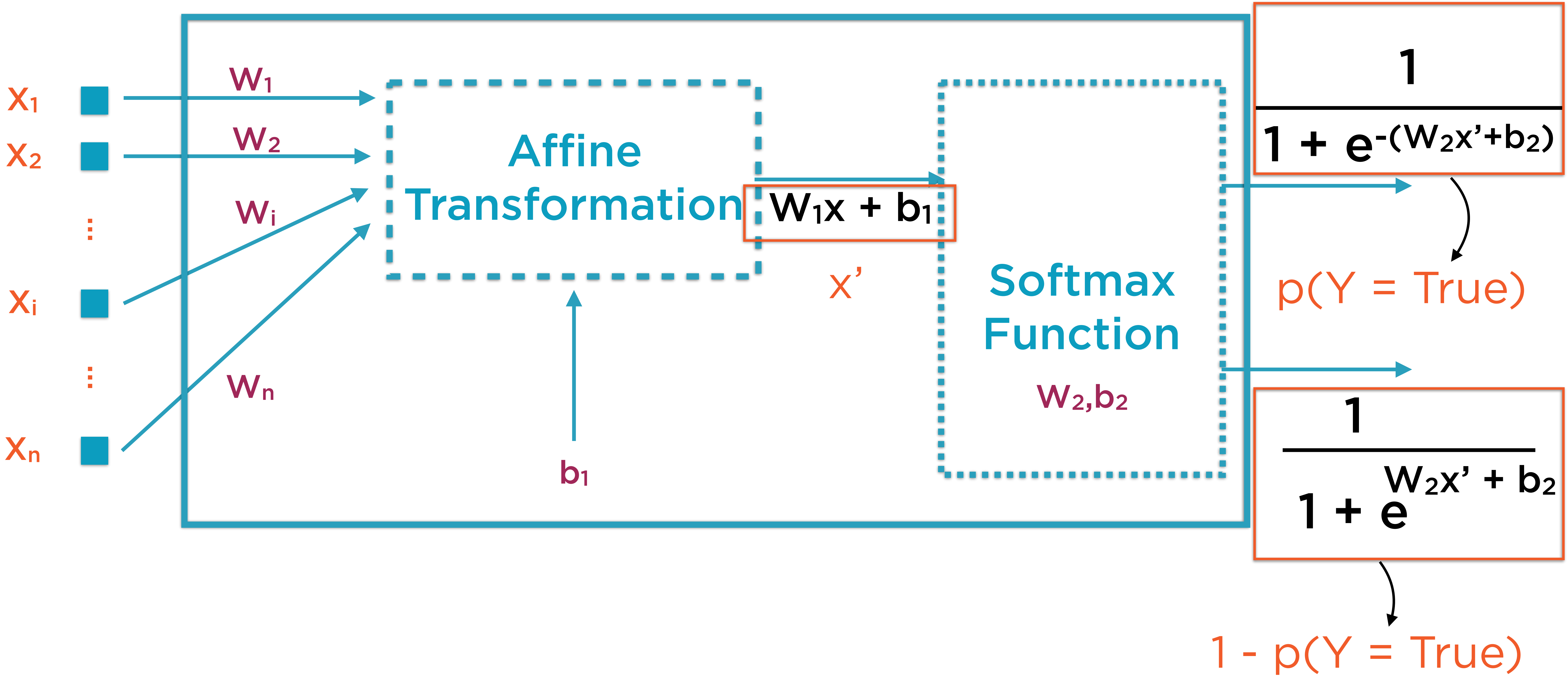
Logistic Regression with One Neuron



Logistic Regression with One Neuron



Logistic Regression with One Neuron



x_1 W_1

x_2 W_2

x_i W_i

x_n W_n

Affine Transformation

b_1

$W_1x + b_1$

x'

Softmax Function

W_2, b_2

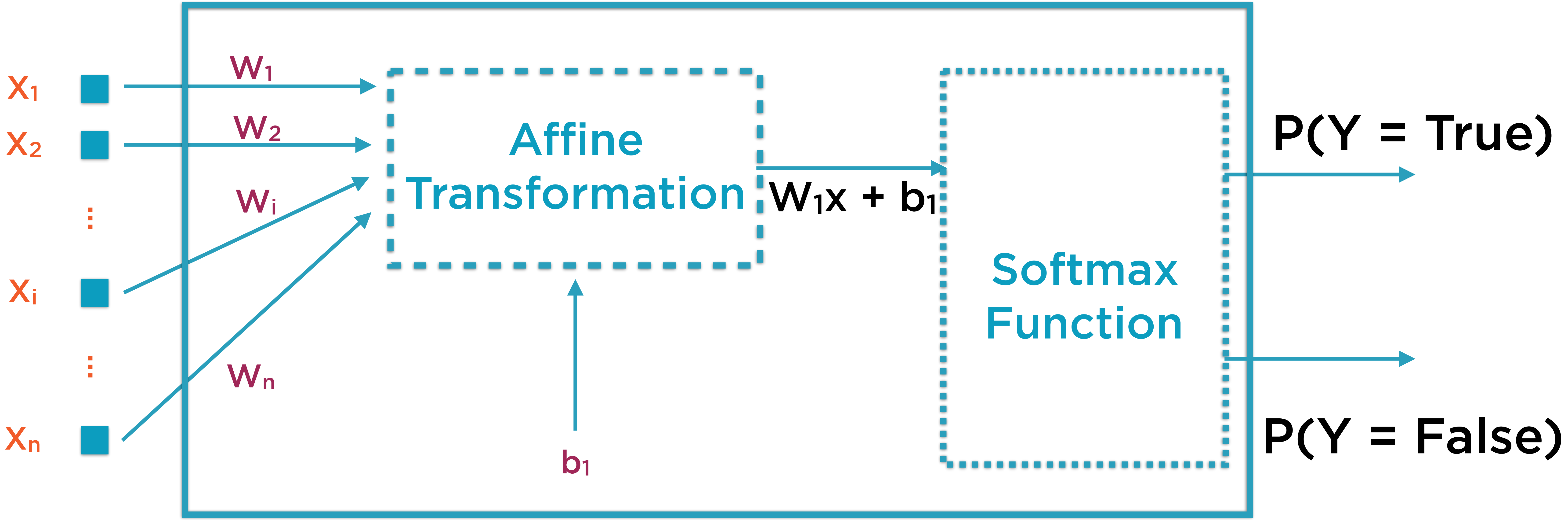
$$\frac{1}{1 + e^{-(W_2x'+b_2)}}$$

$p(Y = True)$

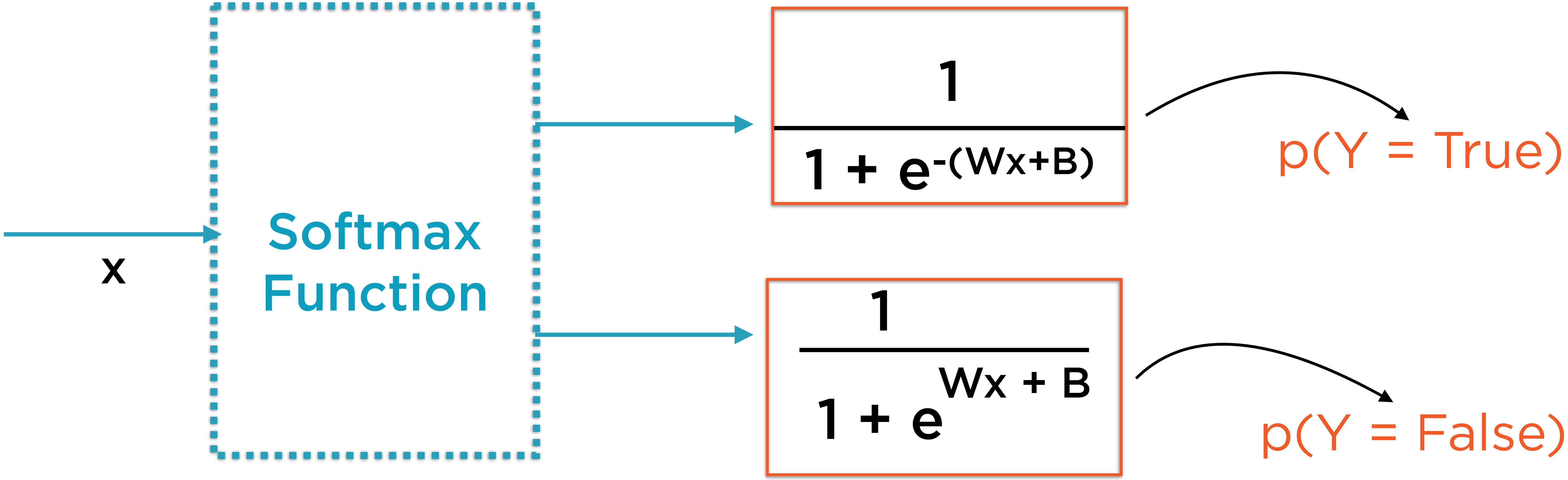
$$\frac{1}{1 + e^{W_2x' + b_2}}$$

$1 - p(Y = True)$

Logistic Regression with One Neuron



SoftMax for True/False Classification



x

Softmax Function

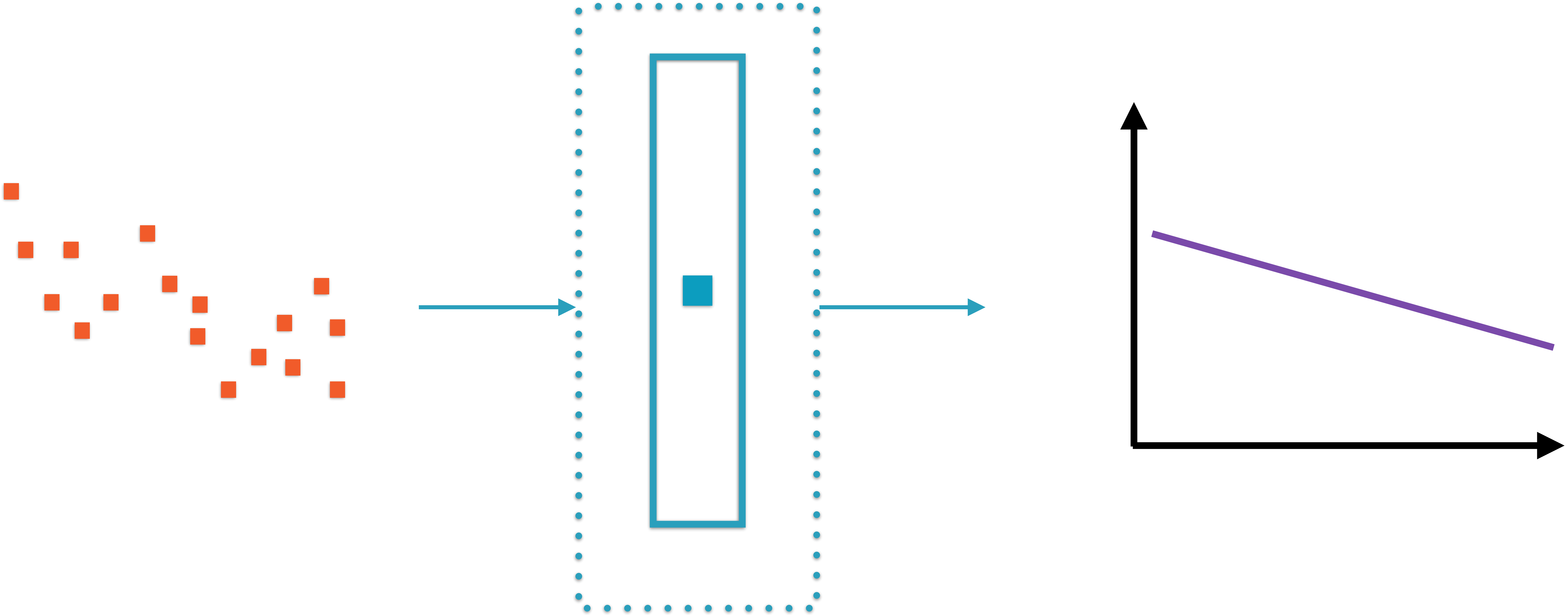
$$\frac{1}{1 + e^{-(Wx+B)}}$$

p(Y = True)

$$\frac{1}{1 + e^{Wx + B}}$$

p(Y = False)

Linear Regression with One Neuron



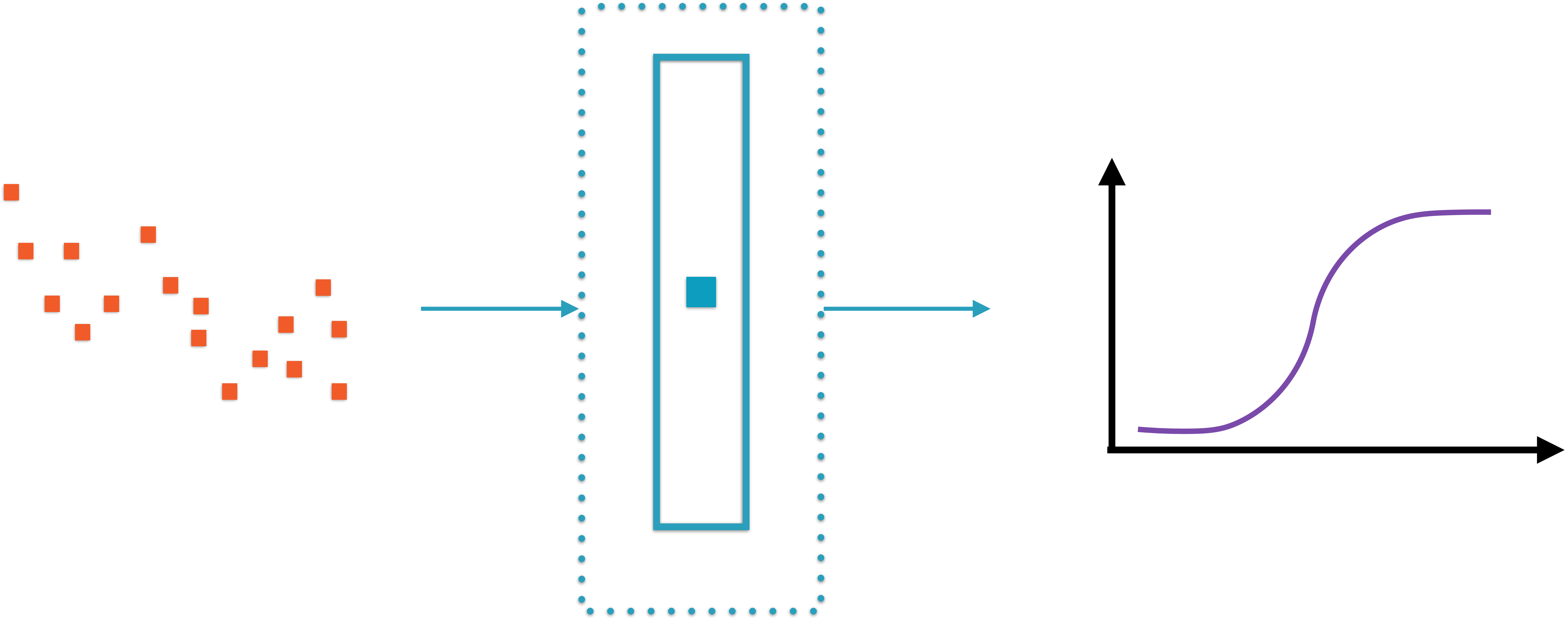
1-dimensional feature vector

Shape (W) = [1,1]

Shape (b) = [1]

Regression Line

Logistic Regression with One Neuron



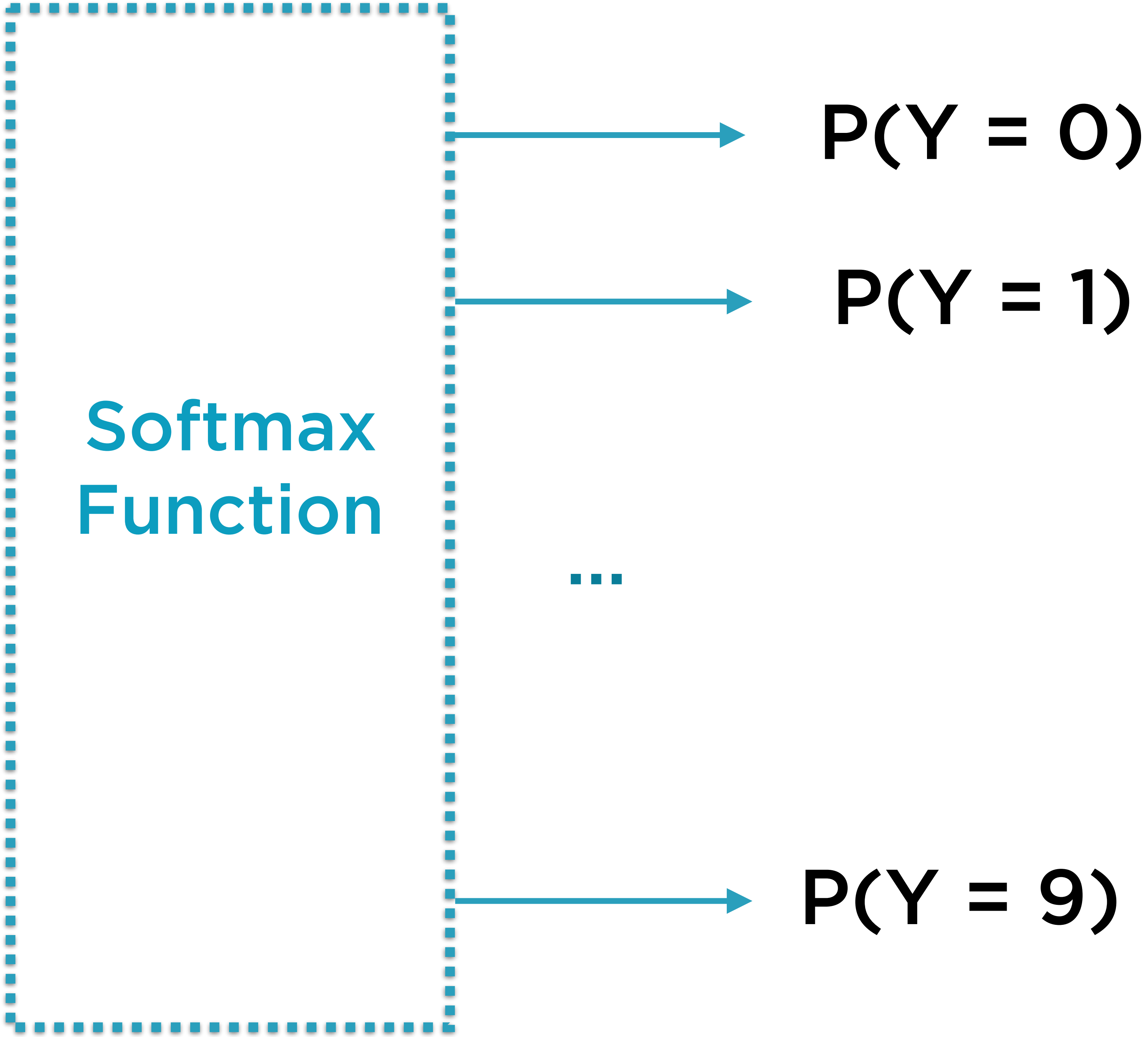
1-dimensional
feature vector

Shape (W) = [1,2]

Shape (b) = [2]

S-Curve

SoftMax for Digit Classification



Softmax Function

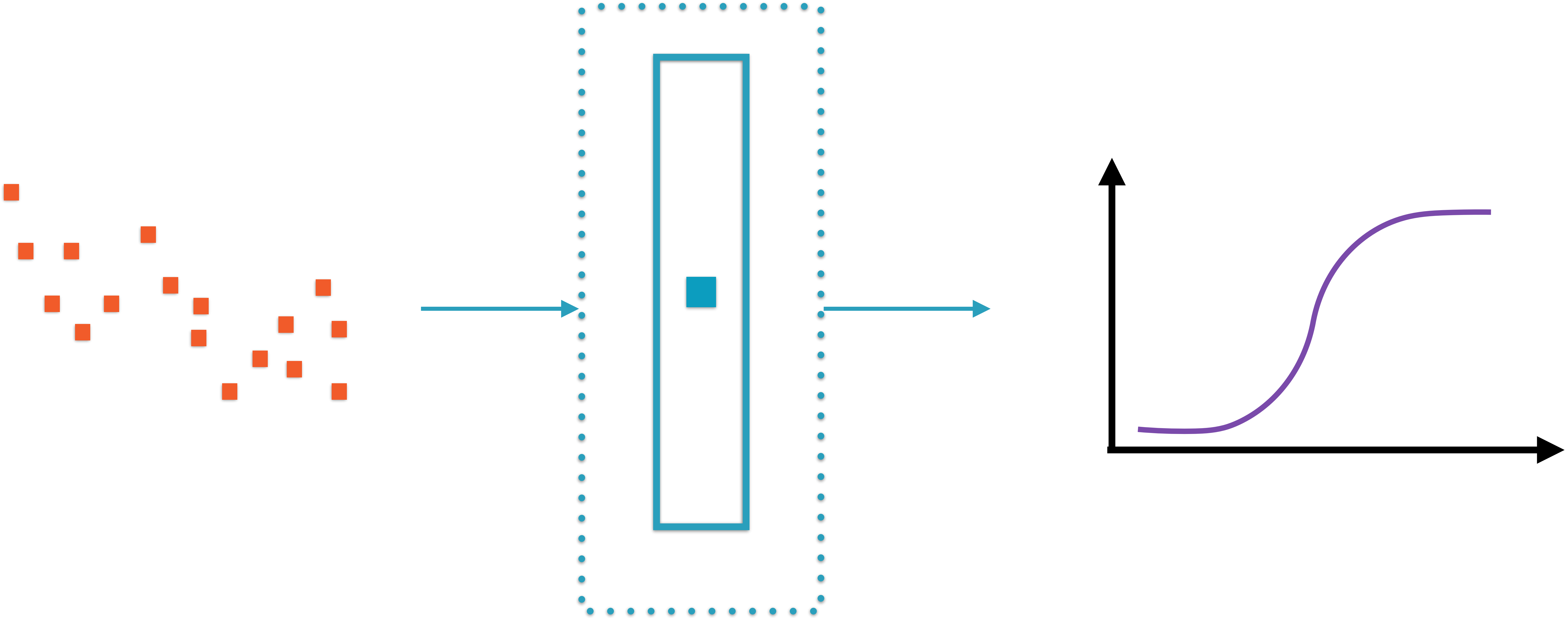
P(Y = 0)

P(Y = 1)

...

P(Y = 9)

SoftMax for Digit Classification



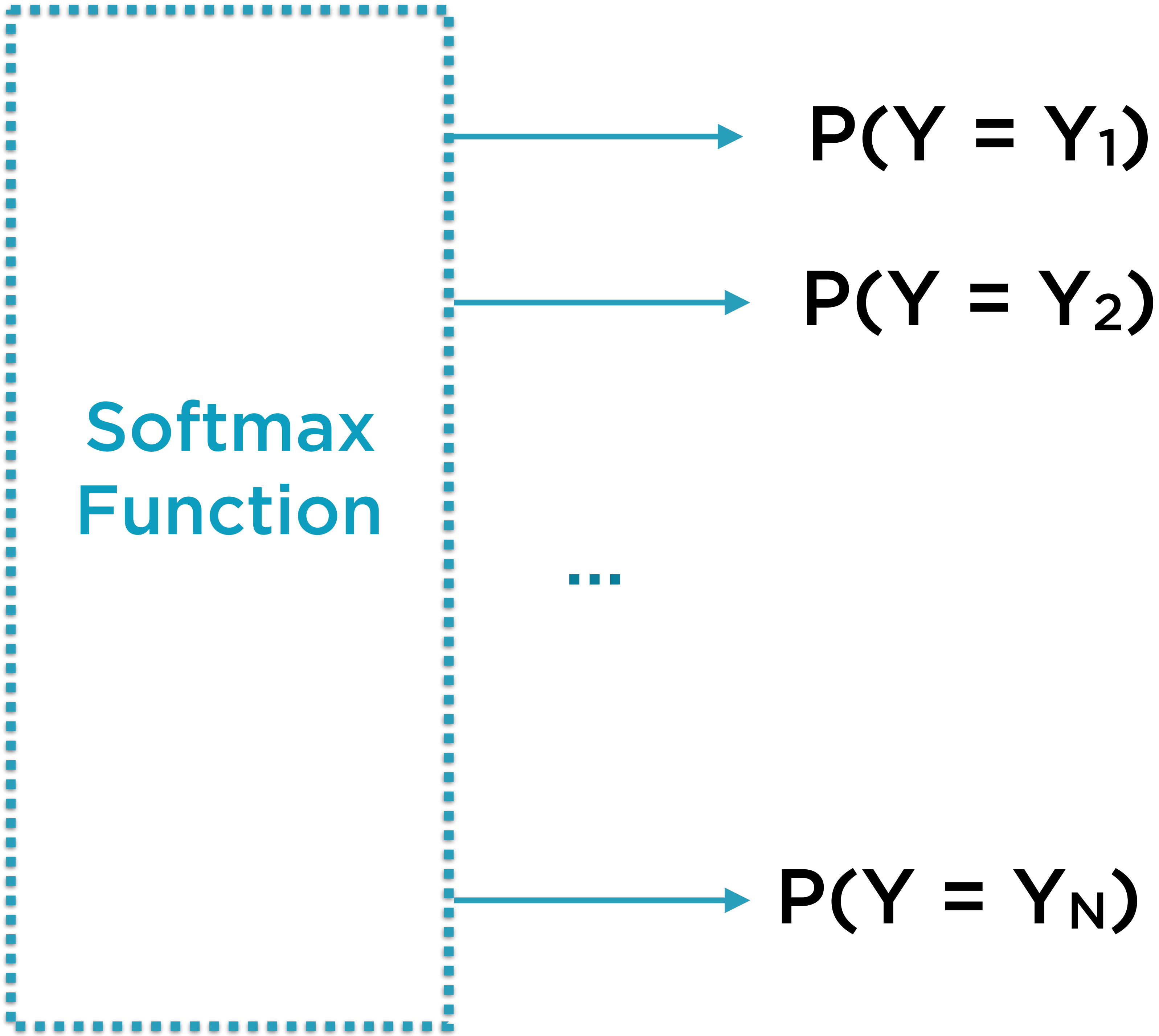
1-dimensional
feature vector

Shape (W) = [1,10]

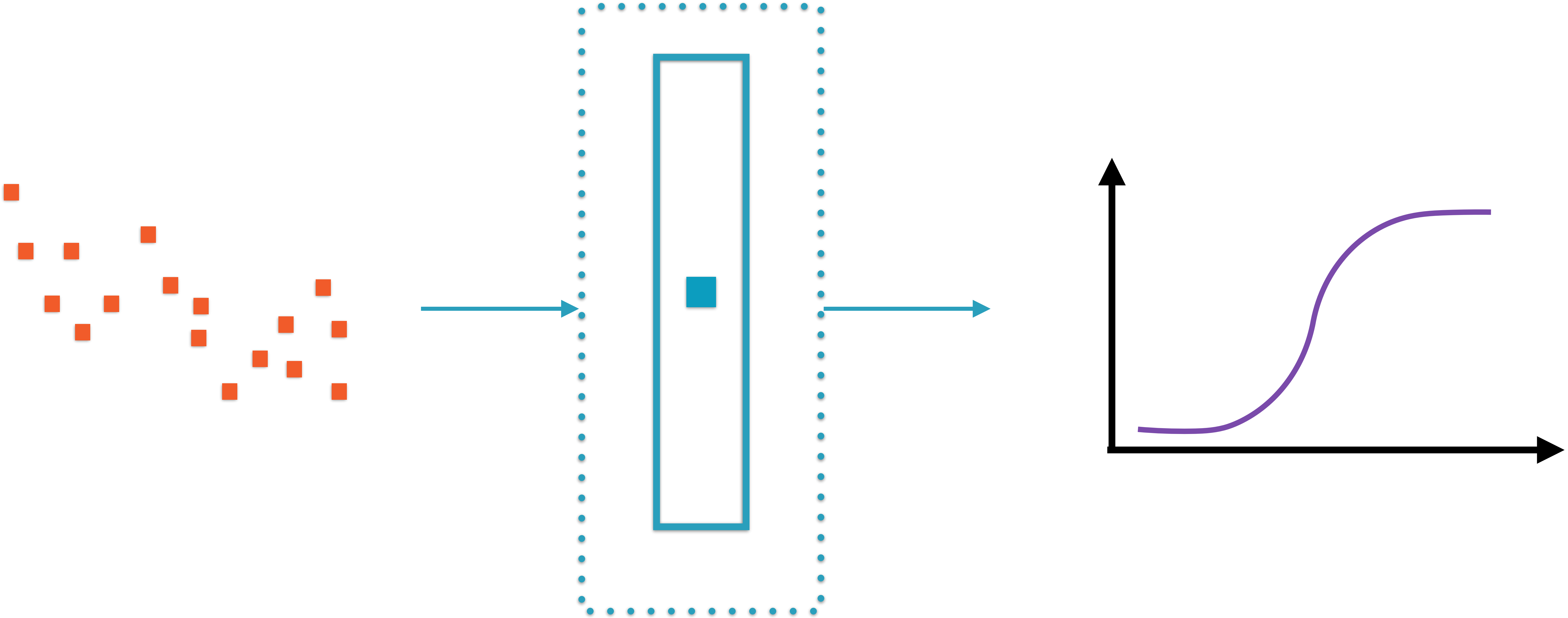
Shape (b) = [10]

S-Curve

SoftMax N-category Classification



SoftMax N-category Classification



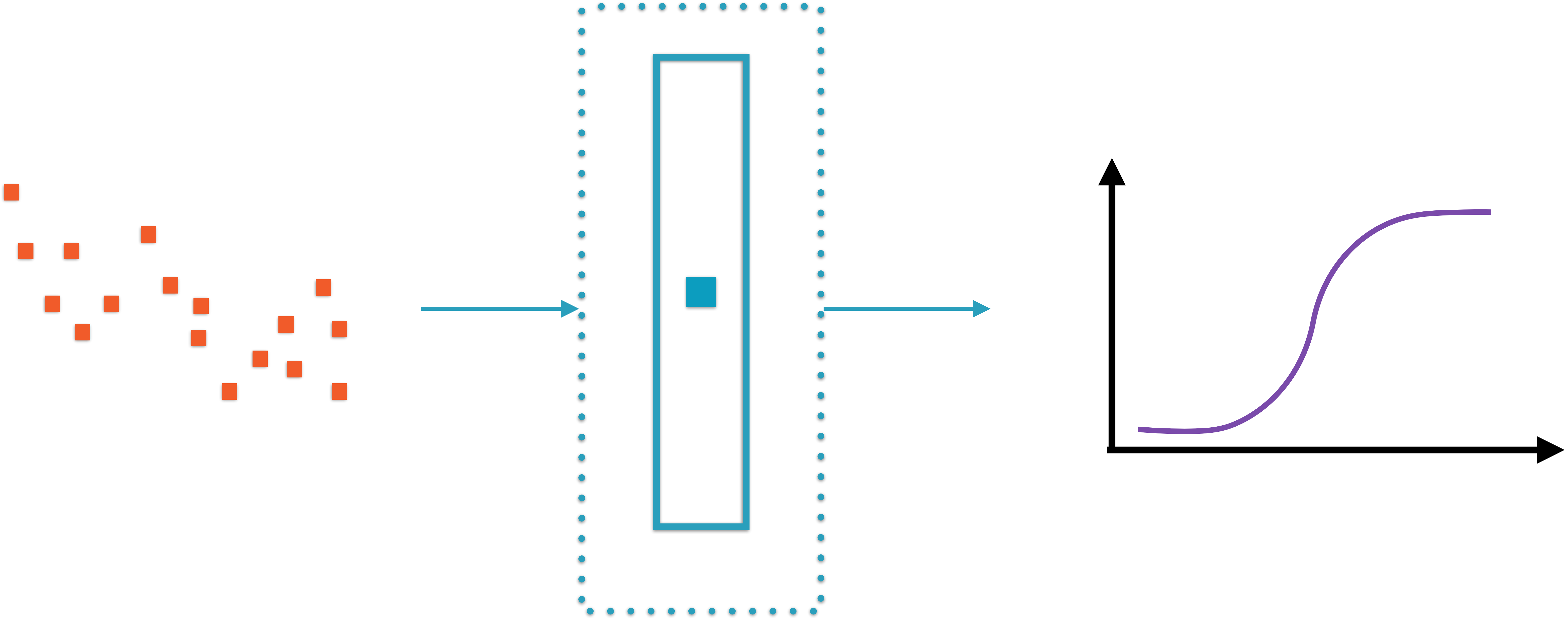
1-dimensional
feature vector

Shape (W) = [1,N]

Shape (b) = [N]

S-Curve

SoftMax N-category Classification



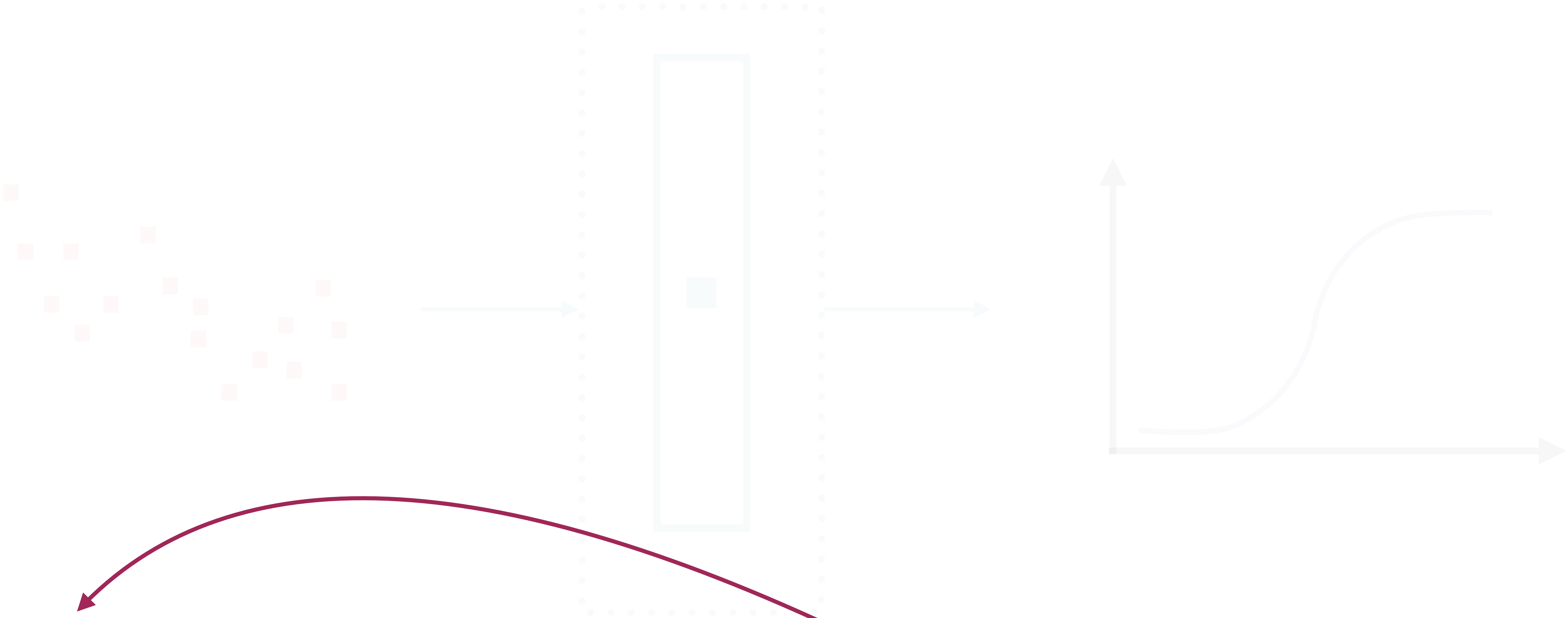
1-dimensional feature vector

Shape (W) = [1,N]

Shape (b) = [N]

S-Curve

SoftMax N-category Classification



1-dimensional
feature vector

Shape (W) = [1,N]

Shape (b) = [N]

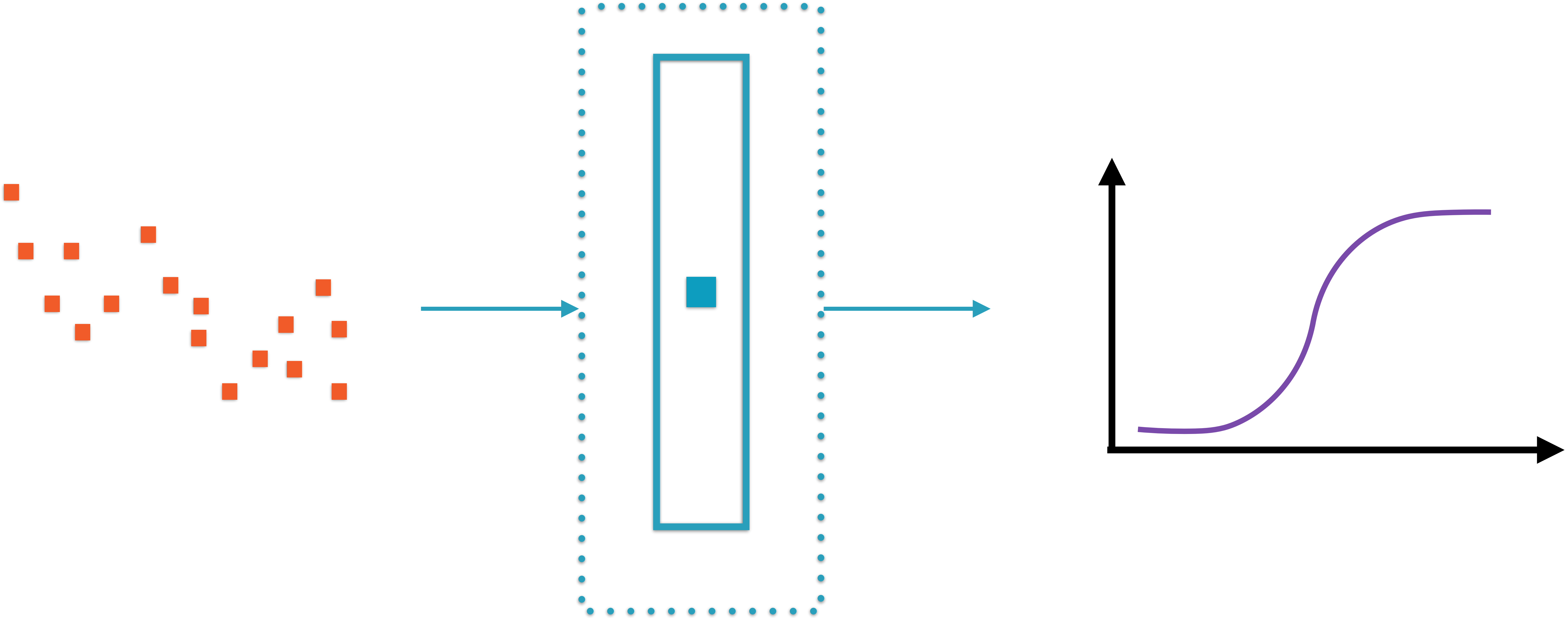
SoftMax N-category Classification

M-dimensional
feature vector

Shape (W) = [M,N]

Shape (b) = [N]

SoftMax N-category Classification



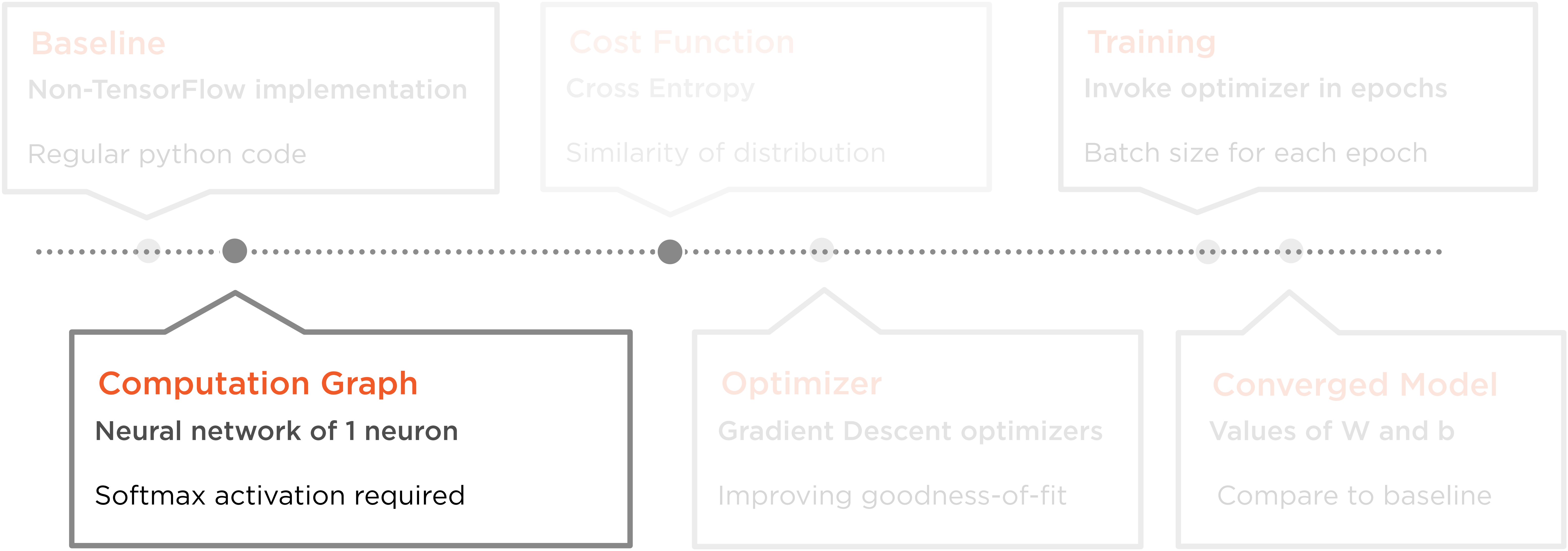
M-dimensional feature vector

Shape (W) = [M,N]

Shape (b) = [N]

S-Curve

Logistic Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Computation Graph

Neural network of 1 neuron

Softmax activation required

Cost Function

Cross Entropy

Similarity of distribution

Optimizer

Gradient Descent optimizers

Improving goodness-of-fit

Training

Invoke optimizer in epochs

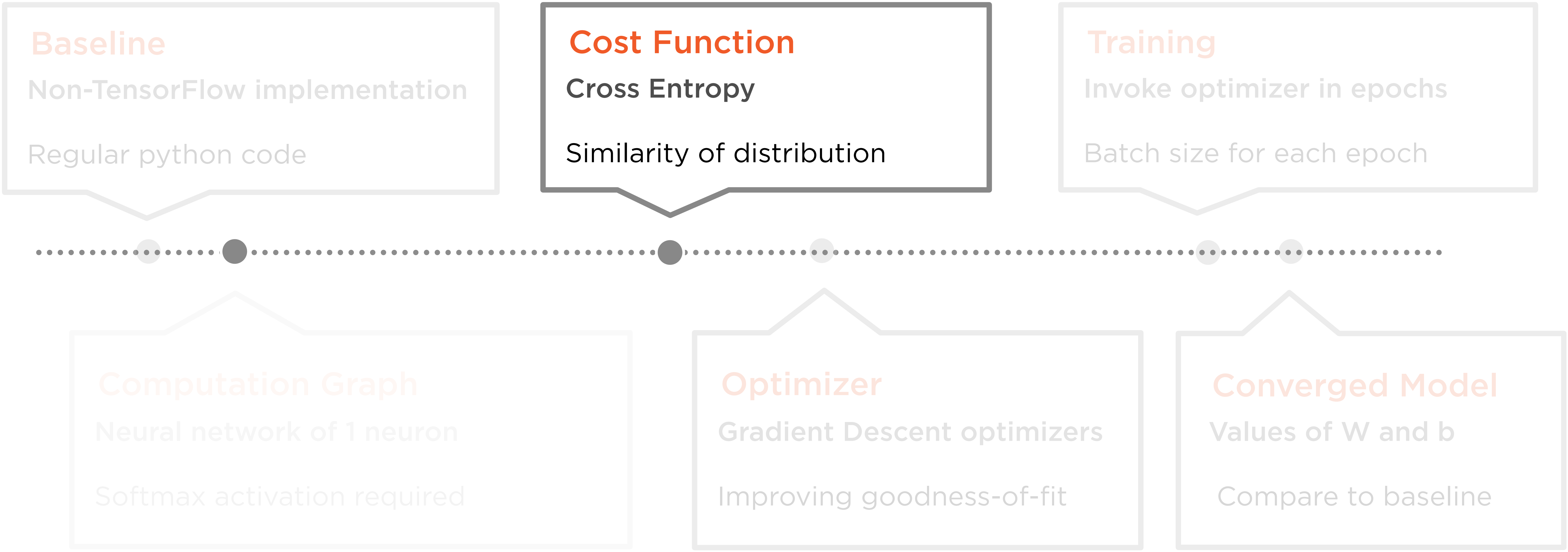
Batch size for each epoch

Converged Model

Values of W and b

Compare to baseline

Logistic Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Computation Graph

Neural network of 1 neuron

Softmax activation required

Cost Function

Cross Entropy

Similarity of distribution

Optimizer

Gradient Descent optimizers

Improving goodness-of-fit

Training

Invoke optimizer in epochs

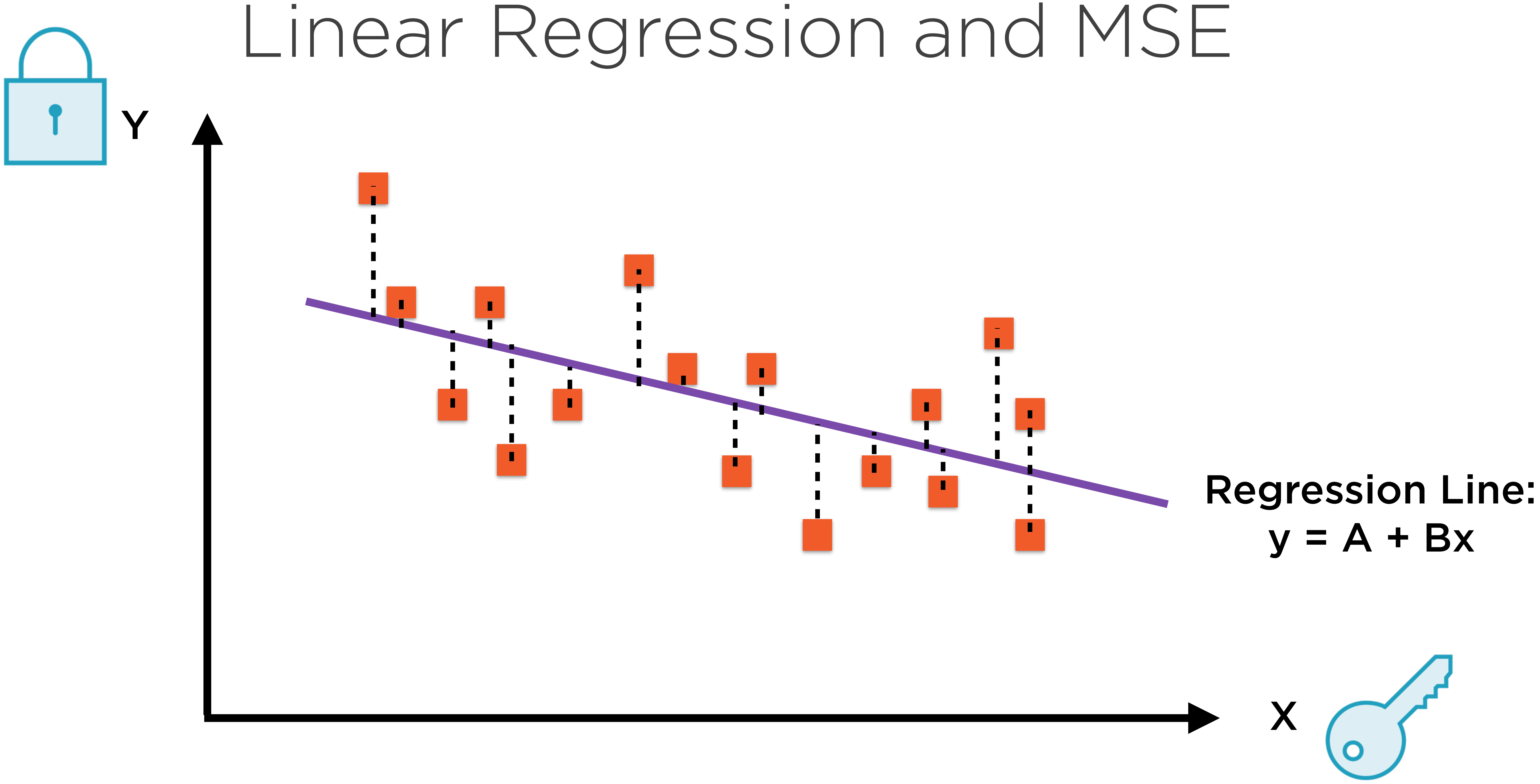
Batch size for each epoch

Converged Model

Values of W and b

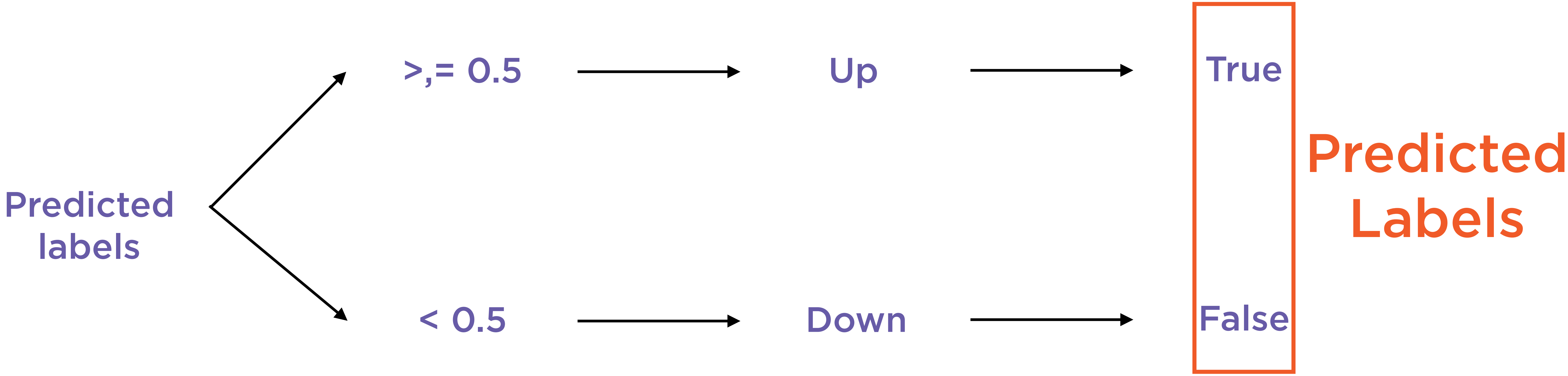
Compare to baseline

Linear Regression and MSE



The “best fit” line is called the regression line

Logistic Regression



Predicted labels

>,= 0.5

< 0.5

Up

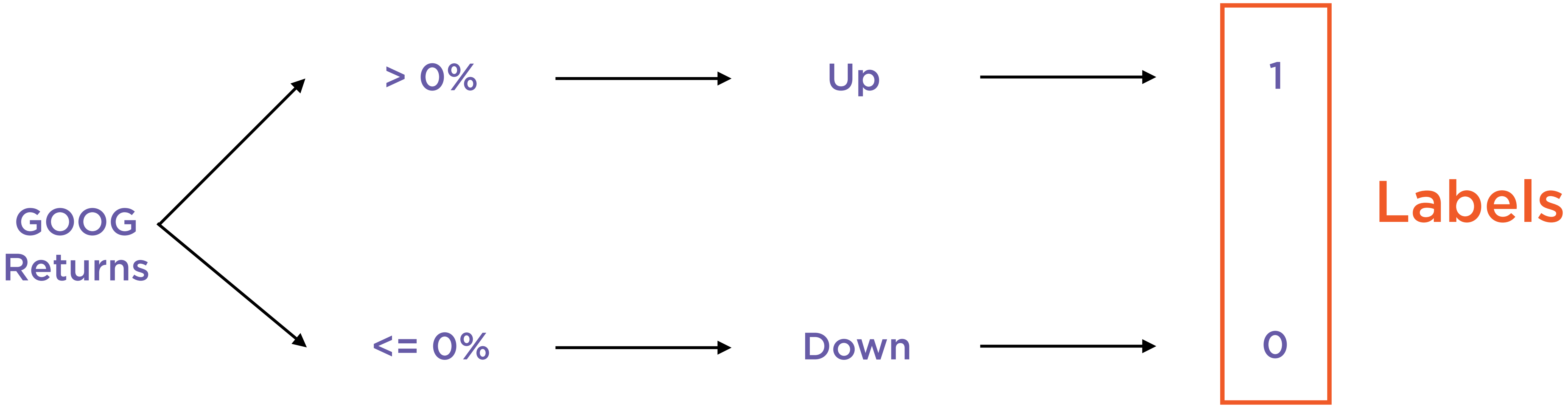
Down

True

False

Predicted Labels

Set up the Problem



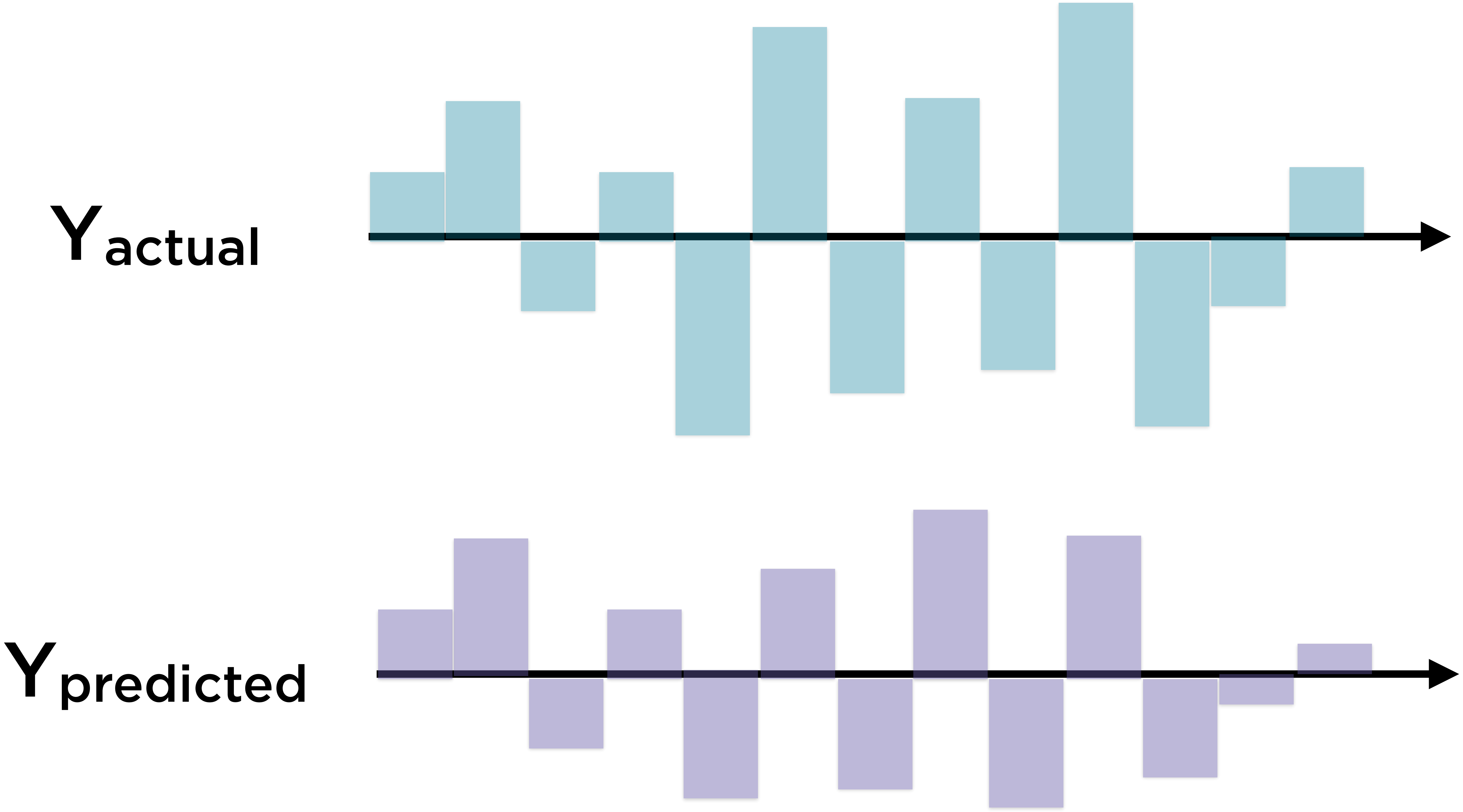
Label GOOG returns as binary (1,0)

Prediction Accuracy

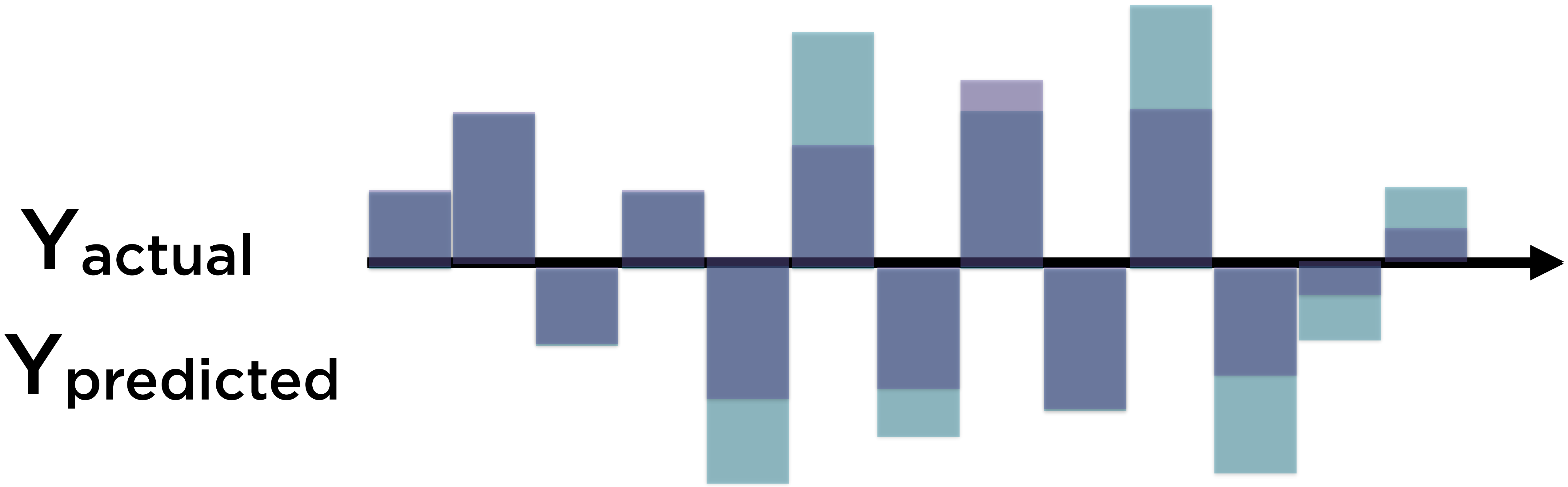
DATE	ACTUAL	PREDICTED
2005-01-01	NA	NA
2005-02-01	0	1
2005-03-01	0	0
2017-01-01	1	1
2017-02-01	1	1

Compare GOOG’s actual labels vs. predicted labels

Intuition: Low Cross Entropy

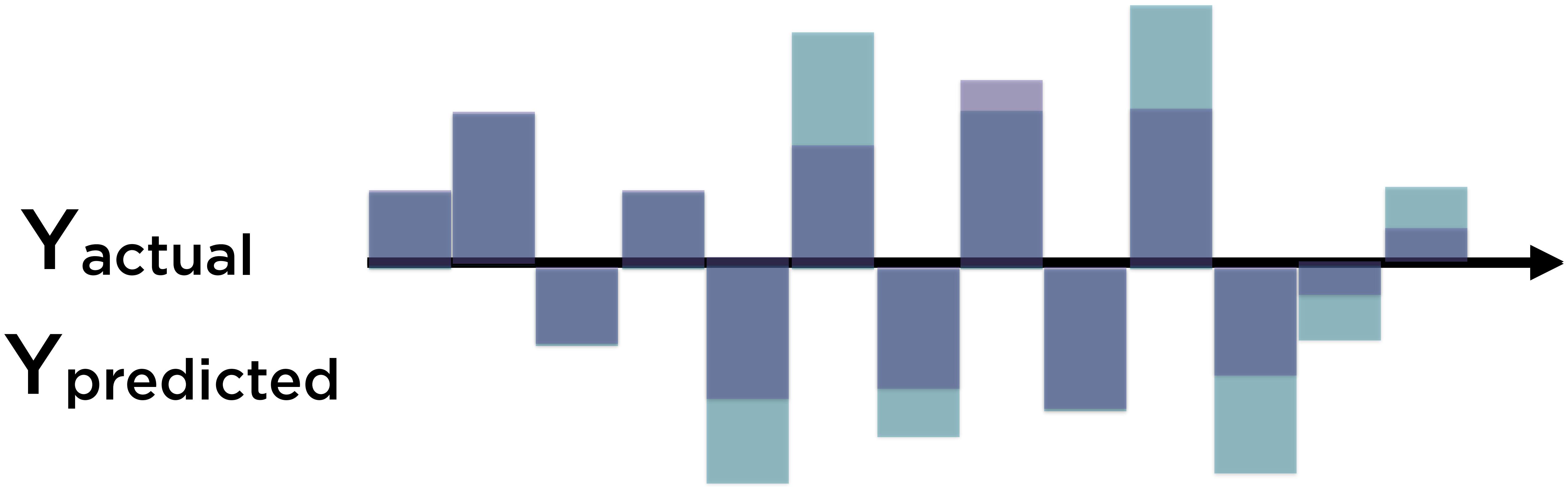


Intuition: Low Cross Entropy



The labels of the two series are in-synch

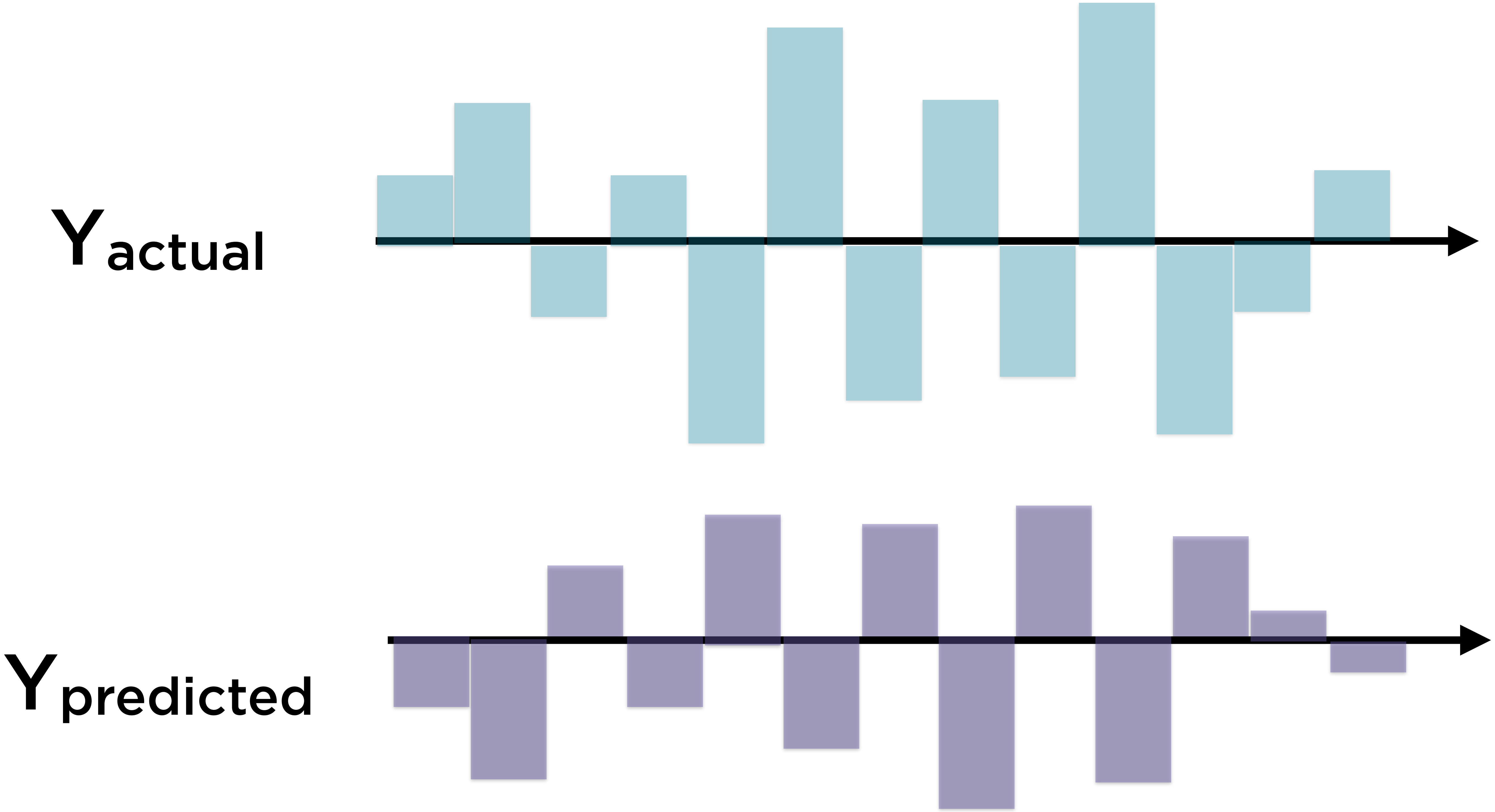
Intuition: Low Cross Entropy



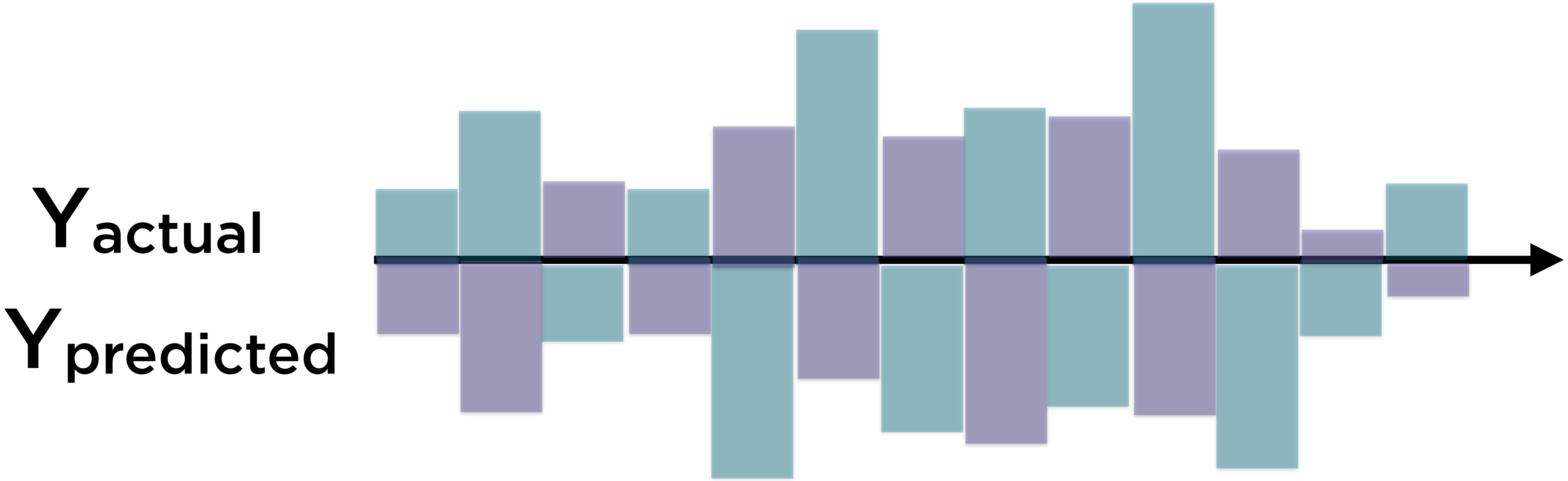
$-Sum(Y_{actual} * log [Y_{predicted}])$ will be small

Cross Entropy

Intuition: High Cross Entropy

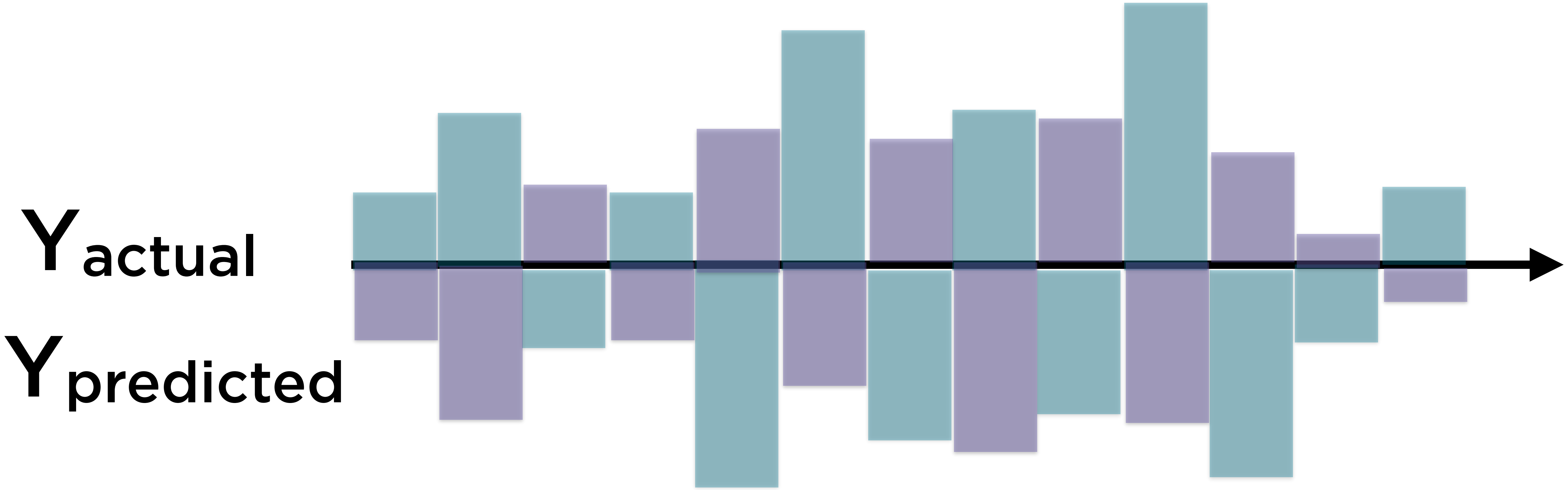


Intuition: High Cross Entropy



The labels of the two series are out-of-synch

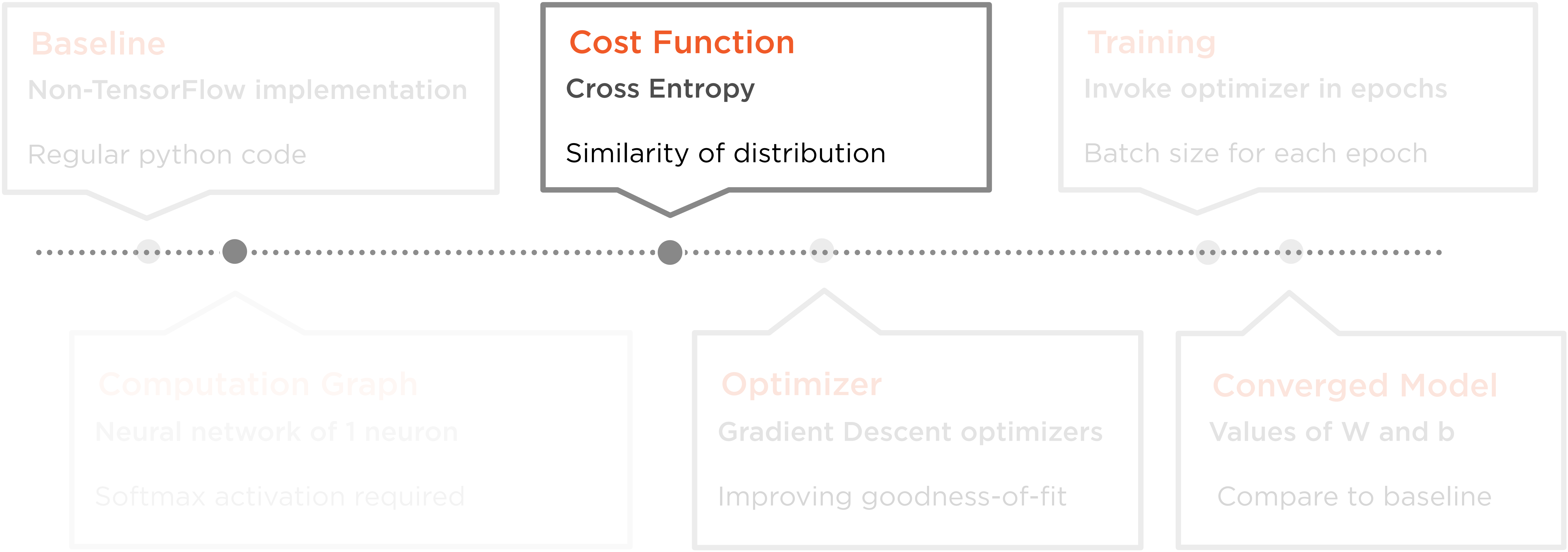
Intuition: High Cross Entropy



$-Sum(Y_{actual} * \log [Y_{predicted}])$ will be large

Cross Entropy

Logistic Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Cost Function

Cross Entropy

Similarity of distribution

Training

Invoke optimizer in epochs

Batch size for each epoch

Computation Graph

Neural network of 1 neuron

Softmax activation required

Optimizer

Gradient Descent optimizers

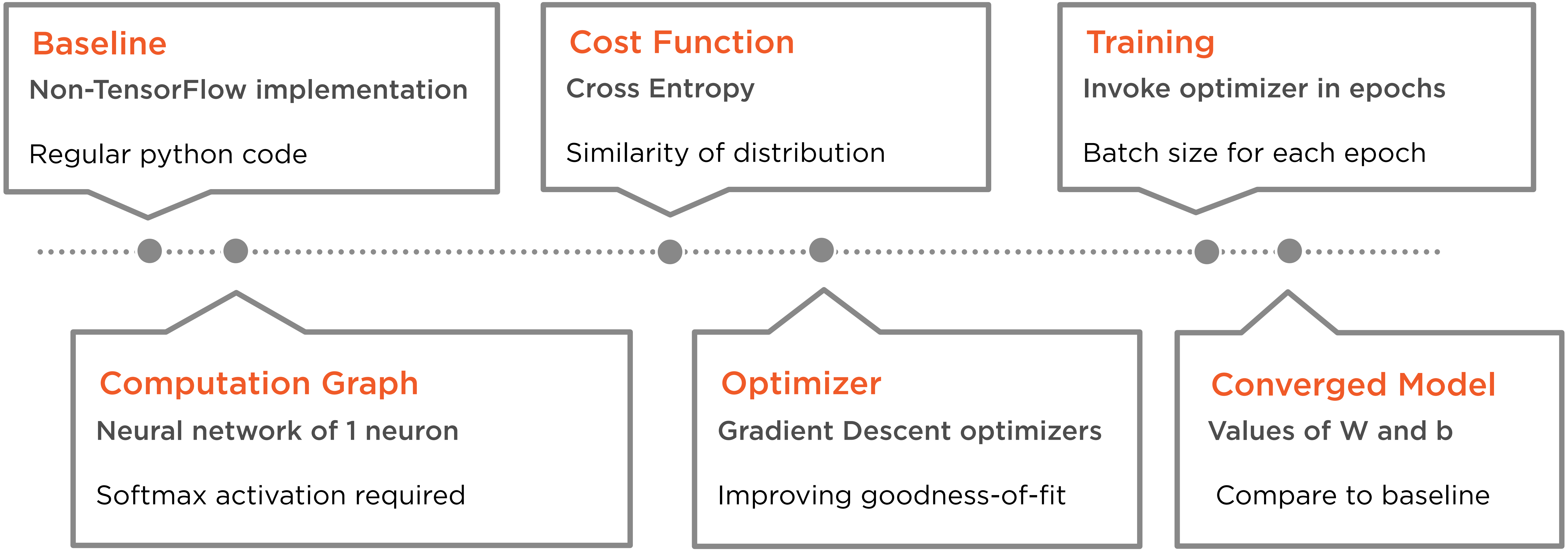
Improving goodness-of-fit

Converged Model

Values of W and b

Compare to baseline

Logistic Regression in TensorFlow



Baseline

Non-TensorFlow implementation

Regular python code

Computation Graph

Neural network of 1 neuron

Softmax activation required

Cost Function

Cross Entropy

Similarity of distribution

Optimizer

Gradient Descent optimizers

Improving goodness-of-fit

Training

Invoke optimizer in epochs

Batch size for each epoch

Converged Model

Values of W and b

Compare to baseline

```
tensorflow.argmax(y,1)
```

Finding the index of the largest element

Return the index of the largest element of tensor y along dimension k

Tensor

```
tensorflow.argmax(y,1)
```

Finding the index of the largest element

Return the index of the largest element of tensor y along dimension k

Dimension

```
tensorflow.argmax(y,1)
```

Finding the index of the largest element

Return the index of the largest element of tensor y along dimension k

tf.argmax

Tensor y

	Dimension 0	Dimension 1
Index = 0		5
1		15
2		12
3		100
4		74
5		33

tf.argmax(y,1)

tf.argmax

Tensor y

	Dimension 0	Dimension 1
Index = 0		5
1		15
2		12
3		100
4		74
5		33

Return value → 3

tf.argmax(y,1)

tf.argmax

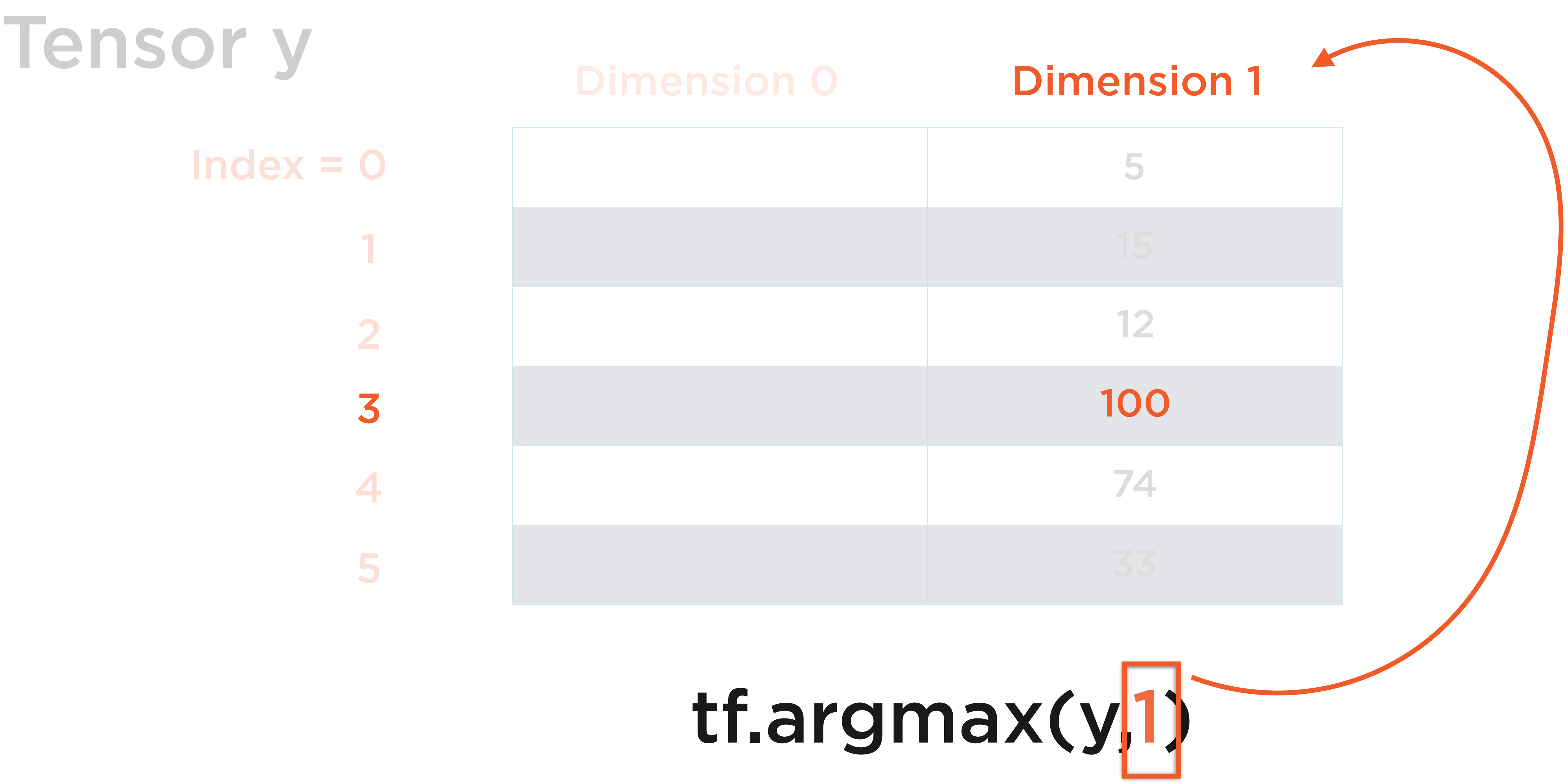
Tensor y

	Dimension 0	Dimension 1
Index = 0		5
1		15
2		12
3		100
4		74
5		33

Return value → 3

tf.argmax(y,1)

tf.argmax

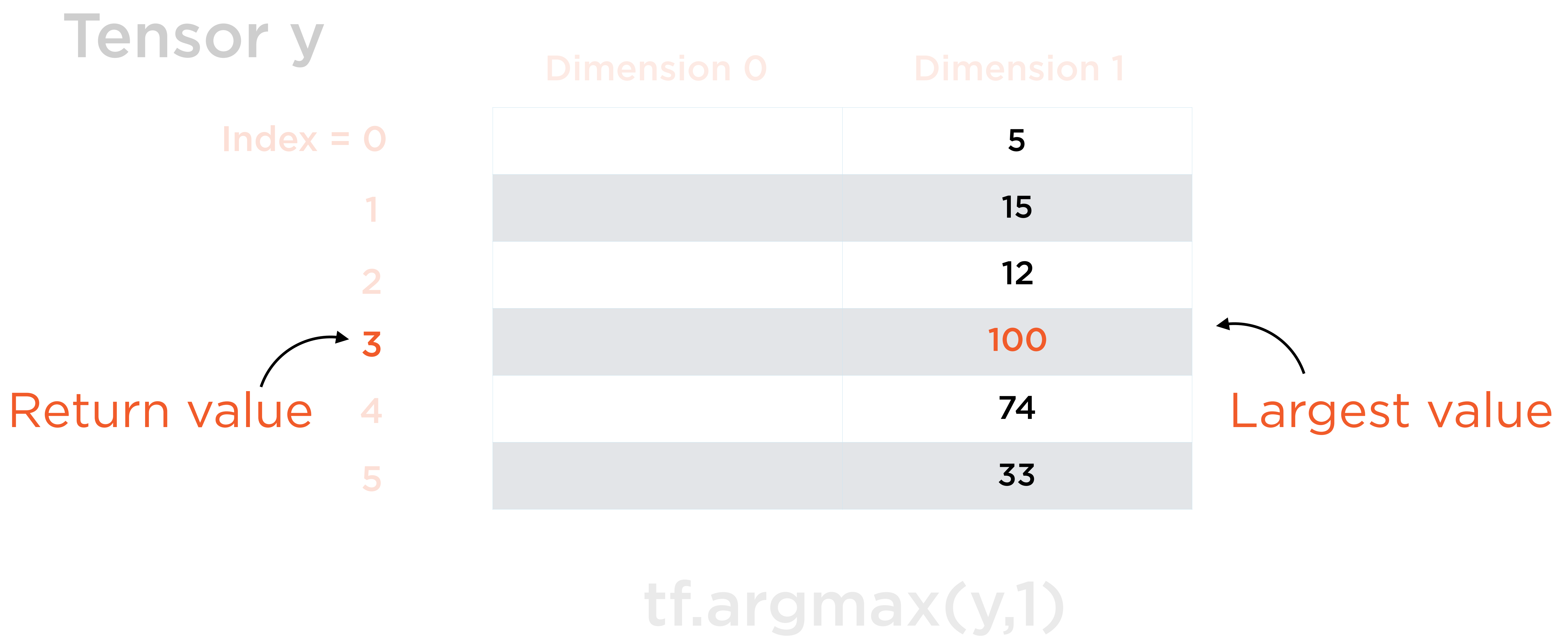


Tensor y

	Dimension 0	Dimension 1
Index = 0		5
1		15
2		12
3		100
4		74
5		33

```
tf.argmax(y,1)
```

tf.argmax



Tensor y

	Dimension 0	Dimension 1
Index = 0		5
1		15
2		12
3		100
4		74
5		33

Return value

Largest value

tf.argmax(y,1)

tf.argmax

Tensor y

	Dimension 1	...	Dimension N
Index = 0			
Index = M			

tf.argmax(y,1)

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

Actual labels

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

Predicted labels

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

One-hot

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

One-hot Representation

TRUE
FALSE
FALSE
...
TRUE

Label Vector

TRUE	FALSE
1	0
0	1
0	1
1	0

One-hot Label Vector

One-hot y_

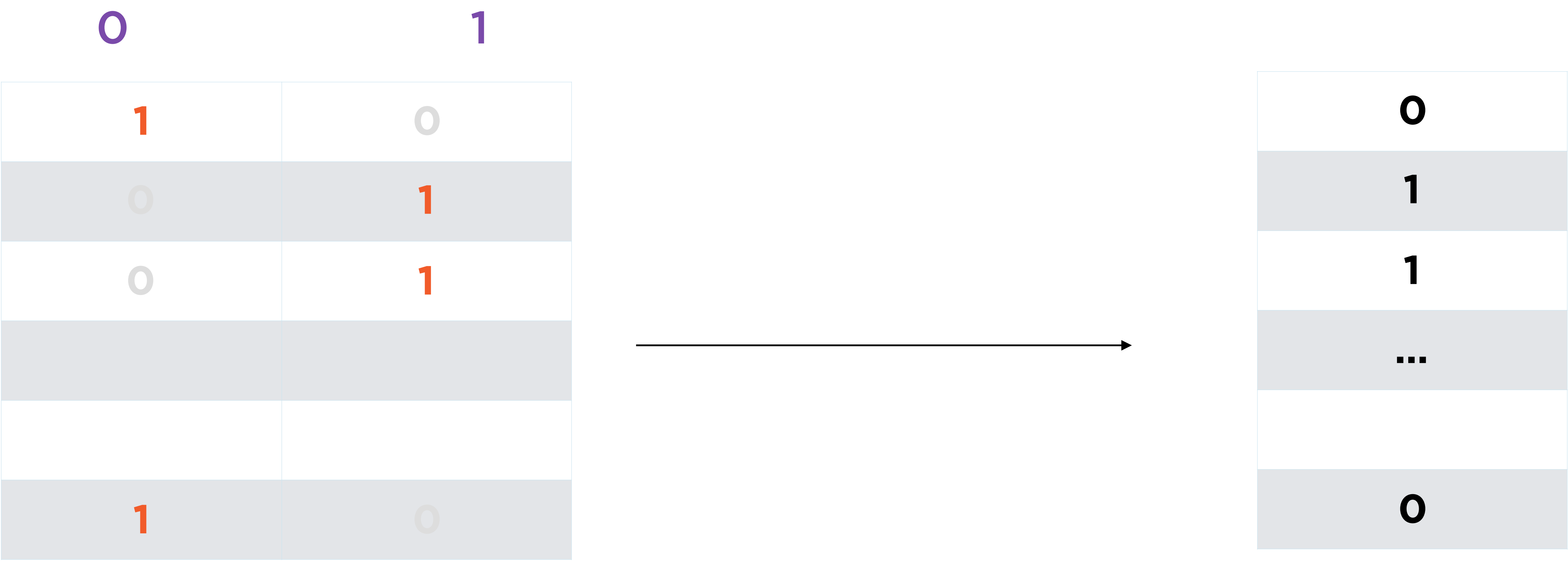
TRUE
FALSE
FALSE
...
TRUE

Label Vector

TRUE	FALSE
1	0
0	1
0	1
1	0

One-hot Label Vector

argmax(y_,1)



0	1
1	0
0	1
0	1
1	0

One-hot Label Vector

0
1
1
...
0

Index of one-hot element

Predicted labels

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

Predicted Probabilities y

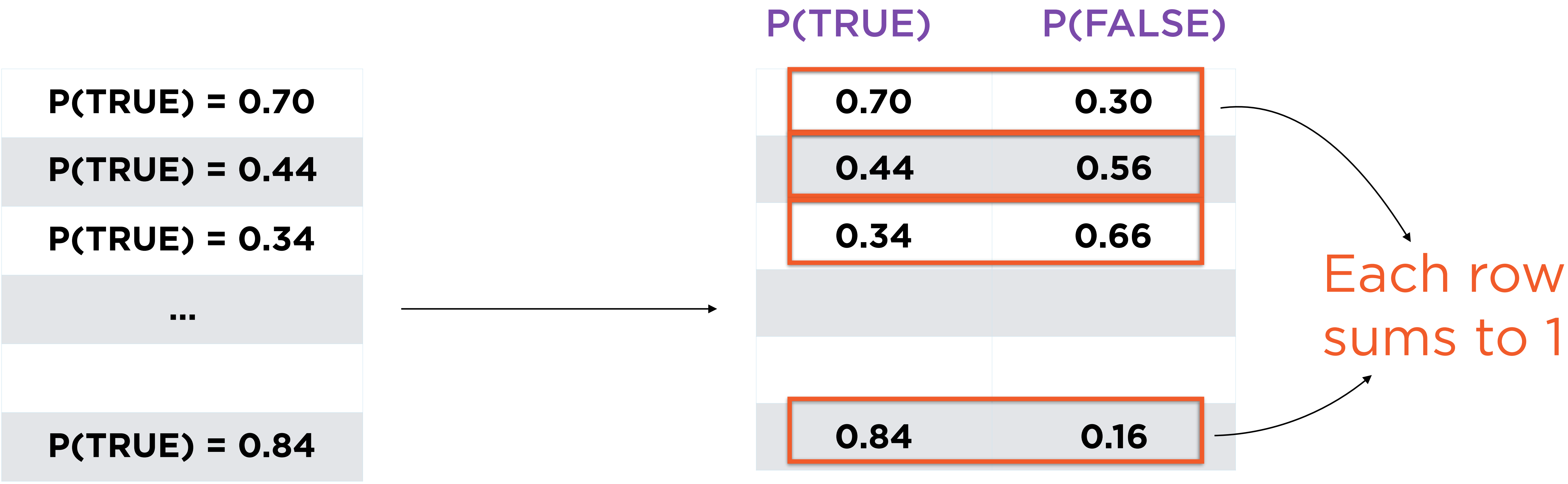
P(TRUE) = 0.70
P(TRUE) = 0.44
P(TRUE) = 0.34
...
P(TRUE) = 0.84

Probabilities

P(TRUE)	P(FALSE)
0.70	0.30
0.44	0.56
0.34	0.66
0.84	0.16

Softmax Output

Predicted Probabilities y



P(TRUE) = 0.70
P(TRUE) = 0.44
P(TRUE) = 0.34
...
P(TRUE) = 0.84

Probabilities

P(TRUE)	P(FALSE)
0.70	0.30
0.44	0.56
0.34	0.66
0.84	0.16

Softmax Output

Each row sums to 1

Predicted Probabilities y

P(TRUE) = 0.70
P(TRUE) = 0.44
P(TRUE) = 0.34
...
P(TRUE) = 0.84

Probabilities

P(TRUE)	P(FALSE)
0.70	0.30
0.44	0.56
0.34	0.66
0.84	0.16

Softmax Output

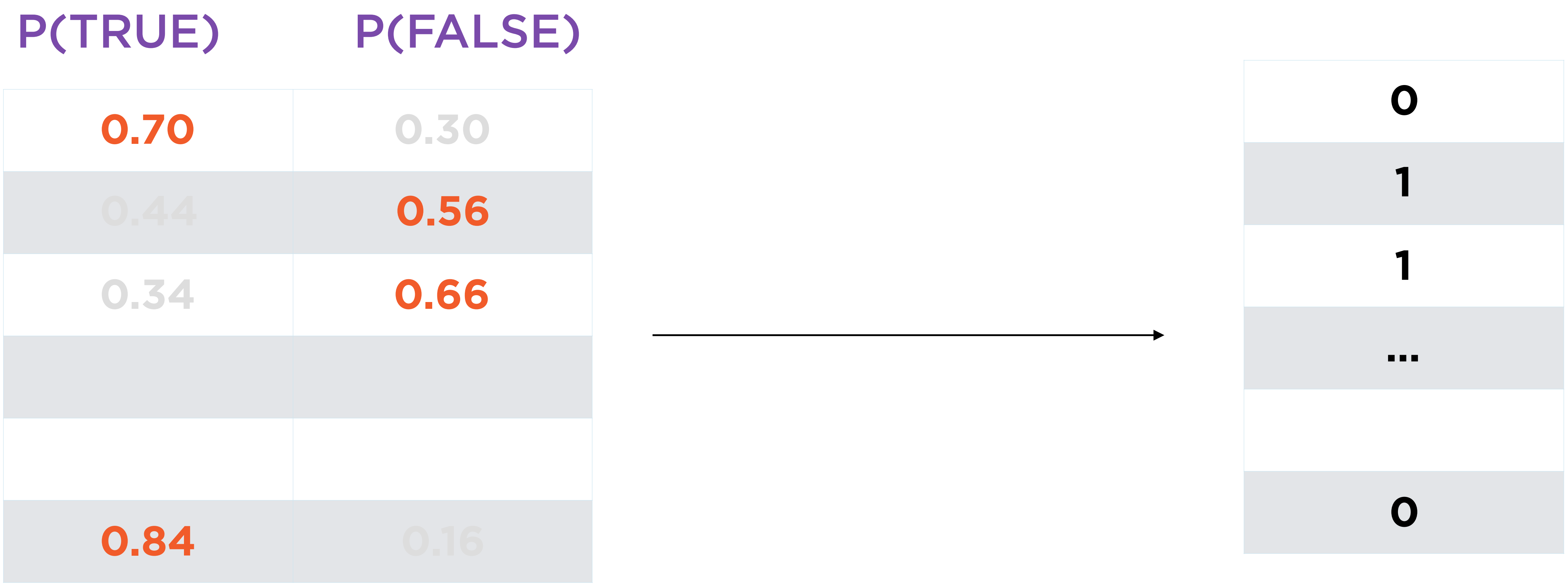
Rule of 50% in Binary Classification

Mammal

Fish

Probability of whales being Fish < 50%

argmax(y,1)



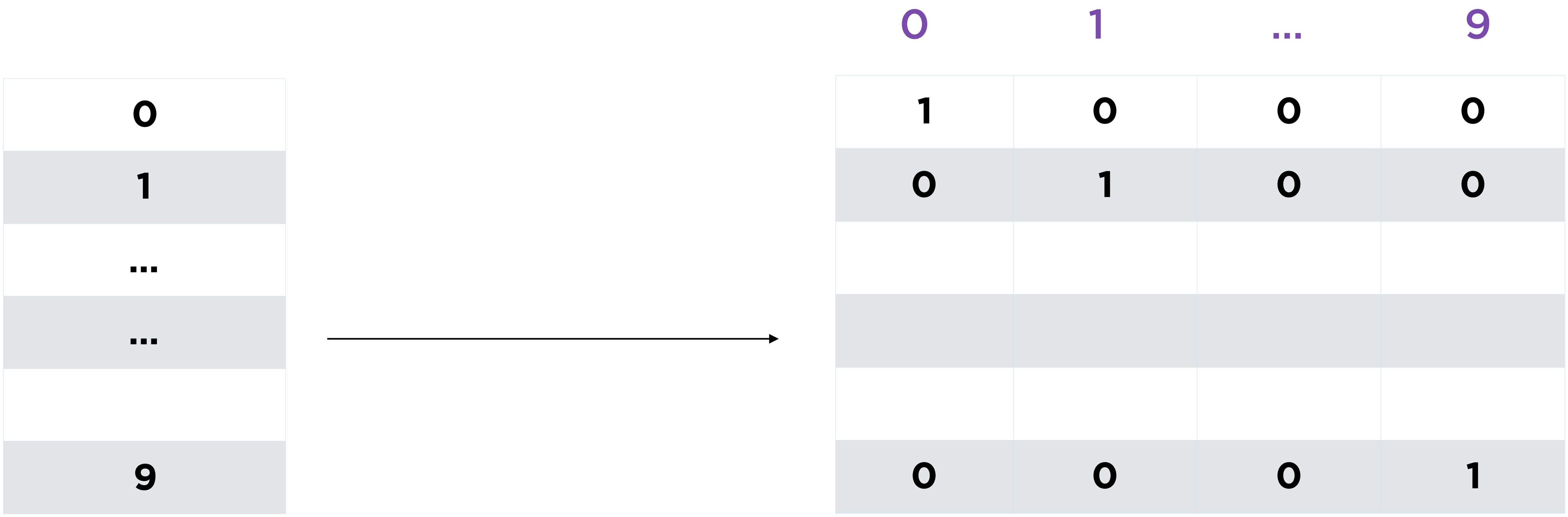
P(TRUE)	P(FALSE)
0.70	0.30
0.44	0.56
0.34	0.66
0.84	0.16

Softmax Output

0
1
1
...
0

argmax(y,1)

One-hot Vectors with Digit Classes



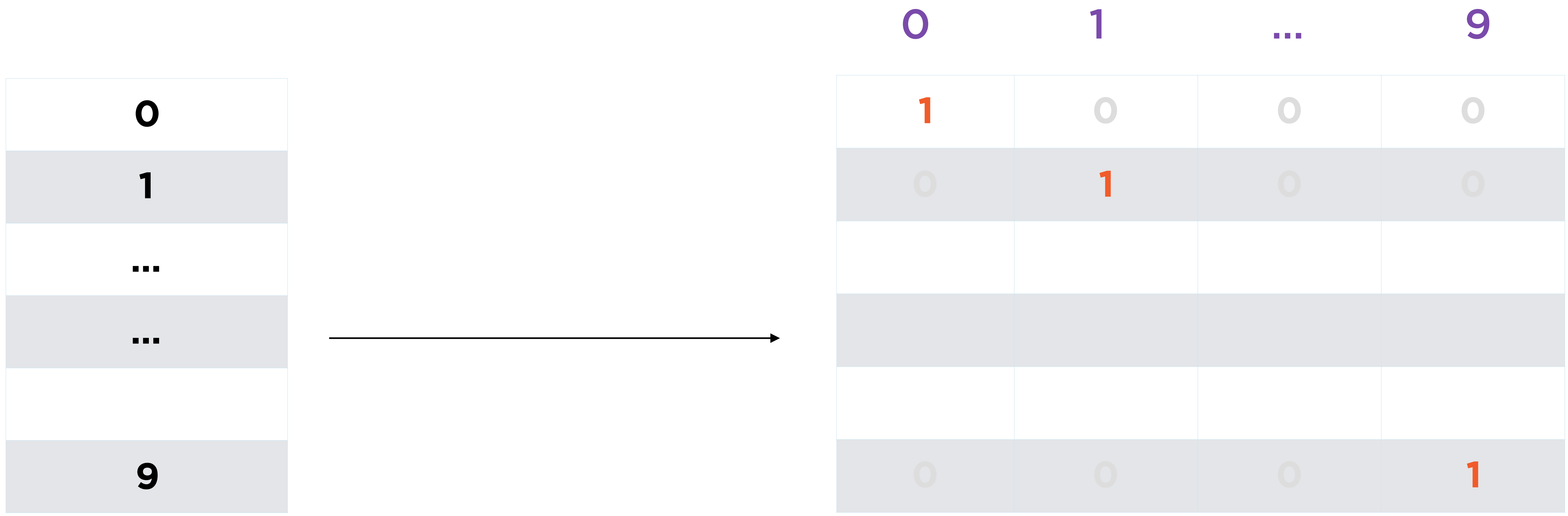
0
1
...
...
9

0	1	...	9
1	0	0	0
0	1	0	0
0	0	0	1

Actual Digits

One-hot Label Vectors

y_:One-hot Vectors with Digit Classes



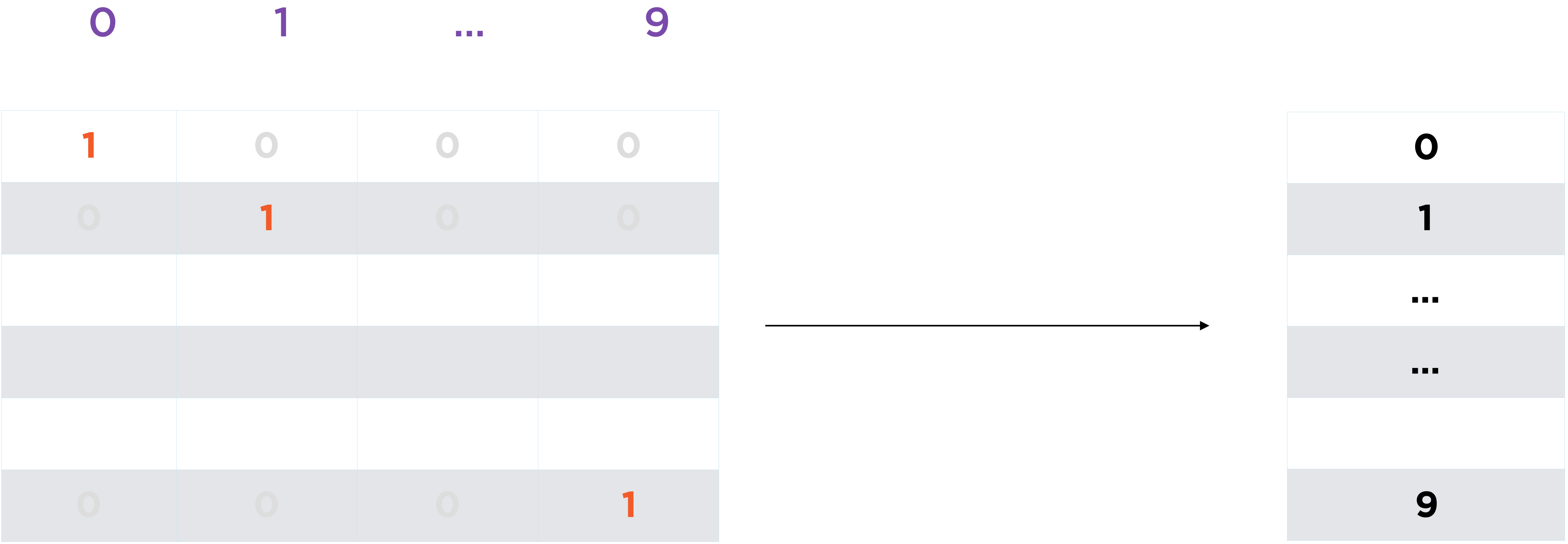
0
1
...
...
9

Actual Digits

0	1	...	9
1	0	0	0
0	1	0	0
0	0	0	1

One-hot Label Vectors

argmax(y_,1)



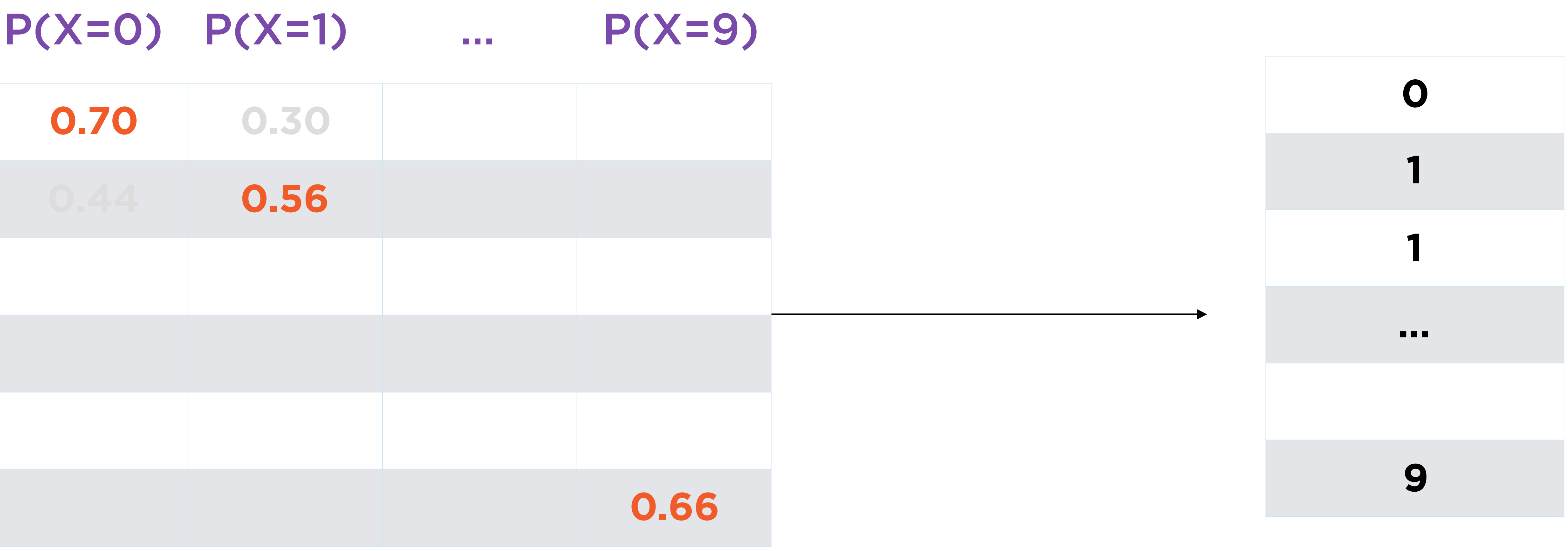
0	1	...	9
1	0	0	0
0	1	0	0
0	0	0	1

0
1
...
...
9

One-hot Label Vectors

argmax(y_,1)

Digit Classification



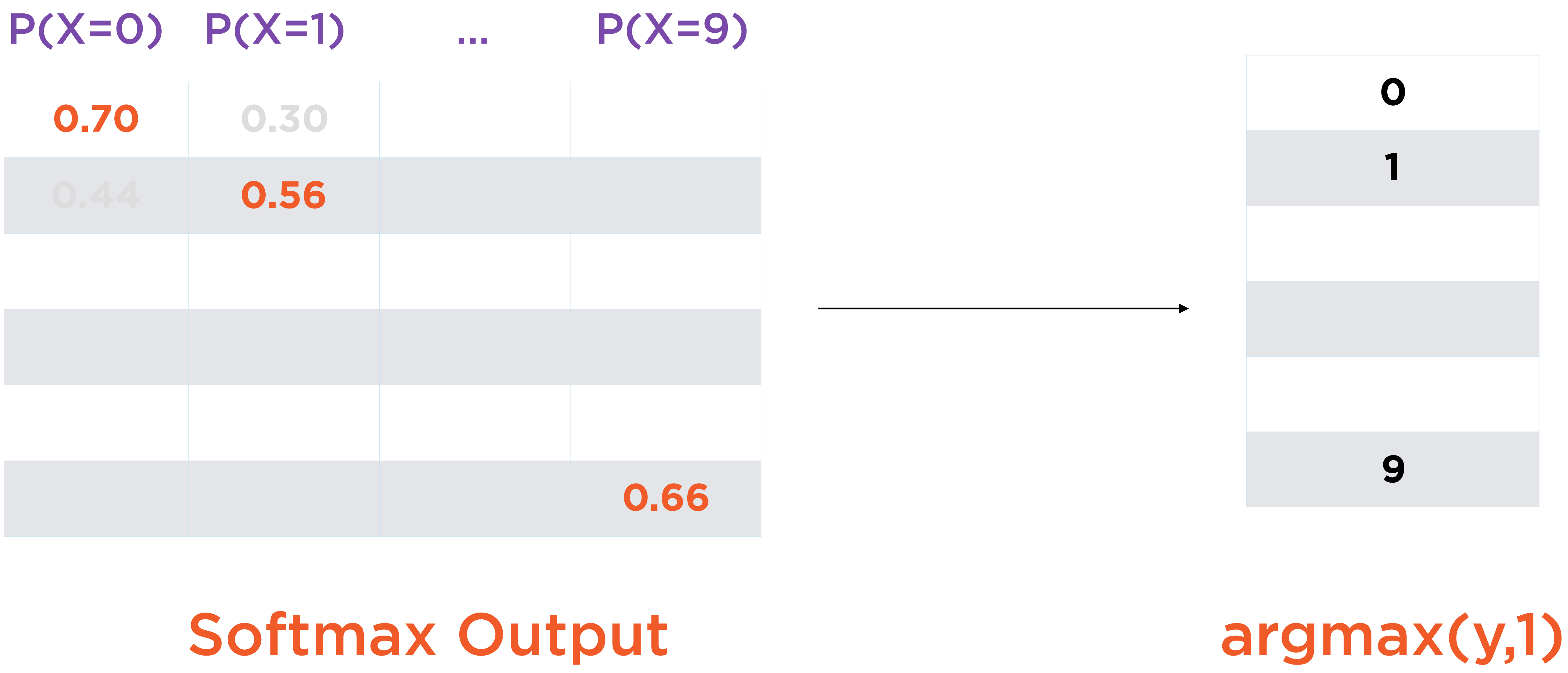
P(X=0)	P(X=1)	...	P(X=9)
0.70	0.30		
0.44	0.56		
			0.66

0
1
1
...
9

Softmax Output

argmax(y,1)

y: Predicted Probabilities



P(X=0)	P(X=1)	...	P(X=9)
0.70	0.30		
0.44	0.56		
			0.66

0
1
9

Softmax Output

argmax(y,1)

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

Actual labels

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

Predicted labels

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

Tensor of actual labels

Tensor of predicted labels

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

List of True, False values

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y

True: Correct prediction
False: Incorrect prediction

```
tf.equal(tf.argmax(y_,1), tf.argmax(y,1))
```

Two invocations of tf.argmax

Once on actual labels y_, once on predicted values y