Linear Models for Regression

Problem Definition

$$\text{Given features}: x_1, x_2, \ldots, x_n$$

$$\text{predict real number y}$$

Simplest Guess

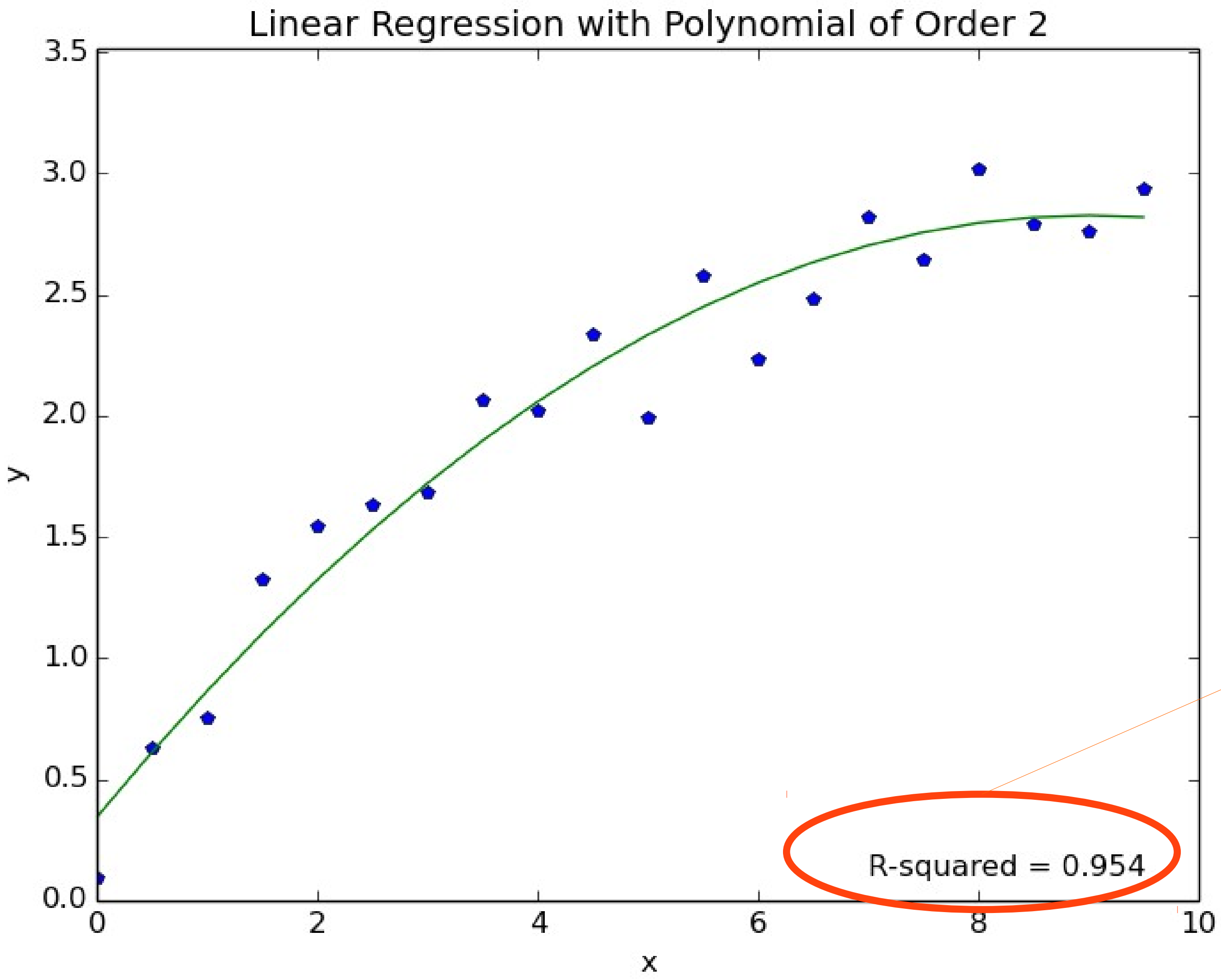
$$y = w_0 + w_1 x_1 + \ldots + w_n x_n$$

Linear in weights, **NOT** features

For example, maybe $x_1 = x$ and $x_n = x^n, n > 1$

Definitely not linear in features x, but linear in weights w_i

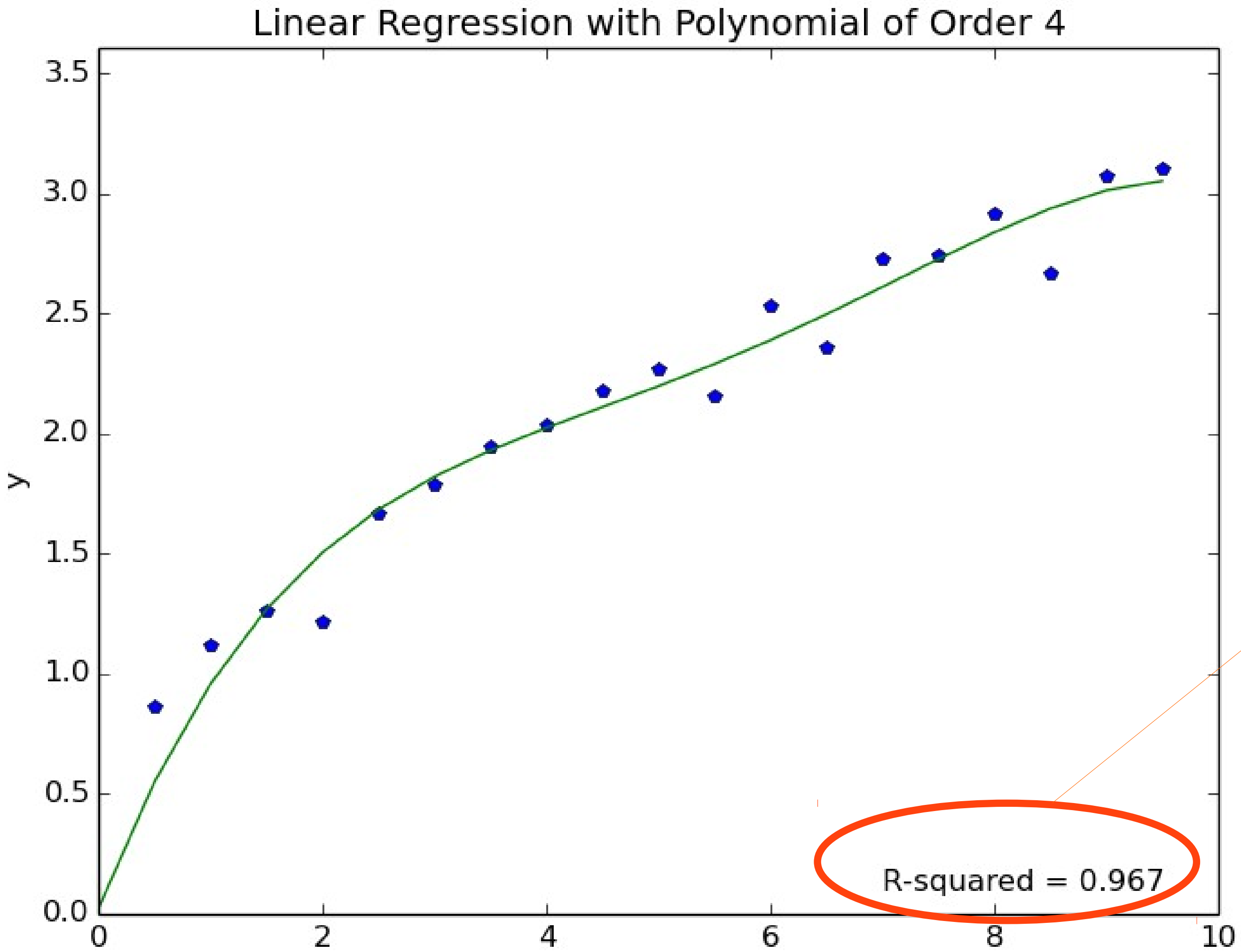
Linear Regression: Example



Will see this again. ~1 means good fit

$$y = w_0 + w_1 x + w_2 x^2 = 0.344 + 0.551x - 0.031x^2$$

Linear Regression: Example



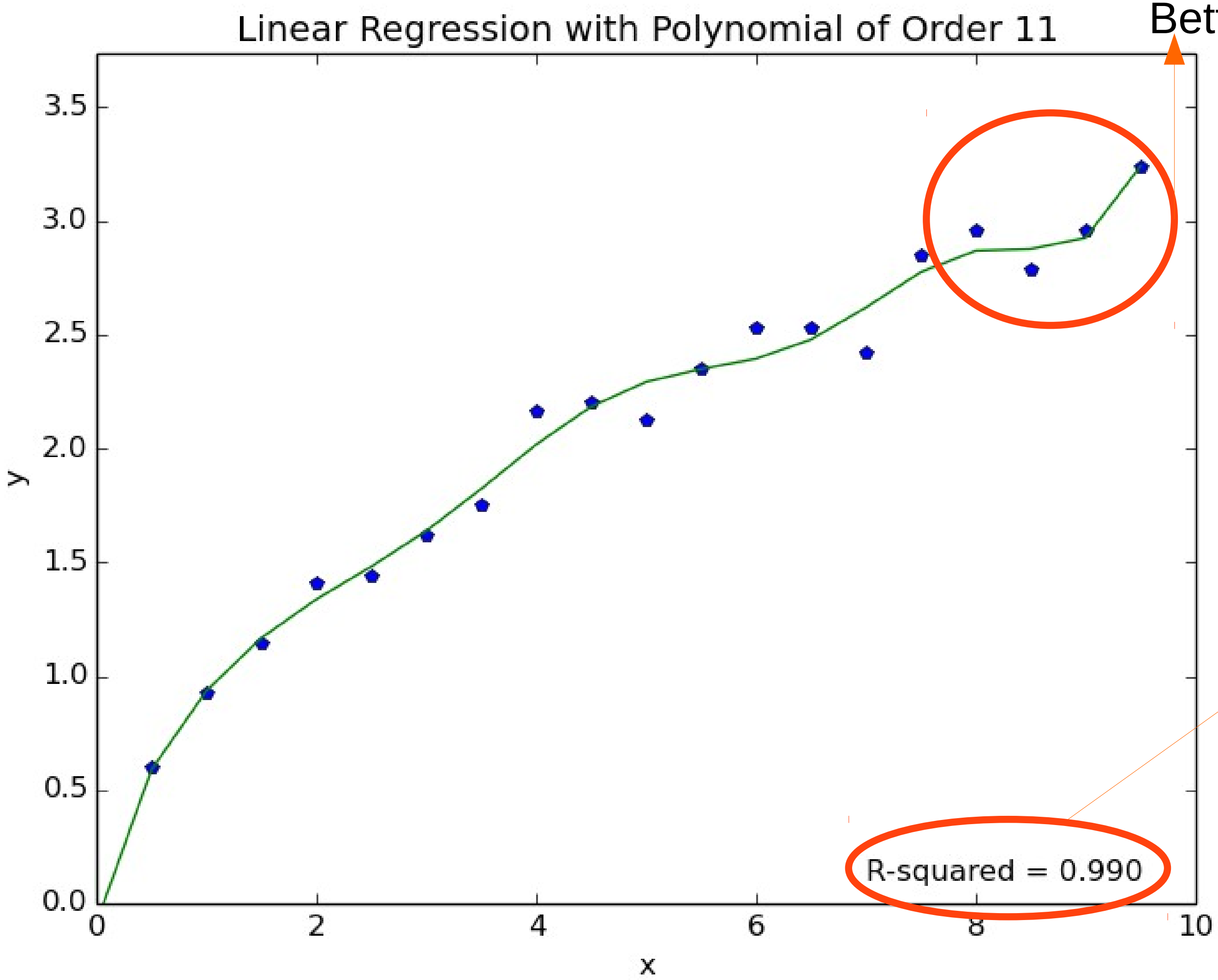
Better

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^2 + w_4 x^4$$

$$y = 0.023 + 1.185x - 0.287x^2 + 0.0345x^3 - 0.001x^4$$

Linear Regression: Example

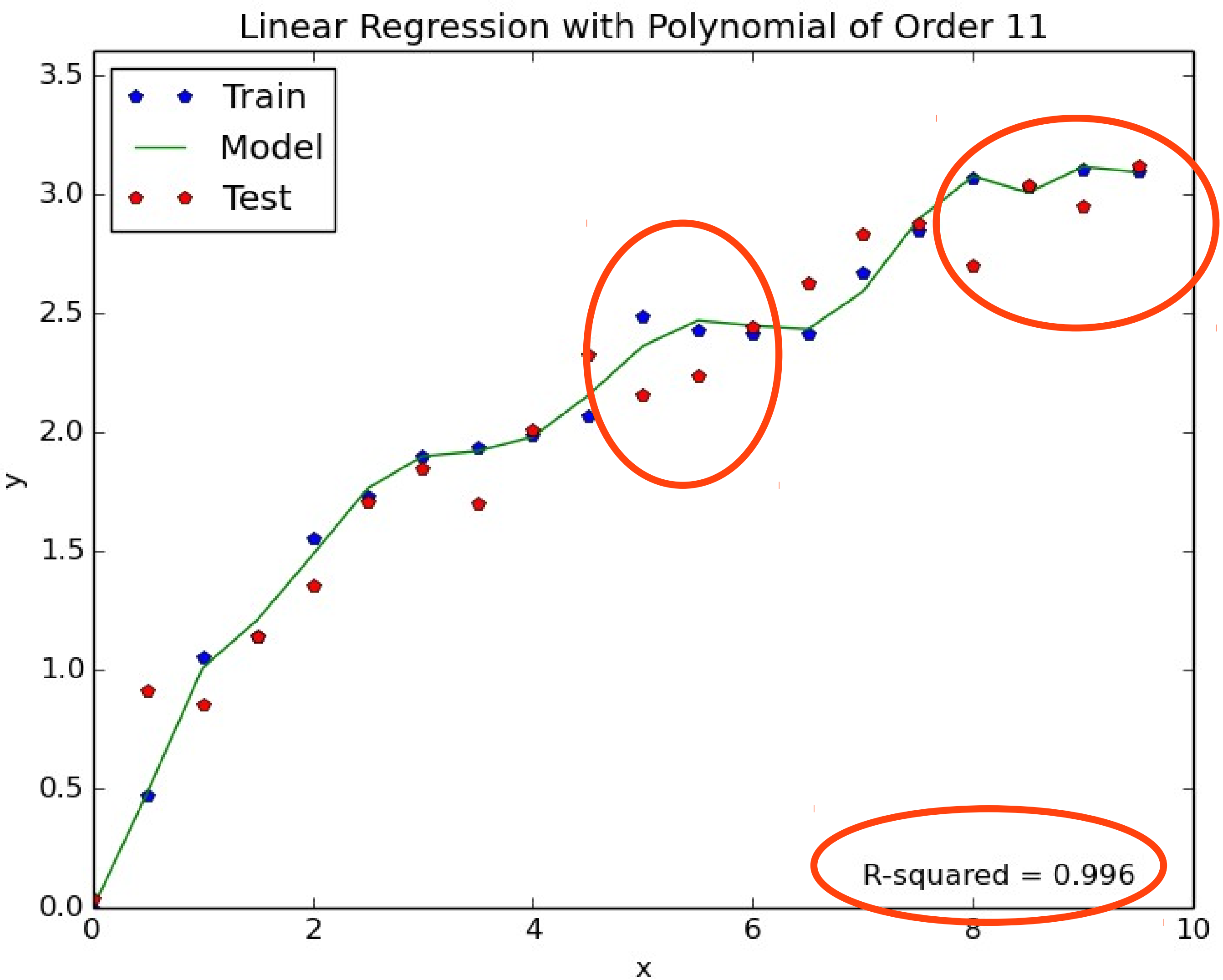
Getting all "squiggles".
Better, right?



Even better?!

$$y = w_0 + w_1 x + \ldots + w_{11} x^{11}$$

Linear Regression: Example



Overfitting: Model too complex and fits too well to training data. Does badly when applied to new data.

Linear Regression: Training

Define cost function to minimize:

M training examples

$$C(w_0, w_1, \ldots, w_n) = \Sigma_{i=1}^{M} C_i(w_0, w_1, \ldots, w_n)$$

Total Cost

Cost for each training example

What should we choose for $C_i(w_0, w_1, \ldots, w_n)$?

Linear Regression: Training

What should we choose for $C_i(w_0, w_1, \ldots, w_n)$?

One choice: $C_i = (y_i - p_i)^2$

$y_i =$ Known value for example i

$p_i =$ Predicted value for example i

$$p_i = w_0 + w_1 x_1^i + \ldots + w_n x_n^i$$

Example i

Linear Regression: Training

$$\text{Minimize cost function: } C(w_0, w_1, \ldots, w_n)$$

$$\text{Solve: } \frac{\partial C}{\partial w_i} = 0, i = 0, \ldots, n$$

$$C = \Sigma_{i=1}^{M} (y_i - (w_0 + w_1 x_1^i + \ldots + w_n x_n^i))^2$$

Linear Regression: Training

$$\Sigma_{i=1}^{M}(y_i - w_0 - w_1 x_1^i - \ldots - w_n x_n^i).1 = 0$$

$$\Sigma_{i=1}^{M}(y_i - w_0 - w_1 x_1^i - \ldots - w_n x_n^i).x_1^i = 0$$

$$\vdots$$

$$\Sigma_{i=1}^{M}(y_i - w_0 - w_1 x_1^i - \ldots - w_n x_n^i).x_n^i = 0$$

Linear Regression: Training

Define:

$$Y \equiv \Sigma_{ex=1}^{M} y_{ex}$$

$$X_{i,j} \equiv \Sigma_{ex=1}^{M} x_i^{ex} x_j^{ex}$$

$$Z_i \equiv \Sigma_{ex=1}^{M} y_{ex} x_i^{ex}$$

$$X_{i,0} = X_{0,i} \equiv \Sigma_{ex=1}^{M} x_i^{ex}$$

$$X_{0,0} = \Sigma_{i=1}^{M} 1 = M$$

$$w_0 X_{0,0} + w_1 X_{0,1} + \ldots + w_n X_{0,n} = Y$$

$$w_0 X_{1,0} + w_1 X_{1,1} + w_2 X_{1,2} + \ldots + w_n X_{1,n} = Z_1$$

$$\vdots$$

$$w_0 X_{n,0} + w_1 X_{n,1} + \ldots + w_n X_{n,n} = Z_n$$

Linear Regression: Training

$$w_0 X_{0,0} + w_1 X_{0,1} + \ldots + w_n X_{0,n} = Y$$

$$w_0 X_1 + w_1 X_{1,1} + w_2 X_{1,2} + \ldots + w_n X_{1,n} = Z_1$$

$$w_0 X_{n,0} + w_1 X_{n,1} + \ldots + w_n X_{n,n} = Z_n$$

Invert matrix to solve for weights

$$\begin{pmatrix} X_{0,0} & X_{0,1} & \ldots & X_{0,n} \\ X_{1,0} & X_{1,1} & \ldots & X_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,0} & X_{n,1} & \ldots & X_{n,n} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} Y \\ Z_1 \\ \vdots \\ Z_n \end{pmatrix}$$

Depend on data

Linear Regression: Assumptions

$$\text{Data:} \quad y = w_0 + w_1 x_1 + \ldots + w_n x_n + \mathcal{N}(0, \sigma^2)$$

Gaussian Noise

$$\textbf{Linear relationship between } \; \mathrm{y} \; \text{ and } \; \mathrm{x}_m, \forall m$$
$$\text{where all other } x_n \text{ are fixed}$$

$\mathcal{N}(0, \sigma^2)$: **Gaussian** noise with **0 mean** and **constant variance**

Might consist of many noise terms

They should be **independent**, **Gaussian** with **constant variance**

Linear Regression: Priors

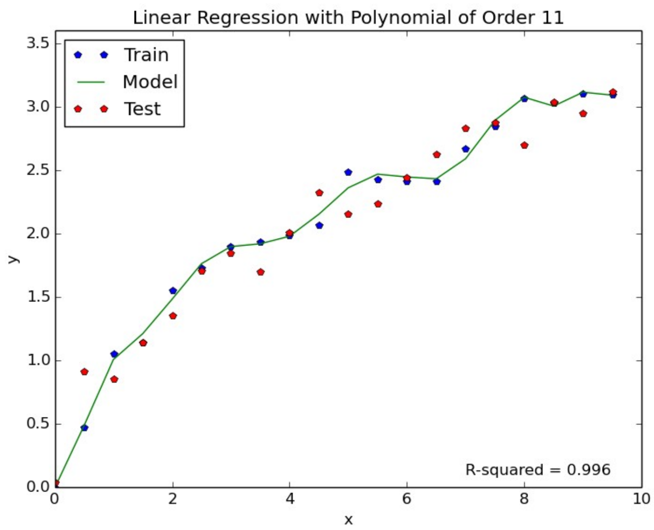
No assumptions made about values that weights take

Minimizing Least Squares Error $\Leftrightarrow$ Maximizing Likelihood assuming Gaussian predictions

$$\text{Likelihood, LL} = \Pi_{i=1}^{M} \frac{1}{2\pi\sigma^2} e^{-\frac{(y_i - w_0 - w_1 x_1 - \ldots - w_n x_n)^2}{2\sigma^2}}$$

Probability, under a Gaussian distribution, that the model will predict the values observed.

Linear Regression: Priors



Recall overfitting example

Large, alternating weights
to account for "squiggliness"

Insert variances

w_0	-0.00908524896296
w_1	-1.68602215
w_2	11.8480971
w_3	-19.3753732
w_4	16.1142291
w_5	-7.88218507
w_6	2.41735237
w_7	-0.477580238
w_8	0.0606973836
w_9	-0.00479240240
w_{10}	0.00023823655
w_{11}	-0.0000041844556

Please excuse significant figures!

Linear Regression: L2 Regularization (Ridge)

Assumption about weights: follow Gaussian distribution.

$$\frac{1}{2\pi\tau^2}e^{-\frac{w_i^2}{2\tau^2}}, i = 1, \ldots, n$$

Penalty on weights away from 0

$$\mathrm{LL} = \Pi_{i=1}^{M}\frac{1}{2\pi\sigma^2}e^{-\frac{(y_i - w_0 - w_1x_1 - \ldots - w_nx_n)^2}{2\sigma^2}}\boxed{\Pi_{i=1}^{n}\frac{1}{2\pi\tau^2}e^{-\frac{w_i^2}{2\tau^2}}}$$

$$\text{Maximizing Likelihood}$$

$$\Updownarrow$$

Prevents overfitting

$$\text{Minimizing new cost function}$$

$$C = \Sigma_{i=1}^{M}(y_i - (w_0 + w_1x_1^i + \ldots + w_nx_n^i))^2 + \boxed{\frac{\sigma^2}{\tau^2}\Sigma_{i=1}^{n}w_i^2}$$

Linear Regression: L1 Regularization (Lasso)

Assumption about weights: follow Laplace distribution.

$$\frac{\lambda}{2}e^{-\lambda|w_i|}, i = 1, \ldots, n$$

Penalty on weights away from 0

$$\mathrm{LL} = \Pi_{i=1}^{M}\frac{1}{2\pi\sigma^2}e^{-\frac{(y_i - w_0 - w_1 x_1 - \ldots - w_n x_n)^2}{2\sigma^2}}\boxed{\Pi_{i=1}^{n}\frac{\lambda}{2}e^{-\lambda|w_i|}}$$

Maximizing Likelihood

$\Updownarrow$

Minimizing new cost function

Prevents overfitting

$$C = \Sigma_{i=1}^{M}(y_i - (w_0 + w_1 x_1^i + \ldots + w_n x_n^i))^2 + \boxed{2\lambda\sigma^2\Sigma_{i=1}^{n}|w_i|}$$

Linear Regression: Python Code

```
from sklearn import linear_model

model = linear_model.LinearRegression() #create instance of linear regression model
model.fit(x_values, y_values) #train model

#DONE!!!

model.predict(new_x_values) #predict result on new inputs

model.coef_ #see weights

model.score(x_values, y_values) #we'll see this soon. Compute R2
```

Linear Regression: Python Code

```
model = linear_model.Ridge(alpha = 0.5) #"Ridge" regression with one free parameter

model = linear_model.Lasso(alpha = 0.5) #"Lasso" regression with one free parameter
```

Use these instead of linear model used previously
Prevent overfitting (coming up soon)

Also, it's very enlightening to look at source code if interested

Linear Regression: Scaling

Generally, features x_m can have different ranges they take values in

Good to normalized each feature to have mean $= 0$ and standard deviation $= 1$

Mean across all examples

$$x_m \rightarrow \frac{x_m - \mu(x_m)}{\sigma(x_m)}$$

Standard dev. across all examples

Mean $= 0$ helps with convergence of algorithms too

according to influence on output

Linear Regression: Performance

$$\text{Residual} \equiv \Sigma_{i=1}^{N}(y_i - f(x_i))^2$$

Result of example i

Model prediction for example i

$$\text{Variance} \equiv \Sigma_{i=1}^{N}(y_i - \bar{y})^2$$

$$\text{Unexplained Variance Ratio} = \frac{\text{Residual}}{\text{Variance}}$$

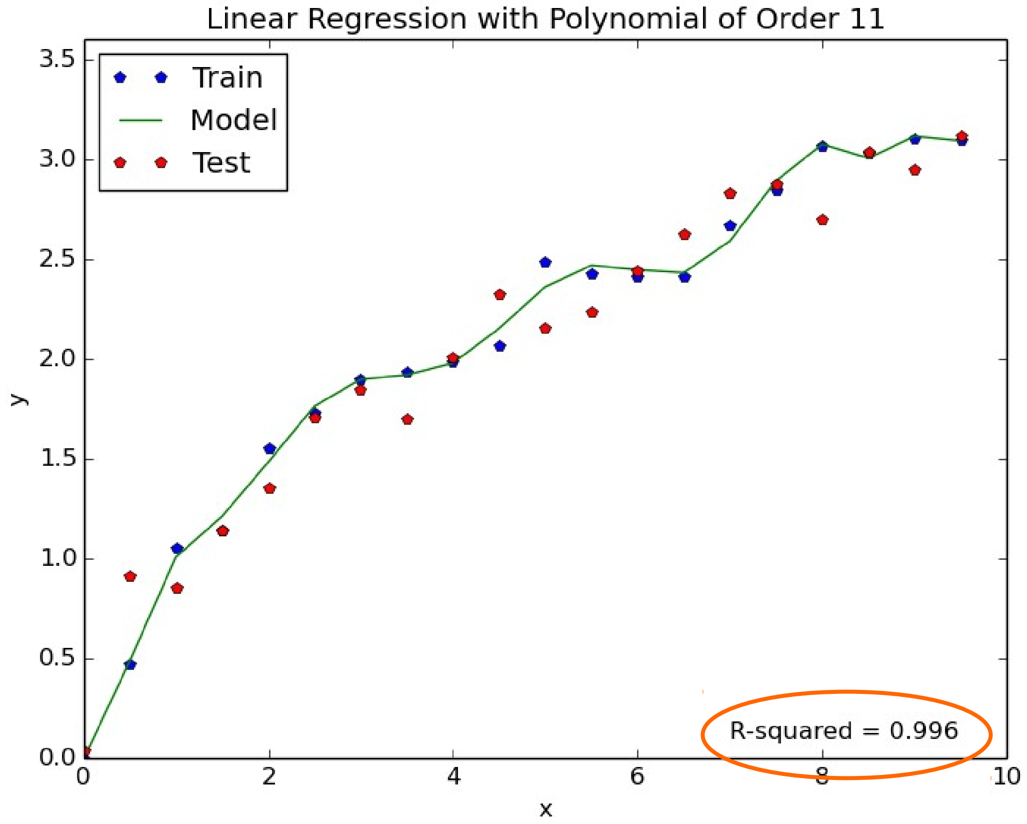
$$\boxed{R^2 \equiv 1 - \frac{\text{Residual}}{\text{Variance}}}$$

Close to 1 = Good(*)

Linear Regression: Performance

$$R^2 \equiv 1 - \frac{\text{Residual}}{\text{Variance}}$$

Close to 1 but might be overfitting



Compare R^2 between train and test sets!!!!!

Linear Regression: Performance

Simpler model is better

R^2_{Adjusted} penalizes model with many parameters

$$R^2 \equiv 1 - \frac{\text{Residual}}{\text{Variance}} \longrightarrow \boxed{R^2_{\text{Adjusted}} \equiv 1 - \frac{\frac{\text{Residual}}{df_e}}{\frac{\text{Variance}}{df_t}}}$$

$$df_t \equiv n - 1$$ → "Degrees of freedom for variance"

$$df_e \equiv n - p - 1$$ → "Degree of freedom for model"

$$n = \text{Number of examples}$$

$$p = \text{number of weights (excluding constant term)}$$

Linear Regression: Performance

Look at distribution of residuals: $y_i - f(x_i)$

Should be Gaussian with constant variance

Linear Models for Classification: Logistic “Regression”

Problem Definition

$$\text{Given features} : x_1, x_2, \ldots, x_n$$

$$\text{predict class or label, y} = 0 \text{ or } 1$$

First Stab

Group together all distinct values of tuples$(x_1, x_2, \dots, x_n)$

For each distinct group, count number of entries with y = 0 and y = 1

Now have Prob[y = 0 $|(x_1, x_2, \dots, x_n)$]

$$Prob[y = 0|(x_1, x_2, \dots, x_n)] > P \rightarrow label = 0$$

$$Prob[y = 0|(x_1, x_2, \dots, x_n)] \leq P \rightarrow label = 1$$

P tuning parameter, e.g., 0.5

Problems with First Stab

$x_1, x_2, \ldots, x_n$continuous - can't count!

Solution: bucket/bin continuous variables in intervals

Receive new features:$(x_1, x_2, \ldots, x_n)$
NOT in training set

Have no data on $Prob[y = 0|(x_1, x_2, \ldots, x_n)]$!!!

Solution: Find closest $(x_1, x_2, \ldots, x_n)$that exists in training set
and use it as a proxy

Linear Model

Can we map:

$$(x_1, x_2, \ldots, x_n)$$

$$\downarrow$$

$$Prob[y|(x_1, x_2, \ldots, x_n)]$$

using some other method?

Linear Model

How about:

$$(x_1, x_2, \ldots, x_n)$$

$$\downarrow$$

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Linear combination:

$$\downarrow ?$$

$$[0, 1]$$

Logistic/Sigmoid Function

Want to map real number to [0,1]

One possible solution: $f(x) = \dfrac{1}{1 + e^{-x}}$

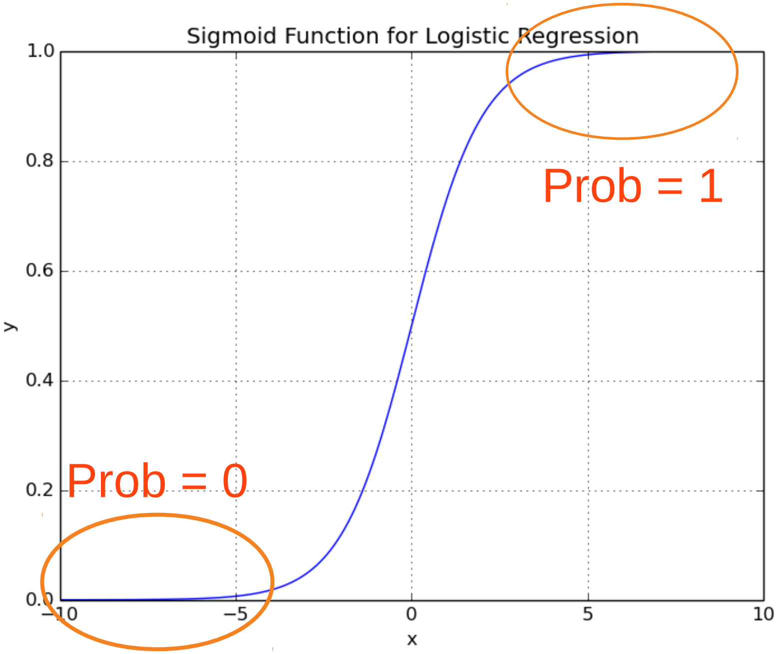
$$x \to \infty, f(x) \to \frac{1}{1+0} = 1$$

$$x \to -\infty, f(x) \to \frac{1}{1+\infty} = 0$$

$$x = 0, f(x) = 0.5$$

Logistic/Sigmoid Function

$$x \to -\infty, f(x) \to \frac{1}{1+\infty} = 0$$



$$x \to \infty, f(x) \to \frac{1}{1+0} = 1$$

Probability

Want to map real number to [0,1]

$$\text{One possible solution: } f(x) = \frac{1}{1+e^{-x}}$$

$x = 0, f(x) = 0.5$ → “Boundary” or “Threshold”

Logistic Regression

$$(x_1, x_2, \ldots, x_n)$$

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Linear combination:

$$f(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n)}}$$

< 0.5

= 0.5

> 0.5

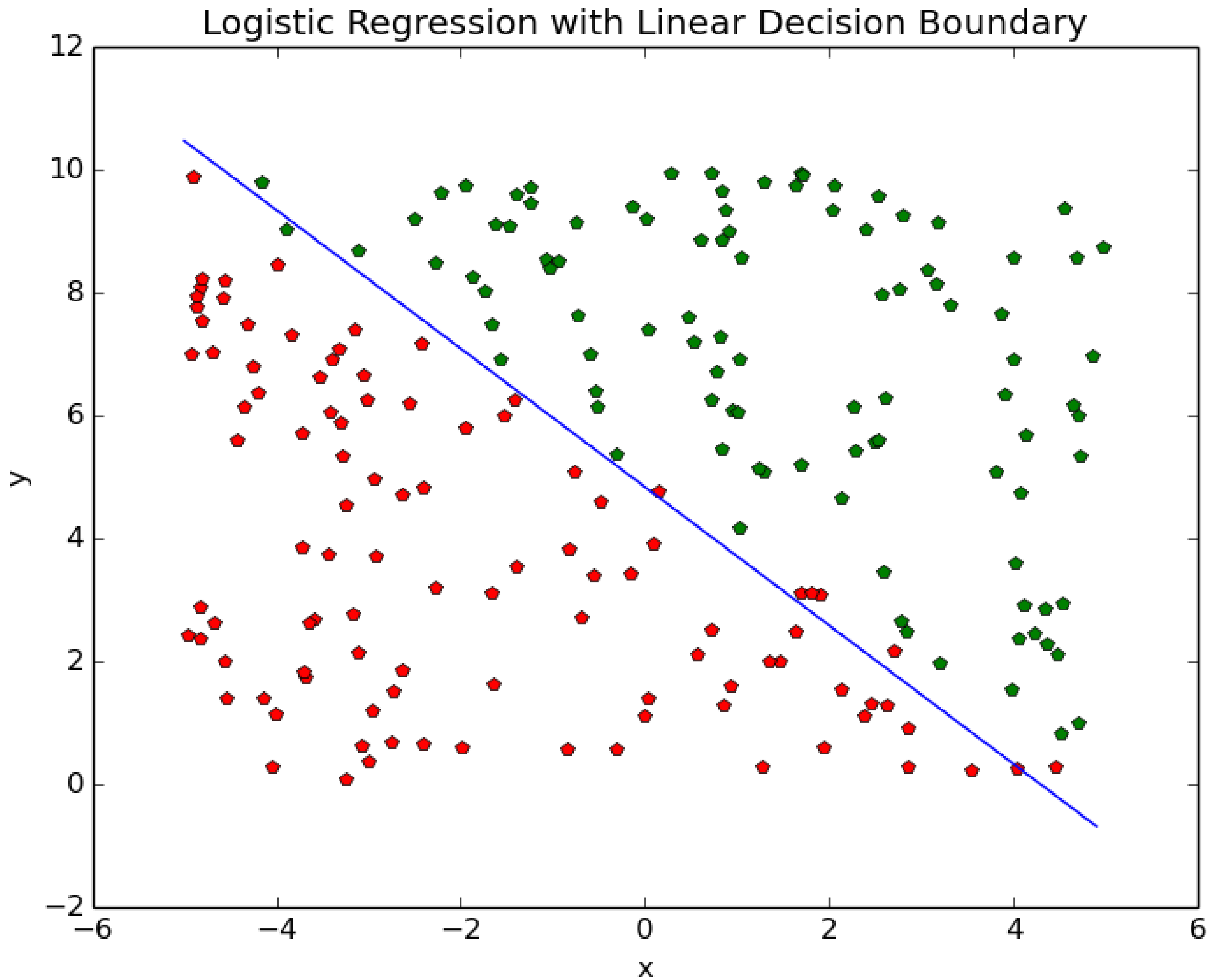
Label 0

Boundary separating classes:

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = 0$$

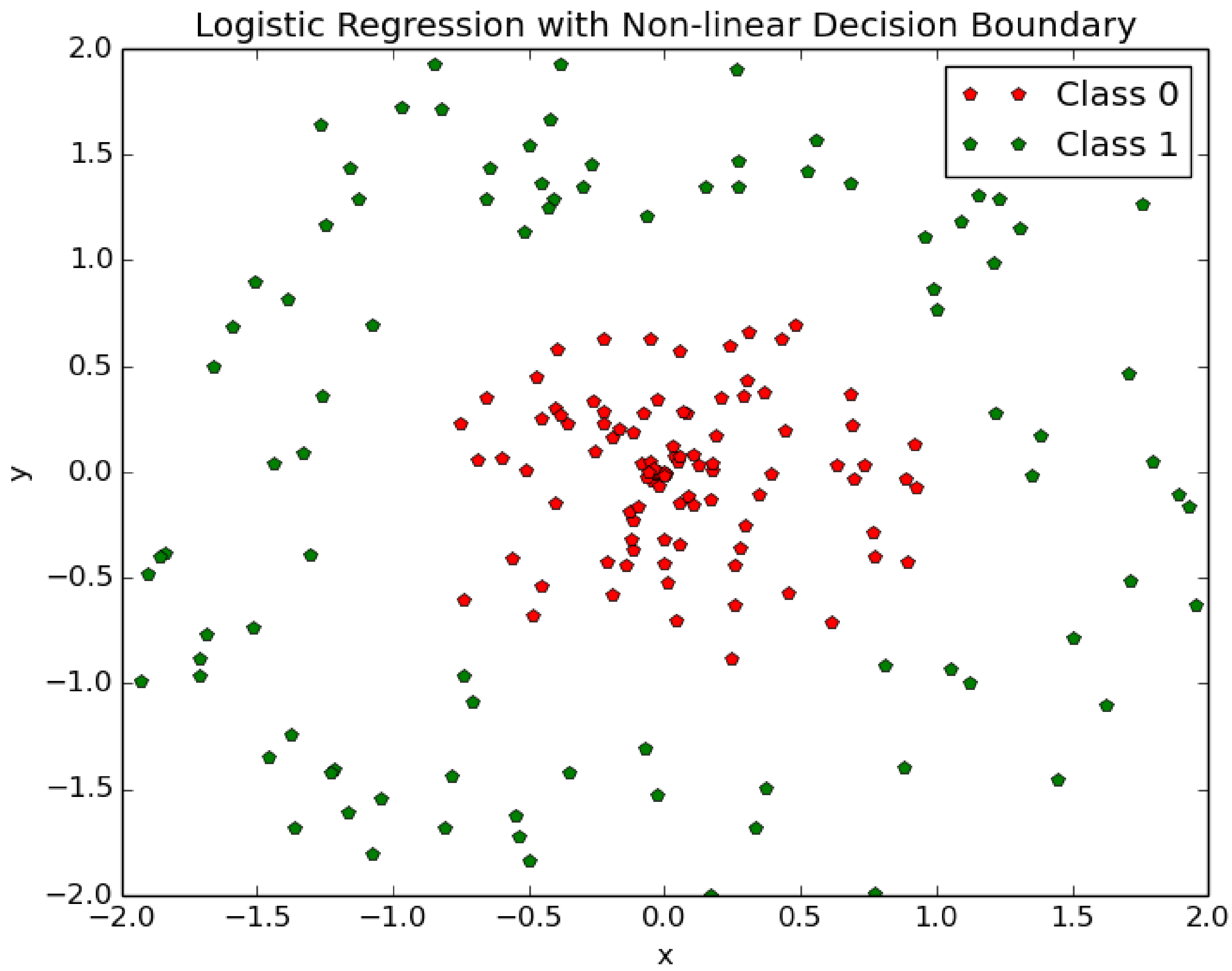
Label 1

Logistic Regression Example 1



$$\text{Decision Boundary} : -4.86 + 1.05x + 1.00y = 0$$

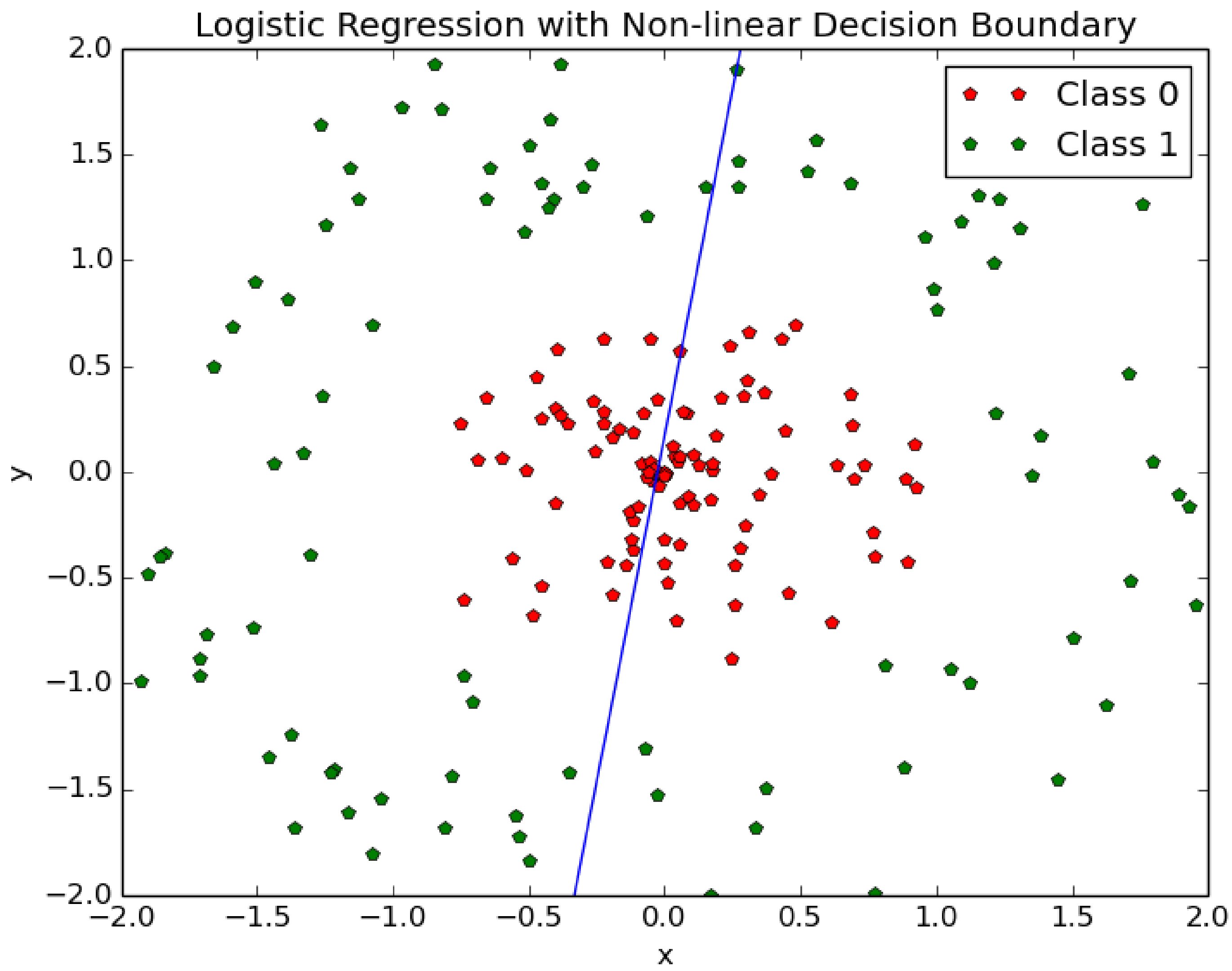
Logistic Regression Example 2



Features are: (x, y)

Accuracy = 64%

Logistic Regression Example 2

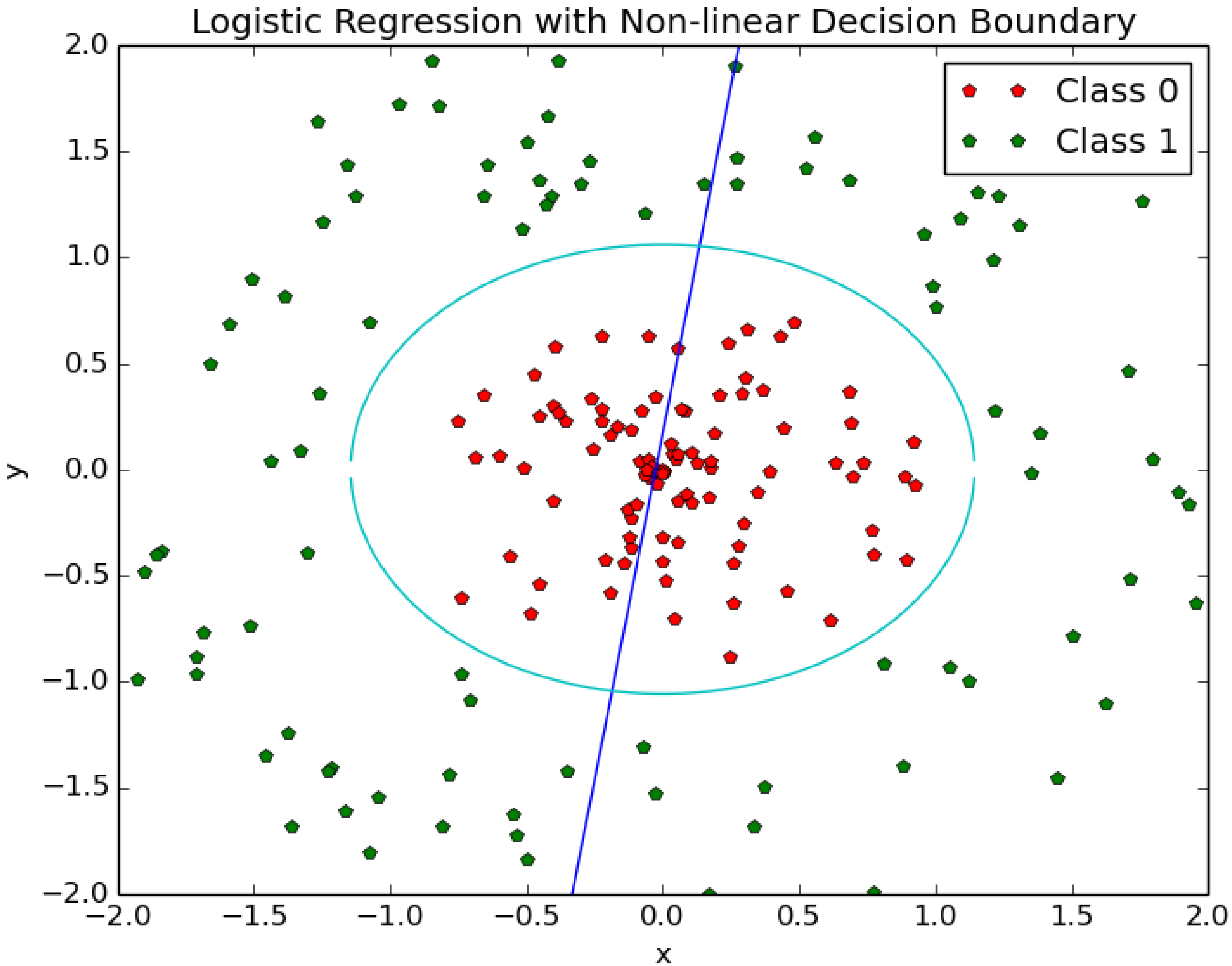


Bad!!!

Decision Boundary: $w_0 + w_1 x + w_2 y = 0 \implies$ Straight Line

Logistic Regression: Example 2

Features are: (x^2, y^2) NOT (x, y)

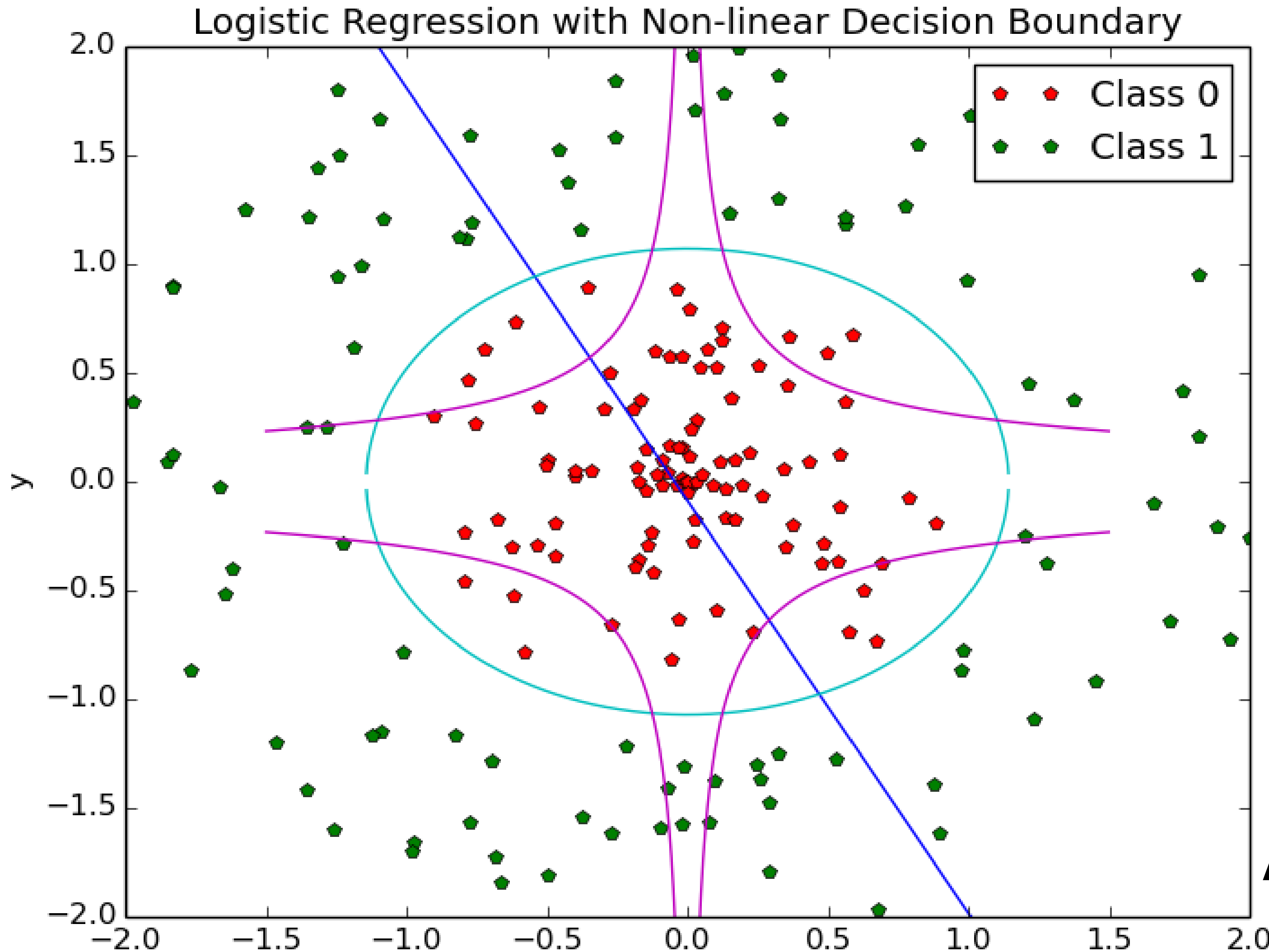


Accuracy = 100%

Decision Boundary: $w_0 + w_1 x^2 + w_2 y^2 = 0 \implies$ Ellipse (in this case at least)

Logistic Regression: Example 2

Features are: $(\log|x|, \log|y|)$ NOT (x, y)



Accuracy = 86%

Decision Boundary: $a\log|x| + b\log|y| + c = 0 \implies y = \pm\frac{e^{-\frac{c}{b}}}{|x|^{\frac{a}{b}}}$

Logistic Regression: Training

Define cost function to minimize:

M training examples

$$C(w_0, w_1, \ldots, w_n) = \Sigma_{i=1}^{M} C_i(w_0, w_1, \ldots, w_n)$$

Total Cost

Cost for each training example

What should we choose for $C_i(w_0, w_1, \ldots, w_n)$?

Logistic Regression: Training

What should we choose for $C_i(w_0, w_1, \ldots, w_n)$?

One choice: $C_i = (y_i - p_i)^2$

$$y_i = \text{Known class of example i} \in \{0, 1\}$$

$$p_i = \text{Predicted class of example i} \in \{0, 1\}$$

Could also try:

$$p_i = \text{Predicted probability of example i} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_n x_n)}}$$

Logistic Regression: Training

$$\text{One choice: } C_i = (y_i - p_i)^2$$

C_i	$y_i = 0$	$y_i = 1$
$p_i = 0$	0	1
$p_i = 1$	1	0

Bounded cost even when result wrong

Unlike linear regression, in logistic regression, the result is binary: right or wrong.

Want to impose high cost for wrong and no cost for right.

Let's try something else.

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

where:

$$y_i = \text{Known class of example i} \in \{0, 1\}$$

$$p_i = \text{Predicted probability of example i} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_n x_n)}}$$

Why this C_i?!!!

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$y_i = 0 \qquad\qquad p_i = 0$$

$$\boxed{C_i = 0?}$$

Logistic Regression: Training

Different Choice: $C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$

$y_i = 0$ $\qquad$ $p_i = 0$

$$\boxed{C_i = 0?}$$

$$C_i = -0 \log 0 - (1 - 0) \log(1 - 0)$$
$$= 0 - \log 1 = \boxed{0}$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$y_i = 1 \qquad\qquad p_i = 1$$

$$\boxed{C_i = 0?}$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$y_i = 1 \qquad\qquad p_i = 1$$

$$\boxed{C_i = 0?}$$

$$C_i = -1 \log 1 - (1 - 1) \log(1 - 1)$$

$$= -\log 1 - 0 \log 0 = \boxed{0}$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$y_i = 1 \qquad\qquad p_i = 0$$

$$\boxed{C_i = \text{large?}}$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$y_i = 1 \qquad\qquad p_i = 0$$

$$\boxed{C_i = \text{large?}}$$

$$C_i = -1 \log 0 - (1 - 1) \log (1 - 0)$$

$$\log 0 = -\infty$$

$$\boxed{C_i = \infty}$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$y_i = 0 \qquad\qquad p_i = 1$$

$$\boxed{C_i = \text{large?}}$$

Logistic Regression: Training

Different Choice: $C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$

$y_i = 0 \qquad\qquad p_i = 1$

$$\boxed{C_i = \text{large?}}$$

$$C_i = -0 \log 1 - (1 - 0) \log (1 - 1)$$

$$\log 0 = -\infty$$

$$\boxed{C_i = \infty}$$

Logistic Regression: Training

$$\text{Different Choice: } C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

Takes values between 0 and 1. Only extreme cases shown.

C_i	$y_i = 0$	$y_i = 1$
$p_i = 0$	0	∞
$p_i = 1$	∞	0

Unbounded cost when result wrong

More appropriate for binary classification

Cost function measures "entropy"

Also, equivalent to negative log likelihood

Logistic Regression: Training

$$\underline{\text{Minimize}}:$$

$$C(w_0, w_1, \ldots, w_n) = \Sigma_{i=1}^{M} C_i(w_0, w_1, \ldots, w_n)$$

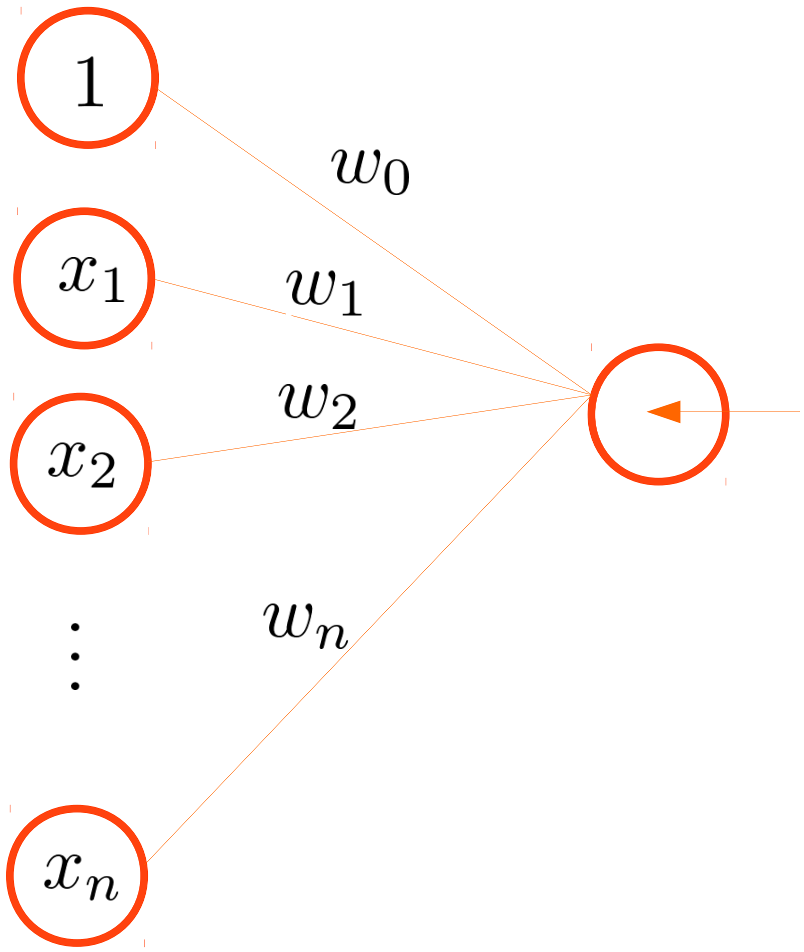
$$C_i = -y_i \log p_i - (1 - y_i) \log (1 - p_i)$$

$$p_i = \text{Predicted probability of example i} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \ldots + w_n x_n)}}$$

$$\underline{\text{with respect to}}:$$

$$w_0, w_1, \ldots, w_n$$

Logistic Regression: Visualization



$$\frac{1}{1+e^{-(w_0+w_1x_1+\ldots+w_nx_n)}}$$

This kind of structure will occur again when we look at neural networks

Logistic Regression: Python Code

```
from sklearn import linear_model

model = linear_model.LogisticRegression()
model.fit(features, labels) #train the model

model.predict(test_features) #make predictions (0/1)
model.predict_proba(test_features) #predict probabilities
```

Logistic Regression: Performance

Have data with binary labels, built a logistic regression model.

How good is the model? How to evaluate its performance?

Logistic Regression: Performance

C_i	$pred = 0$	$pred = 1$
$label = 0$	$0 \rightarrow 0$	$0 \rightarrow 1$
$label = 1$	$1 \rightarrow 0$	$1 \rightarrow 1$

Good

Bad

$$=$$

C_i	$pred = 0$	$pred = 1$
$label = 0$	True Negatives	False Positives
$label = 1$	False Negatives	True Positives

Logistic Regression: Performance

Some metrics to measure on the test set

$$\frac{(0 \to 0) + (1 \to 1)}{(0 \to 0) + (1 \to 1) + (0 \to 1) + (1 \to 0)}$$

$$=$$

$$\frac{\text{examples predicted correctly}}{\text{total examples}}$$

known as **Accuracy**

Logistic Regression: Performance

$$\text{Accuracy} = \frac{(0 \to 0) + (1 \to 1)}{(0 \to 0) + (1 \to 1) + (0 \to 1) + (1 \to 0)}$$

What if 98% of examples are labeled 0 and 2% labeled 1?

Build "dumb" model that predicts 0 for every examples

Accuracy = 98% (Get everything labeled 0 correct)

Bad measure if population of labeled classes very uneven

Logistic Regression: Performance

$$\text{Sensitivity/Recall} = \frac{(1 \to 1)}{(1 \to 1) + (1 \to 0)}$$

$$\text{Ideally, Recall} = 1$$

$$\text{In previous example, Recall } = 0$$

Logistic Regression: Performance

$$\text{Sensitivity/Recall} = \frac{(1 \to 1)}{(1 \to 1) + (1 \to 0)}$$

$$\text{Specificity} = \frac{(0 \to 0)}{(0 \to 0) + (0 \to 1)}$$

Class labels arbitrary

Symmetry under: $0 \leftrightarrow 1$

Sensitivity/Recall $\leftrightarrow$ Specificity

Ideally, both are 1

Logistic Regression: Performance

$$\text{type I error} = \frac{(0 \to 1)}{(0 \to 1) + (0 \to 0)}$$

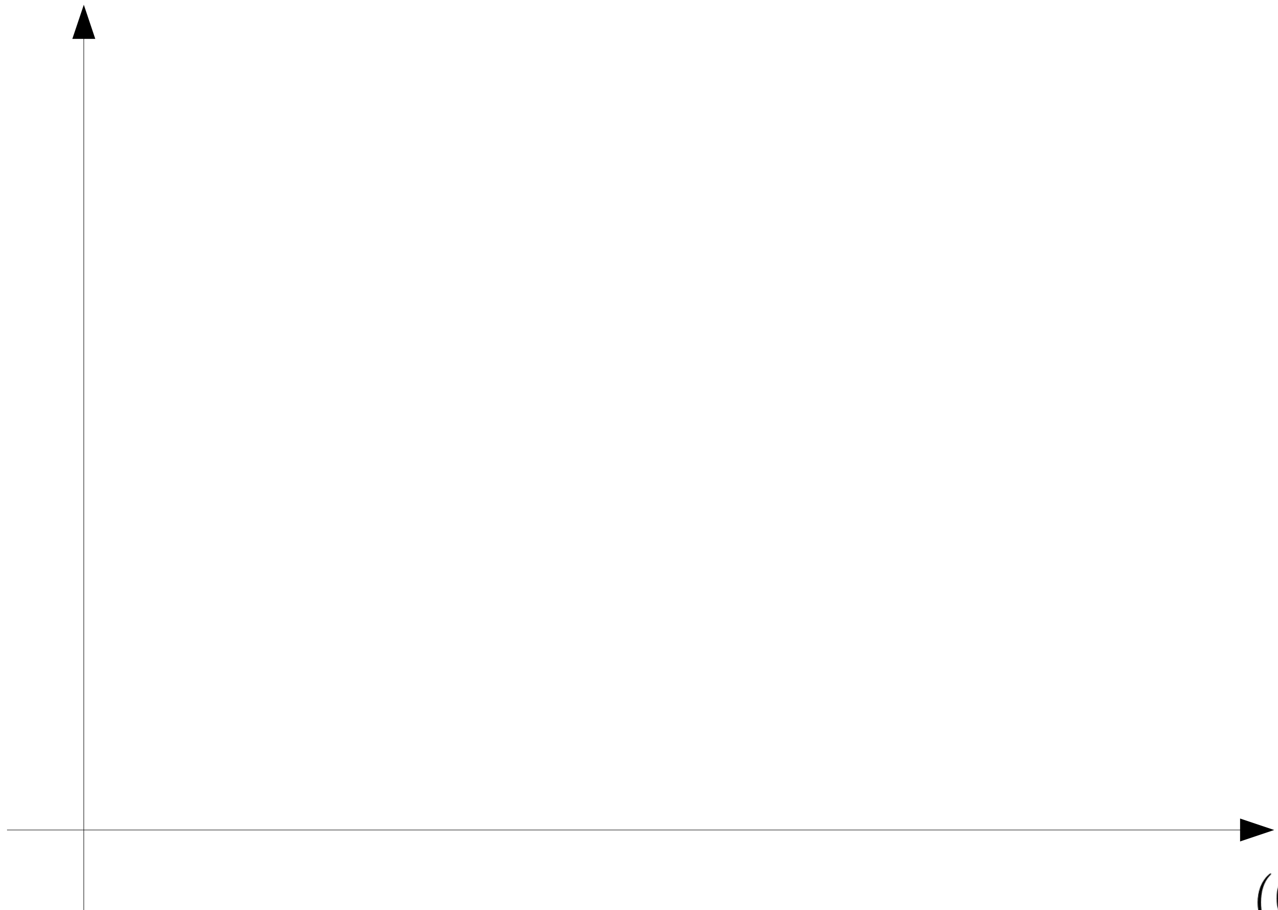
$$0 \leftrightarrow 1 \updownarrow$$

$$\text{type II error} = \frac{(1 \to 0)}{(1 \to 0) + (1 \to 1)}$$

Ideally, both are 0

Logistic Regression: Performance

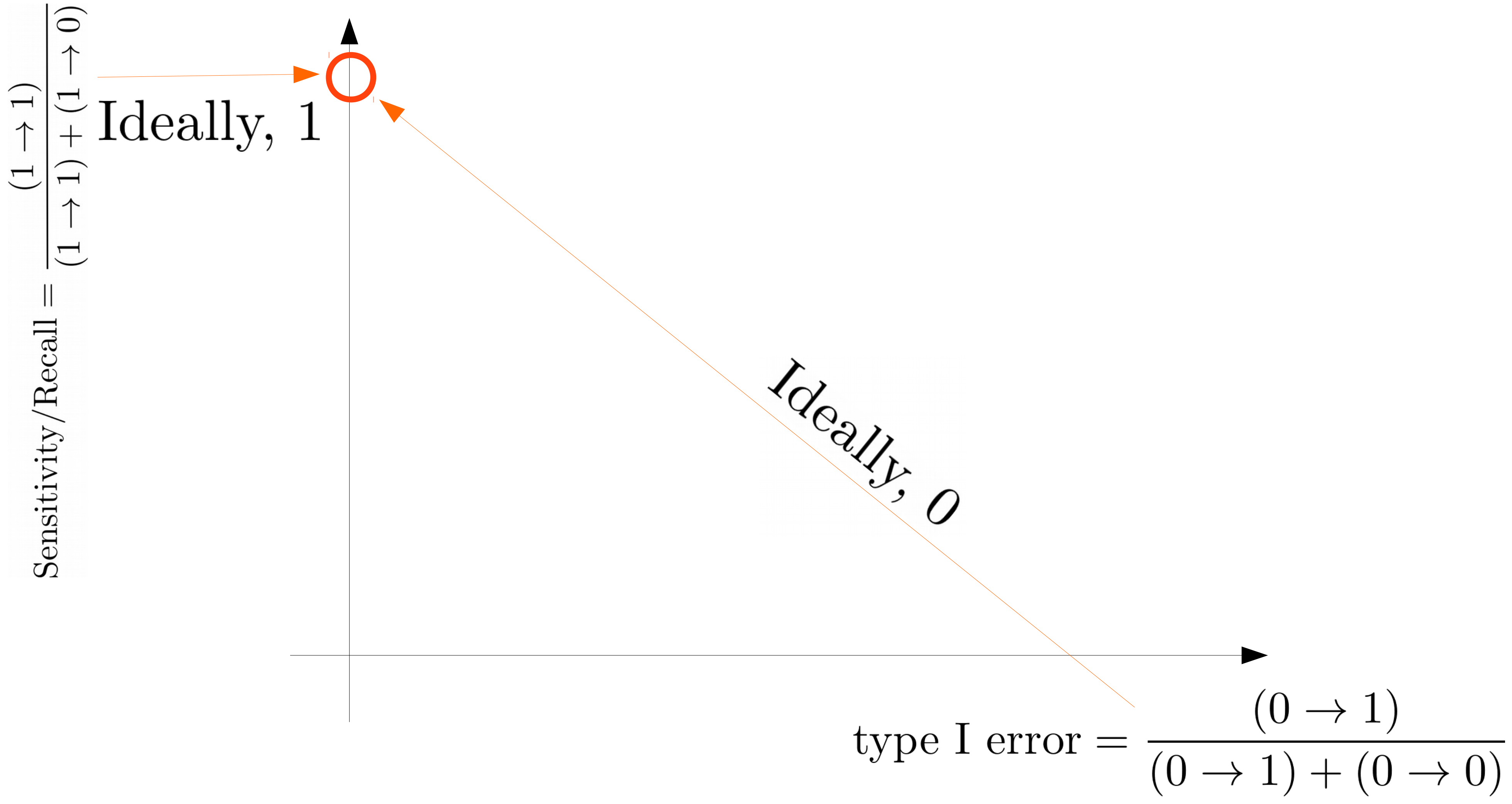
$$\text{Sensitivity/Recall} = \frac{(1 \to 1)}{(1 \to 1) + (1 \to 0)}$$



$$\text{type I error} = \frac{(0 \to 1)}{(0 \to 1) + (0 \to 0)}$$

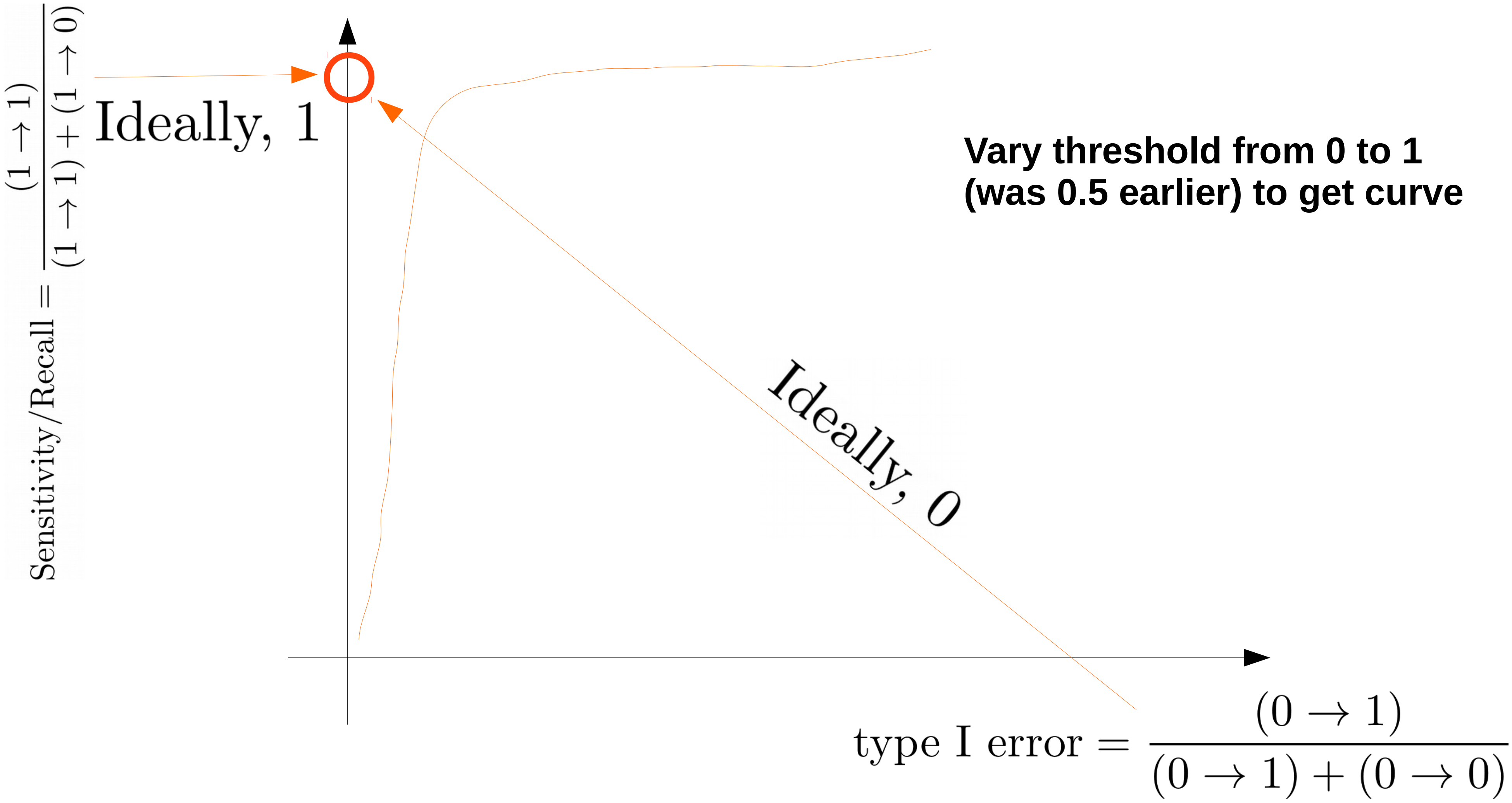
Receiver Operating Characteristic (ROC) Curve

Logistic Regression: Performance



Receiver Operating Characteristic (ROC) Curve

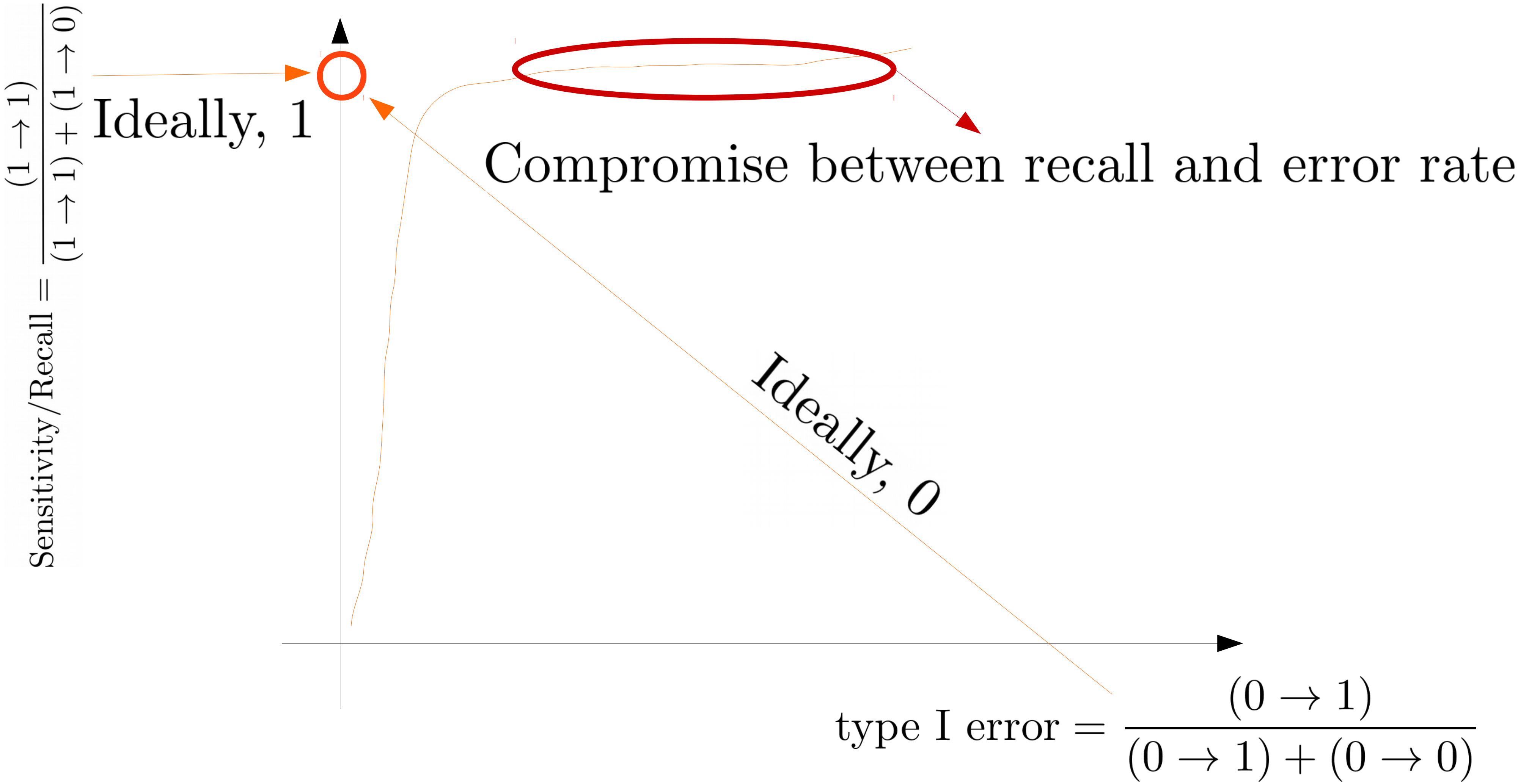
Logistic Regression: Performance



Vary threshold from 0 to 1 (was 0.5 earlier) to get curve

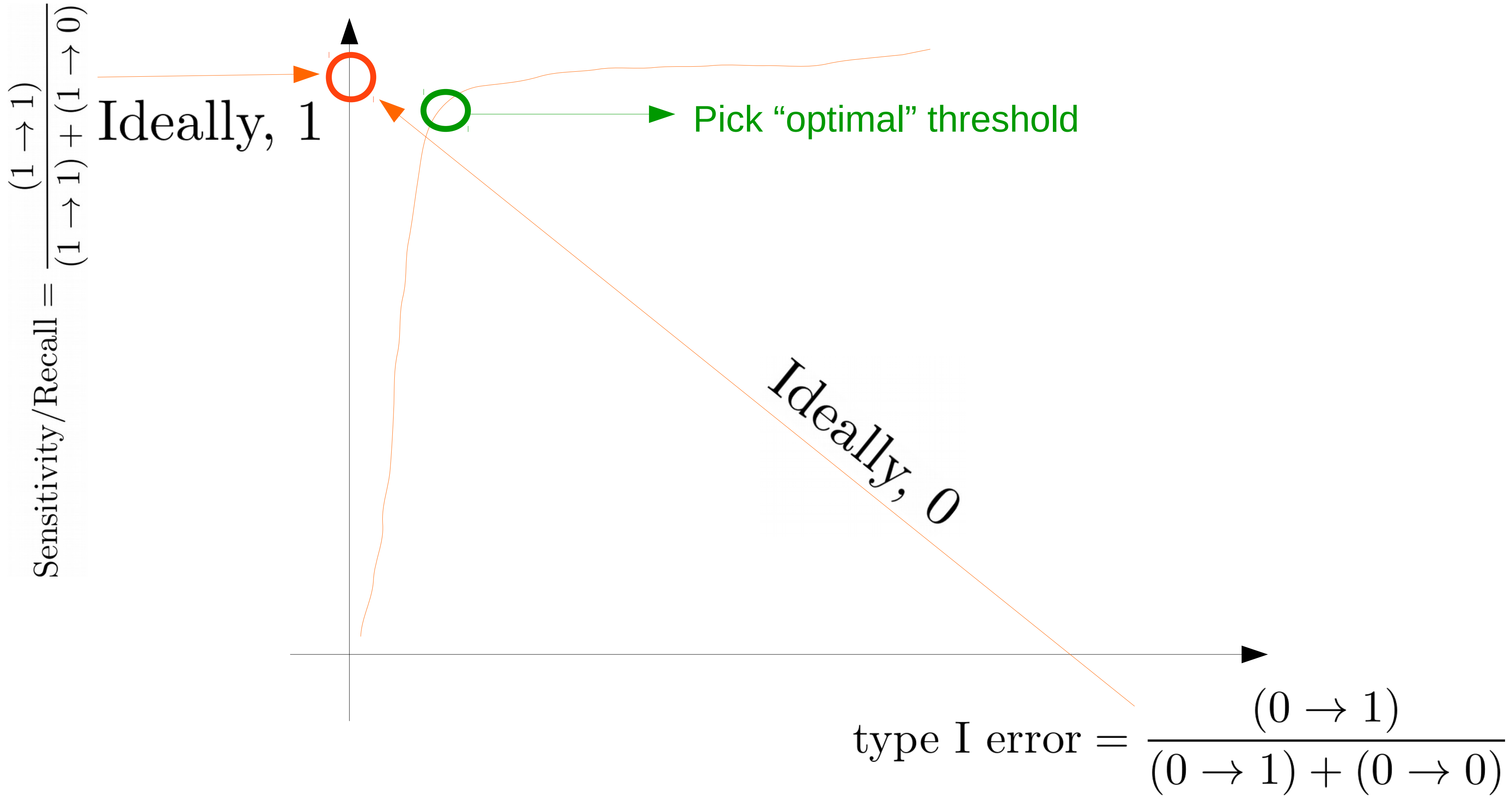
Receiver Operating Characteristic (ROC) Curve

Logistic Regression: Performance



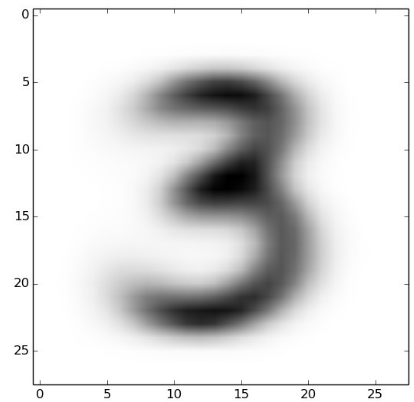
Receiver Operating Characteristic (ROC) Curve

Logistic Regression: Performance



Receiver Operating Characteristic (ROC) Curve

Logistic Regression: Example 3



Hand-written digits dataset: 28 x 28 pixels

Let's try logistic regression to identify the digit 3 (label $= 1$) from other digits (label $= 0$)

Logistic Regression: Example 3

$$\text{Test Set: \# of examples with label } 1 = 1,301$$

$$\text{Test Set: \# of examples with label } 0 = 11,299$$

Assign every test example to label $= 0$:

$$\text{Accuracy} = \frac{11,299}{1,301 + 11,299} = 89.7\%$$

Logistic Regression:

$$\text{Accuracy} = \frac{12,229}{12,600} = 97.1\%$$

Logistic Regression: Example 3

$$\text{Sensitivity/Recall} = \frac{(1 \to 1)}{(1 \to 1) + (1 \to 0)}$$

Assign every test example to label $= 0$:

$$\text{Sensitivity/Recall} = \frac{0}{0 + 1301} = 0$$

Logistic Regression:

$$\text{Sensitivity/Recall} = \frac{1,077}{1,077 + 224} = 82.7\%$$

Logistic Regression: Example 3

$$\text{Specificity} = \frac{(0 \to 0)}{(0 \to 0) + (0 \to 1)}$$

Assign every test example to label $= 0$:

$$\text{Specificity} = \frac{11,299}{11,299 + 0} = 100\%$$

Logistic Regression:

$$\text{Specificity} = \frac{11,152}{11,152 + 147} = 98.7\%$$

Logistic Regression: Example 3

$$\text{type I error} = \frac{(0 \to 1)}{(0 \to 1) + (0 \to 0)}$$

Assign every test example to label $= 0$:

$$\text{type I error} = \frac{0}{0 + 11,299} = 0\%$$

Logistic Regression:

$$\text{type I error} = \frac{147}{147 + 11,152} = 1.3\%$$

Logistic Regression: Example 3

$$\text{type II error} = \frac{(1 \to 0)}{(1 \to 0) + (1 \to 1)}$$

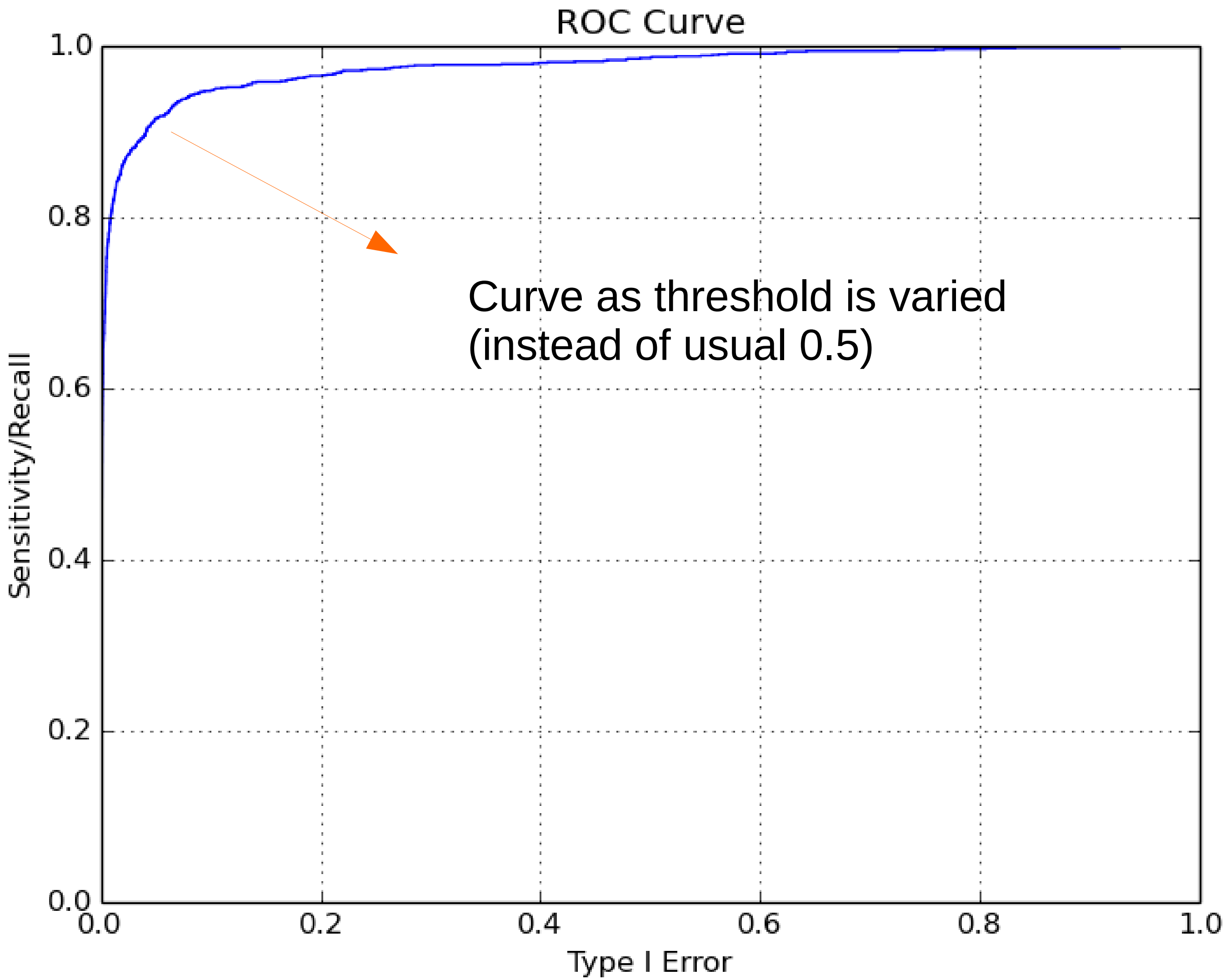
Assign every test example to label $= 0$:

$$\text{type II error} = \frac{1,301}{1,301 + 0} = 100\%$$

Logistic Regression:

$$\text{type II error} = \frac{224}{224 + 1,077} = 17.2\%$$

Logistic Regression: Example 3



$$\text{Area under curve, AUC} = 0.975$$

Questions?

Questions?