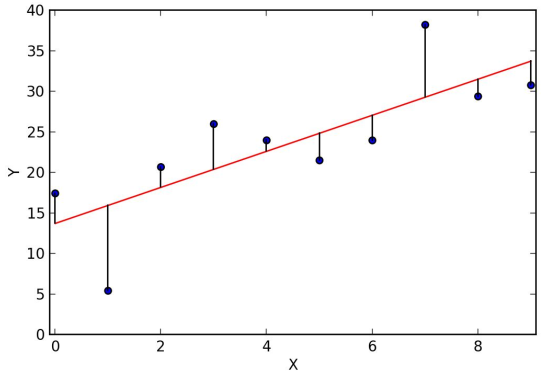
From Linear Modeling to Machine Learning

Statistics for Data Science
CSE357 - Fall 2021

Linear Models

- Ridge Regression (L2 Penalized)
- Lasso Regression (L1 Penalized)
- Feature Selection
- Accuracy Metrics



Linear Regression

Finding a linear function based on *X* to best yield *Y*.

X = "covariate" = "feature" = "predictor" = "regressor" = "independent variable"

Y = "response variable" = "outcome" = "dependent variable"

Linear Regression

Finding a linear function based on *X* to best yield *Y*.

X = “covariate” = “feature” = “predictor” = “regressor” = “independent variable”

Y = “response variable” = “outcome” = “dependent variable”

Regression: $$r(x) = \mathrm{E}(Y|X = x)$$

goal: estimate the function r

Linear Regression

Finding a linear function based on *X* to best yield *Y*.

X = “covariate” = “feature” = “predictor” = “regressor” = “independent variable”

Y = “response variable” = “outcome” = “dependent variable”

Regression: $r(x) = \mathrm{E}(Y|X = x)$

goal: estimate the function r

The **expected** value of *Y*, given that the random variable *X* is equal to some specific value, x.

Linear Regression

Finding a linear function based on *X* to best yield *Y*.

X = "covariate" = "feature" = "predictor" = "regressor" = "independent variable"

Y = "response variable" = "outcome" = "dependent variable"

Regression: $r(x) = \mathrm{E}(Y|X = x)$

goal: estimate the function r

Linear Regression (univariate version): $r(x) = \beta_0 + \beta_1 x$

goal: find β_0, β_1 such that $r(x) \approx \mathrm{E}(Y|X = x)$

Review: Multiple Linear Regression

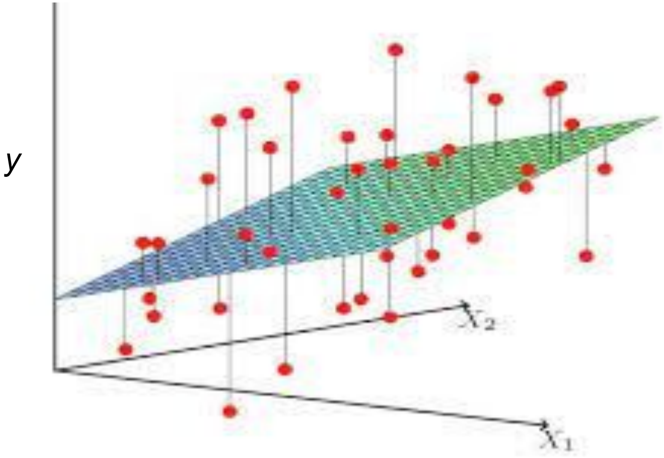
$$Y_i = \sum_{j=0}^{m} \beta_j X_{ij} + \epsilon_i$$

Direct Estimates (Normal Equation)

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

$$\hat{Y}_i = \hat{r}(X_i)$$

Residual: $\hat{\epsilon}_i = Y_i - \hat{Y}_i$



Review: Logistic Regression

What if $Y_i \in \{0, 1\}$? (i.e. we want "classification")

$$p_i \equiv p_i(\beta) \equiv \mathbf{P}(Y_i = 1 | X = x) = \frac{e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}$$

$$logit(p_i) = log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}$$

$P(Y_i = 0 \mid X = x)$

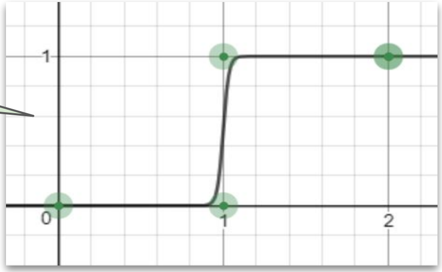
Thus, 0 is class 0

and 1 is class 1.

Logistic Regression with Multiple Feats

Often we want to make a classification based on multiple features:

Y-axis is Y (i.e. 1 or 0)

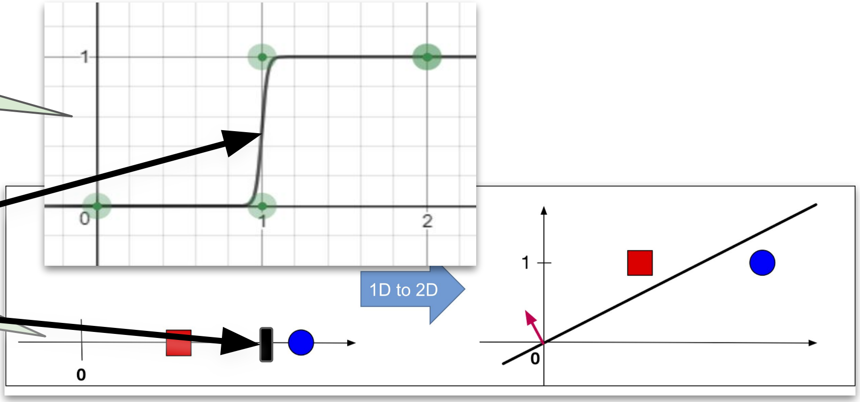


Logistic Regression with Multiple Feats

Often we want to make a classification based on multiple features:

Y-axis is Y (i.e. 1 or 0)

To make room for multiple Xs, let's get rid of y-axis. Instead, show **decision point**.



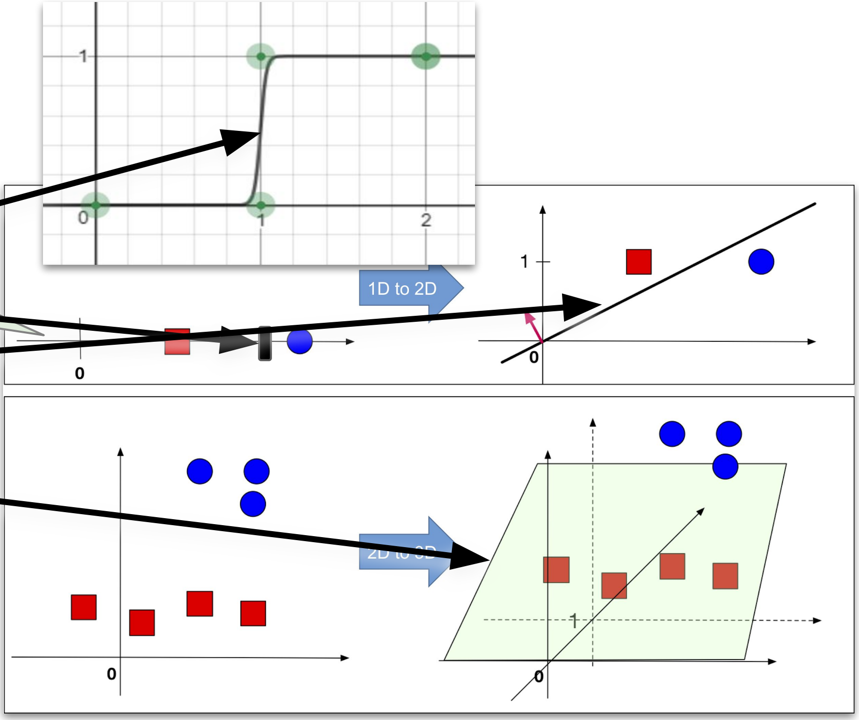
Logistic Regression with Multiple Feats

Often we want to make a classification based on multiple features:

To make room for multiple Xs, let's get rid of y-axis. Instead, show **decision point**.

decision line

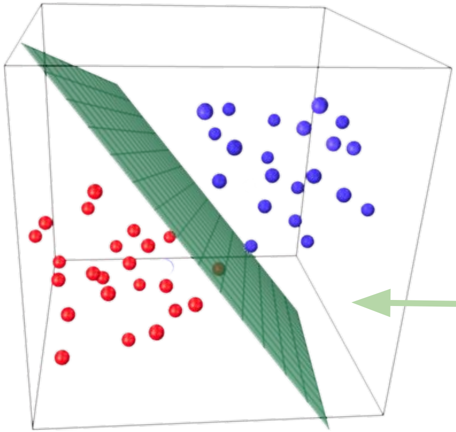
decision hyperplane



Review: Logistic Regression

What if $Y_i \in \{0, 1\}$? (i.e. we want "classification")

$$p_i \equiv p_i(\beta) \equiv \mathbf{P}(Y_i = 1 | X = x) = \frac{e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}$$



$$logit(p_i) = log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}$$

We're learning a linear (i.e. flat) *separating hyperplane*, but fitting it to a *logit* outcome.

(https://www.linkedin.com/pulse/predicting-outcomes-probabilities-logistic-regression-konstantinidis/)

Logistic Regression: Faster Approach

What if $Y_i \in \{0, 1\}$? (i.e. we want "classification")

$$p_i \equiv p_i(\beta) \equiv \mathbf{P}(Y_i = 1|X = x) = \frac{e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}}}$$

$$logit(p_i) = log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + \sum_{j=1}^{m} \beta_j x_{ij}$$

To estimate β, one can use *reweighted least squares:*

(Wasserman, 2005; Li, 2010)

set $\hat{\beta}_0 = ... = \hat{\beta}_m = 0$ (remember to include an intercept)

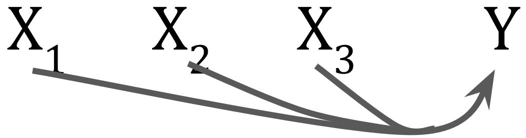
1. Calculate p_i and let W be a diagonal matrix where element$(i, i) = p_i(1 - p_i)$.
2. Set $z_i = logit(p_i) + \frac{Y_i - p_i}{p_i(1 - p_i)} = X\hat{\beta} + \frac{Y_i - p_i}{p_i(1 - p_i)}$
3. Set $\hat{\beta} = (X^T W X)^{-1} X^T W z$ //weighted lin. reg. of Z on Y.
4. Repeat from 1 until $\hat{\beta}$ converges.

Supervised Machine Learning

(genes) (health)

X_1 X_2 X_3 Y

Supervised Machine Learning



Supervised Machine Learning

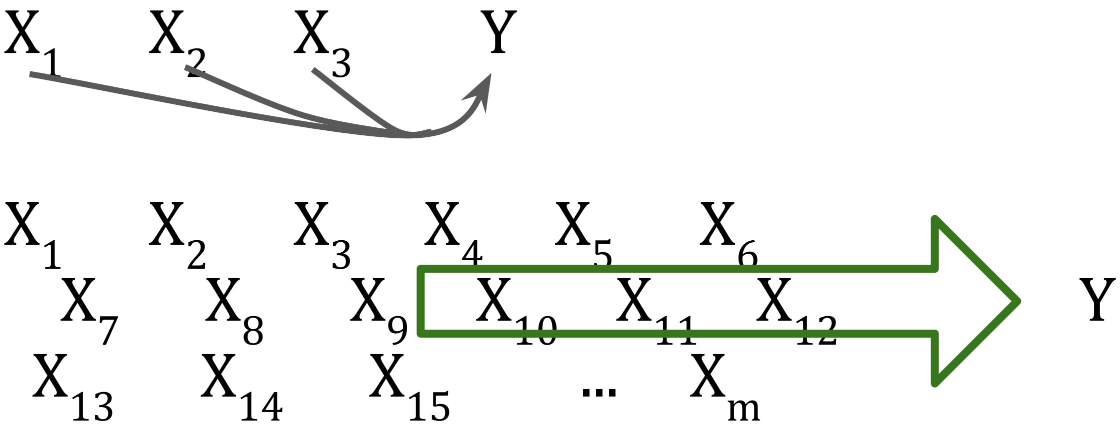
X_1 X_2 X_3 Y

X_1 X_2 X_3 X_4 X_5 X_6

X_7 X_8 X_9 X_{10} X_{11} X_{12} Y

X_{13} X_{14} X_{15} ... X_m

Supervised Machine Learning

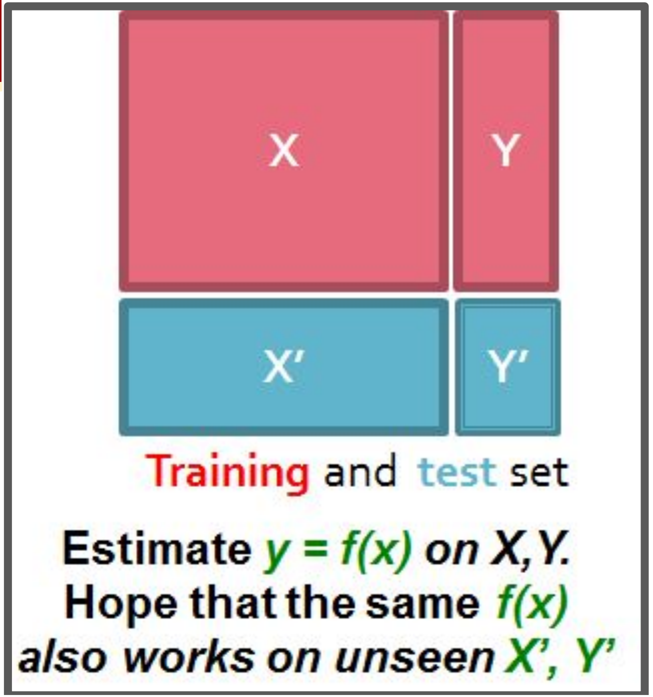


Task: Determine a function, f (or parameters to a function) such that $f(X) = Y$

Supervised Learning

$X_1 \quad X_2 \quad X_3 \qquad Y$

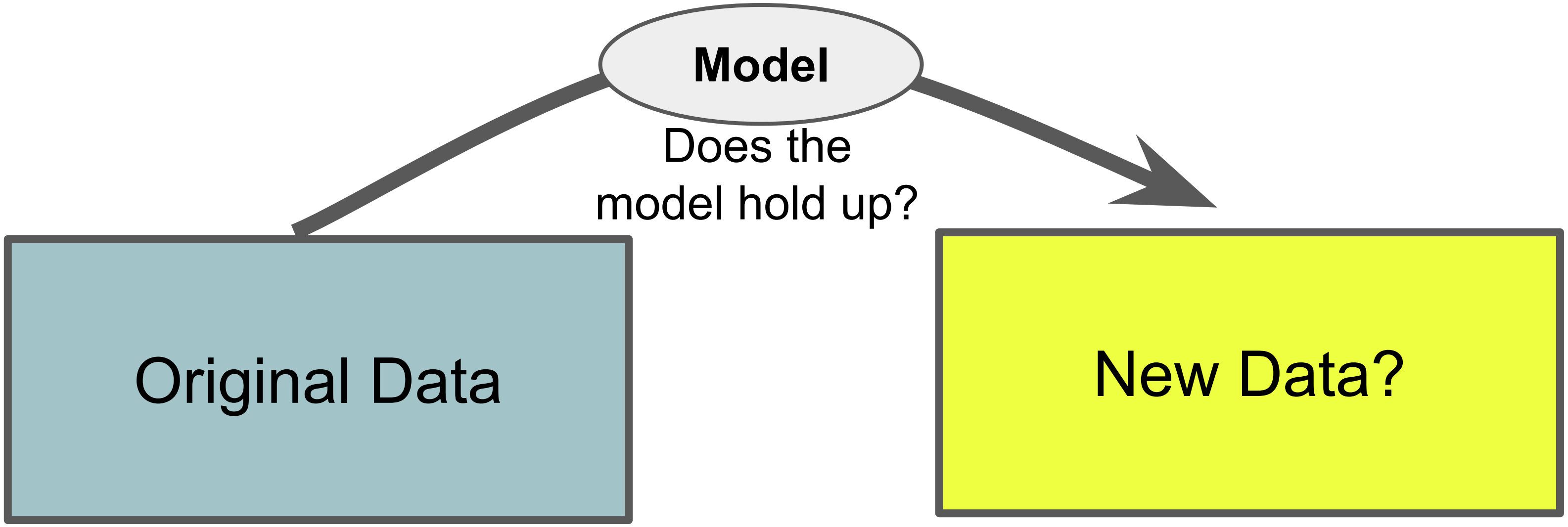
$X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5 \quad X_6$
$X_7 \quad X_8 \quad X_9 \quad X_{10} \quad X_{11} \quad X_{12}$
$X_{13} \quad X_{14} \quad X_{15} \quad \ldots \quad X_m$



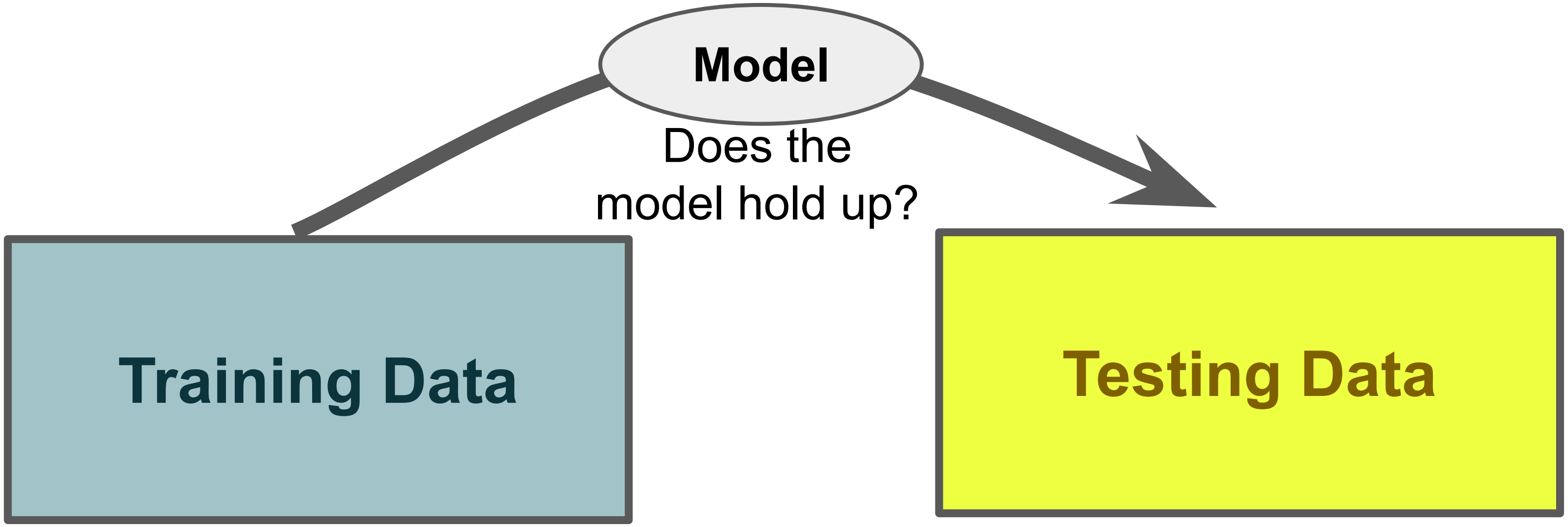
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Task: Determine a function, f (or parameters to a function) such that $f(X) = Y$

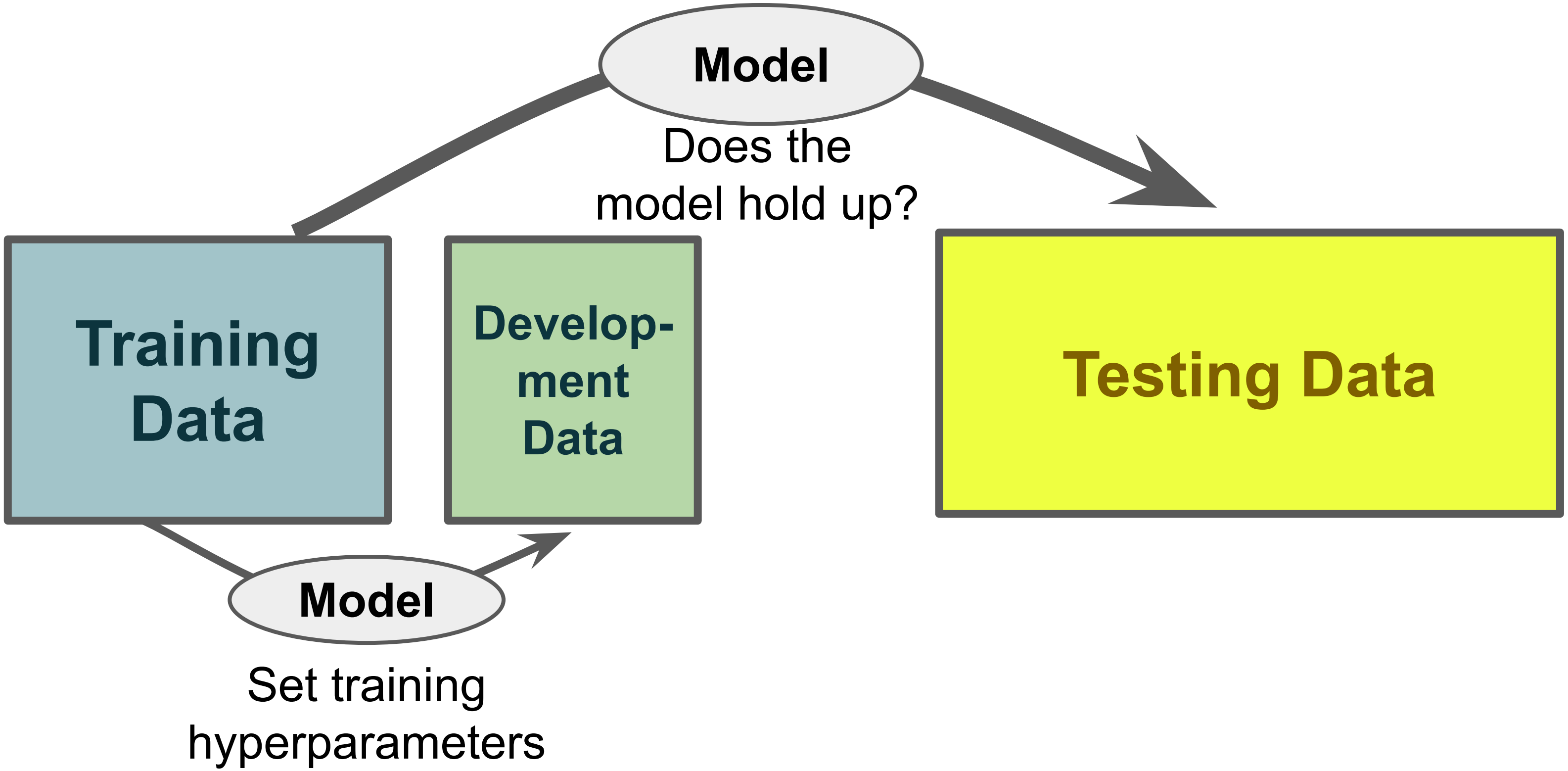
How to Evaluate?



How to Evaluate?

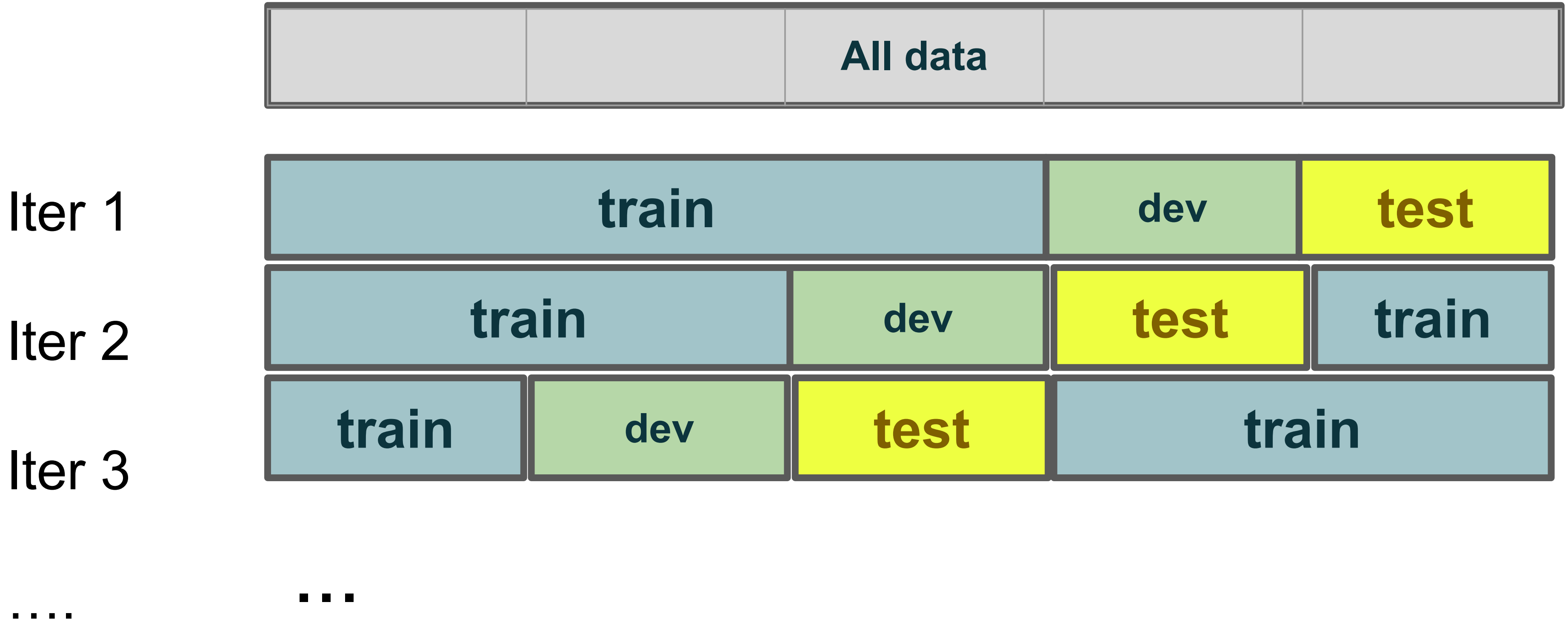


ML: GOAL


Model
Does the
model hold up?
Training
Data
Develop-
ment
Data
Testing Data
Model
Set training
hyperparameters

N-Fold Cross Validation

Goal: Decent estimate of model accuracy



Logistic Regression - Regularization

$$X = Y$$

							Y
0.5	0	0.6	1	0	0.25		1
0	0.5	0.3	0	0	0		1
0	0	1	1	1	0.5		0
0	0	0	0	1	1		0
0.25	1	1.25	1	0.1	2		1

Logistic Regression - Regularization

$$X \quad = \quad Y$$

0.5	0	0.6	1	0	0.25
0	0.5	0.3	0	0	0
0	0	1	1	1	0.5
0	0	0	0	1	1
0.25	1	1.25	1	0.1	2

1
1
0
0
1

Logistic Regression - Regularization

$$X \qquad = \qquad Y$$

0.5	0	0.6	1	0	0.25		1
0	0.5	0.3	0	0	0		1
0	0	1	1	1	0.5		0
0	0	0	0	1	1		0
0.25	1	1.25	1	0.1	2		1

$$1.2 + -63 * x_1 + 179 * x_2 + 71 * x_3 + 18 * x_4 + -59 * x_5 + 19 * x_6 = logit(Y)$$

Logistic Regression - Regularization

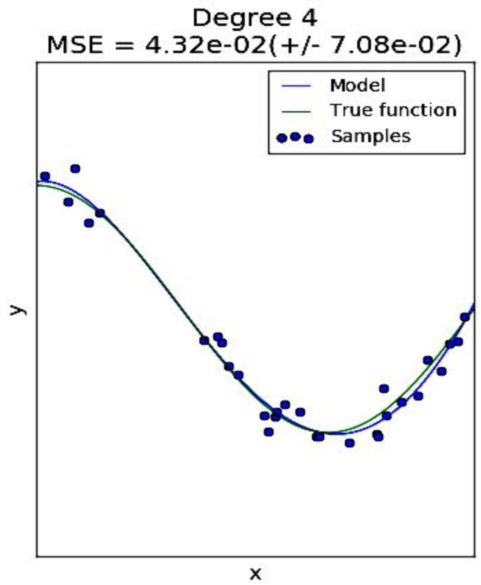
$$X = Y$$

							Y
	0.5	0	0.6	1	0	0.25	1
	0	0.5	0.[illegible]	0	[illegible]	0	1
	0	0	[illegible]	[illegible]	[illegible]	0.5	0
	0	0	0	0	1	1	0
colab	0.25	1	1.25	1	0.1	2	1

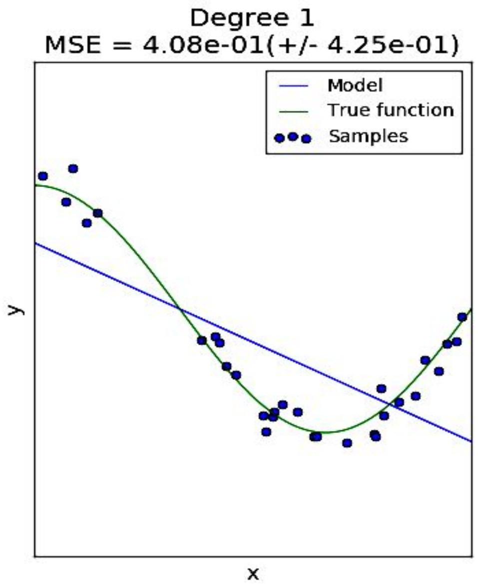
“overfitting”

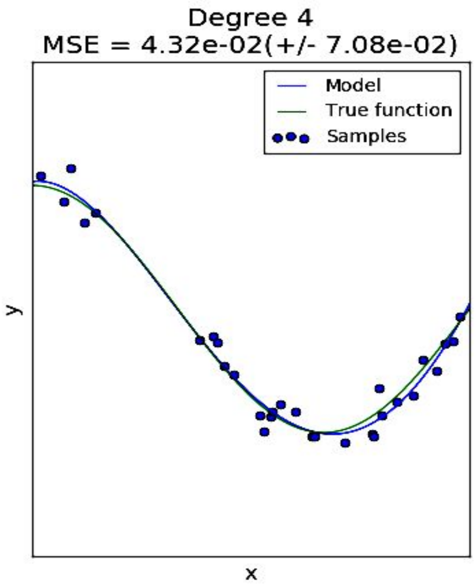
$$1.2 + -63*x_1 + 179*x_2 + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

Overfitting (1-d example)



Overfitting (1-d example)

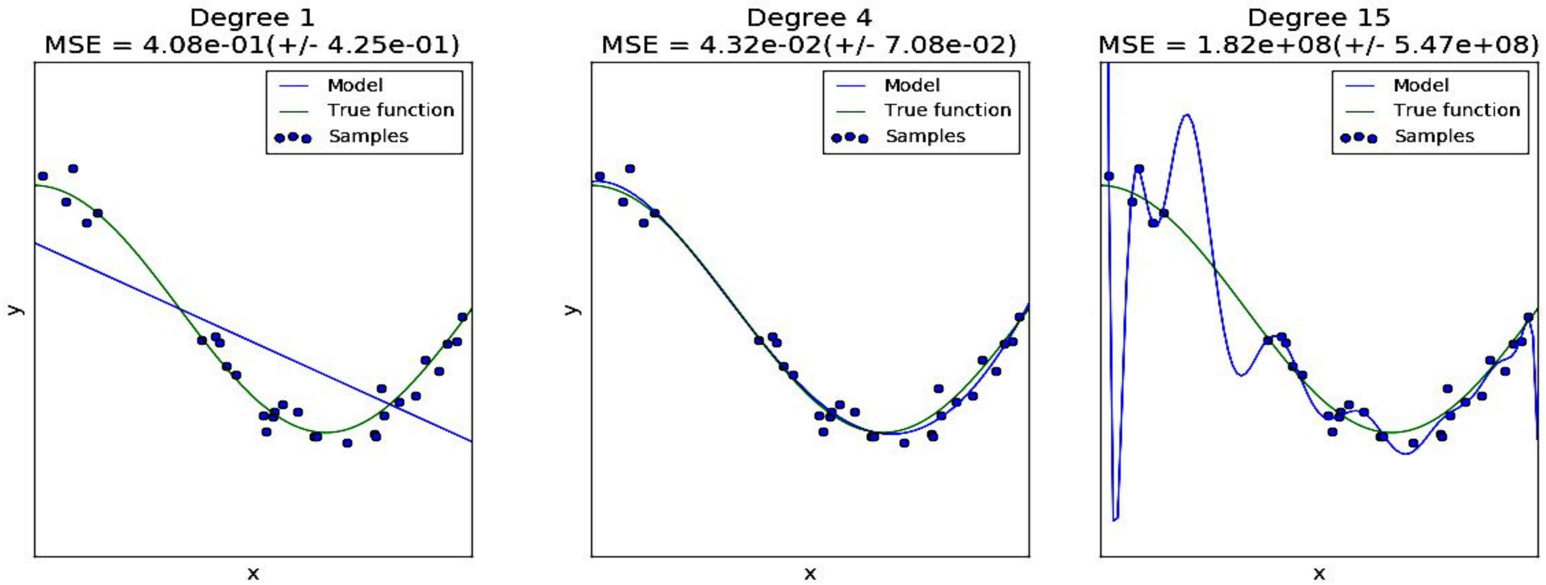




Underfit

(image credit: Scikit-learn; in practice data are rarely this clear)

Overfitting (1-d example)



Underfit

Overfit

(image credit: Scikit-learn; in practice data are rarely this clear)

Overfitting (multidimenstional example)

$$X \qquad = \qquad Y$$

							Y
0.5	0	0.6	1	0	0.25		1
0	0.5	[illegible]	0	[illegible]	0		1
0	0	[illegible]	[illegible]	[illegible]	0.5		0
0	0	0	0	1	1		0
0.25	1	1.25	1	0.1	2		1

"overfitting"

$$1.2 + -63*x_1 + 179*x_2 + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$$

Overfitting (multidimenstional example)

$X \quad = \quad Y$

0.5	0	0.6	[illegible]	[illegible]	[illegible]	1
0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1
0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0
[illegible]	[illegible]	[illegible]	[illegible]	0.1	2	1

$1.2 + -63*x_1 + 17[illegible]*x_2 + 71*x_3 + 18*x_4 + -59*x_5 + 19*x_6 = logit(Y)$

"overfitting": generally due to trying to fit too many features given the number of observations.

Overfitting (multidimenstional example)

X

x_1	x_2
0.5	0
0	0.5
0	0
0	0
0.25	1

What if only 2 predictors?

Y

1
1
0
0
1

Overfitting (multidimenstional example)

X

x_1	x_2
0.5	0
0	0.5
0	0
0	0
0.25	1

What if only 2 predictors?

A: better fit

Y

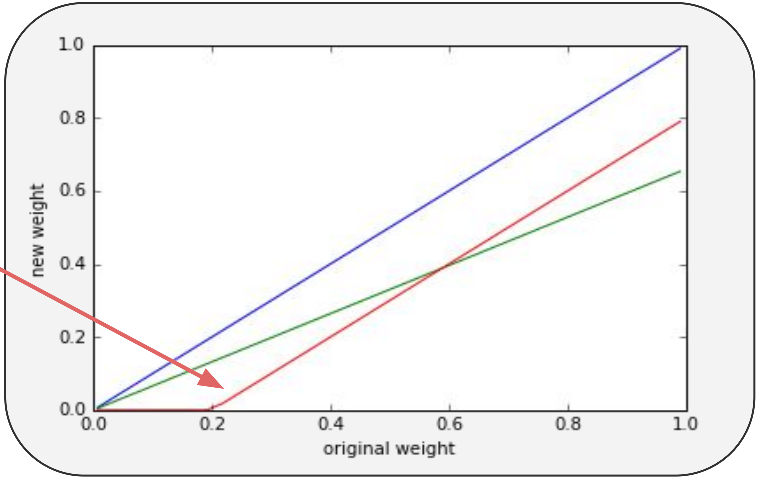
1
1
0
0
1

$$0 + 2 * x_1 + 2 * x_2 = logit(Y)$$

Logistic Regression - Regularization

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.



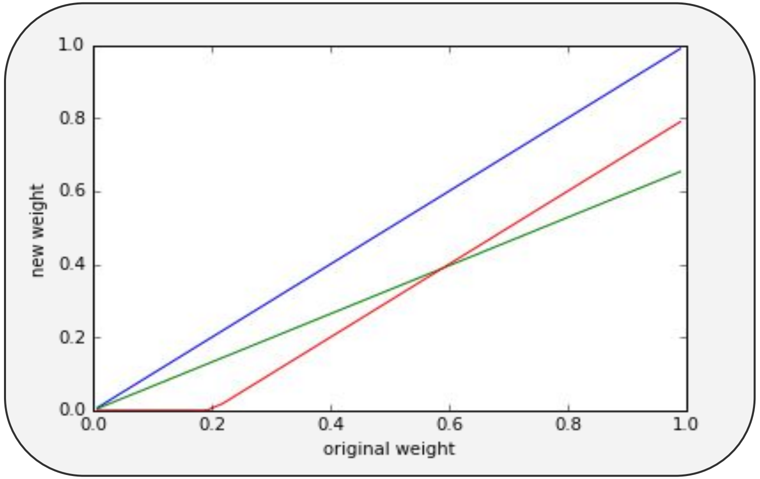
Logistic Regression - Regularization

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

$$J(\beta_0, \beta_1, \ldots, \beta_k | X, y) = -\sum_{i=1}^{N} y_i log\, p(X_i) + (1 - y_i) log(1 - p(X_i))$$

set betas that minimize the loss (J)



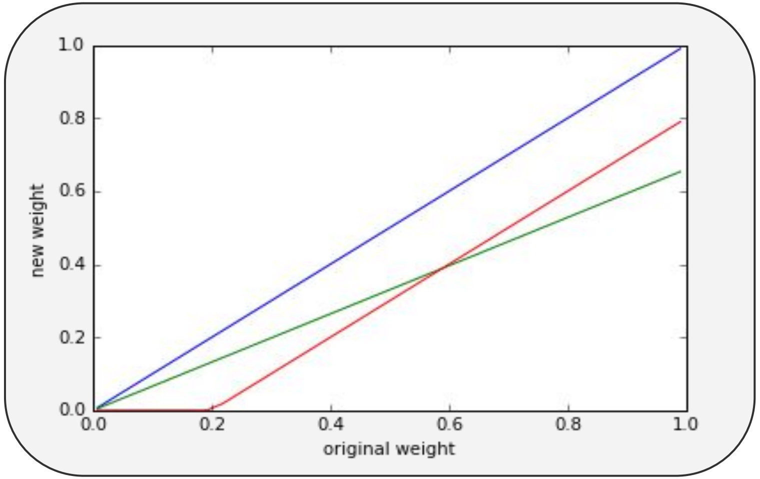
Logistic Regression - Regularization

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

$$J(\beta_0, \beta_1, \ldots, \beta_k | X, y) = -\sum_{i=1}^{N} y_i log\, p(X_i) + (1 - y_i) log(1 - p(X_i)) + \lambda \sum_{j=0}^{k} |\beta_j|$$

set betas that minimize the *penalized* loss (J)



Logistic Regression - Regularization

Sometimes written as:

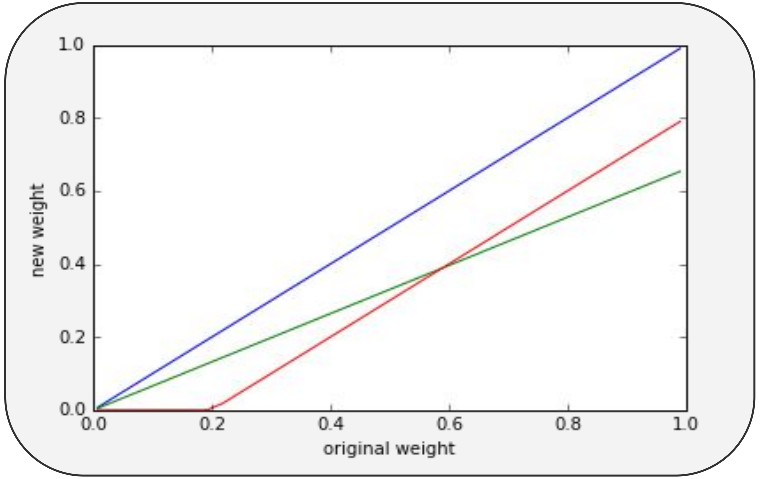
$$||\beta||_1$$

L1 Regularization - "The Lasso"

Zeros out features by adding values that keep from perfectly fitting the data.

$$J(\beta_0, \beta_1, \ldots, \beta_k | X, y) = -\sum_{i=1}^{N} y_i log\, p(X_i) + (1 - y_i) log(1 - p(X_i)) + \lambda \overbrace{\sum_{j=0}^{k} |\beta_j|}$$

set betas that minimize the *L1 penalized* loss (J)



Solved via gradient descent

Logistic Regression - Regularization

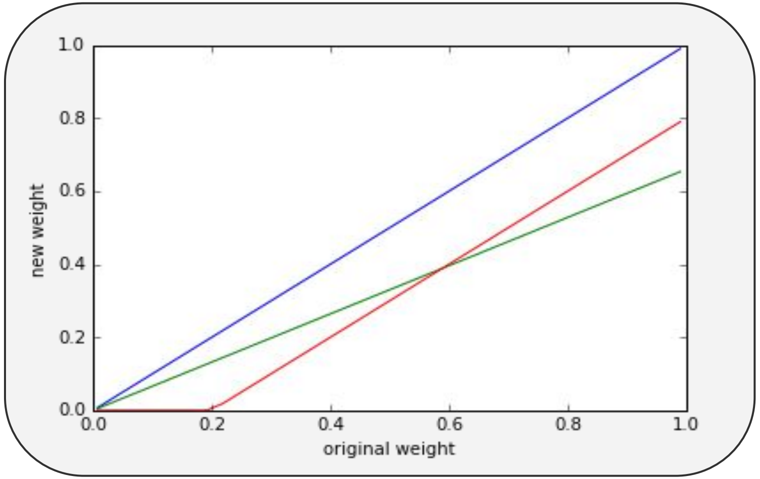
L2 Regularization - “The Lasso”

Shrinks features by adding values that keep from perfectly fitting the data.

$$J(\beta_0, \beta_1, \ldots, \beta_k | X, y) = -\sum_{i=1}^{N} y_i log\, p(X_i) + (1 - y_i) log(1 - p(X_i)) \quad + \lambda \sum_{j=0}^{k} \beta_j^2$$

set betas that minimize the *L2 penalized* loss (J)

Solved via gradient descent



Logistic Regression - Regularization

Sometimes written as: $||\beta||_2^2$

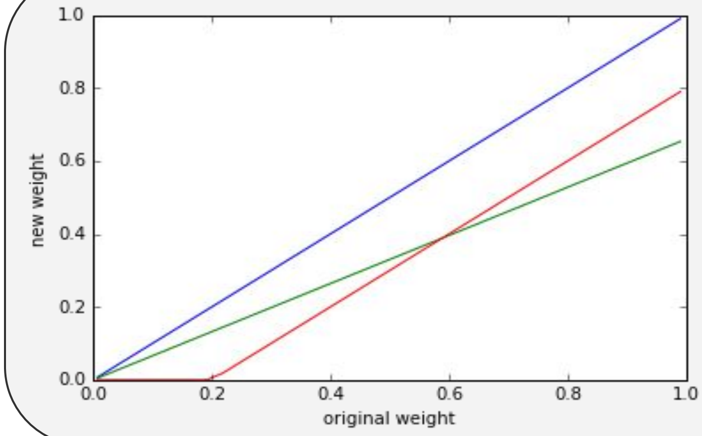
L2 Regularization - "The Lasso"

Shrinks features by adding values that keep from perfectly fitting the data.

$$J(\beta_0, \beta_1, \ldots, \beta_k | X, y) = -\sum_{i=1}^{N} y_i \log p(X_i) + (1 - y_i) \log(1 - p(X_i)) \quad + \lambda \sum_{j=0}^{k} \beta_j^2$$

set betas that minimize the *L2 penalized* loss (J)

Solved via gradient descent



Linear Regression - Regularization

L2 Regularization - “The Lasso”

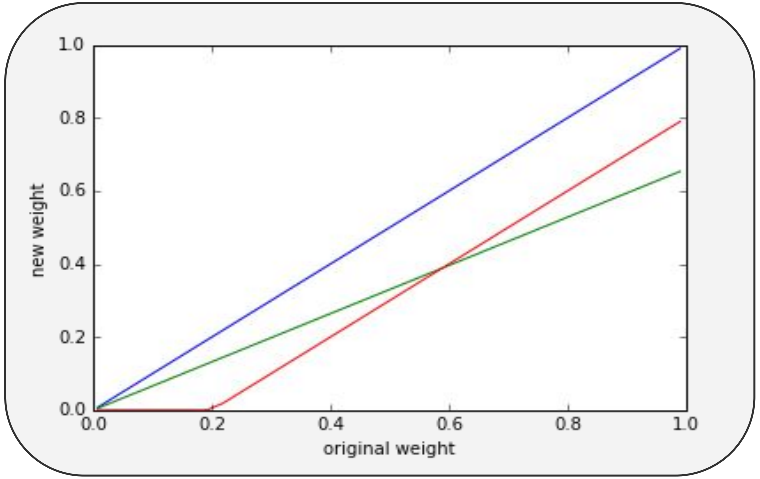
Shrinks features by adding values that keep from perfectly fitting the data.

$$+\lambda \sum_{j=0}^{k} \beta_j^2$$

L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

$$+\lambda \sum_{j=0}^{k} |\beta_j|$$



Linear Regression - Regularization

L2 Regularization - "The Lasso"

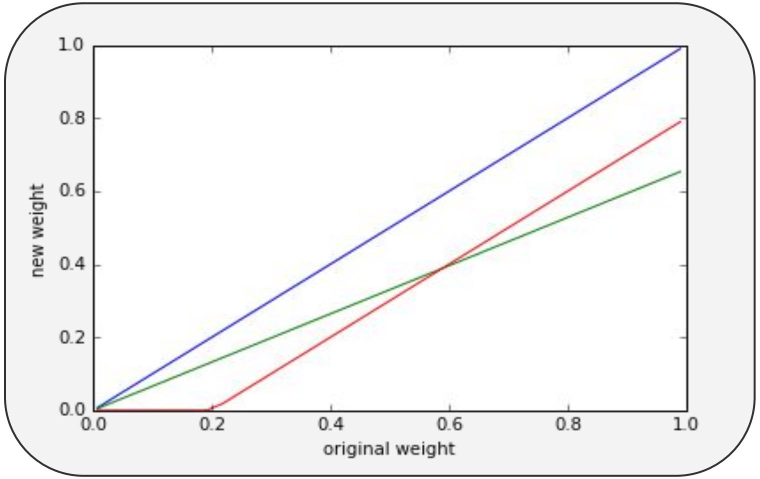
Shrinks features by adding values that keep from perfectly fitting the data.

$$J_{rss}(\beta_0, \beta_1, \ldots, \beta_k | X, y) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{k} \beta_j^2$$

L1 Regularization - "The Lasso"

Zeros out features by adding values that keep from perfectly fitting the data.

$$J_{rss}(\beta_0, \beta_1, \ldots, \beta_k | X, y) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{k} |\beta_j|$$



Linear Regression - Regularization

L2 Regularization - "The Lasso"

Shrinks features by adding values that keep from perfectly fitting the data.

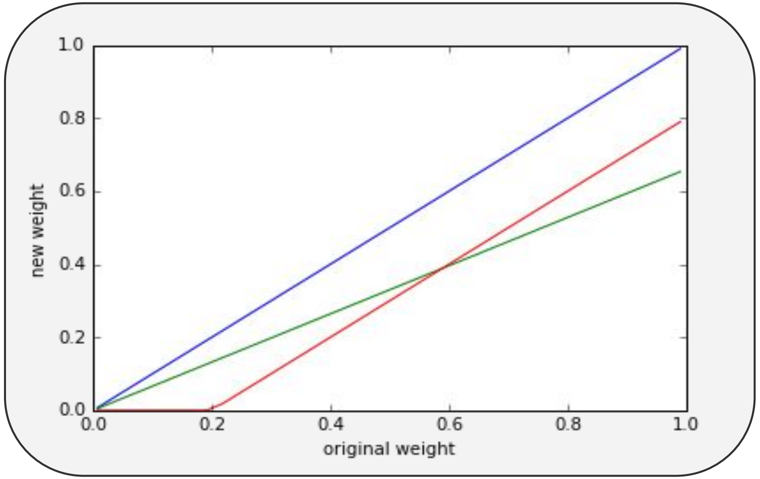
$$J_{mse}(\beta_0, \beta_1, \ldots, \beta_k | X, y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \; + \lambda \sum_{j=0}^{k} \beta_j^2$$

L1 Regularization - "The Lasso"

Zeros out features by adding values that keep from perfectly fitting the data.

$$J_{mse}(\beta_0, \beta_1, \ldots, \beta_k | X, y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{k} |\beta_j|$$

mean squared error: same as *rss* but scales better relative to the penalty *lambda*



Linear Regression - Regularization

L2 Regularization - “The Lasso”

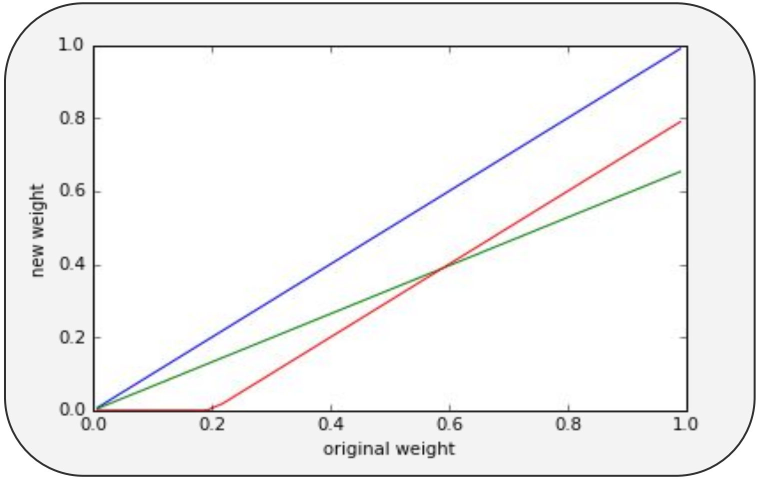
Shrinks features by adding values that keep from perfectly fitting the data.

$$J_{mse}(\beta_0, \beta_1, \ldots, \beta_k | X, y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{k} \beta_j^2$$

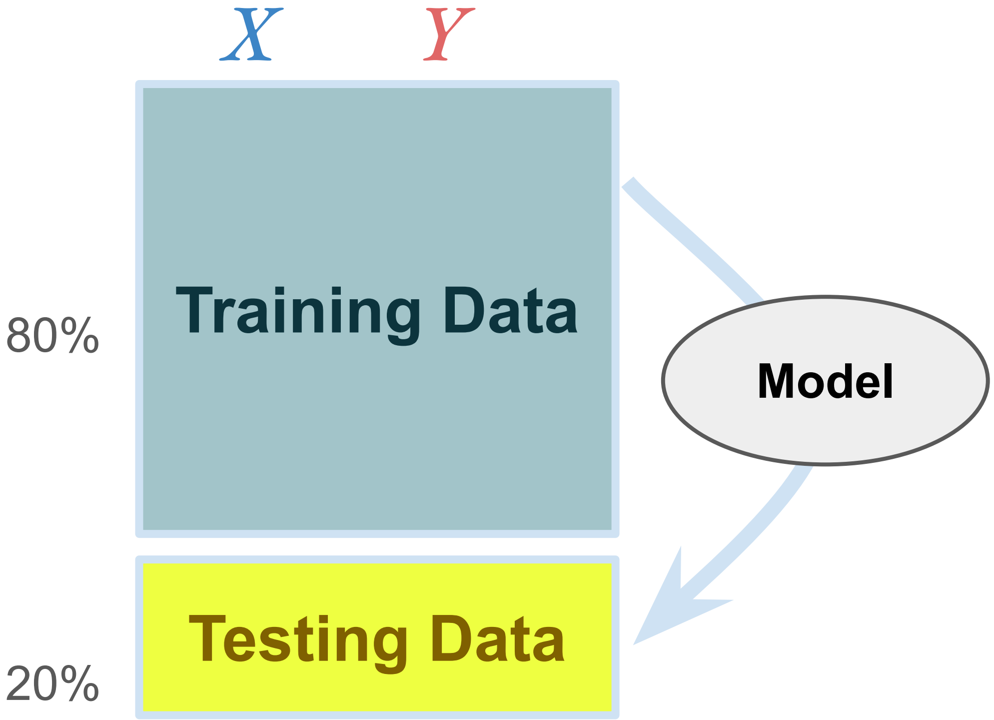
L1 Regularization - “The Lasso”

Zeros out features by adding values that keep from perfectly fitting the data.

$$J_{mse}(\beta_0, \beta_1, \ldots, \beta_k | X, y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=0}^{k} |\beta_j|$$

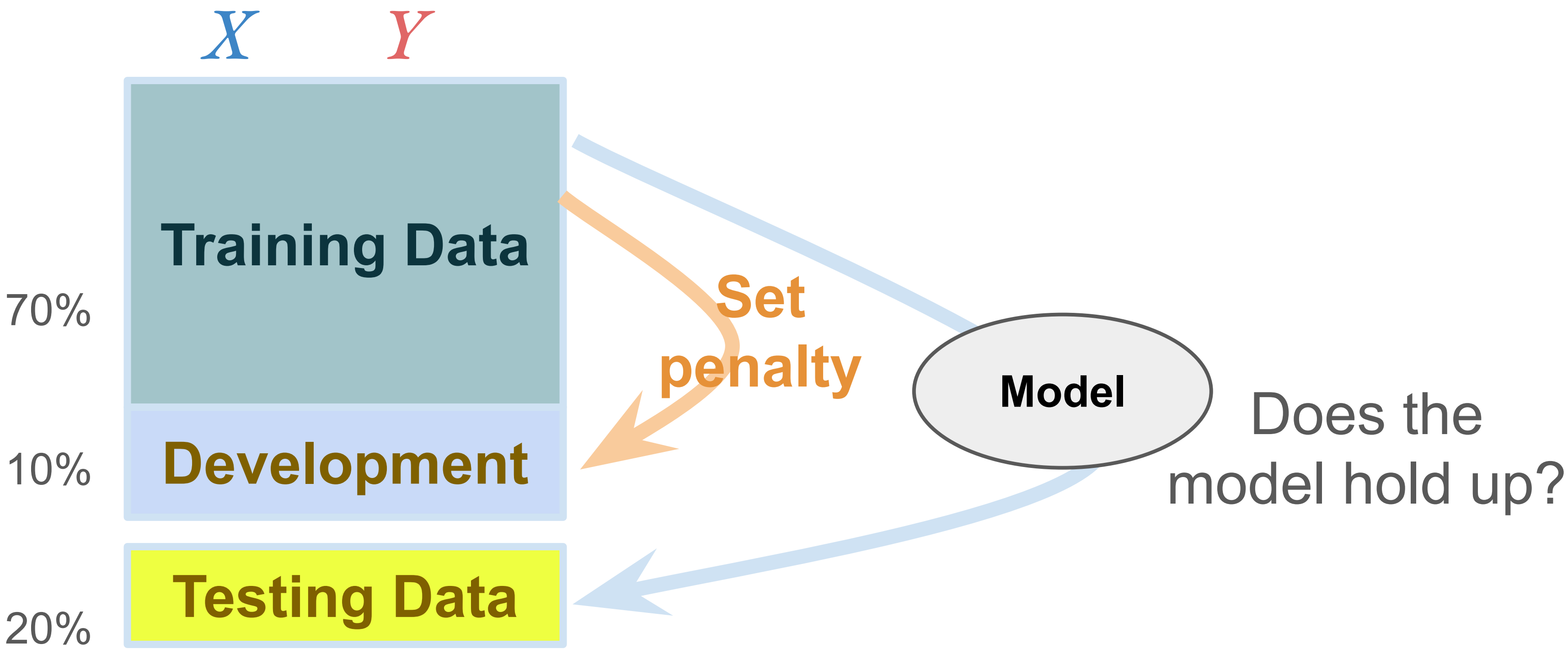


Machine Learning Goal: Generalize to new data



Does the model hold up?

Machine Learning Goal: Generalize to new data



Linear / Logistic Regression - Regularization

Logistic regression: For classification -- when y is binary.

Solving for L1 or L2 regularized loss:

(a) gradient descent, (b) reweighted least squares, (c) coordinate descent

Linear regression: For regression-- when y is continuous.

Solving for L1 regularized loss: gradient descent.

Solving for L2 regularized loss: gradient descent OR normal equation:

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

When L2, L1 work well?

Depends on data, but generally we find...

k: number of features; N: number of observations

	k << N	k < N	k == N	k > N	k >> N
L2	Sometimes	Often	Often	Sometimes	Almost Never
L1	Almost Never	Sometimes	Often	Sometimes	Almost Never

Linear and Logistic Regression Uses:

1. Testing the relationship between variables given other variables. β is an "effect size" -- a score for the magnitude of the relationship; can be tested for significance.

2. Building a predictive model that generalizes to new data. $\hat{Y}$ is an estimate value of Y given X.

Linear and Logistic Regression Uses:

1. Testing the relationship between variables given other variables. β is an "effect size" -- a score for the magnitude of the relationship; can be tested for significance.

2. Building a predictive model that generalizes to new data.
   $\hat{Y}$ is an estimate value of Y given X.
   However, unless $|X| << observatations$ then the model might "overfit".

   -> Regularized linear regression