the shortcut × moz://a

DATA SCIENCE

TRAINING PROGRAM

Learn effective methods to solve business problems

October 7th
@The Shortcut Lab

kodit.io selko SILO.AI

DATA SCIENCE

TRAINING PROGRAM

Modeling: Linear and Logistic Regression and more ...

Erlin Gülbenkoglu from Silo.AI
11.11.19

What are the common Machine Learning methods/approaches?

- Supervised learning
  - Classification (output is a category)
  - Regression (output is continuous number)

- Unsupervised learning
  - Clustering (find similar categories in data)
  - Dimensionality reduction (transform data)
  - Transfer Learning (re-use trained model)
  - Active Learning (query human for outputs)

- Optimization
  - Ensemble methods
  - Gradient descent approaches
  - Genetic algorithms

- Model selection
  - Includes tuning all of the model parameters
  - Performance indicators suitable for task
  - Compare very different models

What are the common Machine Learning methods/approaches?

- Supervised learning
  - Classification (output is a category)
  - Regression (output is continuous number)

Approximating a mapping function (f) from input variables (x) to output variables (y).

What is known? x and y

What is unknown? f (your model)

Training set = $((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$

- Unsupervised learning
  - Clustering (find similar categories in data)
  - Dimensionality reduction (improve data)
  - Transfer Learning (re-use trained model)
  - Active Learning (query human for outputs)

Approximating a mapping function (f) from input variables (x). f transforms data into a desired form.
What is known? x

What is unknown? f

Training set = $((x_1), (x_2), \ldots, (x_n))$

- In this session we will only look at the **supervised linear models.**

Identify your Problem

What kind of target variable (y) do you have? (What do you want to predict?)

Regression: Real valued target variable
(prediction of continuous quantity / numeric variable)

- Example: Amount/size/rate of something
- Evaluated by using root mean squared error, mean absolute error.

Classification: Discrete target variable
(prediction of qualitative variable / categorical variable)

- Variables which do not have an ordinal relationship and belong to a finite set
- Example: object recognition, topic classification
- Evaluated by accuracy, precision, recall, f-1 score and more.

Identify your Problem

! Both regression and classification can have real valued or discrete input variables.

What kind of target variable (y) do you have? (What do you want to predict?)

Regression: Real valued target variable
(prediction of continuous quantity / numeric variable)

- Example: Amount/size/rate of something
- Evaluated by using root mean squared error, mean absolute error.

Note

A problem with one input variable is often called univariate regression problem and with multiple input variables is called a multivariate regression problem.

Classification: Discrete target variables
(prediction of qualitative variable / categorical variable)

- Variables which do not have an ordinal relationship and belong to a finite set
- Example: object recognition, topic classification
- Evaluated by accuracy, precision, recall, f-1 score and more.

Note

A problem with two classes is often called a two-class or binary classification problem.

A problem with more than two classes is often called a multi-class classification problem.

Linear Models - Linear Regression

- Modeling the relationship between Y (dependent variable) and X (independent variables) by fitting a linear equation.
  The vector of inputs: $X = (X_1, X_2, \ldots, X_p)$
  Each X_j represents features (attributes or dimensions) of the dataset. (Would be the columns on an excel sheet.)

- We want to learn the weights of each feature: $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ and the β_0.
- The term β_0 is the intercept, also known as the bias in machine learning.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

Symbols with the hats (^) represent the estimates.

Note

Model has p+1 parameters: p is the amount of features + 1 bias term

Assumptions of Linear Regression

The linear model makes huge assumptions about structure and yields stable but possibly inaccurate predictions.

- Relationship between dependent variable and independent variable(s) is well approximated by a **globally linear function**. To inspect linearity one can plot observed vs. predicted values or residuals vs. predicted values.
- Errors are normally distributed mean μ = 0, variance σ^2 = 1.
- Equal variance around the line (or homoscedasticity of errors). Which means that the standard deviations of the error terms are constant and do not depend on the x-value.
- Independence of the observations. Multicollinearity can be present but shouldn't be so 'high'. We worry more about this in longitudinal datasets!

Linear Models - Linear Regression

- The purpose is to learn a vector of weights and the bias term. We assume :

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{bmatrix}$$

We assume we have p features and n data points.
(p amount of columns and n amount of rows)

- Let's represent our inputs and output in the matrix form as well.

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix} \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

- The analytical solution to calculate the β^ would be like this.

$$X\hat{\beta} = Y$$

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

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

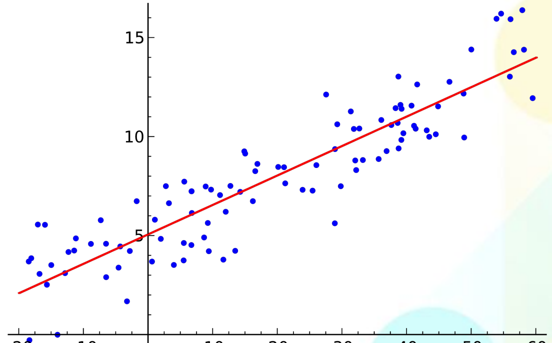
Difficult to calculate inverse of a BIG matrix!

Fitting the model to training data

How do we measure model quality?

Mean squared error (MSE):

$$L(\beta) = \frac{1}{n}\sum_{j=0}^{p}(\beta_j X_{ij} - y_i)^2$$



The less the MSE the better your model.

We find the values for β which minimizes the MSE. Let's assume we have two univariate linear regression problem which has two parameters to learn β_0 and β_1.
We take the first order conditions - we differentiate with respect to β_0 and β_1 and set the derivatives equal to 0. (Ordinary Least Squares Regression)

How do we measure model quality?

Mean squared error (MSE): The most common loss function for regression is the mean squared error. (or root mean squared error)

$$L(\hat{\beta}) = \frac{1}{n}\sum_{j=1}^{p}(\hat{\beta}_0 + X_{ij}\hat{\beta}_j - Y_i)^2$$

Mean absolute error (MAE): Can also be used as loss function or as another metric to describe model performance. The advantage is it is easier to understand. But, has discontinuities in its derivatives, which have hindered its widespread use.

$$L(\hat{\beta}) = \frac{1}{n}\sum_{j=1}^{p}|\hat{\beta}_0 + X_{ij}\hat{\beta}_j - Y_i|$$

How do we measure model quality?

R2 - R-squared - coefficient of determination: A measure to understand how well observed outcomes are replicated by the model.

This is more used by the statistics community as a measure of quality of a linear model. The formula of R2 can be slightly different in different sources. The presented R2 is the one used at sklearn library.

$$R^2 = 1 - \frac{\sum(y - \hat{y})^2}{\sum(y - \frac{y}{n})^2}$$

In sklearn linear regression implementation this is what the score method returns!

Linear Models - Numeric “Dummy coding” of Qualitative Variables

- Linear relationship between X (independent variables) and Y (dependent variable)
  The vector of inputs: $X = (X_1, X_2, . . . , X_p)$

- For example: if variable X_2 has the values {car, bicycle, train, metro}
  - If we just transform these categories into numbers like: {car, bicycle, train, metro} -> {0, 1, 2, 3}
  - Our model will be taught a numeric relationship like car(0) < bicycle(1) < train(2)
- Since the variable X_2 has distinct values (categories) which do not have an ordinal or further rational relationship we will create dummy variables to represent each category of our categorical variable X_2.

X_2	X_2_car	X_2_bicycle	X_2_train	X_2_metro
Bicycle	0	1	0	0
Metro	0	0	0	1
Car	1	0	0	0

We create new dummy variables from X_2 to represent the existence of a category.

What would be the number of necessary dummy variables in k categories?

Linear Models - Linear Classification

- Linear relationship between X (independent variables) and Y (dependent variable)
  The vector of inputs: $X = (X_1, X_2, \ldots, X_p)$
  Output $Y \in \{-1,1\}$ (or $Y \in \{0, 1\}$)

$$\hat{Y} = \text{sign}(X\hat{\beta})$$

Note
Model has p+1 parameters: p is the amount of features + 1 bias term (bias term is included to β vector here)

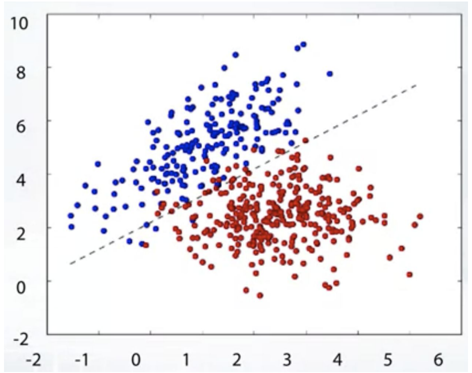


Figure taken from: ABC of Machine Learning. Retrieved from
https://medium.com/@AnIdiotLearner/part-3-abc-of-machine-learning-720f73b7f6a0

Binary Logistic Regression - Sigmoid Function

- The same formula as in linear regression is used: we multiply each x_j by its weight β_j, sums up the weighted features, and adds the bias term β_0. The resulting single number z expresses the weighted sum of the evidence for the class.

$$z = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

- We want to force z to be a legal probability --> to lie between 0 and 1. But, since weights are real-valued, z can range from $-\infty$ to ∞.

- To create a probability, we'll pass z through the logistic function (sigmoid function), σ(z).

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

But, where does this function come from?

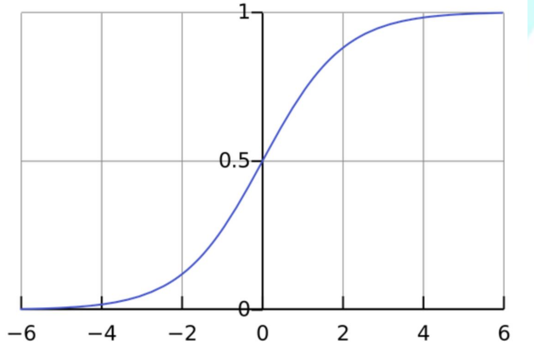


Figure taken from: Sigmoid function. Retrieved from https://en.wikipedia.org/wiki/Sigmoid_function

Logistic Regression

Log Odds

- Probability y: $$0 \leq P(y=1) \leq 1$$
- Odds: $$0 < \frac{P(y=1)}{1-P(y=1)} < \infty$$
- Logit: $$-\infty < \log(\frac{P(y=1)}{1-P(y=1)}) < \infty$$

Logit basically creates a map of probability values from [0 , 1] to (− ∞ , + ∞). The opposite of what we want!

Logistic Function

- Logit: $$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \hat{\beta}_0 + \sum_{j=1}^{p} X_j\hat{\beta}_j = z$$

- Logistic: $$P(Y=1|X) = \frac{e^z}{1+e^z}$$

$$\Updownarrow$$

- Logistic:

(alternative way of showing) $$P(Y=1|X) = \frac{1}{1+e^{-z}}$$

$$\boxed{\mathrm{logit}(z)^{-1} = \mathrm{logistic}(z)}$$

$$\Downarrow$$

Also can be called expit or sigmoid function.

Multinomial Logistic Regression - Softmax function

Linear relationship between X (independent variables) and Y (dependent variable)
The vector of inputs: X = (X1, X2, . . . , Xp)
Output Y ∈ {1, … K}

$$\hat{Y} = \arg \max_{k \in \{1,\ldots,K\}} (X\hat{\beta}_k)$$

Note

Simply we rerun the binary classification multiple times, once for each class.
Model has K*(p+1) parameters.

Example: If we had 3 classes:

$z = (\beta_1 X, \beta_2 X, \beta_3 X)$

Our score vector:

z = (6, -4.5, 8) scores -> y=3

Logistic Regression

- Class scores from linear models:

$$z = (X\hat{\beta}_1, X\hat{\beta}_2, ..., X\hat{\beta}_K)$$

logits

$$\Downarrow$$

$$(e^{z_1}, e^{z_2}, ..., e^{z_K})$$

expits

$$\Downarrow$$

$$\left(\frac{e^{z_1}}{\sum_{k=1}^{K} e^{z_K}}, \frac{e^{z_2}}{\sum_{k=1}^{K} e^{z_K}}, \ldots, \frac{e^{z_K}}{\sum_{k=1}^{K} e^{z_K}}\right)$$

softmax transformation

Logistic Regression

- Class scores from a linear model:

$$z = (X\hat{\beta}_1, X\hat{\beta}_2, ..., X\hat{\beta}_K)$$ logits

$$\Downarrow$$

$$(e^{z_1}, e^{z_2}, ..., e^{z_K})$$ expits

$$\Downarrow$$

$$(\frac{e^{z_1}}{\sum_{k=1}^{K} e^{z_K}}, \frac{e^{z_2}}{\sum_{k=1}^{K} e^{z_K}}, ..., \frac{e^{z_K}}{\sum_{k=1}^{K} e^{z_K}})$$ softmax transformation

- Example: Do the softmax transformation for the following z scores:

  z = (6, -4.5, 8)

Logistic Regression

- Class scores from a linear model:

$$z = (X\hat{\beta}_1, X\hat{\beta}_2, ..., X\hat{\beta}_K)$$ logits

$$\Downarrow$$

$$(e^{z_1}, e^{z_2}, ..., e^{z_K})$$ expits

$$\Downarrow$$

$$\left(\frac{e^{z_1}}{\sum_{k=1}^{K} e^{z_K}}, \frac{e^{z_2}}{\sum_{k=1}^{K} e^{z_K}}, ..., \frac{e^{z_K}}{\sum_{k=1}^{K} e^{z_K}}\right)$$ softmax transformation

- Example: Do the softmax transformation for the following z scores:

z = (6, -4.5, 8)

$$\Downarrow$$

e = (403.43, 0.011, 2980.96)

$$\Downarrow$$

s(z) = (0.12, 0.0, 0.88)

Sigmoid vs. Softmax Transformation

We use them to map a vector to a likelihoods. Softmax is basically a generalization of the sigmoid function.

Sigmoid Function (Logistic Function)

- Used for binary classification in logistic regression model.
- The likelihoods sum will be 1.
- The high value will have the high probability but not the higher probability.
- Useful for cases when your outputs are independent of one another.

Softmax Function

- Used for multi-classification in logistic regression model.
- The likelihoods sum will be 1.
- The high value will have the higher probability than other values.
- Useful for cases when each sample can belong to exactly one class.

How do we measure model quality?

Classification accuracy: Is the most common technique to measure model performance in classification tasks. It is basically the fraction of predictions our model predicted correctly (# correct predictions / # all predictions)

	Predicted	
Actual	True positive	False positive
	False negative	True negative

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1 Score = 2 x (Precision x Recall) / (Precision + Recall)

How do we measure model quality?

Classification accuracy: Is the most common technique to measure model performance in classification tasks. It is basically the fraction of predictions our model predicted correctly (# correct predictions / # all predictions)

Can we use it as our loss function?

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i = y_i)$$

[P] Iverson bracket:

$$[P] = \begin{cases} 1, & \text{P is true,} \\ 0, & \text{P is false.} \end{cases}$$

Problems

- Not differentiable,
- Does not asses model confidence

Loss function - Cross Entropy

$$\left(\frac{e^{z_1}}{\sum_{k=1}^{K} e^{z_K}}, \frac{e^{z_2}}{\sum_{k=1}^{K} e^{z_K}}, ..., \frac{e^{z_K}}{\sum_{k=1}^{K} e^{z_K}}\right)$$

Predicted class probabilities

$$p = ([y = 1], [y = 2], ..., [y = K])$$

Target values for class probabilities

$$-\sum_{k=1}^{K} [y = k] \log \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} = -\log \frac{e^{z_y}}{\sum_{j=1}^{K} e^{z_j}}$$

Cross entropy measures the similarity between class scores z and class probabilities p.

Loss function - Cross Entropy

$$-\sum_{k=1}^{K}[y=k]\log\frac{e^{z_k}}{\sum_{j=1}^{K}e^{z_j}}=-\log\frac{e^{z_y}}{\sum_{j=1}^{K}e^{z_j}}$$

Note

As you may have guessed logs are actually ln. But for some reason everyone writes log.

Example:

Let’s consider that we have four classes K = 4 and the example we picked belongs to the second class y = 2.

Ex 1: Our prediction is y=2 with 100% probability.

- -0*log0 - 1*log1 - 0*log0 - 0*log0 = 0

Cross entropy is 0! Perfect model. :)

Ex 2: Our prediction is y=2 with 65% probability.

-

Ex 3: Our prediction is y=1.

-

Loss function - Cross Entropy

$$-\sum_{k=1}^{K}[y=k]\log\frac{e^{z_k}}{\sum_{j=1}^{K}e^{z_j}} = -\log\frac{e^{z_y}}{\sum_{j=1}^{K}e^{z_j}}$$

Note

As you may have guessed logs are actually ln. But for some reason everyone writes log.

Example:

Let's consider that we have four classes K = 4 and the example we picked belongs to the second class y = 2.

Ex 1: Our prediction is y=2 with 100% probability.

- -0*log0 - 1*log1 - 0*log0 - 0*log0 = 0

Cross entropy is 0! Perfect model. :)

Ex 2: Our prediction is y=2 with 65% probability.

- -0*log0.2 - 1*log0.65 - 0*log0.1 - 0*log0.05 = 0.431

Ex 3: Our prediction is y=1.

- -0*log0.44 - 1*log0 - 0*log0.36 - 0*log0.2 = + ∞

We can sum all the cross entropies in the training set and use it as our loss function. Cross entropy is differentiable! Difficult to find analytical solution so instead we will use some optimization techniques.

Take a break, Archimedes. A nice soak in the bath will work wonders.

Note: in the following slides β and β^ are used interchangeably. But, all of the weights shown represent what we estimated.

Loss functions

Linear Regression

$$L(\beta) = \frac{1}{n}\sum_{j=1}^{p}(\beta_0 + X_{ij}\beta_j - Y_i)^2$$

Logistic Regression

$$L(\beta) = -\sum_{k=1}^{K}[y_i = k]\log\frac{e^{\beta_k x_i}}{\sum_{j=1}^{K} e^{\beta_j x_i}}$$

Gradient Descent

Can optimize any differentiable loss function!

Optimization problem: minimize β

Gradient Descent

Optimization problem: L(β) -> minimize β

We have some approximation of β - how could we refine them?

$$\nabla L(\beta^0) = (\frac{\partial L(\beta^0)}{\partial \beta_0}, \frac{\partial L(\beta^0)}{\partial \beta_1}, ..., \frac{\partial L(\beta^0)}{\partial \beta_p})$$

Vector of partial derivatives w.r.t all our p parameters

In order to minimize the value of loss we compute the gradient at initialized point β^0 and step at the direction of anti gradient.

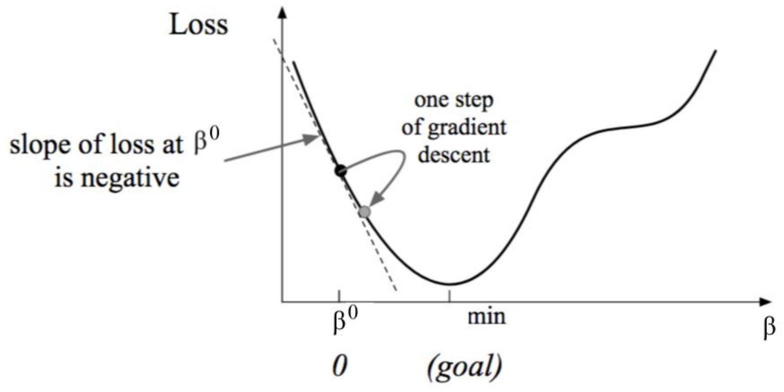
Gradient step - $$\underbrace{\beta^1}_{\text{Next approximation}} = \beta^0 - \underbrace{\eta_1}_{\text{Learning rate}} \nabla L(\beta^0)$$

Gradient Descent

```
$\beta^0$ initialization
while True
    $\beta^t = \beta^{t-1} - \eta_t \nabla L(\beta^{t-1})$
    if $||\beta^t - \beta^{t-1}|| < \epsilon$ then break
```

Learning rate

Stopping criteria



Gradient Descent - Linear Regression

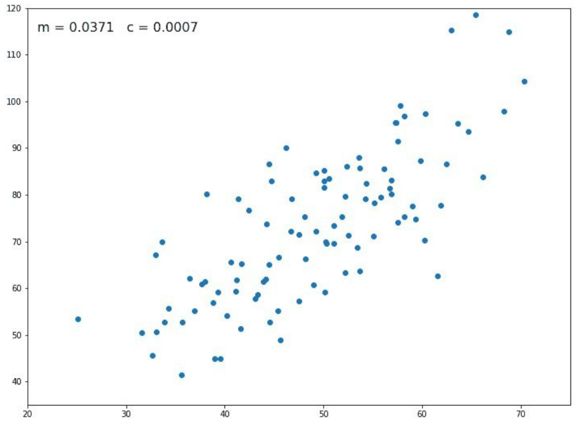


Figure taken from: Menon, A. (2018, Sep, 16). Linear Regression using Gradient Descent. Retrieved from https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

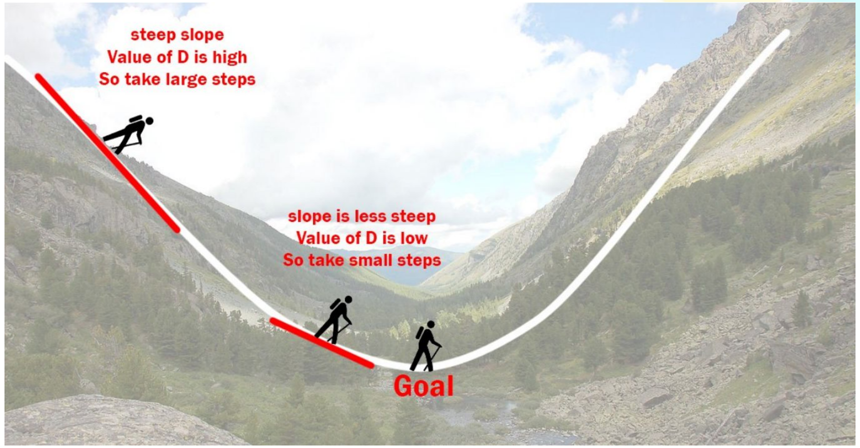


Figure taken from: Menon, A. (2018, Sep, 16). Linear Regression using Gradient Descent. Retrieved from https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

Here m is our β_1 and c is our β_0.

Bias-Variance Tradeoff

There are two kinds of prediction errors we can differentiate to understand better:

- **Error due to Bias**

  Error of limited flexibility of the learning algorithm to learn the true signal from the training dataset. High bias may cause your model to miss the relations between your inputs and outputs. In ML we call this underfitting!

- **Error due to Variance**

  Error of the learning algorithm's sensitivity to specific fluctuations in a training set. High variance may cause your model to even learn the random noise in your training data. In ML we call this overfitting!

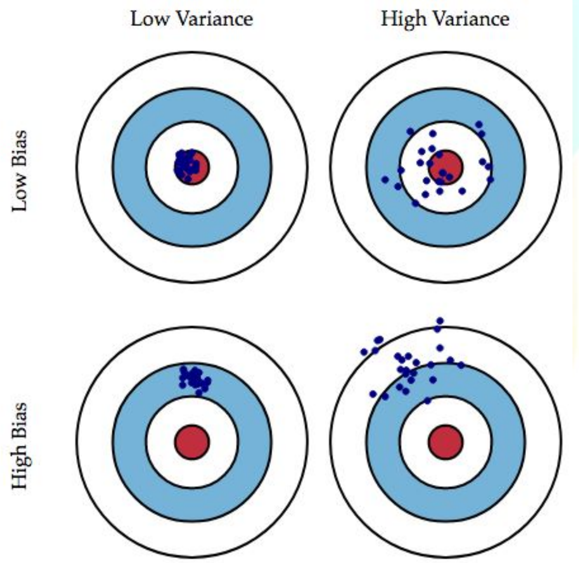


Figure taken from: From Understanding the Bias-Variance Tradeoff, by Scott Fortmann-Roe.

Bias-Variance Tradeoff

There are two kinds of prediction errors we can differentiate to understand better:

- **Error due to Bias**

  Error of limited flexibility of the learning algorithm to learn the true signal from the training dataset. High bias may cause your model to miss the relations between your inputs and outputs. In ML we call this underfitting!

- Relatively simpler models
- Consistent but rather inaccurate models
- Linear models would have more of an underfitting problem then overfitting.

- **Error due to Variance**

  Error of the learning algorithm's sensitivity to specific fluctuations in a training set. High variance may cause your model to even learn the random noise in your training data. In ML we call this overfitting!

- More complex models
- Accurate but inconsistent
- Linear models can be *regularized* to reduce the model complexity.

Regularization (Shrinkage Methods)

Linear models can be regularized to decrease their variance at the cost of increasing their bias (hopefully not too much!).
Works with assumption that smaller weights generate simpler models thus helps with the overfitting problem.
We would like to shrink the regression coefficients towards zero by imposing a penalty on their size.

A shrinkage quantity is added to the loss function.

$$\underbrace{L_{reg}(\beta)}_{\text{Regularized Loss Function}} = \underbrace{L(\beta)}_{\text{Loss Function}} + \overbrace{\lambda}^{\text{Regularization parameter (strength)}} \underbrace{R(\beta)}_{\text{Regularizer}} \rightarrow \overbrace{\min_{\beta}}^{\text{Just minimize the new loss!}}$$

- Parameter λ basically controls the impact on bias and variance.
- λ >= 0
- The larger the value of λ, the greater the amount of regularization.
- When λ = 0 there is no regularization.
- When λ→∞ the regression coefficient estimates approach to zero.

Regularization (Shrinkage Methods)

There are two popular techniques: Ridge Regression and Lasso where a shrinkage quantity is added to the loss function.

Ridge Regression (L2 Penalty)

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{j=1}^{p}(\beta_0 + \beta_j X_{ij} - y_i)^2 + \lambda \underbrace{\sum_{j=1}^{p}\beta_j^2}_{\text{L2 norm}}$$

- Drives all weights closer to zero.
- Works better when output is a function of all input variables.
- Can be optimized with gradient methods.

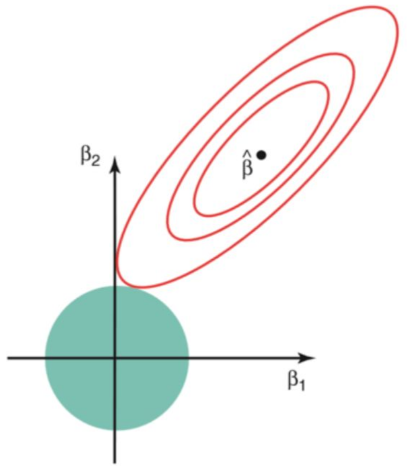


Figure taken from: An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Regularization (Shrinkage Methods)

There are two popular techniques: Ridge Regression and Lasso where a shrinkage quantity is added to the loss function.

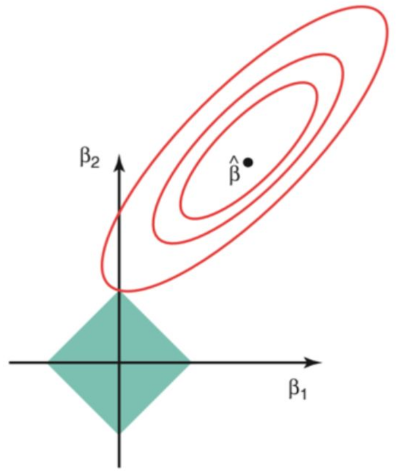


Figure taken from: An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Lasso (L1 Penalty)

$$L_{Lasso}(\beta) = \frac{1}{n}\sum_{j=1}^{p}(\beta_0 + \beta_j X_{ij} - y_i)^2 + \lambda \underbrace{\sum_{j=1}^{p}|\beta_j|}_{\text{L1 norm}}$$

- Derives some weights exactly to zero (basically does feature selection).
- Learns sparse models (majority of features have zero weights and minority have nonzero weights)
- Cannot be optimized with simple gradient methods

These two approaches are used a lot in neural networks!

Regularization (Shrinkage Methods)

There are two popular techniques: Ridge Regression and Lasso where a shrinkage quantity is added to the loss function.

Ridge Regression (L2 Penalty)

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{j=1}^{p}(\beta_0 + \beta_j X_{ij} - y_i)^2 + \lambda \underbrace{\sum_{j=1}^{p}\beta_j^2}_{\text{L2 norm}}$$

- Drives all weights closer to zero.
- Works better when output is a function of all input variables.
- Can be optimized with gradient methods.

Lasso (L1 Penalty)

$$L_{Lasso}(\beta) = \frac{1}{n}\sum_{j=1}^{p}(\beta_0 + \beta_j X_{ij} - y_i)^2 + \lambda \underbrace{\sum_{j=1}^{p}|\beta_j|}_{\text{L1 norm}}$$

- Derives some weights exactly to zero (basically does feature selection).
- Learns sparse models (majority of features have zero weights and minority have nonzero weights)
- Cannot be optimized with simple gradient methods.

Why linear regression is great?

Interpretation! It is awesome to say that for example: for every one unit of change in X_{10}, the change in Y is by β_{10}.

Sometimes it is ***more important*** to have an interpretable model than a more accurate model.

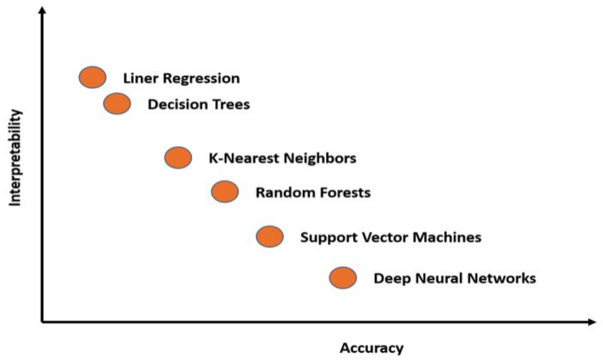


Figure taken from: Rodriguez, J. (2018, June, 6). Interpretability vs. Accuracy: The Friction that Defines Deep Learning. Retrieved from https://towardsdatascience.com/interpretability-vs-accuracy-the-friction-that-defines-deep-learning-dae16c84db5c