

## Machine Learning

### Machine Learning Resources

[Online Resources \(/machine-learning-resources/\)](/machine-learning-resources/)

[Journal Library \(/machine-learning/journals-library/\)](/machine-learning/journals-library/)

[Datasets for Machine Learning \(/machine-learning/datasets\)](/machine-learning/datasets)

### Machine Learning and Econometrics

[Resources \(/machine-learning/econometrics/resources\)](/machine-learning/econometrics/resources)

### Supervised Learning Theory

[Overview \(/machine-learning/\)](/machine-learning/)

[One Variable Linear Regression \(/one-variable-linear-regression/\)](/one-variable-linear-regression/)

[Linear Algebra \(/linear-algebra-machine-learning/\)](/linear-algebra-machine-learning/)

[Multiple Variable Linear Regression \(/multi-variable-linear-regression/\)](/multi-variable-linear-regression/)

[Logistic Regression \(/logistic-regression/\)](/logistic-regression/)

[Neural Networks \(Representation\) \(/neural-networks-representation/\)](/neural-networks-representation/)

[Neural Networks \(Learning\) \(/neural-networks-learning/\)](/neural-networks-learning/)

[Applying Machine Learning \(/applying-machine-learning/\)](/applying-machine-learning/)

[Machine Learning Systems Design \(/machine-learning-systems-design/\)](/machine-learning-systems-design/)

[Support Vector Machines \(/machine-learning-svms-support-vector-machines/\)](/machine-learning-svms-support-vector-machines/)

### Unsupervised Learning Theory

[Unsupervised Learning \(/machine-learning-unsupervised-learning/\)](/machine-learning-unsupervised-learning/)

[Dimensionality Reduction \(/machine-learning-dimensionality-reduction/\)](/machine-learning-dimensionality-reduction/)

[Anomaly Detection \(/machine-learning-anomaly-detection/\)](/machine-learning-anomaly-detection/)

[Recommender Systems \(/machine-learning-recommender-systems/\)](/machine-learning-recommender-systems/)

[Large Scale Machine Learning \(/machine-learning-large-scale/\)](/machine-learning-large-scale/)

[Photo OCR \(/machine-learning-photo-ocr/\)](/machine-learning-photo-ocr/)

## Reinforcement Learning Theory

[Markov Decision Processes \(/machine-learning-markov-decision-processes/\)](/machine-learning-markov-decision-processes/)

[Reinforcement Learning \(/machine-learning-reinforcement-learning/\)](/machine-learning-reinforcement-learning/)

[Game Theory \(/machine-learning-game-theory/\)](/machine-learning-game-theory/)

## Deep Learning Theory

[Deep Learning Terms \(/machine-learning/deep-learning/terms/\)](/machine-learning/deep-learning/terms/)

[Deep Learning Intro \(/machine-learning/deep-learning/intro/\)](/machine-learning/deep-learning/intro/)

[Deep Neural Networks Intro \(/machine-learning/deep-learning/neural-nets/\)](/machine-learning/deep-learning/neural-nets/)

[Deep Convolutional Networks Intro \(/machine-learning/deep-learning/convnets/\)](/machine-learning/deep-learning/convnets/)

## Deep Learning with TensorFlow

[Exploring NotMNIST \(/machine-learning/deep-learning/tensorflow/notmnist/\)](/machine-learning/deep-learning/tensorflow/notmnist/)

[Deep Neural Networks \(/machine-learning/deep-learning/tensorflow/deep-neural-nets/\)](/machine-learning/deep-learning/tensorflow/deep-neural-nets/)

[Regularization \(/machine-learning/deep-learning/tensorflow/regularization/\)](/machine-learning/deep-learning/tensorflow/regularization/)

[Deep Convolutional Networks \(/machine-learning/deep-learning/tensorflow/convnets/\)](/machine-learning/deep-learning/tensorflow/convnets/)

## Machine Learning with Scikit-Learn

[Introduction to Machine Learning \(/machine-learning-intro-easy/\)](/machine-learning-intro-easy/)

[IPython Introduction \(/ipython-introduction/\)](/ipython-introduction/)

[Iris Dataset \(/machine-learning-iris-dataset/\)](/machine-learning-iris-dataset/)

[Linear Regression Model \(/machine-learning-linear-regression/\)](/machine-learning-linear-regression/)

[Linear Regression Model Evaluation \(/machine-learning-evaluate-linear-regression-model/\)](/machine-learning-evaluate-linear-regression-model/)

[Polynomial Regression \(/machine-learning-polynomial-regression/\)](/machine-learning-polynomial-regression/)

[Vectorization, Multinomial Naive Bayes Classifier and Evaluation \(/machine-learning-multinomial-naive-bayes-vectorization/\)](/machine-learning-multinomial-naive-bayes-vectorization/)

[Gaussian Naive Bayes \(/machine-learning-gaussian-naive-bayes/\)](/machine-learning-gaussian-naive-bayes/)[K-nearest Neighbors \(KNN\) Classification Model \(/machine-learning-k-nearest-neighbors-knn/\)](/machine-learning-k-nearest-neighbors-knn/)[Ensemble Learning and Adaboost \(/machine-learning-ensemble-of-learners-adaboost/\)](/machine-learning-ensemble-of-learners-adaboost/)[Decision Trees \(/machine-learning-decision-trees/\)](/machine-learning-decision-trees/)[Support Vector Machines \(/machine-learning-svms/\)](/machine-learning-svms/)[Clustering with KMeans \(/machine-learning-clustering-kmeans/\)](/machine-learning-clustering-kmeans/)[Dimensionality Reduction and Feature Transformation \(/machine-learning-dimensionality-reduction-feature-transform/\)](/machine-learning-dimensionality-reduction-feature-transform/)[Feature Engineering and Scaling \(/machine-learning-feature\\_engineering\\_scaling/\)](/machine-learning-feature_engineering_scaling/)[Cross-Validation for Parameter Tuning, Model Selection, and Feature Selection \(/machine-learning-cross-validation/\)](/machine-learning-cross-validation/)[Efficiently Searching Optimal Tuning Parameters \(/machine-learning-efficiently-search-tuning-param/\)](/machine-learning-efficiently-search-tuning-param/)[Evaluating a Classification Model \(/machine-learning-evaluate-classification-model/\)](/machine-learning-evaluate-classification-model/)[One Hot Encoding \(/machinelearning-one-hot-encoding/\)](/machinelearning-one-hot-encoding/)[F1 Score \(/machinelearning-f1-score/\)](/machinelearning-f1-score/)[Learning Curve \(/machinelearning-learning-curve/\)](/machinelearning-learning-curve/)

## Machine Learning Projects

[Titanic Survival Data Exploration \(/machine-learning-project-titanic-survival/\)](/machine-learning-project-titanic-survival/)[Boston House Prices Prediction and Evaluation \(Model Evaluation and Prediction\) \(/machine-learning-project-boston-home-prices/\)](/machine-learning-project-boston-home-prices/)[Building a Student Intervention System \(Supervised Learning\) \(/machine-learning-project-student-intervention/\)](/machine-learning-project-student-intervention/)[Identifying Customer Segments \(Unsupervised Learning\) \(/machine-learning-project-customer-segments/\)](/machine-learning-project-customer-segments/)[Training a Smart Cab \(Reinforcement Learning\) \(/machine-learning-proj-smart-cab/\)](/machine-learning-proj-smart-cab/)

# Learning Curve

**Summary:** Evaluate bias and variance with a learning curve

## Learning Curve Theory

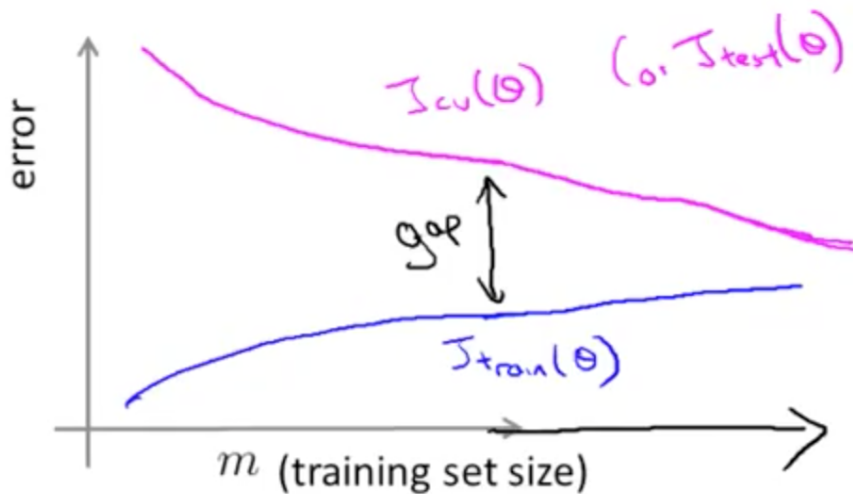
- Graph that compares the performance of a model on training and testing data over a varying number of training instances
- We should generally see performance improve as the number of training points increases
- When we separate training and testing sets and graph them individually
  - We can get an idea of how well the model can generalize to new data
- Learning curve allows us to verify when a model has learned as much as it can about the data
- When it occurs
  1. The performances on the training and testing sets reach a plateau
  2. There is a consistent gap between the two error rates
- The key is to find the sweet spot that minimizes bias and variance by finding the right level of model complexity
- Of course with more data any model can improve, and different models may be optimal
- For a more in-depth theoretical coverage of learning curves, you can view a guide by Andrew Ng that I have compiled [here](http://www.ritchieng.com/applying-machine-learning/) (<http://www.ritchieng.com/applying-machine-learning/>)

## Types of learning curves

- Bad Learning Curve: High Bias
  - When training and testing errors converge and are high
    - No matter how much data we feed the model, the model cannot represent the underlying relationship and has high systematic errors
    - Poor fit
    - Poor generalization
- Bad Learning Curve: High Variance

- When there is a large gap between the errors
  - Require data to improve
  - Can simplify the model with fewer or less complex features
- Ideal Learning Curve
  - Model that generalizes to new data
  - Testing and training learning curves converge at similar values
  - Smaller the gap, the better our model generalizes

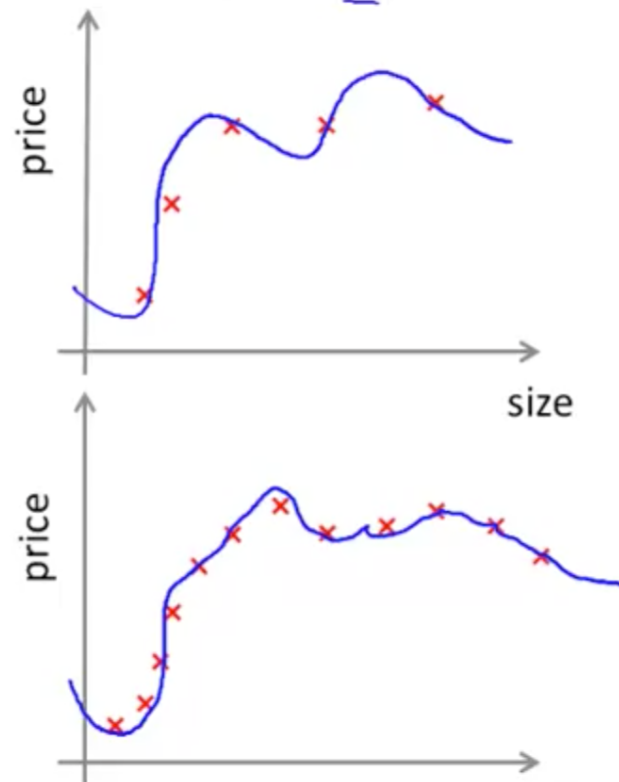
## High variance



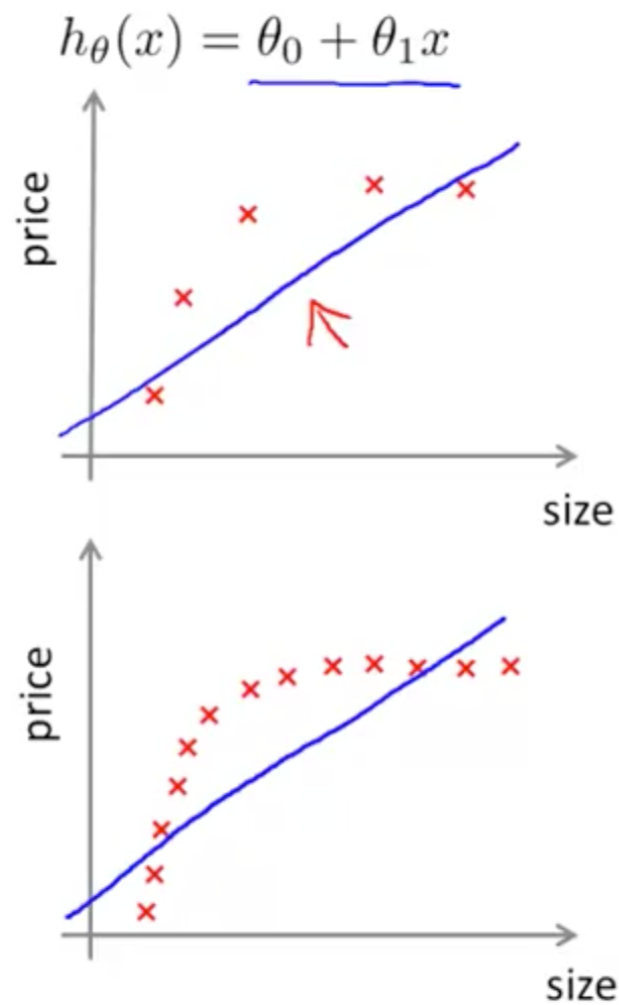
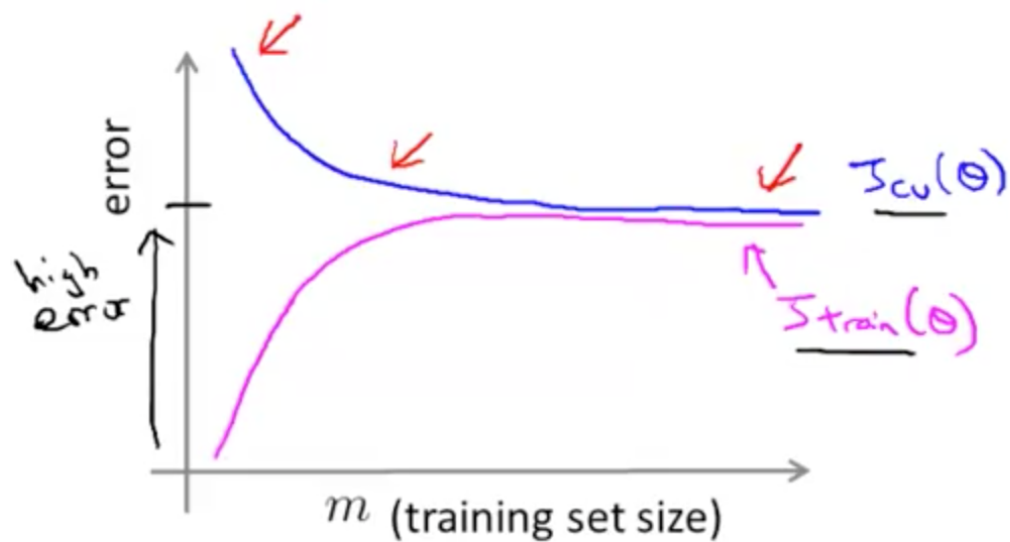
If a learning algorithm is suffering from high variance, getting more training data is likely to help.  $\leftarrow$

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \dots + \theta_{100} x^{100}$$

(and small  $\lambda$ )  $\nwarrow$



## High bias



### Example 1: High Bias

- In this example, you'll see that we'll be using a linear learner on quadratic data
- The result is that we've high bias
- We'll have a low score (high error)

```
In [1]: # imports
from sklearn.linear_model import LinearRegression
from sklearn.learning_curve import learning_curve
import matplotlib.pyplot as plt
from sklearn.metrics import explained_variance_score, make_scorer
from sklearn.cross_validation import KFold
import numpy as np
```

```
In [2]: size = 1000
cv = KFold(size, shuffle=True)
```

### Create X array

```
In [3]: #np.reshape(old_shape, new_shape)

# new array (-1, 1)
# -1 implies to take shape from original, hence 1000
# this creates a 1000 x 1 array
X = np.reshape(np.random.normal(scale=2,size=size),(-1,1))
X.shape
```

```
Out[3]: (1000, 1)
```



```
In [4]: # np.random.normal(scale=2,size=size) creates a 1000 x 1 matrix  
# scale=2 is the standard deviation of the distribution  
np.random.normal(scale=2,size=size).shape
```

Out[4]: (1000,)

### Create y array

```
In [5]: y = np.array([[1 - 2*x[0] +x[0]**2] for x in X])  
y.shape
```

Out[5]: (1000, 1)

### Plot learning curve

```
In [6]: def plot_curve():
        # instantiate
        lg = LinearRegression()

        # fit
        lg.fit(X, y)

        """
        Generate a simple plot of the test and training learning curve.

        Parameters
        -----
        estimator : object type that implements the "fit" and "predict" methods
            An object of that type which is cloned for each validation.

        title : string
            Title for the chart.

        X : array-like, shape (n_samples, n_features)
            Training vector, where n_samples is the number of samples and
            n_features is the number of features.

        y : array-like, shape (n_samples) or (n_samples, n_features), optional
            Target relative to X for classification or regression;
            None for unsupervised learning.

        ylim : tuple, shape (ymin, ymax), optional
            Defines minimum and maximum yvalues plotted.

        cv : integer, cross-validation generator, optional
            If an integer is passed, it is the number of folds (defaults to 3).
            Specific cross-validation objects can be passed, see
            sklearn.cross_validation module for the list of possible objects

        n_jobs : integer, optional
            Number of jobs to run in parallel (default 1).
```

```
x1 = np.linspace(0, 10, 8, endpoint=True) produces
8 evenly spaced points in the range 0 to 10
"""

train_sizes, train_scores, test_scores = learning_curve(lg, X, y, n_jobs=-1, cv=cv, train_sizes=np.linspace(.1
, 1.0, 5), verbose=0)

train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.figure()
plt.title("RandomForestClassifier")
plt.legend(loc="best")
plt.xlabel("Training examples")
plt.ylabel("Score")
plt.gca().invert_yaxis()

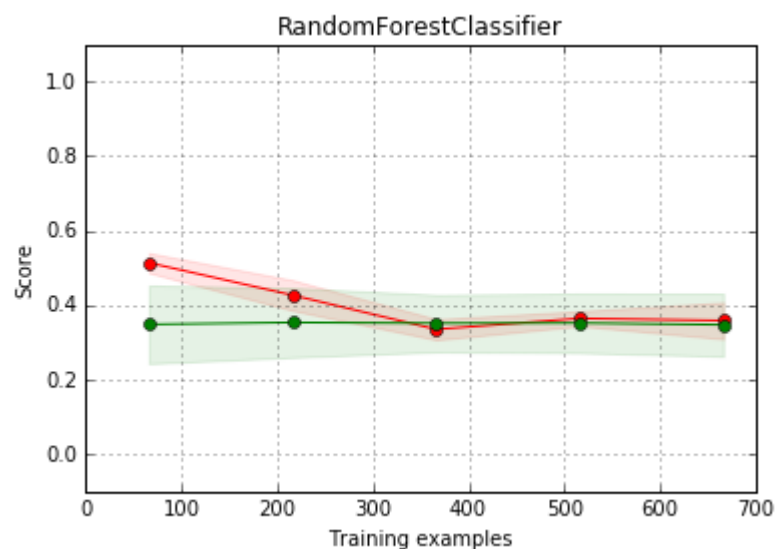
# box-like grid
plt.grid()

# plot the std deviation as a transparent range at each training set size
plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alph
a=0.1, color="r")
plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.
1, color="g")

# plot the average training and test score lines at each training set size
plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")

# sizes the window for readability and displays the plot
# shows error from 0 to 1.1
plt.ylim(-.1,1.1)
plt.show()
```

```
In [8]: %matplotlib inline  
plot_curve()
```



***Compared to the theory we covered, here our y-axis is 'score', not 'error', so the higher the score, the better the performance of the model.***

- Training score (red line) decreases and plateau
  - Indicates underfitting
  - High bias
- Cross-validation score (green line) stagnating throughout
  - Unable to learn from data
- Low scores (high errors)
  - Should tweak model (perhaps increase model complexity)

**Example 2: High Variance**

- Noisy data and complex model

There're no inline notes here as the code is exactly the same as above and are already well explained.

```
In [12]: from sklearn.tree import DecisionTreeRegressor

X = np.round(np.reshape(np.random.normal(scale=5,size=2*size),(-1,2)),2)
y = np.array([[np.sin(x[0])+np.sin(x[1])]] for x in X])

def plot_curve():
    # instantiate
    dt = DecisionTreeRegressor()

    # fit
    dt.fit(X, y)

    """
    Generate a simple plot of the test and training learning curve.

    Parameters
    -----
    estimator : object type that implements the "fit" and "predict" methods
        An object of that type which is cloned for each validation.

    title : string
        Title for the chart.

    X : array-like, shape (n_samples, n_features)
        Training vector, where n_samples is the number of samples and
        n_features is the number of features.

    y : array-like, shape (n_samples) or (n_samples, n_features), optional
        Target relative to X for classification or regression;
        None for unsupervised learning.

    ylim : tuple, shape (ymin, ymax), optional
        Defines minimum and maximum yvalues plotted.

    cv : integer, cross-validation generator, optional
        If an integer is passed, it is the number of folds (defaults to 3).
        Specific cross-validation objects can be passed, see
```

*sklearn.cross\_validation module for the list of possible objects*

*n\_jobs : integer, optional*

*Number of jobs to run in parallel (default 1).*

*x1 = np.linspace(0, 10, 8, endpoint=True) produces*

*8 evenly spaced points in the range 0 to 10*

*"""*

```
train_sizes, train_scores, test_scores = learning_curve(dt, X, y, n_jobs=-1, cv=cv, train_sizes=np.linspace(.1, 1.0, 5), verbose=0)
```

```
train_scores_mean = np.mean(train_scores, axis=1)
```

```
train_scores_std = np.std(train_scores, axis=1)
```

```
test_scores_mean = np.mean(test_scores, axis=1)
```

```
test_scores_std = np.std(test_scores, axis=1)
```

```
plt.figure()
```

```
plt.title("RandomForestClassifier")
```

```
plt.legend(loc="best")
```

```
plt.xlabel("Training examples")
```

```
plt.ylabel("Score")
```

```
plt.gca().invert_yaxis()
```

```
# box-like grid
```

```
plt.grid()
```

```
# plot the std deviation as a transparent range at each training set size
```

```
plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1, color="r")
```

```
plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean + test_scores_std, alpha=0.1, color="g")
```

```
# plot the average training and test score lines at each training set size
```

```
plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
```

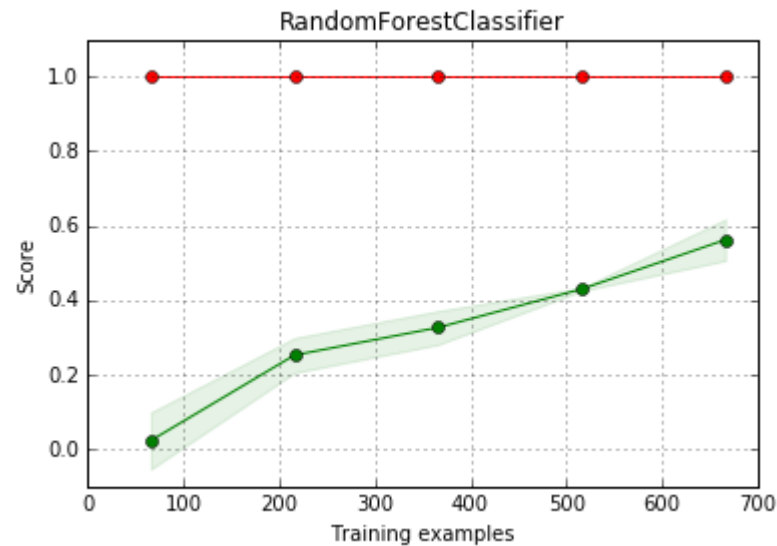
```
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")
```

```
# sizes the window for readability and displays the plot
```

```
# shows error from 0 to 1.1
```

```
plt.ylim(-.1,1.1)
plt.show()

plot_curve()
```



***Compared to the theory we covered, here our y-axis is 'score', not 'error', so the higher the score, the better the performance of the model.***

- Training score (red line) is at its maximum regardless of training examples
  - This shows severe overfitting
- Cross-validation score (green line) increases over time
- Huge gap between cross-validation score and training score indicates high variance scenario
  - Reduce complexity of the model or gather more data



Tags: machine\_learning (/tag\_machine\_learning)

3 Comments Ritchie Ng | Deep Learning & Computer Vision Engineer

1 Login ▾

♥ Recommend 3  Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



**Datasec** • a year ago

I think this is a very good article showing the practical considerations of choosing ML models.

Question: in your example where the training score is near perfect throughout (and thus indicates over-fitting). What would be the obvious next step? Choose a simpler model and bring down the training score? What is a good range for the training score?

2 ^ | ▾ • Reply • Share ›



**AYUSH JAIN** • 2 years ago

Awesome tutorial Ritchie!

1 ^ | ▾ • Reply • Share ›



**Ritchie Ng** Moderator ➔ AYUSH JAIN • 2 years ago

Thanks!

^ | ▾ • Reply • Share ›

ALSO ON RITCHIE NG | DEEP LEARNING & COMPUTER VISION ENGINEER

## Using "inplace" parameter | Machine Learning for Trading & Public Policy

3 comments • 2 years ago



**robert risk** — Thanks for explaining the inplace parameter

## Convolutional Neural Networks with TensorFlow | Machine Learning, Deep Learning, Reinforcement ...

7 comments • 2 years ago



**ayush kaul** — I have tried problem 1. I got Kernal Restart error in jupyter notebook. I am running on CPU i3 4 GB memory

## Selecting Pandas Series | Machine Learning for Trading & Public Policy

2 comments • 2 years ago



**Dominic Lawrence Mtaki** — very nice blog ,it real assist me to get familiar with panda

## I am an NVIDIA Deep Learning Institute Instructor | Machine Learning, Deep Learning, and Computer ...

1 comment • 10 months ago



**Aitor Lopez Beltran** — First of all, congratulations! Excuse me, but I'm having a hard time finding out how to get certified as a DLI Instructor. Could you help me, please? Thanks ...

---

[Subscribe](#) [Add Disqus to your site](#)[Add Disqus](#)[Add](#) [Disqus' Privacy Policy](#)[Privacy Policy](#)[Privacy Policy](#)

© 2018 Ritchie Ng. All rights reserved.

Site last updated: Sep 2, 2018

[Github \(https://github.com/ritchieng\)](https://github.com/ritchieng) | [Linkedin \(https://www.linkedin.com/in/ritchieng\)](https://www.linkedin.com/in/ritchieng) | [Facebook \(https://www.facebook.com/ritchiengz\)](https://www.facebook.com/ritchiengz) | [Twitter \(https://twitter.com/ritchieng\)](https://twitter.com/ritchieng) | [Tech in Asia \(https://www.techinasia.com/profile/ritchieng\)](https://www.techinasia.com/profile/ritchieng)