Machine Learning with Python

Sebastian Raschka, Ph.D.
MSU Data Science workshop
East Lansing, Michigan State University • Feb 21, 2018
Today’s focus:

And if we have time, a quick overview ...
Tutorial Material on GitHub:

https://github.com/rasbt/msu-datascience-ml-tutorial-2018

Contact:

- E-mail: mail@sebastianraschka.com
- Website: http://sebastianraschka.com
- Twitter: @rasbt
- GitHub: rasbt
Machine learning is used & useful (almost) anywhere
The Traditional Programming Paradigm

Inputs (observations)

Programmer $\rightarrow$ Program $\rightarrow$ Computer $\rightarrow$ Outputs

*Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed*  
– Arthur Samuel (1959)

Machine Learning

Inputs $\rightarrow$ Computer $\rightarrow$ Program

Outputs $\leftarrow$
3 Types of Learning

- Supervised
- Unsupervised
- Reinforcement
Working with Labeled Data

Supervised Learning

Regression

Classification
Working with Unlabeled Data

Unsupervised Learning

Clustering

Compression
Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning
Simple Linear Regression

\[ \hat{y} = w_0 + w_1 x \]

- \( w_0 \) (intercept)
- \( w_1 \) (slope)
- \( \Delta y / \Delta x \)
- \( |\hat{y} - y| \)
- \( (x_i, y_i) \)
- \( \Delta x \)
- \( \Delta y \)

vertical offset
# Data Representation

**Columns: features** (explanatory variables, independent variables, covariates, predictors, variables, inputs, attributes)

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>( x_1 )</th>
<th>( \ldots )</th>
<th>( x_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{0,0} )</td>
<td>( x_{0,1} )</td>
<td>( \ldots )</td>
<td>( x_{0,m} )</td>
</tr>
<tr>
<td>( x_{1,0} )</td>
<td>( x_{1,1} )</td>
<td>( \ldots )</td>
<td>( x_{1,m} )</td>
</tr>
<tr>
<td>( x_{2,0} )</td>
<td>( x_{2,1} )</td>
<td>( \ldots )</td>
<td>( x_{2,m} )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( x_{n,0} )</td>
<td>( x_{n,1} )</td>
<td>( \ldots )</td>
<td>( x_{n,m} )</td>
</tr>
</tbody>
</table>

**Rows: training examples** (observations, records, instances, samples)

**Targets** (target variable, response variable, dependent variable, labels, ground truth)

\[ X = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \quad y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \]
“Basic” Supervised Learning Workflow

1. Data
   - Labels
   - Training Data
   - Training Labels
   - Test Data
   - Test Labels

2. Training Data
   - Training Labels
   - Hyperparameter Values
   - Learning Algorithm
   - Model

3. Test Data
   - Prediction
   - Test Labels
   - Performance
   - Model

4. Data
   - Labels
   - Hyperparameter Values
   - Learning Algorithm
   - Final Model
Jupyter Notebook
Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning
Scikit-learn API

class SupervisedEstimator(...):
    def __init__(self, hyperparam, ...):
        ...
    def fit(self, X, y):
        ...
        return self
    def predict(self, X):
        ...
        return y_pred
    def score(self, X, y):
        ...
        return score
    ...
Iris Dataset

Iris-Setosa

Iris-Versicolor

Iris-Virginica
### Iris Dataset

#### Features (columns)

<table>
<thead>
<tr>
<th></th>
<th>sepal length [cm]</th>
<th>sepal width [cm]</th>
<th>petal length [cm]</th>
<th>petal width [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
</tr>
<tr>
<td>50</td>
<td>6.4</td>
<td>3.5</td>
<td>4.5</td>
<td>1.2</td>
</tr>
<tr>
<td>150</td>
<td>5.9</td>
<td>3.0</td>
<td>5.0</td>
<td>1.8</td>
</tr>
</tbody>
</table>

#### Samples (rows)

- **X=**
  - setosa
  - setosa
  - versicolor
  - virginica

- **y=**
Note about Non-Stratified Splits

- training set → 38 x Setosa, 28 x Versicolor, 34 x Virginica
- test set → 12 x Setosa, 22 x Versicolor, 16 x Virginica
Linear Regression Recap

![Diagram of Linear Regression](image)

- **Bias unit**: 1
- **Input values**: $x_1, x_2, \ldots, x_m$
- **Weight coefficients**: $w_0, w_1, w_2, \ldots, w_m$
- **Net input function**: $z = \sum w_i x_i + w_0$
- **Activation function**: $a = \sigma(z)$
- **Predicted output**: $\hat{y} = a$
Linear Regression Recap

\[ z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \mathbf{w}^\mathsf{T} \mathbf{x} \]

Here: Identity function

Bias unit

Activation function

Net input function

Predicted output
Logistic Regression, a Generalized Linear Model (a Classifier)
A “Lazy Learner:” K-Nearest Neighbors Classifier

\[
d(x^{(i)}, x^{(j)}) = \sqrt{\sum_{k} |x_{k}^{(i)} - x_{k}^{(j)}|^p}
\]
Jupyter Notebook
There are many, many more classification and regression algorithms ...

Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning
## Categorical Variables

<table>
<thead>
<tr>
<th>color</th>
<th>size</th>
<th>price</th>
<th>class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>M</td>
<td>$10.49</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>XL</td>
<td>$15.00</td>
<td>1</td>
</tr>
<tr>
<td>green</td>
<td>L</td>
<td>$12.99</td>
<td>1</td>
</tr>
</tbody>
</table>
### Encoding Categorical Variables (Ordinal vs Nominal)

<table>
<thead>
<tr>
<th>color</th>
<th>size</th>
<th>price</th>
<th>class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>M</td>
<td>$10.49</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>XL</td>
<td>$15.00</td>
<td>1</td>
</tr>
<tr>
<td>green</td>
<td>L</td>
<td>$12.99</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>color</th>
<th>size</th>
<th>price</th>
<th>class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>M</td>
<td>$10.49</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>XL</td>
<td>$15.00</td>
<td>1</td>
</tr>
<tr>
<td>green</td>
<td>L</td>
<td>$12.99</td>
<td>1</td>
</tr>
</tbody>
</table>

- **red**: 1, 0, 0
- **blue**: 0, 1, 0
- **green**: 0, 0, 1

- **size**: 0, 0, 1
- **class label**: 0, 1, 1
Feature Normalization

Min-max scaling

\[ X_{norm} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \]

Z-score standardization

\[ z = \frac{x - \mu}{\sigma} \]

<table>
<thead>
<tr>
<th>feature</th>
<th>minmax</th>
<th>z-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.0</td>
<td>-1.46385</td>
</tr>
<tr>
<td>2.0</td>
<td>0.2</td>
<td>-0.87831</td>
</tr>
<tr>
<td>3.0</td>
<td>0.4</td>
<td>-0.29277</td>
</tr>
<tr>
<td>4.0</td>
<td>0.6</td>
<td>0.29277</td>
</tr>
<tr>
<td>5.0</td>
<td>0.8</td>
<td>0.87831</td>
</tr>
<tr>
<td>6.0</td>
<td>1.0</td>
<td>1.46385</td>
</tr>
</tbody>
</table>
class UnsupervisedEstimator(...):
    def __init__(self, ...):
        ...
    def fit(self, X):
        ...
        return self
    def transform(self, X):
        ...
        return X_transf
    def predict(self, X):
        ...
        return pred
Scikit-learn Pipelines

Class labels
Training data

Pipeline

Scaling

Dimensionality Reduction

Learning Algorithm

Model

fit

fit & transform

fit & transform

fit & transform

fit

Test data

predict

transform

transform

transform

predict

Class labels

Scikit-learn Pipelines
Jupyter Notebook
Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. **Dimensionality Reduction: Feature Selection & Extraction**
6. Model Evaluation & Hyperparameter Tuning
Dimensionality Reduction – why?
Dimensionality Reduction – why?

- Predictive performance
- Storage & speed
- Visualization & interpretability
Recursive Feature Elimination

available features: [ f1  f2  f3  f4 ]

[ w1  w2  w3  w4 ]

fit model, remove lowest weight, repeat

[ w1  w2  w4 ]

fit model, remove lowest weight, repeat

[ w1  w4 ]

fit model, remove lowest weight, repeat

[ w4 ]

fit model, remove lowest weight, repeat
Sequential Feature Selection

available features: [ f1  f2  f3  f4 ]

- [ f1 ]
- [ f1  f3 ]
- [ f1  f3  f4 ]
- [ f1  f2 ]
- [ f1  f4 ]
- [ f1  f3  f2 ]

fit model, pick best, repeat

fit model, pick best, repeat
Principal Component Analysis
Jupyter Notebook
Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning
“Basic” Supervised Learning Workflow

1. Data
   - Training Data
   - Training Labels
   - Test Data
   - Test Labels

2. Hyperparameter Values
   - Learning Algorithm
   - Model

3. Prediction
   - Test Data
   - Model
   - Performance
   - Test Labels

4. Hyperparameter Values
   - Final Model
   - Learning Algorithm
   - Data
   - Labels
Holdout Method and Hyperparameter Tuning 1-3
Holdout Method and Hyperparameter Tuning

4. Training Data → Validation Data → Validation Labels → Best Hyperparameter Values → Learning Algorithm → Model

5. Test Data → Model → Prediction → Performance → Test Labels

6. Data → Labels → Best Hyperparameter Values → Learning Algorithm → Final Model

This work by Sebastian Raschka is licensed under a Creative Commons Attribution 4.0 International License.
K-fold Cross-Validation

Learning Algorithm

Hyperparameter Values

Model

Training Fold Data
Training Fold Labels

Validation Fold Data
Validation Fold Labels

Performance

Prediction

Performance

Model

\[ \text{Performance} = \frac{1}{K} \sum_{i=1}^{K} \text{Performance}_i \]
K-fold Cross-Validation Workflow 1-3

1. Data
   - Training Data
   - Test Data
   - Training Labels
   - Test Labels

2. 
   - Training Data
   - Training Labels
   - Hyperparameter values

3. 
   - Training Data
   - Training Labels
   - Best Hyperparameter Values
   - Learning Algorithm

Final Model
K-fold Cross-Validation Workflow 4-5

4

5

Test Data → Prediction → Performance

Model

Test Labels

Data

Labels

Best Hyperparameter Values

Learning Algorithm

Final Model
More info about model evaluation (one of the most important topics in ML):

https://sebastianraschka.com/blog/index.html

- Model evaluation, model selection, and algorithm selection in machine learning Part I - The basics
- Model evaluation, model selection, and algorithm selection in machine learning Part II - Bootstrapping and uncertainties
- Model evaluation, model selection, and algorithm selection in machine learning Part III - Cross-validation and hyperparameter tuning
Jupyter Notebook
BONUS SLIDES
TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)


Google Research*

While TensorFlow can be run entirely on a CPU or multiple CPUs, one of the core strengths of this library is its support of GPUs (Graphical Processing Units) that are very efficient at performing highly parallelized numerical computations. In addition, TensorFlow also supports distributed systems as well as mobile computing platforms, including Android and Apple’s iOS.

But what is a tensor? In simplifying terms, we can think of tensors as multidimensional arrays of numbers, as a generalization of scalars, vectors, and matrices.

1. Scalar: \( \mathbb{R} \)
2. Vector: \( \mathbb{R}^n \)
3. Matrix: \( \mathbb{R}^n \times \mathbb{R}^m \)
4. 3-Tensor: \( \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \)
5. …

When we describe tensors, we refer to its “dimensions” as the rank (or order) of a tensor, which is not to be confused with the dimensions of a matrix. For instance, an \( m \times n \) matrix, where \( m \) is the number of rows and \( n \) is the number of columns, would be a special case of a rank-2 tensor. A visual explanation of tensors and their ranks is given below.

https://sebastianraschka.com/pdf/books/dlb/appendix_g_tensorflow.pdf
## GPUs

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Intel® Core™ i7-6900K Processor Extreme Ed.</th>
<th>NVIDIA GeForce® GTX™ 1080 Ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Clock Frequency</td>
<td>3.2 GHz</td>
<td>&lt; 1.5 GHz</td>
</tr>
<tr>
<td>Cores</td>
<td>8</td>
<td>3584</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>64 GB/s</td>
<td>484 GB/s</td>
</tr>
<tr>
<td>Floating-Point Calculations</td>
<td>409 GFLOPS</td>
<td>11300 GFLOPS</td>
</tr>
<tr>
<td>Cost</td>
<td>~ $1000.00</td>
<td>~ $700.00</td>
</tr>
</tbody>
</table>
Vectorization

\[
X = \text{np.random.random((num_train_examples, num_features))}
\]

\[
W = \text{np.random.random((num_features, num_hidden))}
\]
Vectorization

\[
\begin{array}{c}
\text{Green} \\
\text{Blue} \\
\text{Orange}
\end{array}
\times
\begin{array}{c}
\text{White} \\
\text{White} \\
\text{White}
\end{array}
=\begin{array}{c}
\text{White} \\
\text{White} \\
\text{White}
\end{array}
\]
Computation Graphs

\[ a(x, w, b) = \text{relu}(w \cdot x + b) \]

\[ u = wx \]

\[ v = u + b \]

\[ a = \text{relu}(v) \]
Computation Graphs

```python
import tensorflow as tf

g = tf.Graph()
with g.as_default() as g:
    x = tf.placeholder(dtype=tf.float32, shape=None, name='x')
    w = tf.Variable(initial_value=2, dtype=tf.float32, name='w')
    b = tf.Variable(initial_value=1, dtype=tf.float32, name='b')

    u = x * w
    v = u + b
    a = tf.nn.relu(v)

print(x, w, b, u, v, a)
```

Tensor("x:0", dtype=float32) <tf.Variable 'w:0' shape=() dtype=float32_ref> <tf.Variable 'b:0' shape=() dtype=float32_ref> Tensor("mul:0", dtype=float32) Tensor("add:0", dtype=float32) Tensor("Relu:0", dtype=float32)
Computation Graphs

\[
\begin{align*}
    b &= 1 \\
    x \\
    w &= 2 \\
    u &= wx \\
    v &= u + b \\
    a &= \text{relu}(v)
\end{align*}
\]

with tf.Session(graph=g) as sess:
    sess.run(init_op)
    b_res = sess.run('b:0')

print(b_res)

1.0
\[
\frac{\partial a}{\partial b} = \frac{\partial v}{\partial b} = 1 \\
\frac{\partial v}{\partial b} = 1 \\
\frac{da}{dv} = 1 \\
\frac{da}{wb} = 1 \\
\frac{\partial v}{\partial w} = 3 \\
\frac{\partial u}{\partial w} = 3 \\
\frac{\partial a}{\partial w} = \frac{\partial u}{\partial w} \frac{\partial a}{\partial u} = 3 \times 1 \times 1 = 3 \\
\]

\[u = wx \]
\[v = u + b \]
\[a = \text{relu}(v) \]

https://github.com/rasbt/pydata-annarbor2017-dl-tutorial
```python
with tf.Session(graph=g) as sess:
    sess.run(tf.global_variables_initializer())
    res = sess.run([d_a_w, d_b_w], feed_dict={'x:0': 3})
```

```
[3.0] [1.0]
```
```python
import torch
import torch.nn.functional as F
from torch.autograd import Variable
from torch.autograd import grad

x = Variable(torch.Tensor([[3]]))
w = Variable(torch.Tensor([[2]]), requires_grad=True)
b = Variable(torch.Tensor([[1]]), requires_grad=True)

u = x * w
v = u + b
a = F.relu(v)

partial_derivatives = grad(a, (w, b))

for name, grad in zip("wb", (partial_derivatives)):
    print('d_a_%s: %s

61
```
Multilayer Perceptron
```python
g = tf.Graph()
with g.as_default():

    # Input data
    tf_x = tf.placeholder(tf.float32, [None, n_input], name='features')
    tf_y = tf.placeholder(tf.float32, [None, n_classes], name='targets')

    # Model parameters
    weights = {
        'hl1': tf.Variable(tf.truncated_normal([n_input, n_hidden_1], stddev=0.1)),
        'out': tf.Variable(tf.truncated_normal([n_hidden_2, n_classes], stddev=0.1))
    }
    biases = {
        'b1': tf.Variable(tf.zeros([n_hidden_1])),
        'out': tf.Variable(tf.zeros([n_classes]))
    }

    # Multilayer perceptron
    layer_1 = tf.add(tf.matmul(tf_x, weights['hl1']), biases['b1'])
    layer_1 = tf.nn.relu(layer_1)
    out_layer = tf.matmul(layer_1, weights['out']) + biases['out']

    # Loss and optimizer
    loss = tf.nn.softmax_cross_entropy_with_logits(logits=out_layer, labels=tf_y)
    cost = tf.reduce_mean(loss, name='cost')
    optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
    train = optimizer.minimize(cost, name='train')

    # Prediction
    correct_prediction = tf.equal(tf.argmax(tf_y, 1), tf.argmax(out_layer, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32), name='accuracy')

with tf.Session(graph=g) as sess:
sess.run(tf.global_variables_initializer())

for epoch in range(training_epochs):
    avg_cost = 0.
    total_batch = mnist.train.num_examples // batch_size

    for i in range(total_batch):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        _, c = sess.run(['train', 'cost:0'], feed_dict={
            'features:0': batch_x,
            'targets:0': batch_y})

    if torch.cuda.is_available():
        model.cuda()

for epoch in range(num_epochs):
    for batch_idx, (features, targets) in enumerate(train_loader):
        features = Variable(features.view(-1, 28*28))
        targets = Variable(targets)

        if torch.cuda.is_available():
            features, targets = features.cuda(), targets.cuda()

        logits, probs = model(features)
        cost = cost_fn(logits, targets)
        optimizer.zero_grad()

        cost.backward()

        optimizer.step()
```

```
Further Resources

- Math-heavy
- Math-free scikit-learn intro
- Mix of code & math (~60% scikit-learn)
Thanks for attending!

Tutorial Material on GitHub:

https://github.com/rasbt/msu-datascience-ml-tutorial-2018

Contact:

- E-mail: mail@sebastianraschka.com
- Website: http://sebastianraschka.com
- Twitter: @rasbt
- GitHub: rasbt