Big Data and Machine Learning with Python

Michael E. Rose, PhD

Course at LMU, March 2019

Supervised Machine Learning

Examples in Economics

- Monica Andini et al. (2018), "Targeting with machine learning: An application to a tax rebate program in Italy", Journal of Economic Behavior and Organization 156.
- Joshua Blumenstock et al. (2015): "Predicting Poverty and Wealth from Mobile Phone Metadata," Science, 350(6264).
- Neal Jean et al. (2016): "Combining satellite imagery and machine learning to predict poverty," Science 353(6301).

How does it work?

- Making predictions of <u>known</u> variable from provided dataset

How does it work?

- Making predictions of <u>known</u> variable from provided dataset

1. Split sample randomly
2. Train algorithm on training set
3. Evaluate on test set (= "generalization")
4. (Tweak model parameters, repeat 2 and 3)
5. Use on unseen data

Translation: Econometrics to Machine Learning

Term in Econometrics	Term in Machine Learning
Variable	Feature
Variable construction	Feature engineering
fit	learn
coefficient	weight
Non-binary regression	Prediction
Binary regression	Classification
Dummy	One-hot encoding
Fit	Learn

1. Split sample randomly

1. Use function `train_test_split()` (→ Documentation)
2. Two mandatory parameters: Data (`X`) and labels or targets (`y`)

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

2. Train algorithm on training set

1. With `sklearn`, there is one class for each algorithm
2. Class API always the same: Initiate object with algorithm parameters, `.fit()` on it

```
from sklearn.neighbors import KNeighborsClassifier
knc = KNeighborsClassifier(n_neighbors=1)
knc.fit(X_train, y_train)
```

3. Evaluate on test set

1. Test set is data with labels or targets, not used for training
2. `sklearn` provides all evluation measures
3. Standard score is *accuracy score*: Number of correct predictions divided by the number of all samples

```
print(knc.score(X_test, y_test))
```

4. Tweak model parameters, repeat 2 and 3

```
from sklearn.neighbors import KNeighborsClassifier
knc = KNeighborsClassifier(n_neighbors=3)
knc.fit(X_train, y_train)
print(knc.score(X_test, y_test))
```

5. Predict labels of unseen data

```
y_pred = knc.predict(X_test)
print(y_pred)
```

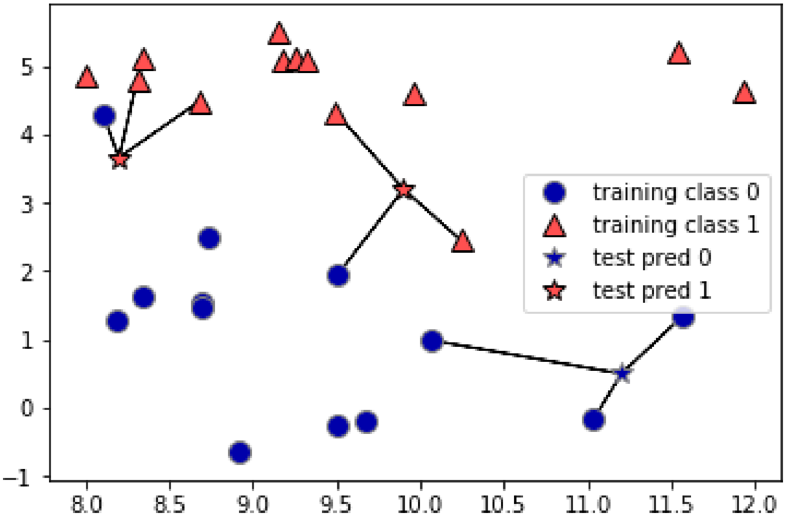
k-Nearest Neighbor

- 2 parameters:
  1. How many neighbors?
  2. How to measure distance?

\+ Easy to understand

\- Slow on large set and often preprocessing necessary

k-Nearest Neighbor, cont.

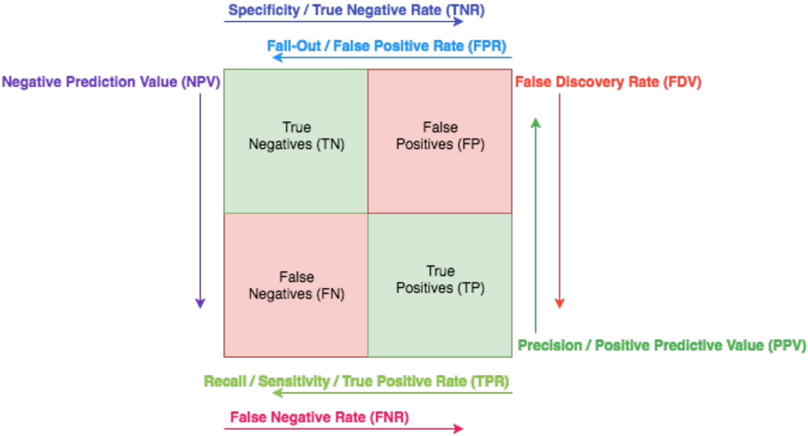


from: Andreas Müller and Sarah Guido (2016): Introduction to Machine Learning with Python, O'Reilly

What is distance?

- Multiple ways to compute distance between two points in multi-dimensional space
- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

Confusion matrix



from: Sanyam Kapoor (2017): "Visualizing the Confusion Matrix"

Precision and Recall

- Precision
  - What proportion of positive *identifications* was actually correct?
  - $\frac{TP}{FP+TP}$

Precision and Recall

- Precision
  - What proportion of positive *identifications* was actually correct?
  - $\frac{TP}{FP+TP}$
- Recall
  - What proportion of *actual positives* was identified correctly?
  - $\frac{TP}{TP+FN}$

Precision and Recall

- Precision
  - What proportion of positive *identifications* was actually correct?
  - $\frac{TP}{FP+TP}$
- Recall
  - What proportion of *actual positives* was identified correctly?
  - $\frac{TP}{TP+FN}$

What happens when you predict all entries to be in one class?

Other measures

- f-score
  - Harmonic mean of precision and recall: $2 \times \frac{\mathit{precision} \times \mathit{recall}}{\mathit{precision} + \mathit{recall}}$
- See sklearn documentation

Linear models

- 2 parameters:
  1. How complex should the model be?
  2. Which regularization?

\+ Fast and easy

\- Sometimes intransparent

Variance-Bias-Tradeoff

- Both Variance and Bias of an estimator are desired to be low
- OLS is unbiased but has huge variance, specifically when
  - ... features are highly correlated with each other
  - ... there are many predictors

$\rightarrow$ Regularization: Reduce variance at the cost of introducing some bias!

Regularization

- Ridge (L2 regularization)
  - To push coefficients towards 0

$$L_{\text{ridge}}(\hat{\beta}) = \sum_{i=1}^{N}(y_i - x'\hat{\beta})^2 + \lambda \sum_{j=1}^{m} \hat{\beta}_j^2$$

- Lasso (L1 regularization)
  - To reduce some coefficients to 0

$$L_{\text{lasso}}(\hat{\beta}) = \sum_{i=1}^{N}(y_i - x'\hat{\beta})^2 + \lambda \sum_{j=1}^{m} |\hat{\beta}_j|$$