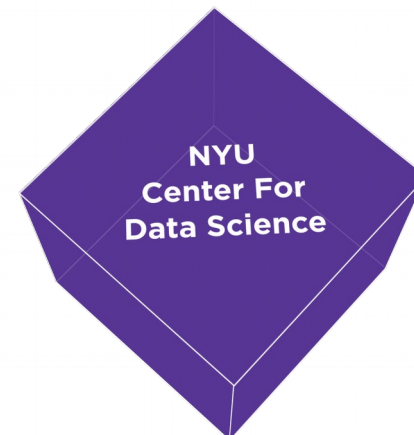
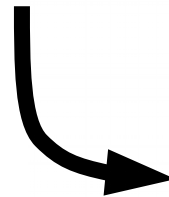
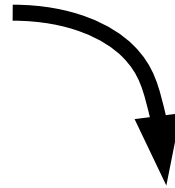




Advanced Scikit-Learn

Andreas Mueller (NYU Center for Data Science, scikit-learn)

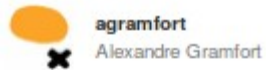
Me



Classification
Regression
Clustering
Semi-Supervised Learning
Feature Selection
Feature Extraction
Manifold Learning
Dimensionality Reduction
Kernel Approximation
Hyperparameter Optimization
Evaluation Metrics
Out-of-core learning

.....





agramfort
Alexandre Gramfort



AlexanderFabisch
Alexander Fabisch



alextp
Alexandre Passos



amueller
Andreas Mueller



arjoly
Arnaud Joly



bdholt1
Brian Holt



GaelVaroquaux
Gael Varoquaux



glouppe
Gilles Louppe



jakevdp
Jake Vanderplas



jaquesgrobler
Jaques Grobler



jnothman



kastnerkyle
Kyle Kastner



bthirion
bthirion



chrisfilo
Chris Filo Gorgole...



cournape
David Cournapeau



duchesnay
Duchesnay



dwf
David Warde-Farley



fabianp
Fabian Pedregosa



kuantkid
Wei Li



larsmans
Lars



lucidfrontier45
Shiqiao Du



mblondel
Mathieu Blondel



MechCoder
Manoj Kumar



ndawe
Noel Dawe



NelleV
Varoquaux



ogrisel
Olivier Grisel



paolo-losi
Paolo Losi



pprett
Peter Prettenhofer



robertlayton
Robert Layton



ronw
Ron Weiss



satra
Satrajit Ghosh



sklearn-ci



vene
Vlad Niculae



VirgileFritsch
Virgile Fritsch



vmichel
Vincent Michel



yarikoptic
Yaroslav Halchenko



Overview

- Reminder: Basic scikit-learn concepts
- Working with text data
- Model building and evaluation:
 - Pipelines
 - Randomized Parameter Search
 - Scoring Interface
- Out of Core learning
 - Feature Hashing
 - Kernel Approximation
- New stuff in 0.17 and 0.18-dev
 - Overview
 - Calibration

Documentation of scikit-learn 0.17

Quick Start

A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

User Guide

The main documentation. This contains an in-depth description of all algorithms and how to apply them.

Other Versions

- [scikit-learn 0.18 \(development\)](#)
- [scikit-learn 0.17 \(stable\)](#)
- [scikit-learn 0.16](#)
- [scikit-learn 0.15](#)

Tutorials

Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

API

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Additional Resources

Talks given, slide-sets and other information relevant to scikit-learn.

Contributing

Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.

Flow Chart

A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.

FAQ

Frequently asked questions about the project and contributing.

<http://scikit-learn.org/>

Representing Data

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

Representing Data

one sample

$X =$

1.1	2.2	3.4	5.6	1.0
6.7	0.5	0.4	2.6	1.6
2.4	9.3	7.3	6.4	2.8
1.5	0.0	4.3	8.3	3.4
0.5	3.5	8.1	3.6	4.6
5.1	9.7	3.5	7.9	5.1
3.7	7.8	2.6	3.2	6.3

Representing Data

one sample

$X =$

1.1	2.2	3.4	5.6	1.0
6.7	0.5	0.4	2.6	1.6
2.4	9.3	7.3	6.4	2.8
1.5	0.0	4.3	8.3	3.4
0.5	3.5	8.1	3.6	4.6
5.1	9.7	3.5	7.9	5.1
3.7	7.8	2.6	3.2	6.3

one feature

Representing Data

one sample

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

one feature

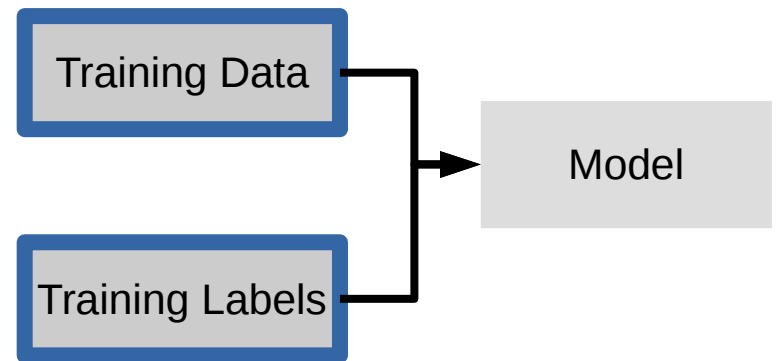
$$y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

outputs / labels

Supervised Machine Learning

```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

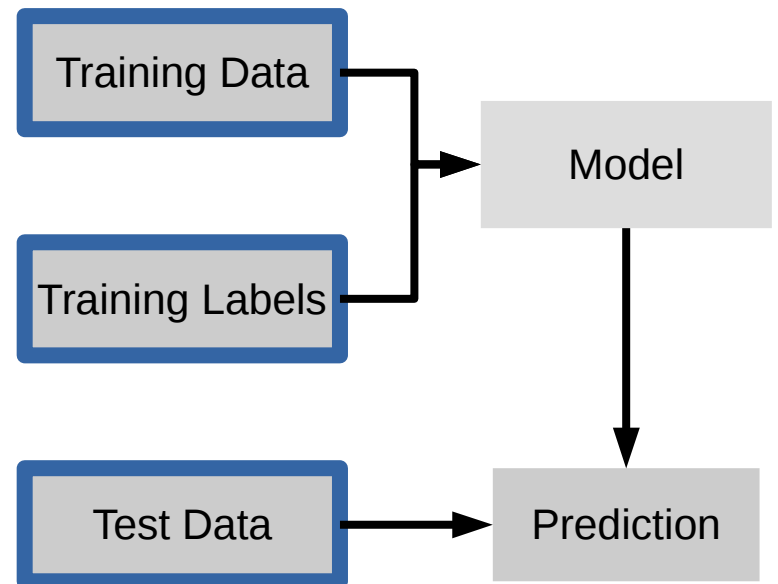


Supervised Machine Learning

```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
```



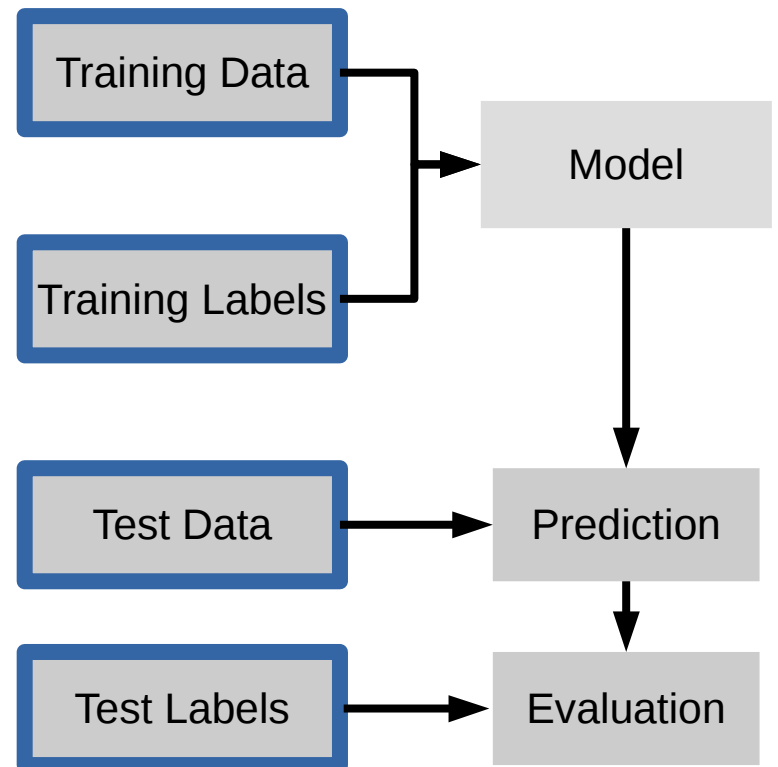
Supervised Machine Learning

```
clf = RandomForestClassifier()
```

```
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
```

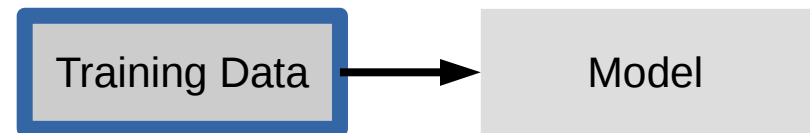
```
clf.score(X_test, y_test)
```



Unsupervised Transformations

```
pca = PCA(n_components=3)
```

```
pca.fit(X_train)
```

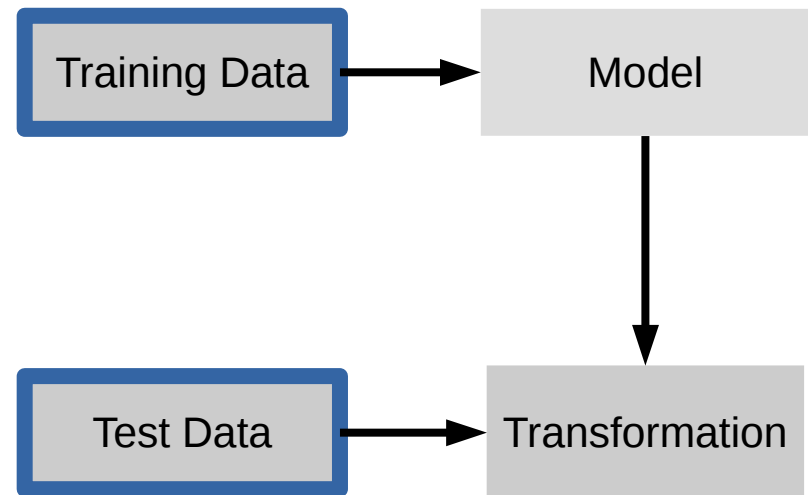


Unsupervised Transformations

```
pca = PCA(n_components=3)
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```



Basic API

```
estimator.fit(X, [y])
```

```
estimator.predict
```

```
estimator.transform
```

Classification

Preprocessing

Regression

Dimensionality reduction

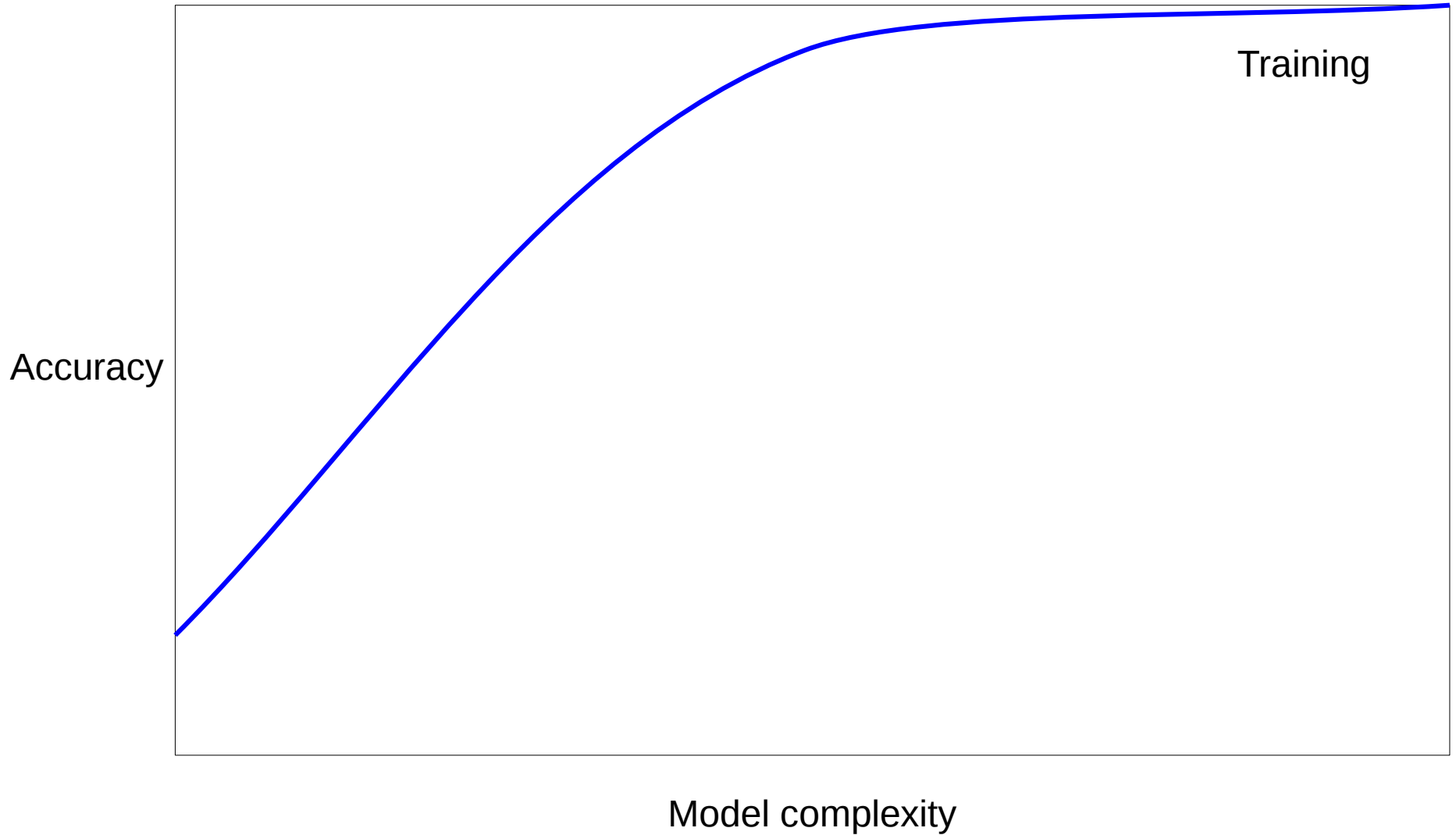
Clustering

Feature selection

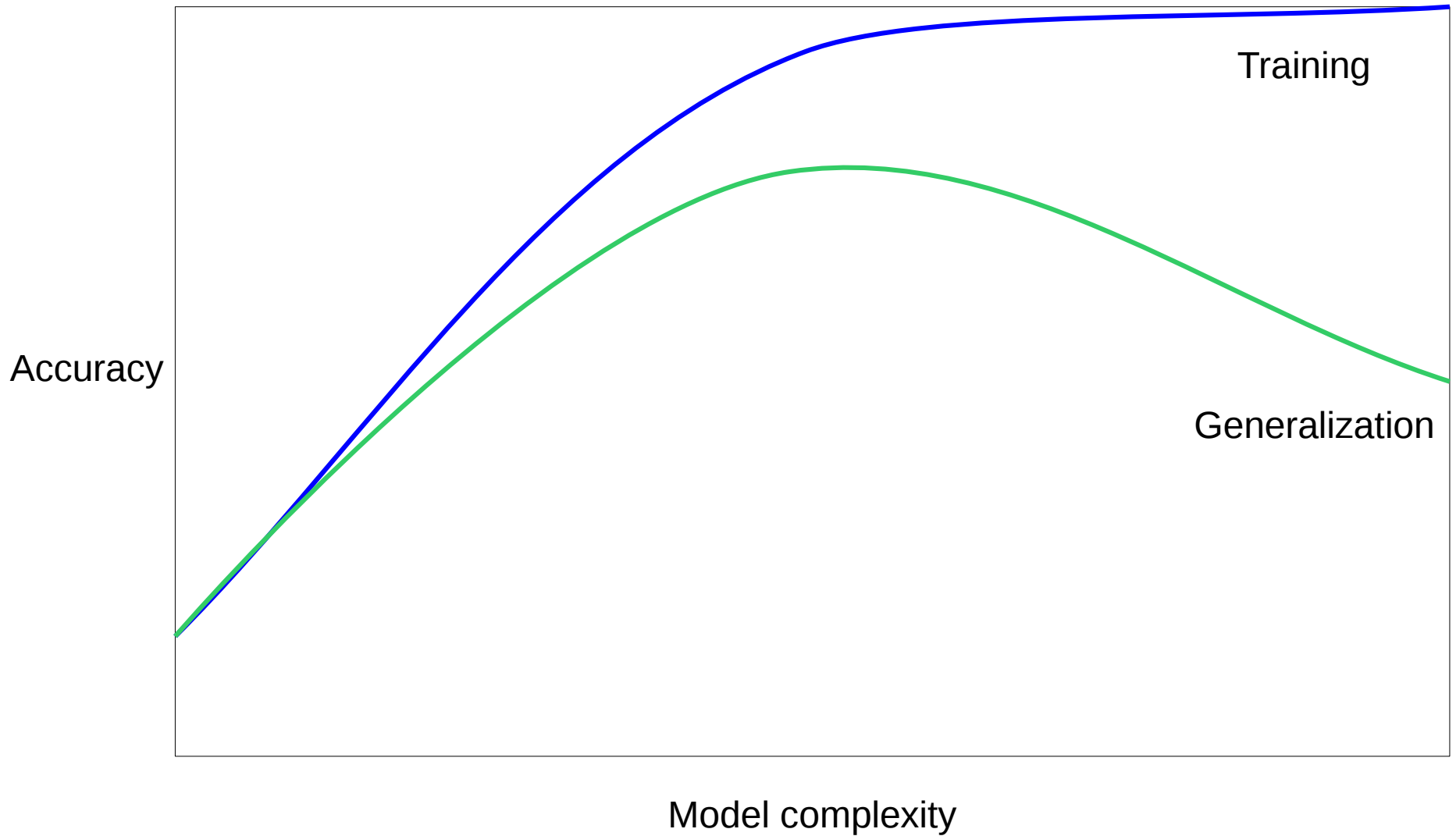
Feature extraction

Model selection and model complexity (aka bias-variance tradeoff)

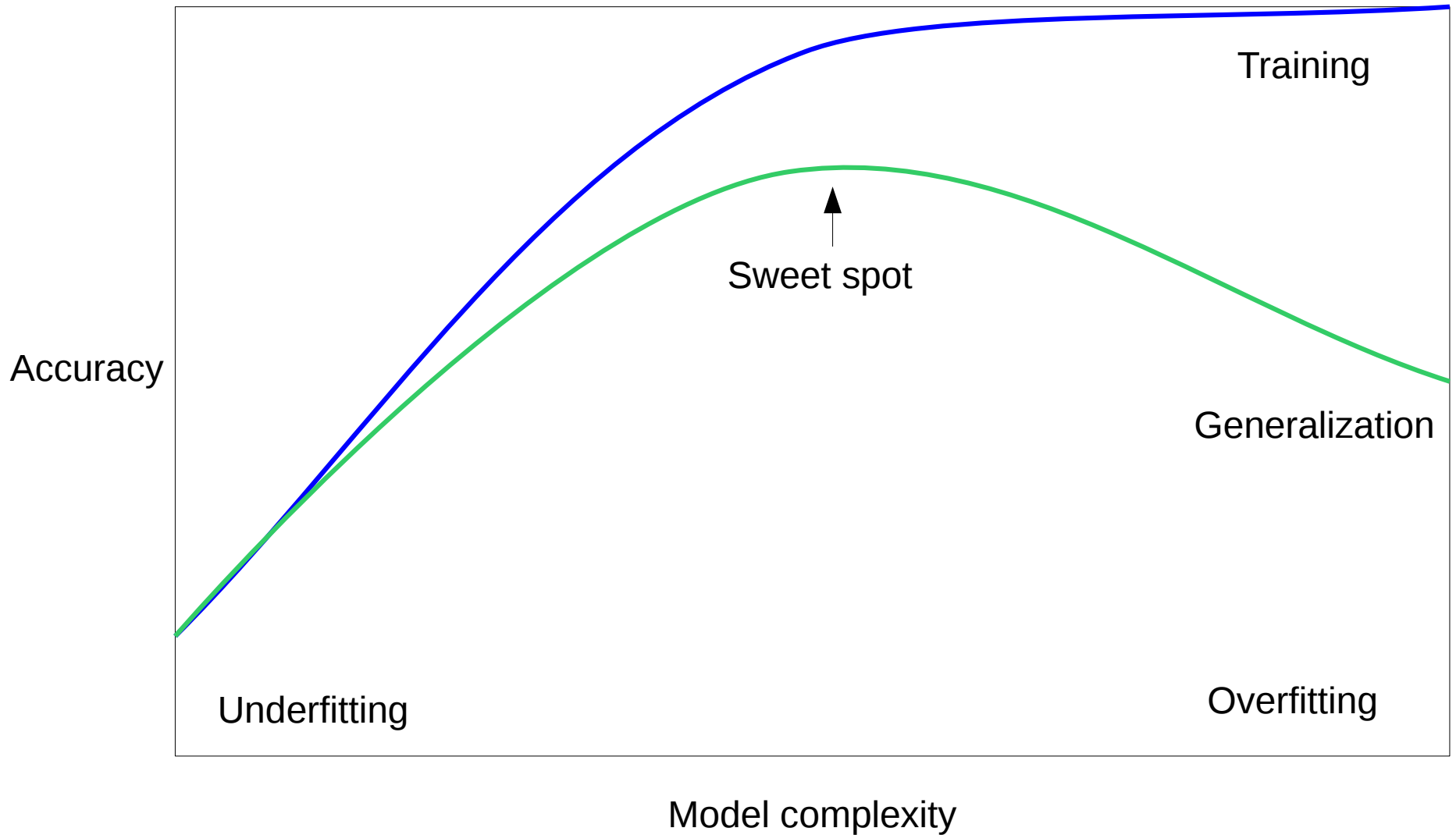
Overfitting and Underfitting



Overfitting and Underfitting



Overfitting and Underfitting



Cross-Validation

```
from sklearn.cross_validation import cross_val_score  
  
scores = cross_val_score(SVC(), X, y, cv=5)  
print(scores)  
  
>> [ 0.92  1.    1.    1.    1. ]
```

Cross-Validation

```
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(SVC(), X, y, cv=5)
print(scores)

>> [ 0.92  1.    1.    1.    1. ]

cv_ss = ShuffleSplit(len(X_train), test_size=.3,
                    n_iter=10)
scores_shuffle_split = cross_val_score(SVC(), X, y,
                                       cv=cv_ss)
```

Cross-Validation

```
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(SVC(), X, y, cv=5)
print(scores)

>> [ 0.92  1.    1.    1.    1. ]

cv_ss = ShuffleSplit(len(X_train), test_size=.3,
                    n_iter=10)
scores_shuffle_split = cross_val_score(SVC(), X, y,
                                       cv=cv_ss)

cv_labels = LeaveOneLabelOut(labels)
scores_pout = cross_val_score(SVC(), X, y, cv=cv_labels)
```

Cross -Validated Grid Search

```
In [2]: clf = SVC()  
        clf.fit(X_train, y_train)
```

```
SVC(self, C=1.0, kernel='rbf', degree=3, gamma=0.0, coef0=0.0,  
     shrinking=True, probability=False, tol=0.001, cache_size=200,  
     class_weight=None, verbose=False, max_iter=-1, random_state=None)
```


All Data

Training data

Test data

All Data

Training data Test data

Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 1 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 2 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 3 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 4 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 5 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Test data

All Data

Training data Test data

Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Finding Parameters

Final evaluation

Test data

Cross -Validated Grid Search

```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.predict(X_test)
grid.score(X_test, y_test)
```

Sample application: Sentiment Analysis

IMDB Movie Reviews Data

Review:

One of the worst movies I've ever rented. Sorry it had one of my favorite actors on it (Travolta) in a nonsense role. In fact, anything made sense in this movie.

Who can say there was true love between Eddy and Maureen?
Don't you remember the beginning of the movie ?

Is she so lovely? Ask her daughters. I don't think so.

Label: negative

Training data: 12500 positive, 12500 negative

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

`"This is how you get ants."`

Bag Of Word Representations

CountVectorizer / TfidfVectorizer

"This is how you get ants."

tokenizer



['this', 'is', 'how', 'you', 'get', 'ants']

Bag Of Word Representations

CountVectorizer / TfidfVectorizer

"This is how you get ants."

tokenizer

['this', 'is', 'how', 'you', 'get', 'ants']

Build a vocabulary over all documents

['aardvak', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']

Bag Of Word Representations

CountVectorizer / TfidfVectorizer

"This is how you get ants."

tokenizer

['this', 'is', 'how', 'you', 'get', 'ants']

Build a vocabulary over all documents

['aardvak', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']

Sparse matrix encoding

aardvak	ants	get	you	zyxst
[0, ..., 0, 1, 0, ... , 0, 1 , 0, ..., 0, 1, 0,	0]			

N-grams (unigrams and bigrams)

`CountVectorizer` / `TfidfVectorizer`

N-grams (unigrams and bigrams)

`CountVectorizer / TfidfVectorizer`

`"This is how you get ants."`

N-grams (unigrams and bigrams)

`CountVectorizer` / `TfidfVectorizer`

"This is how you get ants."

Unigram tokenizer



`['this', 'is', 'how', 'you', 'get', 'ants']`

N-grams (unigrams and bigrams)

CountVectorizer / TfidfVectorizer

"This is how you get ants."

Unigram tokenizer

↓
['this', 'is', 'how', 'you', 'get', 'ants']

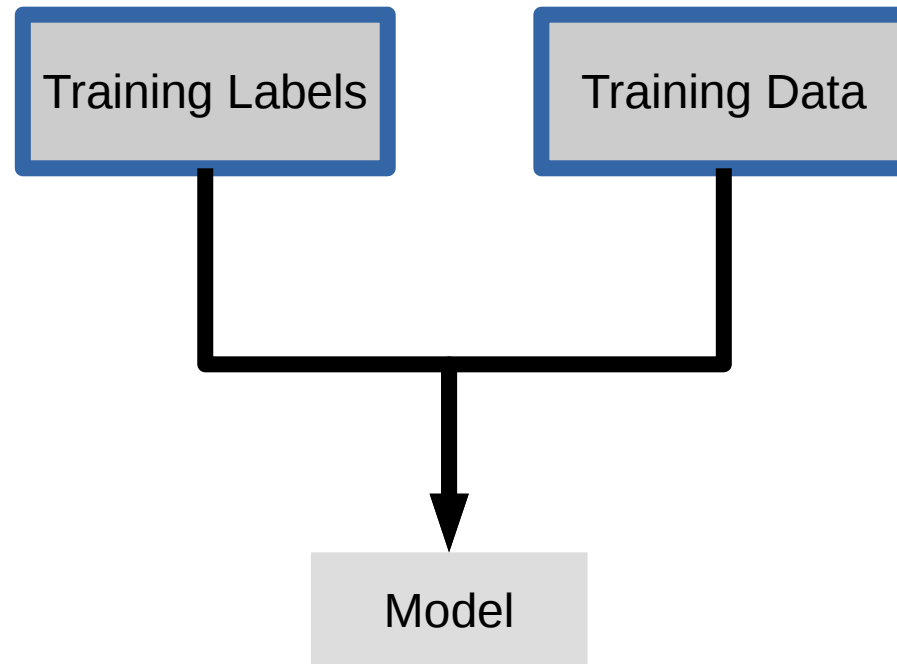
"This is how you get ants."

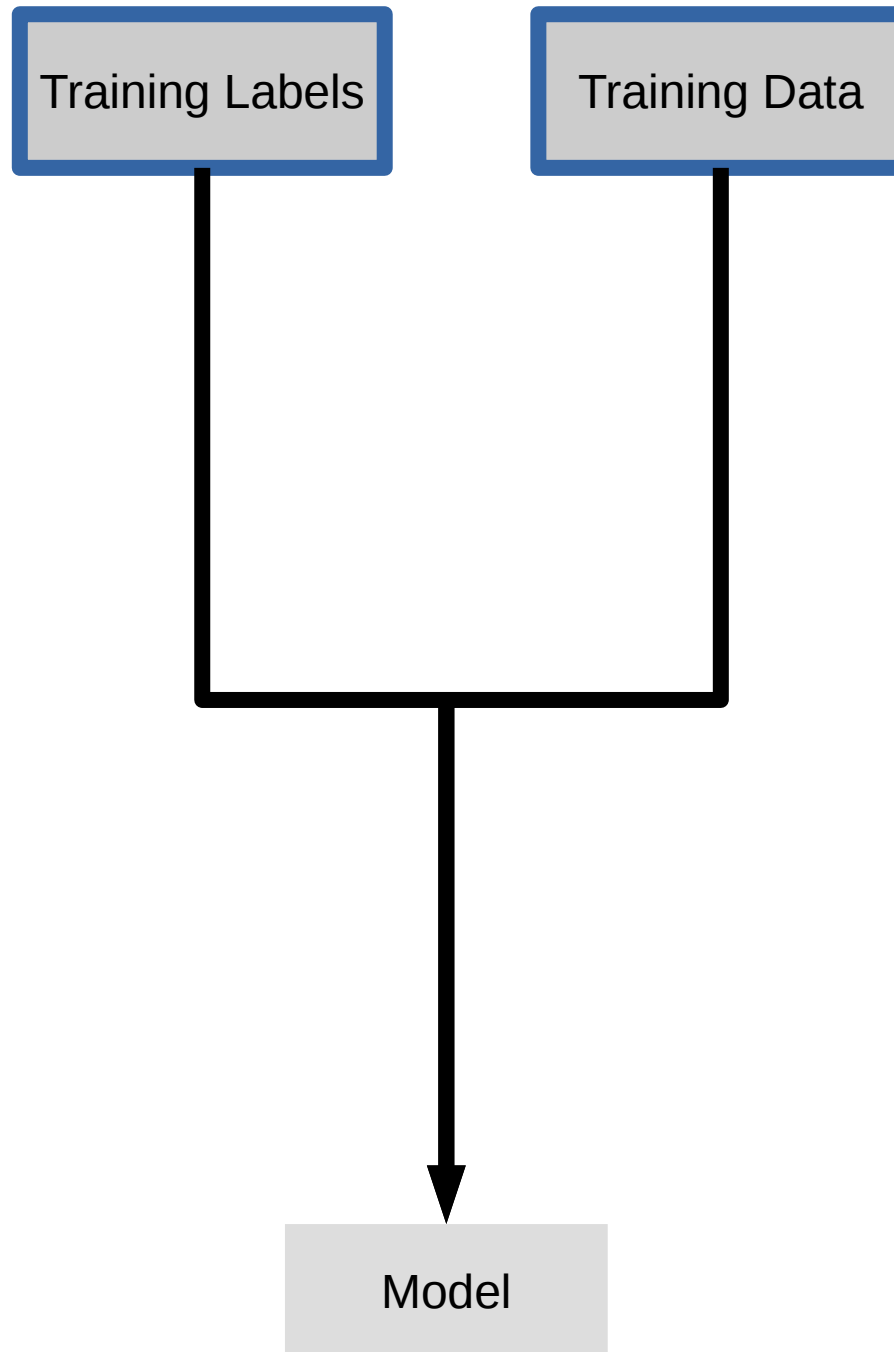
Bigram tokenizer

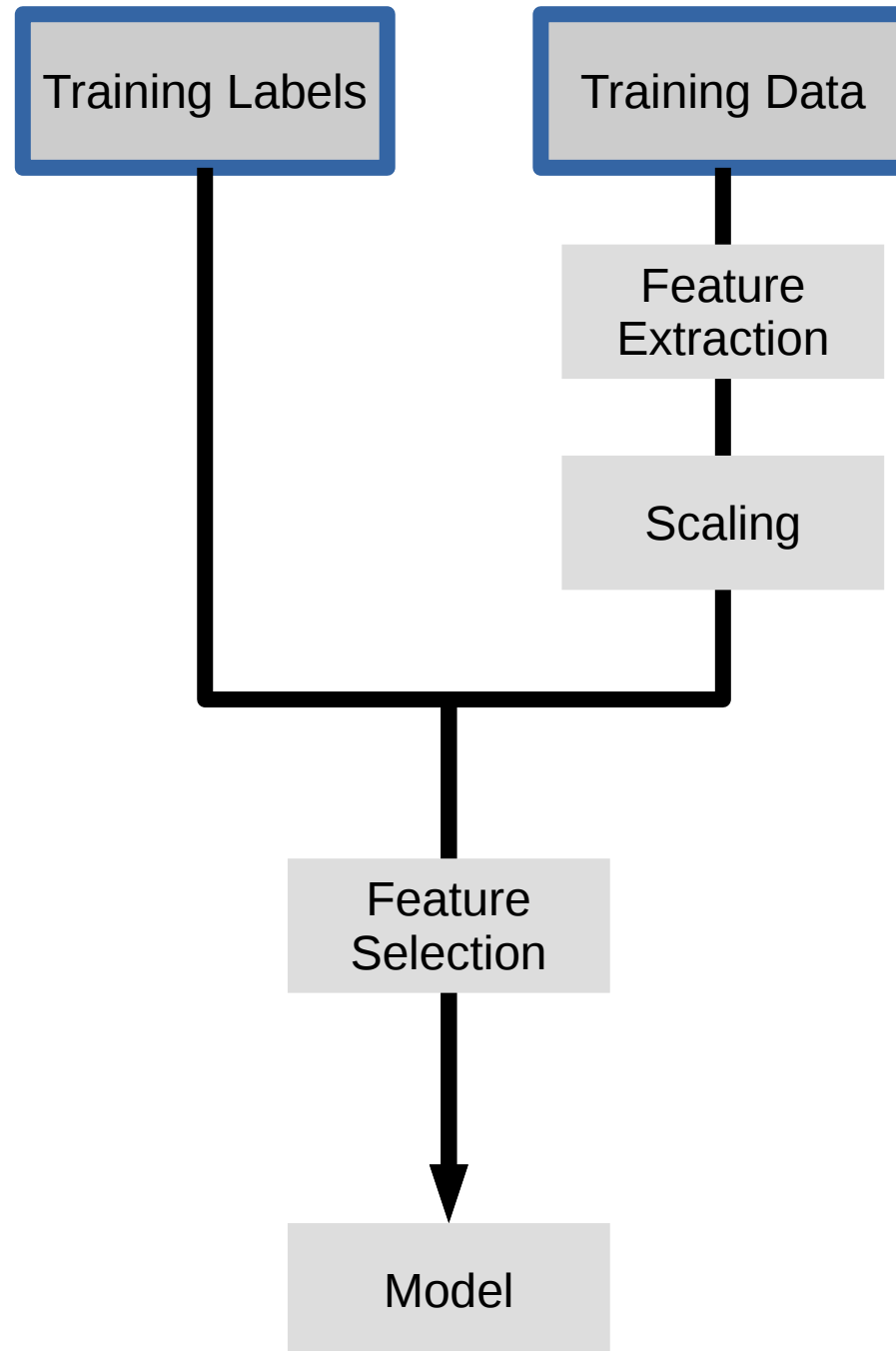
↓
['this is', 'is how', 'how you', 'you get', 'get ants']

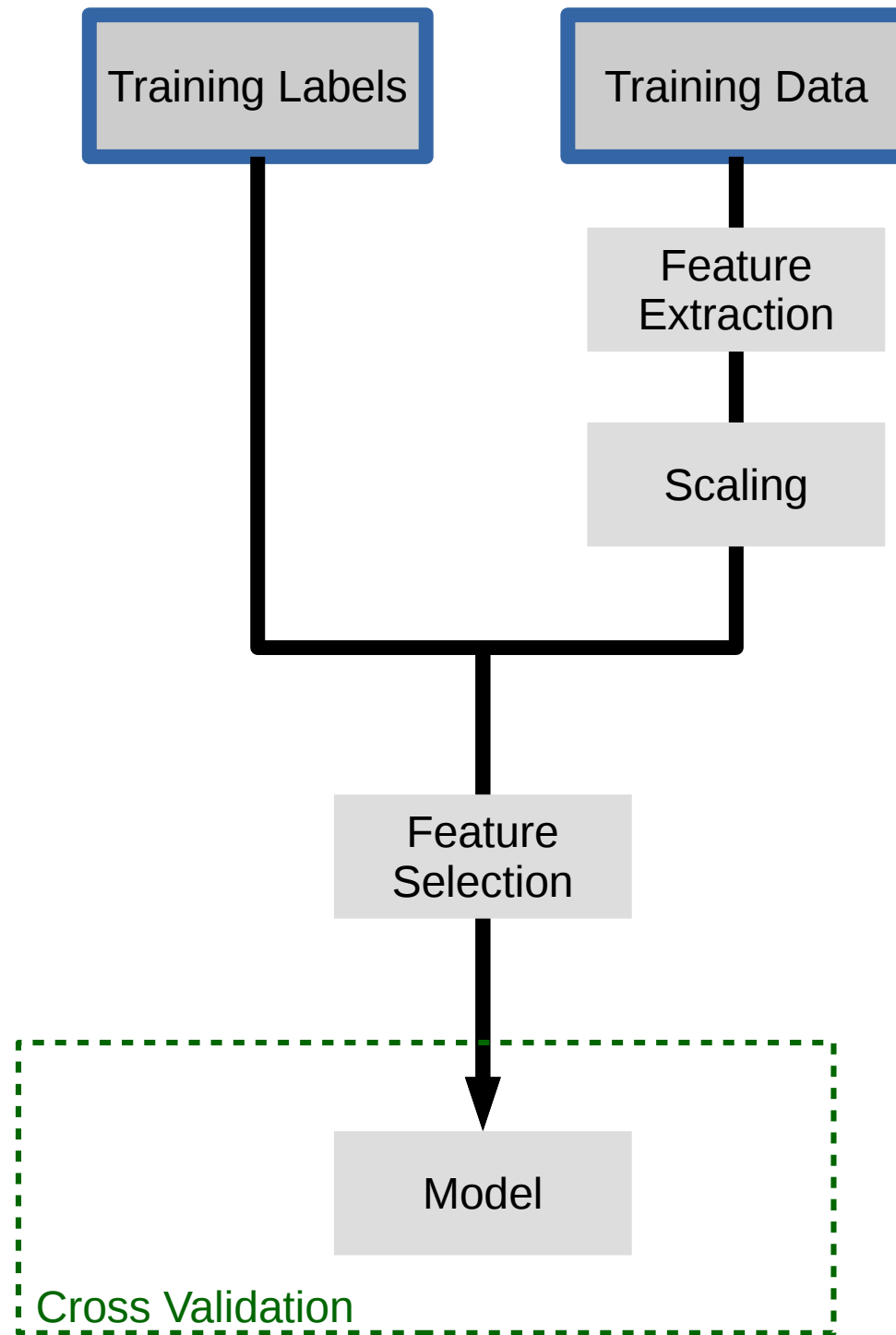
Notebook Working With Text Data

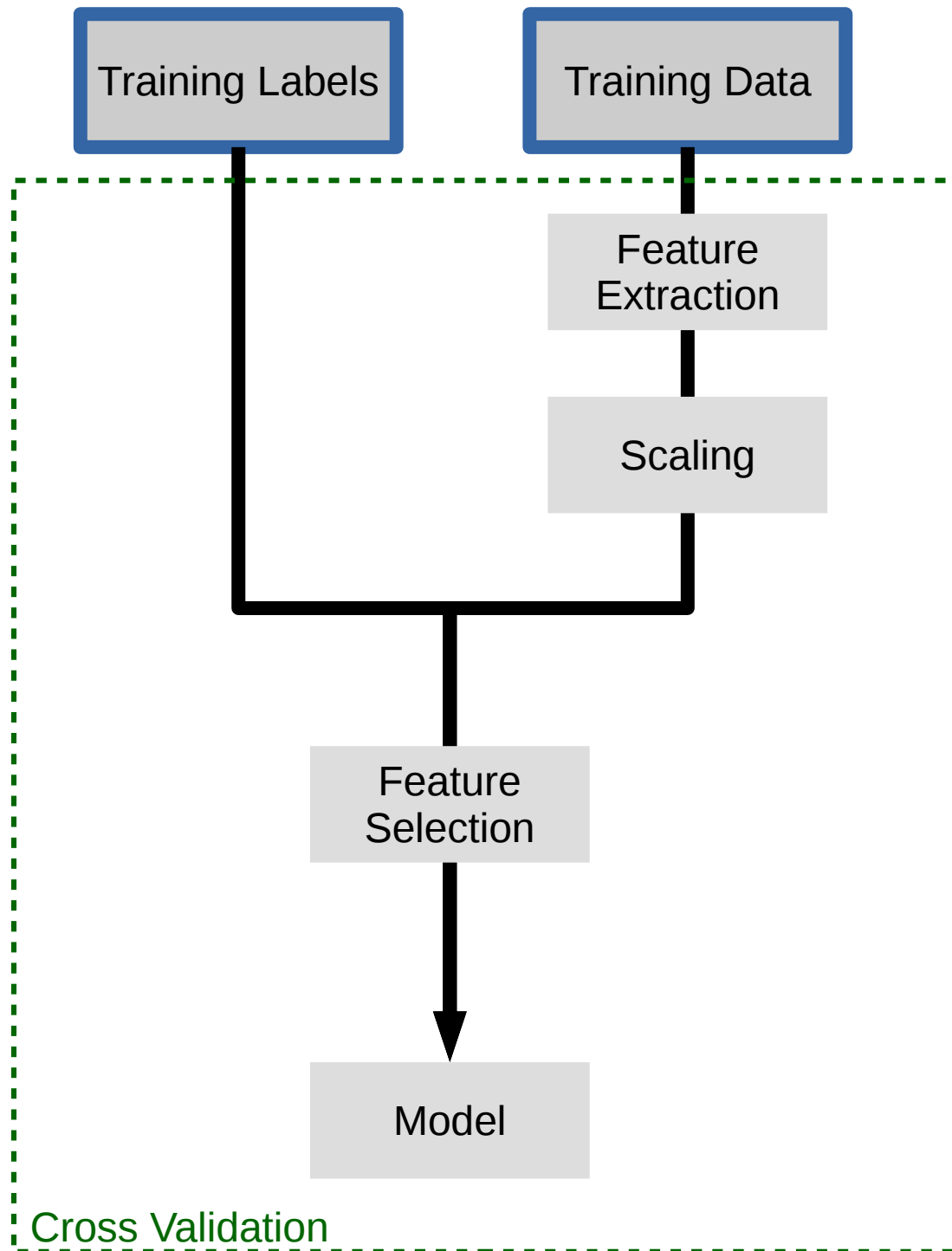
Pipelines









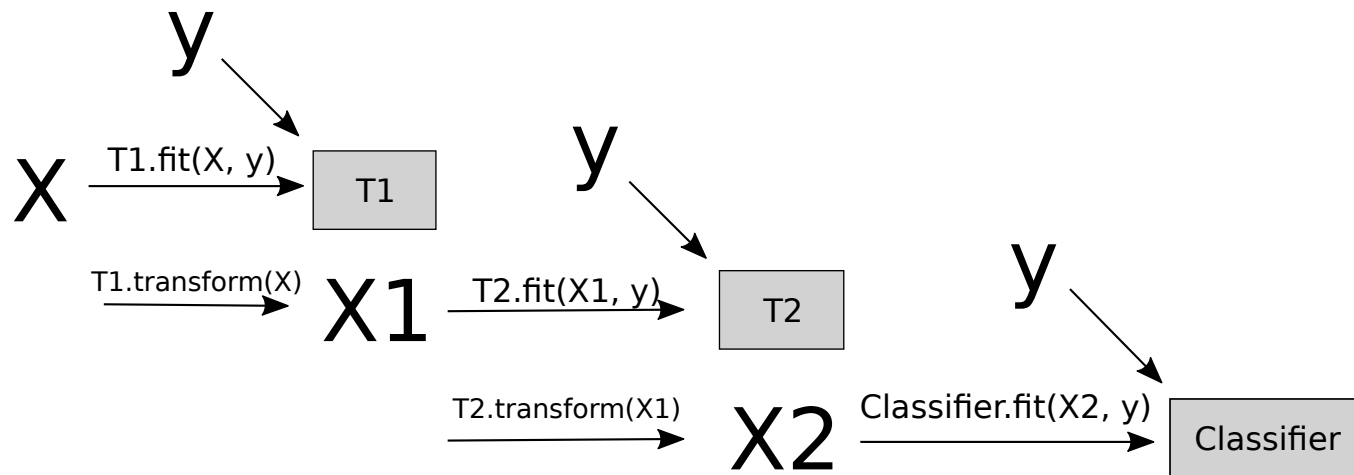


Pipelines

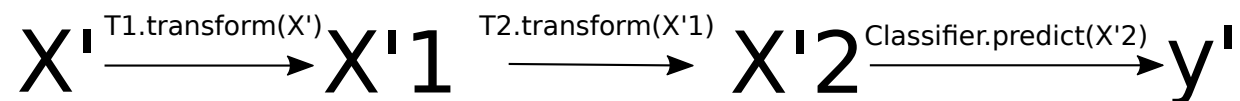
```
pipe = make_pipeline(T1(), T2(), Classifier())
```



```
pipe.fit(X, y)
```



```
pipe.predict(X')
```



Pipelines

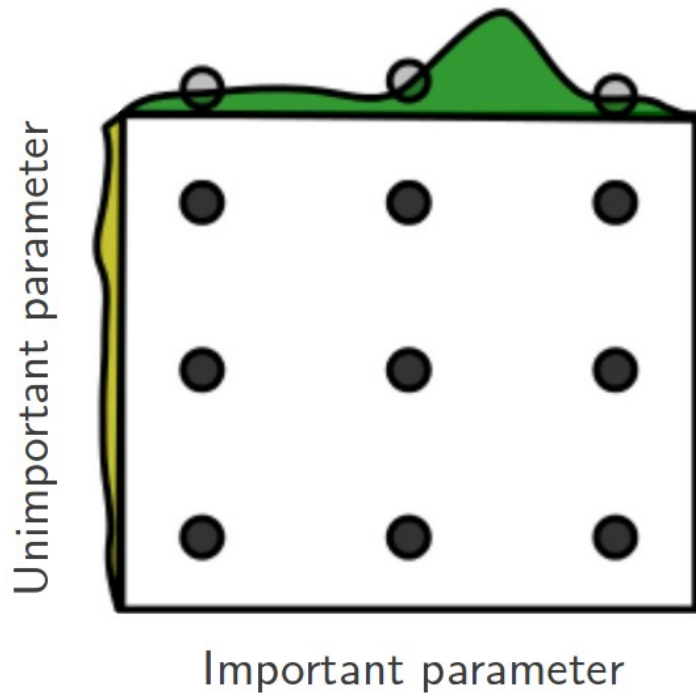
```
from sklearn.pipeline import make_pipeline  
  
pipe = make_pipeline(StandardScaler(), SVC())  
pipe.fit(X_train, y_train)  
pipe.predict(X_test)
```


Continue Notebook Working with Text Data

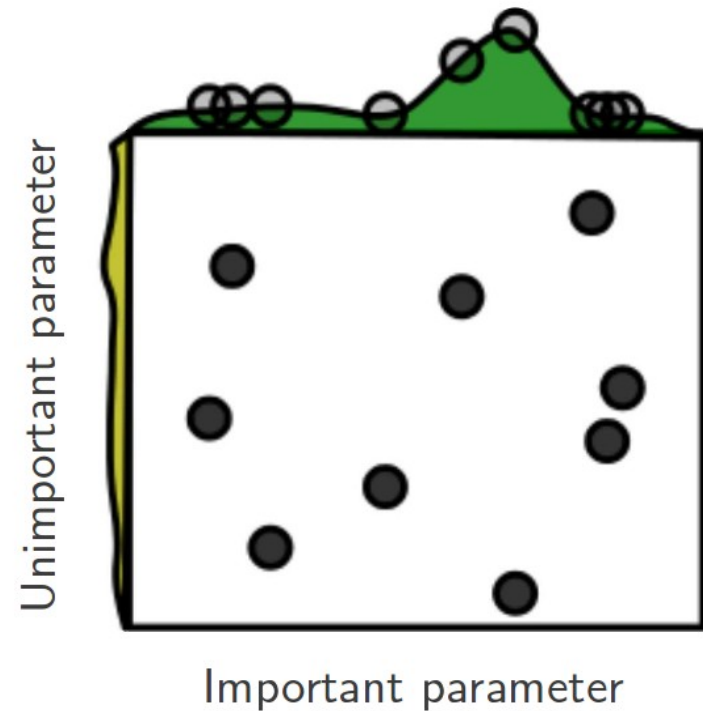
Randomized Parameter Search

Randomized Parameter Search

Grid Layout

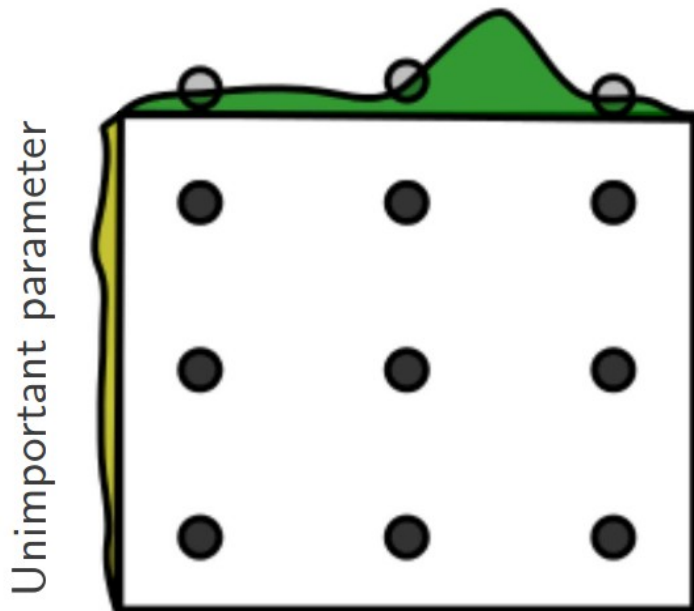


Random Layout

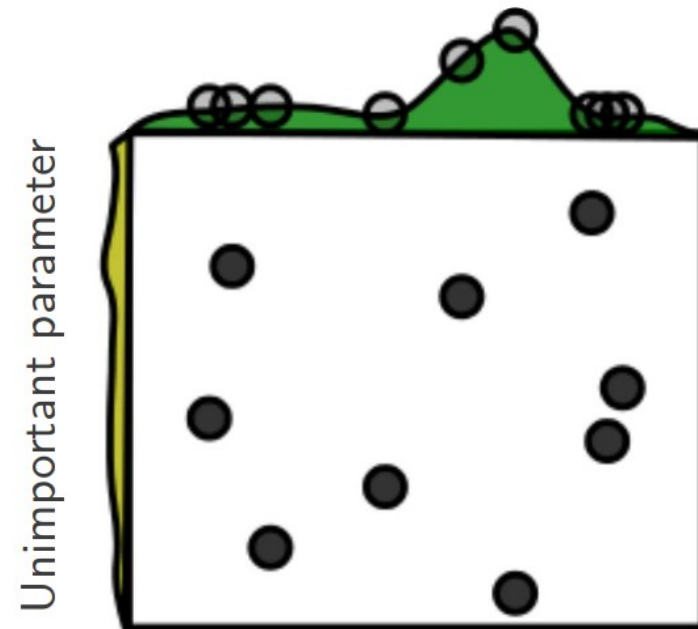


Randomized Parameter Search

Grid Layout



Random Layout



Step-size free for continuous parameters
Decouples runtime from search-space size
Robust against irrelevant parameters

Randomized Parameter Search

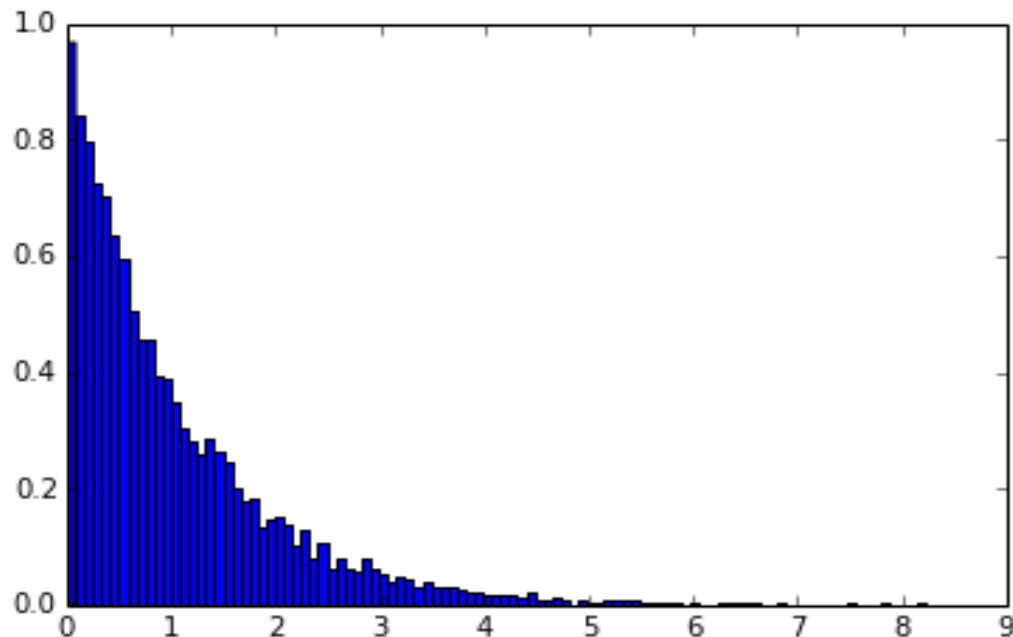
```
params = {'featureunion__countvectorizer-1__ngram_range':  
          [(1, 3), (1, 5), (2, 5)],  
          'featureunion__countvectorizer-2__ngram_range':  
          [(1, 1), (1, 2), (2, 2)],  
          'linearsvc__C': 10. ** np.arange(-3, 3)}
```

Randomized Parameter Search

```
params = {'featureunion__countvectorizer-1__ngram_range':  
          [(1, 3), (1, 5), (2, 5)],  
          'featureunion__countvectorizer-2__ngram_range':  
          [(1, 1), (1, 2), (2, 2)],  
          'linearsvc__C': expon()}
```

Randomized Parameter Search

```
params = {'featureunion__countvectorizer-1__ngram_range':  
          [(1, 3), (1, 5), (2, 5)],  
          'featureunion__countvectorizer-2__ngram_range':  
          [(1, 1), (1, 2), (2, 2)],  
          'linearsvc__C': expon()}
```

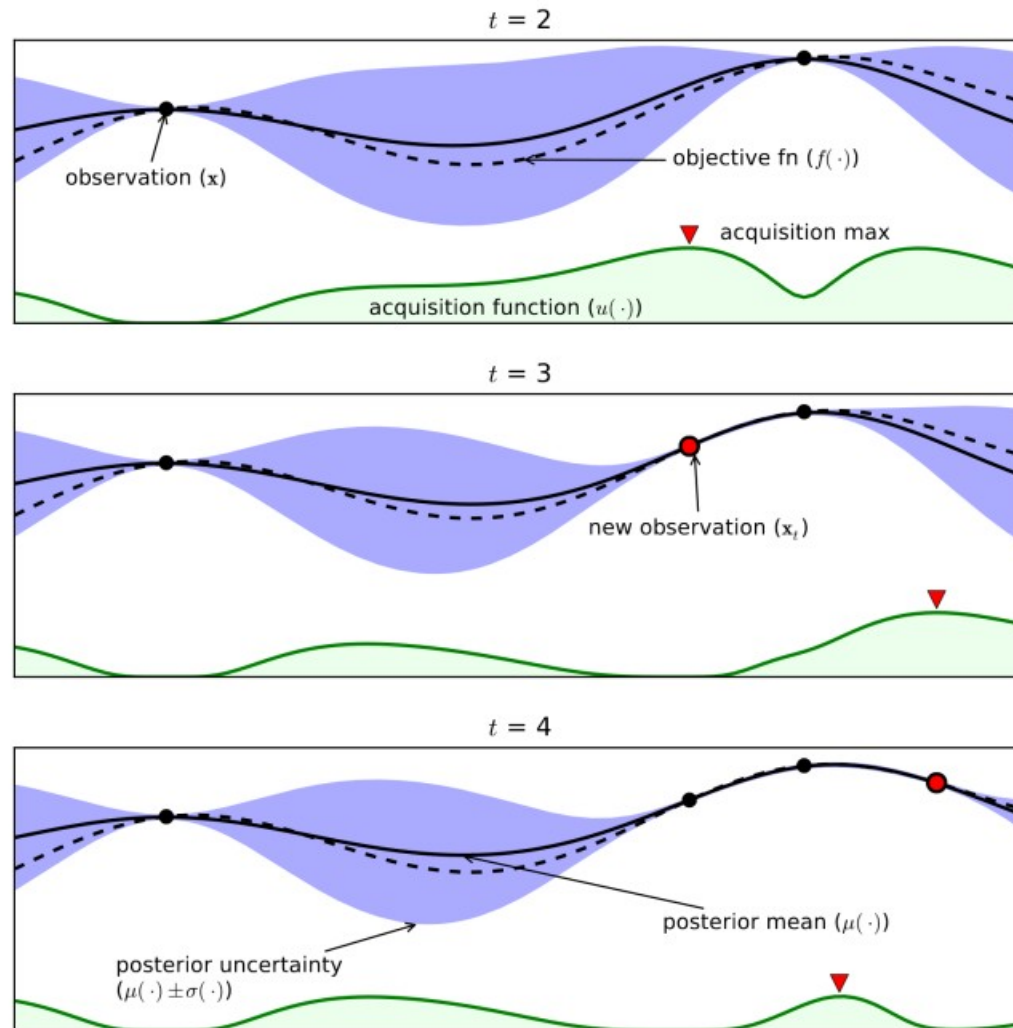


```
rs = RandomizedSearchCV(text_pipe,  
    param_distributions=param_distributins, n_iter=50)
```

Randomized Parameter Search

- Always use distributions for continuous variables.
- Don't use for low dimensional spaces.

GP based parameter optimization (coming soon)




From Eric Brochu, Vlad M. Cora and Nando de Freitas


Efficient Parameter Search and Path Algorithms

```
rfe = RFE(LogisticRegression())
```

```
rfe = RFE(LogisticRegression())  
param_grid = {'n_features_to_select': range(1, n_features)}  
gridsearch = GridSearchCV(rfe, param_grid)  
grid.fit(X, y)
```




```
rfe = RFE(LassoRegression())  
param_grid = {'n_features_to_select': range(1, n_features)}  
gridsearch = GridSearchCV(rfe, param_grid)  
grid.fit(X, y)
```



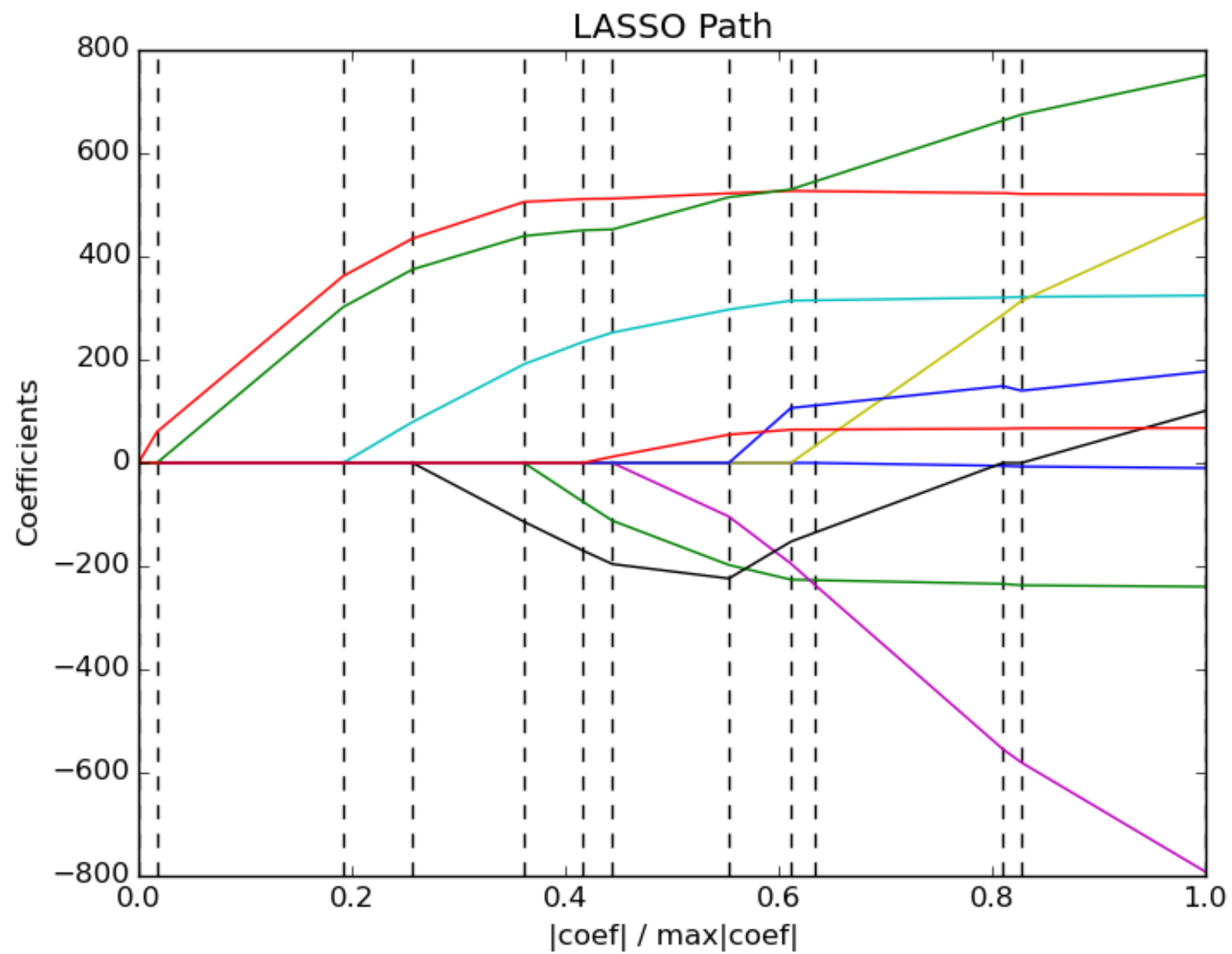
```
rfe = RFE(LogisticRegression())  
param_grid = {'n_features_to_select': range(1, n_features)}  
gridsearch = GridSearchCV(rfe, param_grid)  
grid.fit(X, y)
```

```
rfecv = RFECV(LogisticRegression())
```



```
rfe = RFE(LogisticRegression())  
param_grid = {'n_features_to_select': range(1, n_features)}  
gridsearch = GridSearchCV(rfe, param_grid)  
grid.fit(X, y)
```

```
rfecv = RFECV(LogisticRegression())  
rfecv.fit(X, y)
```



Linear Models	Feature Selection	Tree-Based models [possible]
LogisticRegressionCV [new]	RFECV	[DecisionTreeCV]
RidgeCV		[RandomForestClassifierCV]
RidgeClassifierCV		[GradientBoostingClassifierCV]
LarsCV		
ElasticNetCV		
...		

Notebook Efficient Parameter Search

Scoring Functions

GridSeachCV
RandomizedSearchCV
cross_val_score
...CV

Default:
Accuracy (classification)
R2 (regression)

Notebook scoring metrics

Out of Core Learning

- Large Scale – “Out of core: Fits on a hard disk but in RAM”

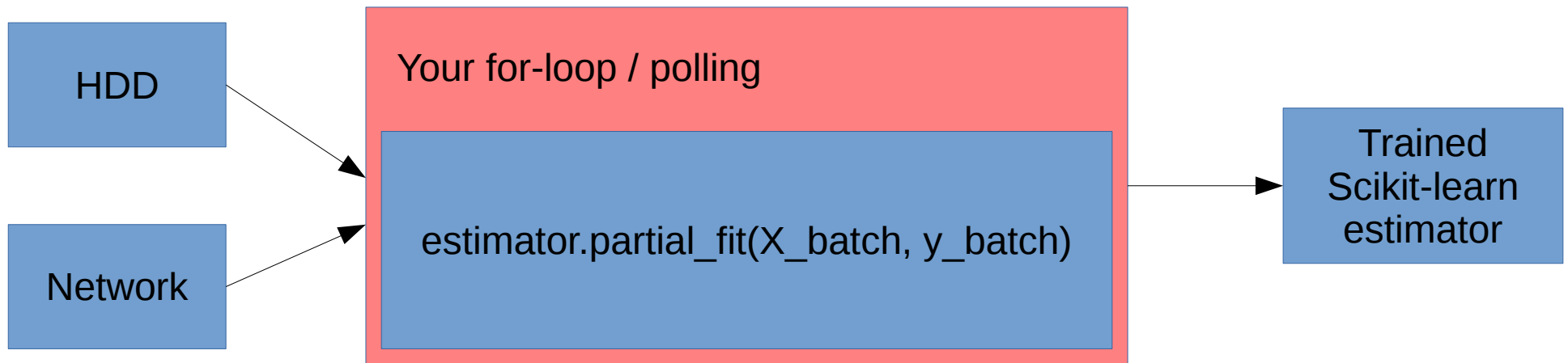
- Large Scale – “Out of core: Fits on a hard disk but in RAM”
- Non-linear – because real-world problems are not.

- Large Scale – “Out of core: Fits on a hard disk but in RAM”
- Non-linear – because real-world problems are not.
- Single CPU – Because parallelization is hard (and often unnecessary)

Think twice!

- Old laptop: 4GB Ram
- 1073741824 float32
- Or 1mio data points with 1000 features
- EC2 : 256 GB Ram
- 68719476736 float32
- Or 68mio data points with 1000 features

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
Memory Optimized - Current Generation					
r3.large	2	6.5	15	1 x 32 SSD	\$0.195 per Hour
r3.xlarge	4	13	30.5	1 x 80 SSD	\$0.39 per Hour
r3.2xlarge	8	26	61	1 x 160 SSD	\$0.78 per Hour
r3.4xlarge	16	52	122	1 x 320 SSD	\$1.56 per Hour
r3.8xlarge	32	104	244	2 x 320 SSD	\$3.12 per Hour
Storage Optimized - Current Generation					
i2.xlarge	4	14	30.5	1 x 800 SSD	\$0.938 per Hour
i2.2xlarge	8	27	61	2 x 800 SSD	\$1.876 per Hour
i2.4xlarge	16	53	122	4 x 800 SSD	\$3.751 per Hour
i2.8xlarge	32	104	244	8 x 800 SSD	\$7.502 per Hour



Supported Algorithms

- All `SGDClassifier` derivatives
- Naive Bayes
- `MinibatchKMeans`
- `IncrementalPCA`
- `MiniBatchDictionaryLearning`
- `MultilayerPerceptron` (dev branch)
- `Scalers`

Out of Core Learning

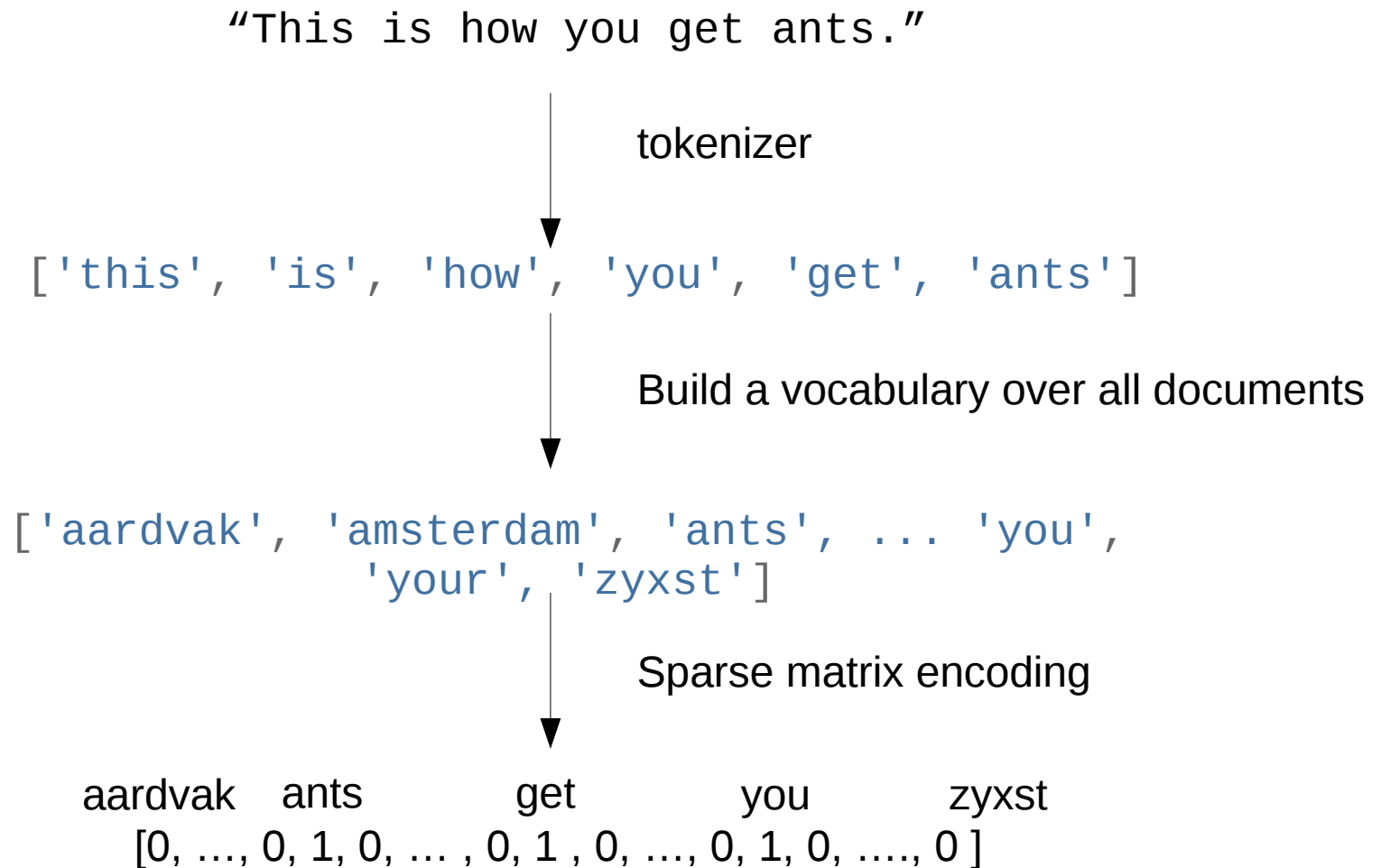
```
sgd = SGDClassifier()

for i in range(9):
    X_batch, y_batch = cPickle.load(open("batch_%02d" % i))
    sgd.partial_fit(X_batch, y_batch, classes=range(10))
```

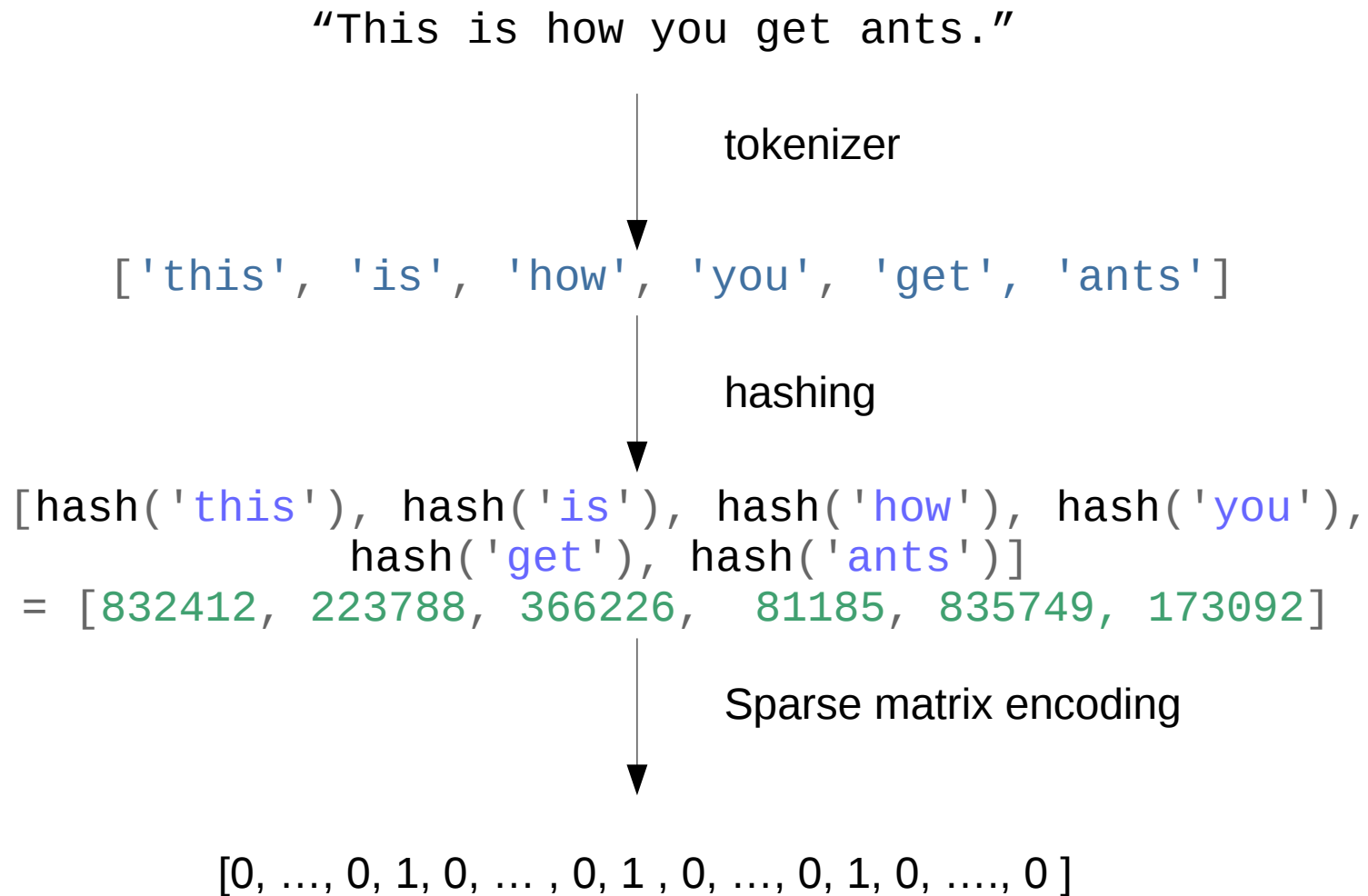
Possibly go over the data multiple times.

The hashing trick for text data

Text Classification: Bag Of Word

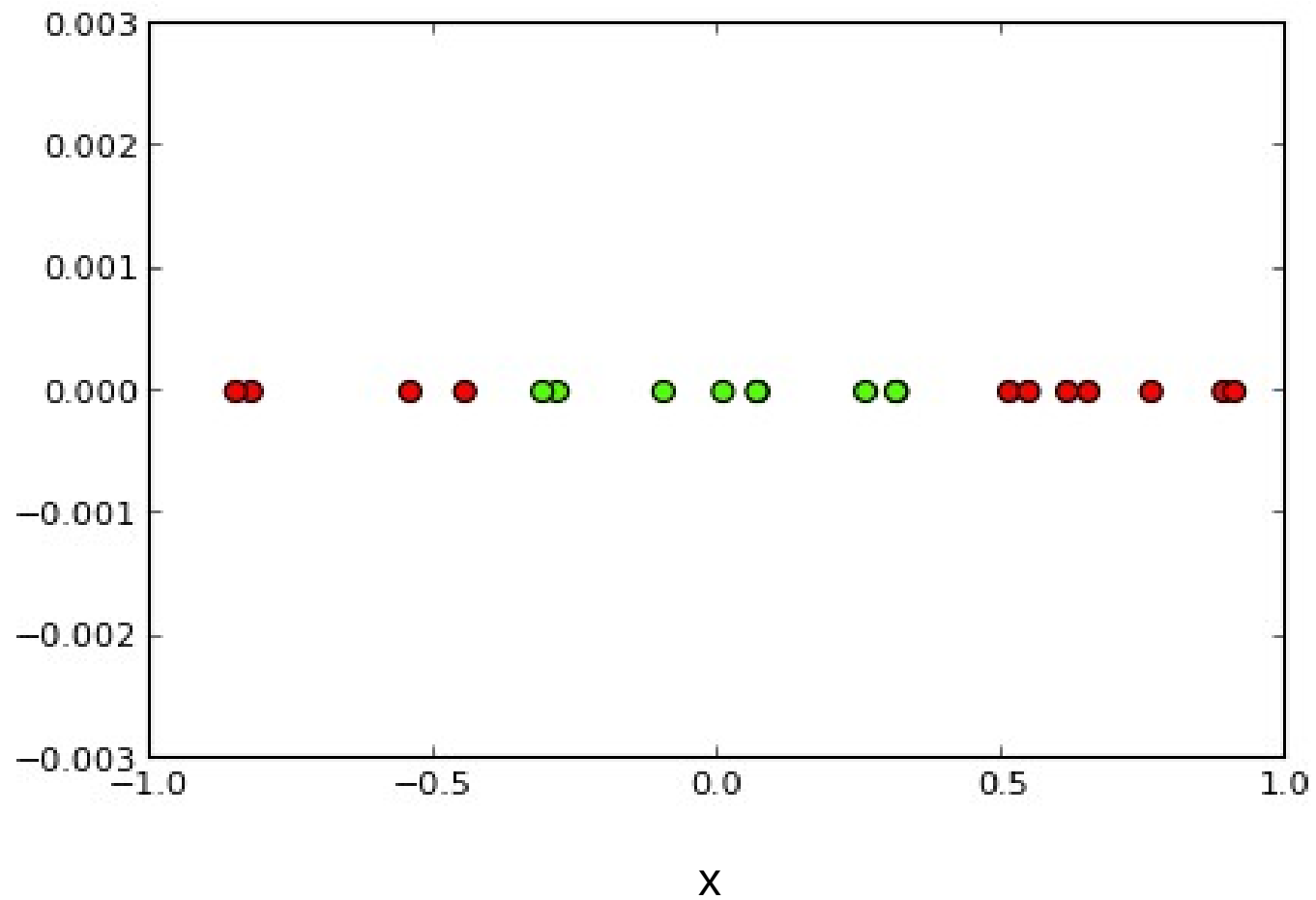


Text Classification: Hashing Trick

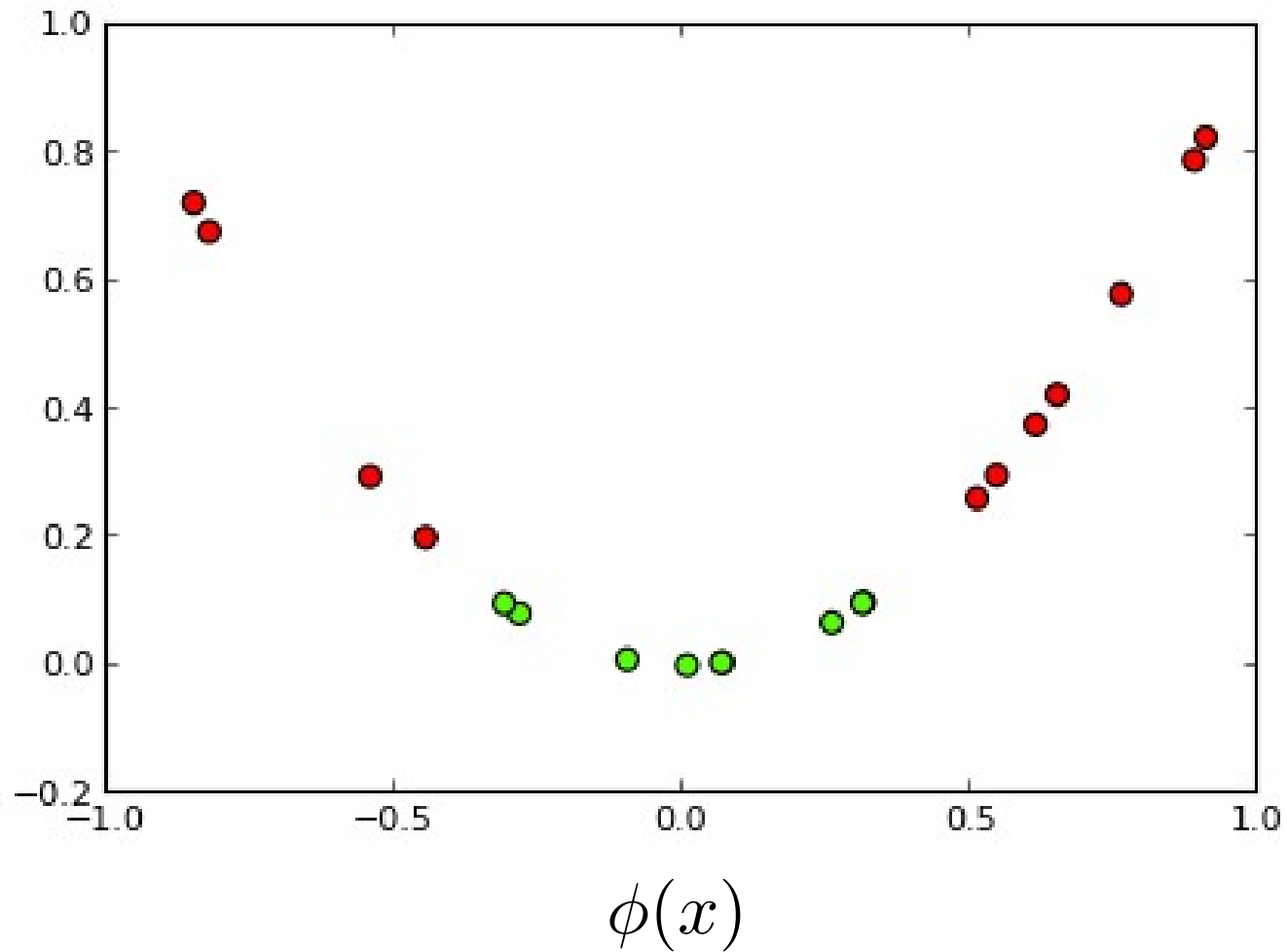


Kernel Approximations

Reminder: Kernel Trick



Reminder: Kernel Trick



Reminder: Kernel Trick

Classifier linear \rightarrow need only

$$\langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$$

Reminder: Kernel Trick

Classifier linear \rightarrow need only

$$\langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$$

Linear: $\langle x, x' \rangle$

Polynomial: $(\gamma \langle x, x' \rangle + r)^d$

RBF: $\exp(-\gamma |x - x'|^2)$

Sigmoid: $\tanh(\gamma \langle x, x' \rangle + r)$

Complexity

- Solving kernelized SVM:
 $\sim O(n_{\text{samples}}^3)$
- Solving linear (primal) SVM:
 $\sim O(n_{\text{samples}} * n_{\text{features}})$

n_{samples} large? Go primal!

Undoing the Kernel Trick

- Kernel approximation:

$$\langle \hat{\phi}(x_i), \hat{\phi}(x_j) \rangle \approx k(x_i, x_j)$$

- $k = \exp(-\gamma |x - x'|^2)$
 $\hat{\phi} = \text{RBFSampler}$

Usage

```
sgd = SGDClassifier()
kernel_approximation = RBFSampler(gamma=.001, n_components=400)

for i in range(9):
    X_batch, y_batch = cPickle.load(open("batch_%02d" % i))
    if i == 0:
        kernel_approximation.fit(X_batch)
    X_transformed = kernel_approximation.transform(X_batch)
    sgd.partial_fit(X_transformed, y_batch, classes=range(10))
```

How (and why) to build your own estimator

Why?

GridSearchCV
cross_val_score
Pipeline

How

- “fit” method
- set_params and get_params (or inherit)
- Run check_estimator

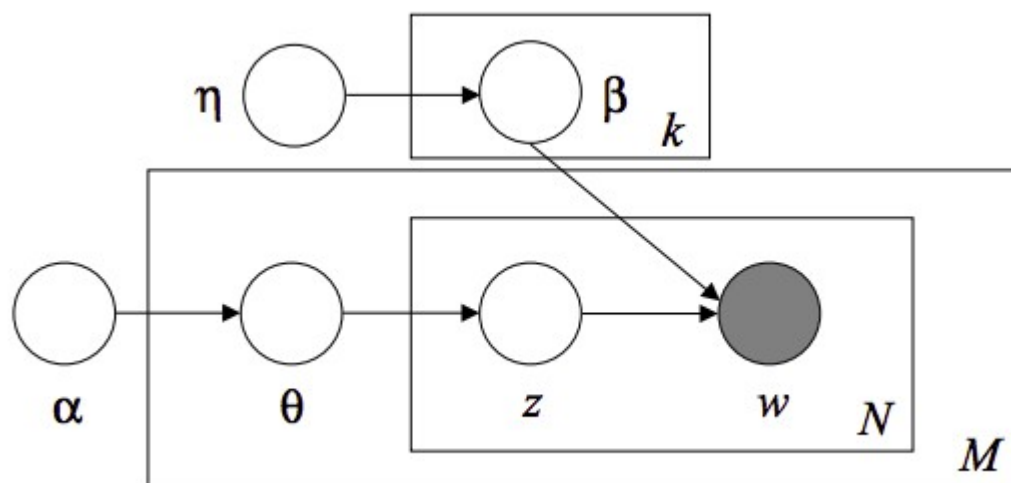
See the “build your own estimator” docs!

Notebook Building your own estimator

What's new in 0.17

Latent Dirichlet Allocation

using online variational inference



Topic #0:

government people mr law gun state president states public use right
rights national new control american security encryption health
united

Topic #1:

drive card disk bit scsi use mac memory thanks pc does video hard
speed apple problem used data monitor software

Topic #2:

said people armenian armenians turkish did saw went came women
killed children turkey told dead didn left started greek war

Topic #3:

year good just time game car team years like think don got new play
games ago did season better ll

Topic #4:

10 00 15 25 12 11 20 14 17 16 db 13 18 24 30 19 27 50 21 40

Topic #5:

windows window program version file dos use files available display
server using application set edu motif package code ms software

Topic #6:

edu file space com information mail data send available program ftp
email entry info list output nasa address anonymous internet

Topic #7:

ax max b8f g9v a86 pl 145 1d9 0t 34u 1t 3t giz bhj wm 2di 75u 2tm
bxn 7ey

Topic #8:

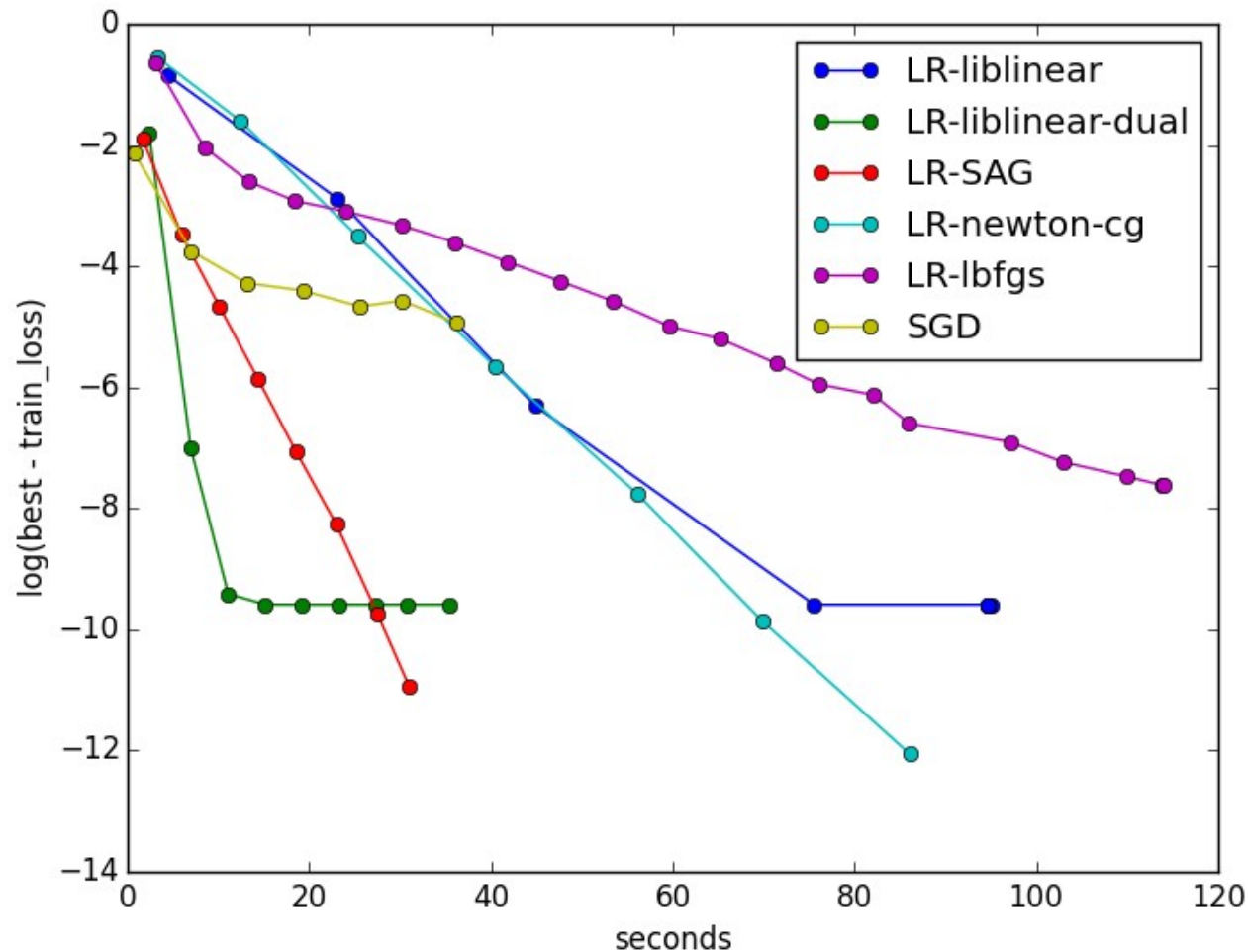
god people jesus believe does say think israel christian true life jews
did bible don just know world way church

Topic #9:

don know like just think ve want does use good people key time way
make problem really work say need

By Chyi-Kwei Yau, based on code by Matt Hoffman

SAG for Logistic Regression and Ridge Regression



By Danny Sullivan and Tom Dupre la Tour

Coordinate Descent Solver for Non-Negative Matrix Factorization

Topics in NMF model:

Topic #0:

don people just like think know time good right ve make say want did really way new use going said

Topic #1:

windows file dos files window program use running using version ms problem server pc screen ftp run application os software

Topic #2:

god jesus bible christ faith believe christians christian heaven sin hell life church truth lord say belief does existence man

Topic #3:

geb dsl n3jxp chastity cadre shameful pitt intellect skepticism surrender gordon banks soon edu lyme blood weight patients medical probably

Topic #4:

key chip encryption clipper keys escrow government algorithm secure security encrypted public des nsa enforcement bit privacy law secret use

Topic #5:

drive scsi ide drives disk hard controller floppy hd cd mac boot rom cable internal tape bus seagate bios quantum

Topic #6:

game team games players year hockey season play win league teams nhl baseball player detroit toronto runs pitching best playoffs

Topic #7:

thanks mail does know advance hi info looking anybody address appreciated help email information send ftp post interested list appreciate

Topic #8:

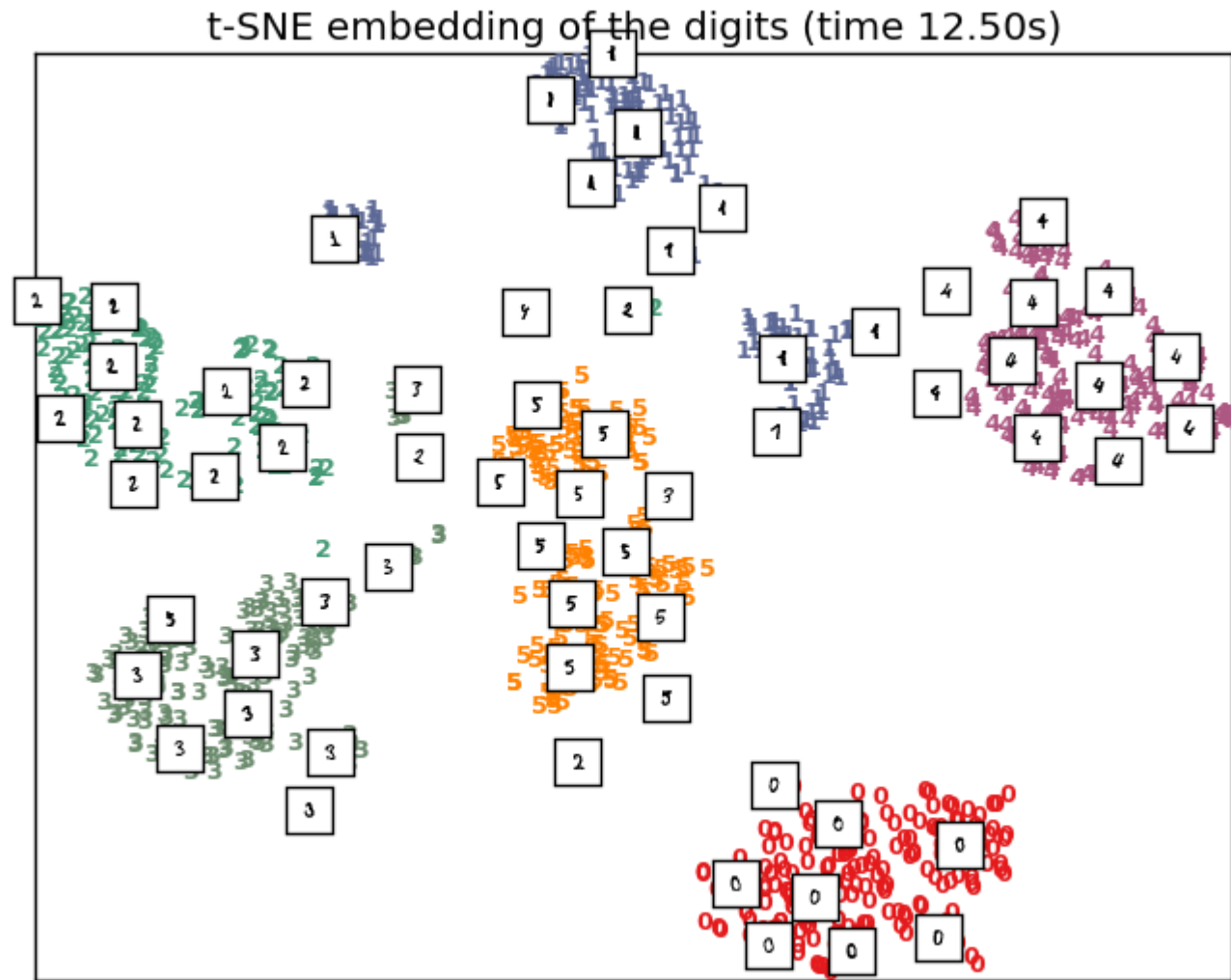
card video monitor vga bus drivers cards color driver ram ati mode memory isa graphics vesa pc vlb diamond bit

Topic #9:

00 sale 50 shipping 20 10 price 15 new 25 30 dos offer condition 40 cover asking 75 interested 01

By Tom Dupre la Tour and Mathieu Blondel

Barnes-Hut Approximation for T-SNE manifold learning



FunctionTransformer

```
>>> import numpy as np
>>> from sklearn.preprocessing import FunctionTransformer
>>> transformer = FunctionTransformer(np.log1p)
>>> X = np.array([[0, 1], [2, 3]])
>>> transformer.transform(X)
array([[ 0.          ,  0.69314718],
       [ 1.09861229,  1.38629436]])
```

VotingClassifier

```
clf1 = LogisticRegression()  
clf2 = RandomForestClassifier()  
clf3 = GaussianNB()  
  
ecf = VotingClassifier(  
    estimators=[('lr', clf1), ('rf', clf2), ('gbn', clf3)],  
    voting="hard")
```

Scalers

- RobustScaler
- MaxAbsScaler

By Thomas Unterthiner.

Add Backlinks to Docs



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Google™ Custom Search

Search x

Previous
sklearn.ense
m...

Up
API
Reference

This documentation is for
scikit-learn **version**
0.18.dev0 — [Other](#)
[versions](#)

If you use the software,
please consider [citing](#)
[scikit-learn](#).

3.2.4.3.1.
`sklearn.ensemble.RandomForestC`
`lassifier`
3.2.4.3.1.1. Examples using
`sklearn.ensemble.RandomForestClas`
`sifier`

3.2.4.3.1. `sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,  
bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)  
\[source\]
```

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the [User Guide](#).

Parameters: `n_estimators` : integer, optional (default=10)

The number of trees in the forest.

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_features : int, float, string or None, optional (default="auto")

Add Backlinks to Docs



[Home](#) [Installation](#) [Documentation](#) ▾ [Examples](#)

Google™ Custom Search

Search ×

[Previous](#)
sklearn.ense
m...

[Up](#)
API
Reference

This documentation is for
scikit-learn **version**
0.18.dev0 — [Other](#)
[versions](#)

If you use the software,
please consider [citing](#)
[scikit-learn](#).

3.2.4.3.1.
`sklearn.ensemble.RandomForestC`
`lassifier`
3.2.4.3.1.1. Examples using
`sklearn.ensemble.RandomForestClas`
`sifier`

3.2.4.3.1. `sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,  
bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)
```

[\[source\]](#)

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Parameters: `n_estimators` : integer, optional (default=10)

The number of trees in the forest.

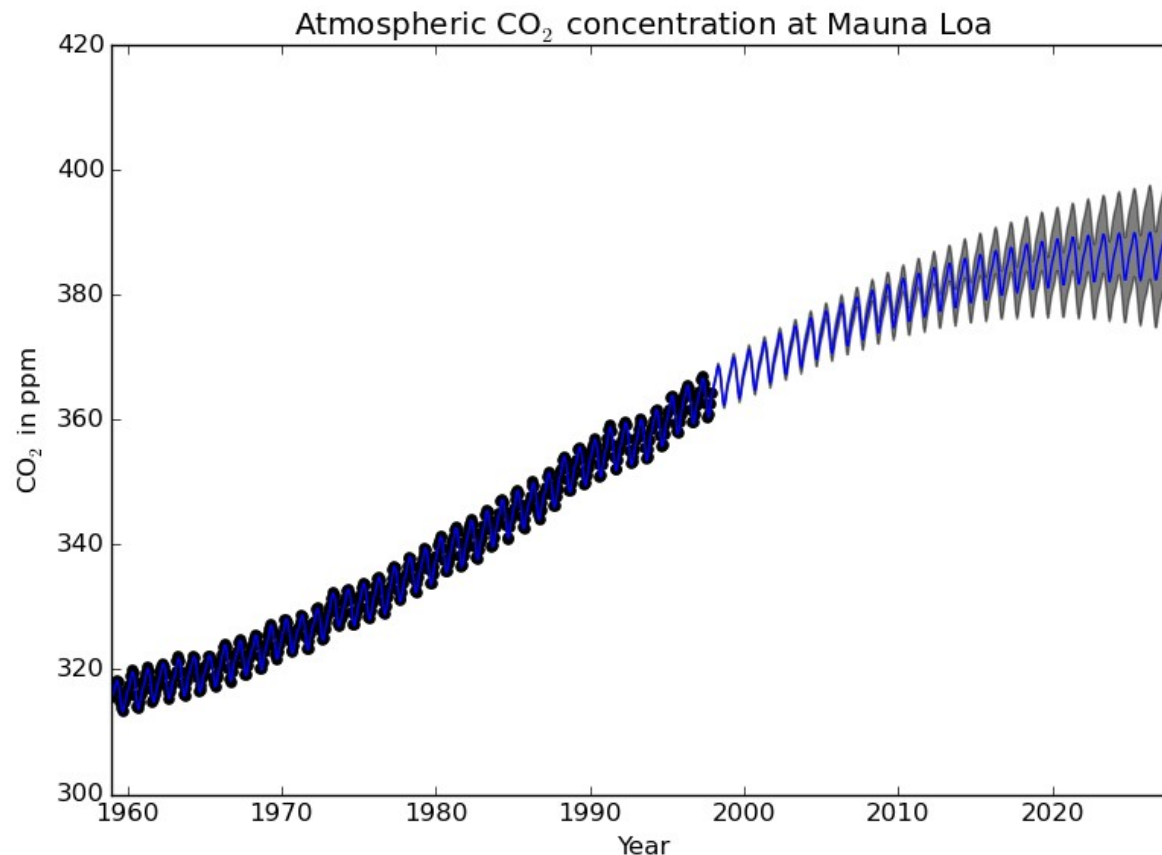
`criterion` : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

`max_features` : int, float, string or None, optional (default="auto")

What the future will bring (0.18)

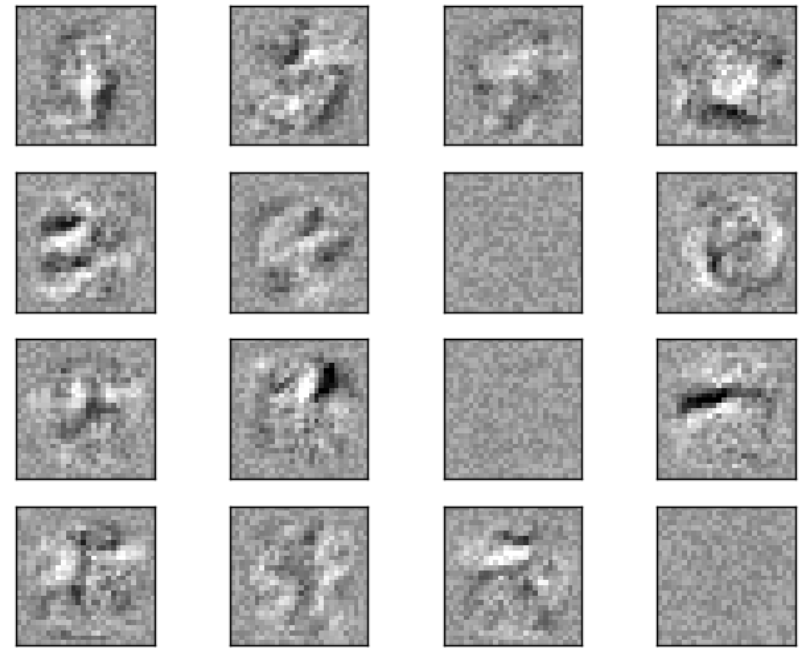
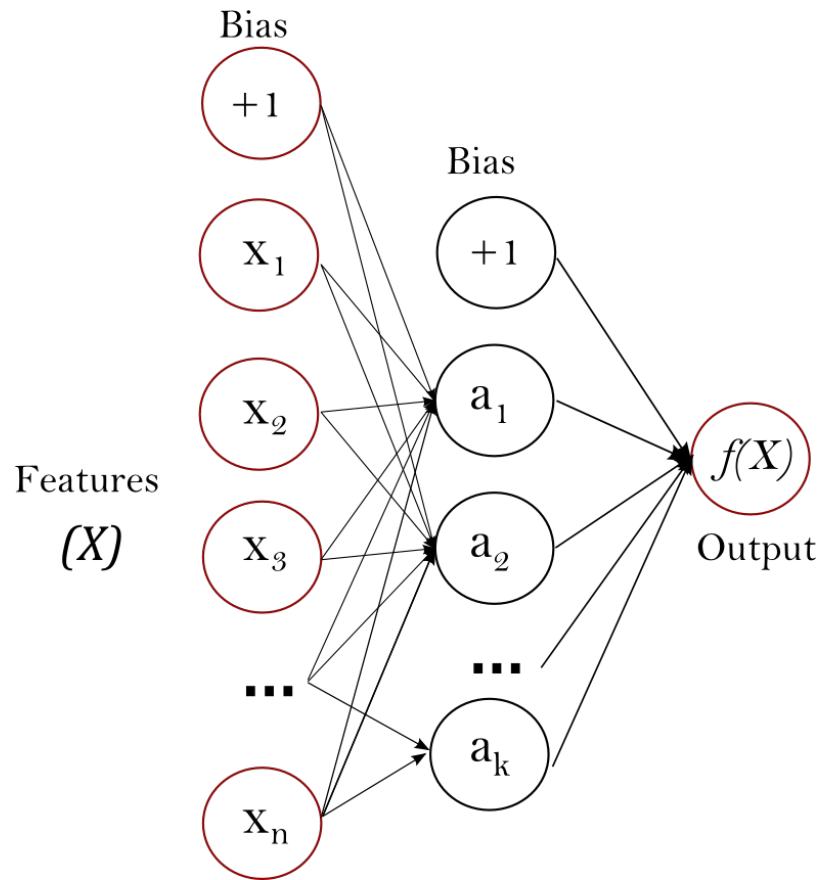
Gaussian Process Rewrite



```
34.4**2 * RBF(length_scale=41.8)
+ 3.27**2 * RBF(length_scale=180)
    * ExpSineSquared(length_scale=1.44, periodicity=1)
+ 0.446**2 * RationalQuadratic(alpha=17.7, length_scale=0.957)
+ 0.197**2 * RBF(length_scale=0.138) +
WhiteKernel(noise_level=0.0336)
```

By Jan Hendrik Metzen.

Neural Networks



By Jiyuan Qian and Issam Laradji

Improved Cross-Validation

current

```
>>> import numpy as np
>>> from sklearn.cross_validation import KFold

>>> kf = KFold(4, n_folds=2)
>>> for train, test in kf:
...     print("%s %s" % (train, test))
[2 3] [0 1]
[0 1] [2 3]
```

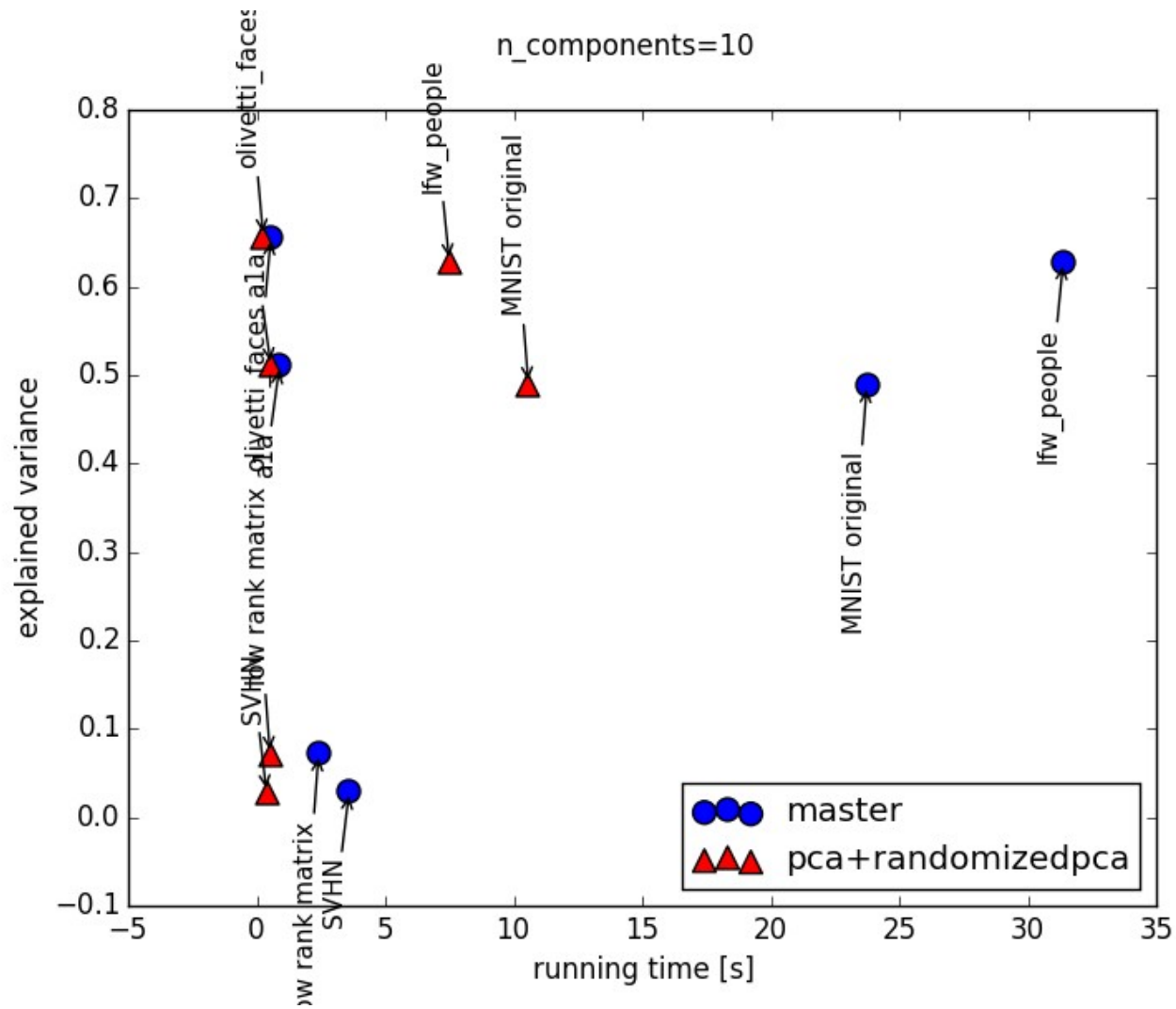
future

```
>>> import numpy as np
>>> from sklearn.model_selection import KFold

>>> X = ["a", "b", "c", "d"]
>>> kf = KFold(n_folds=2)
>>> for train, test in kf.split(X):
...     print("%s %s" % (train, test))
[2 3] [0 1]
[0 1] [2 3]
```

```
gs = GridSearchCV(LinearSVC(random_state=0), param_grid={'C': [1, 10]},
                  cv=inner_cv)
cross_val_score(gs, X=X, y=y, labels=labels, cv=outer_cv,
                fit_params={'labels': labels})
```

Faster PCA



By Giorgio Patrini

O'REILLY®



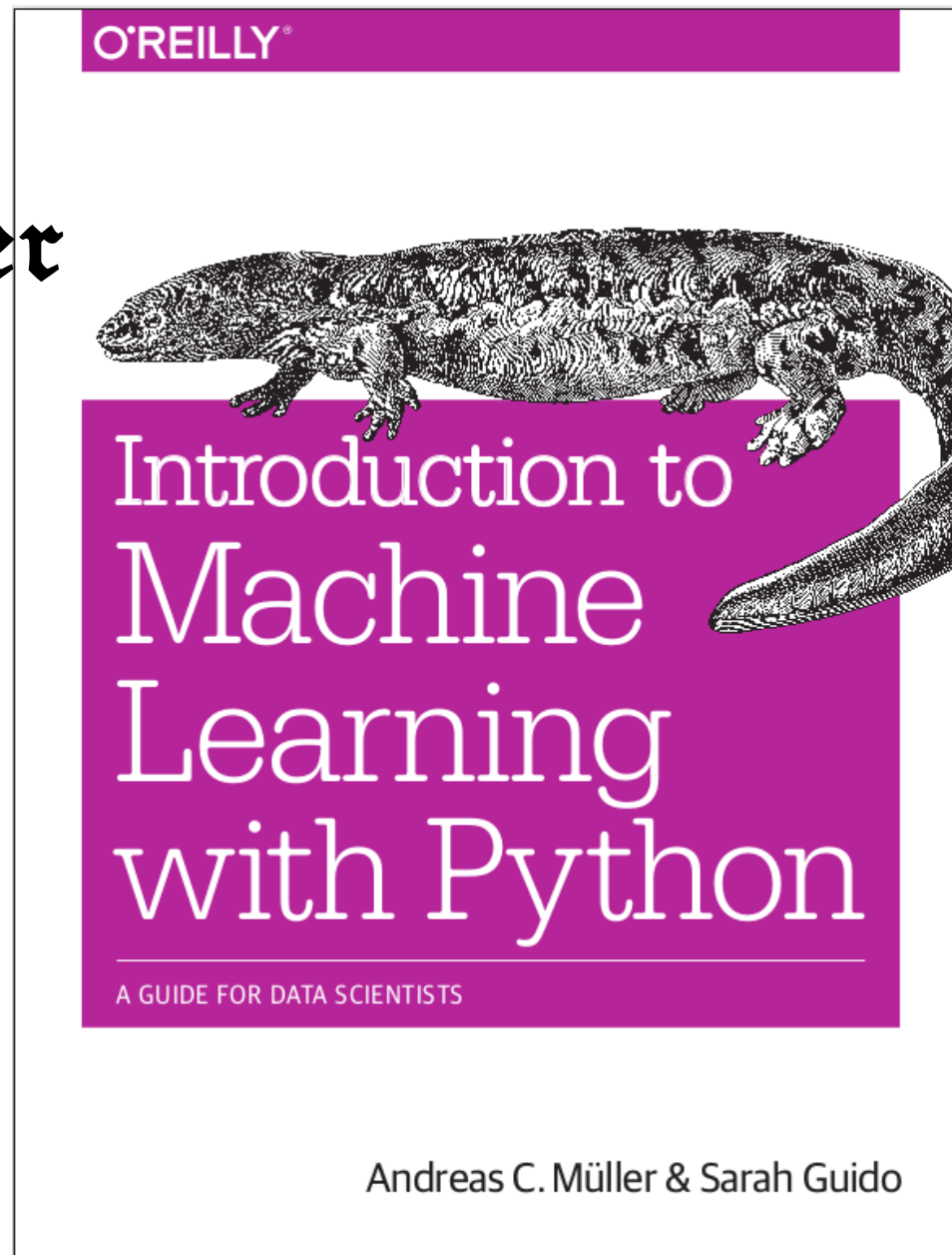
Introduction to Machine Learning with Python

A GUIDE FOR DATA SCIENTISTS

Andreas C. Müller & Sarah Guido

Release June 2016

Hellbender



Release June 2016

Thank you!



@amuellerm1



@amueller



importamueller@gmail.com



<http://amueller.github.io>