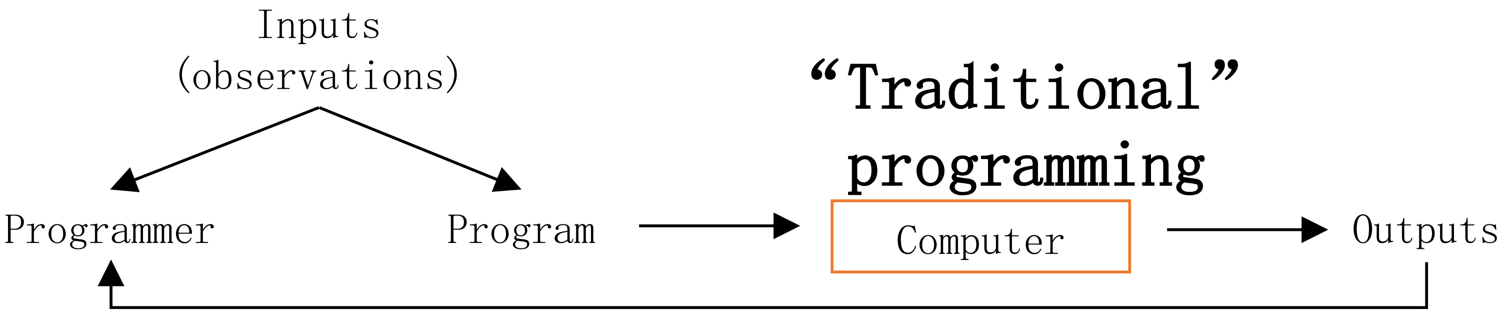
机器学习和量化交易实战

第四讲

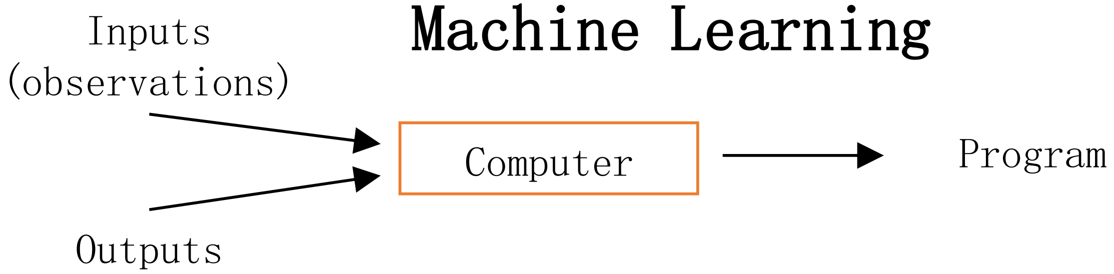
Outline

- From OLS to kernel machines and beyond
  - OLS
  - Ridge
  - Ｌａｓｓｏ
  - Kernels
  - Cross-validation
  - Hands on: sklearn

What is Machine Learning?



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



Examples of Machine Learning

https://flic.kr/p/5BLW6G [CC BY 2.0]

http://commons.wikimedia.org/wiki/File:American_book_company_1916._letter_envelope-2.JPG#filelinks [public domain]

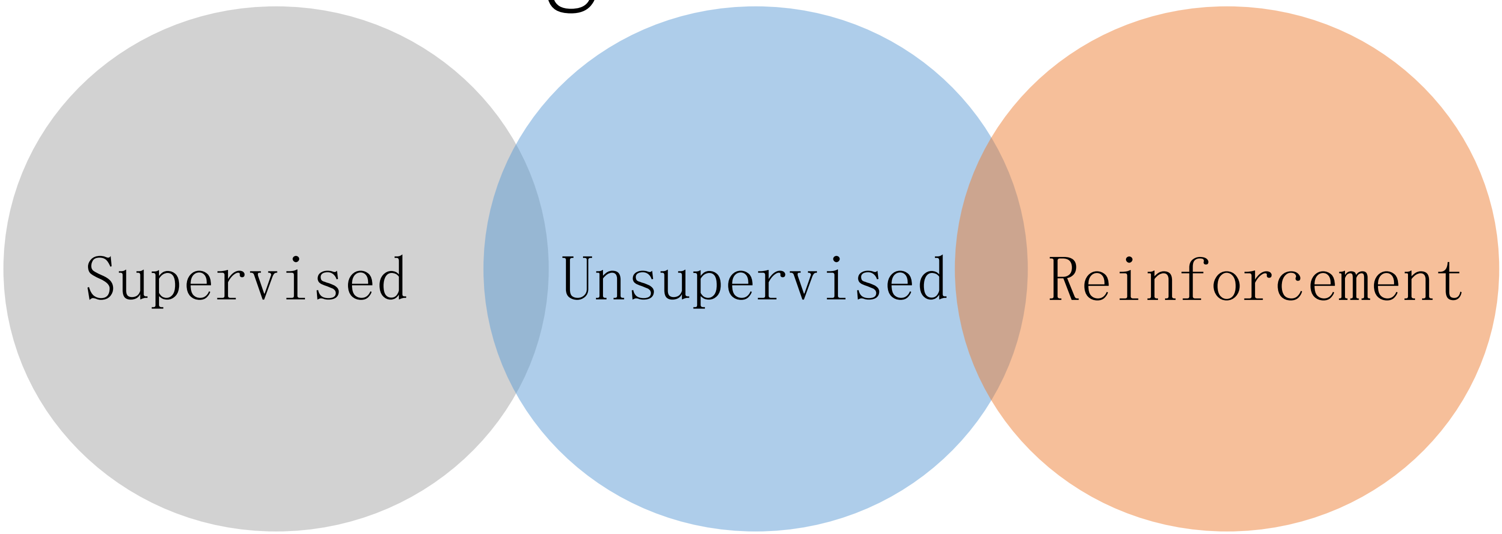
NETFLIX

http://commons.wikimedia.org/wiki/File:Netflix_logo.svg [public domain]

By Steve Jurvetson [CC BY 2.0]

And many, many more …

3 Types of Learning

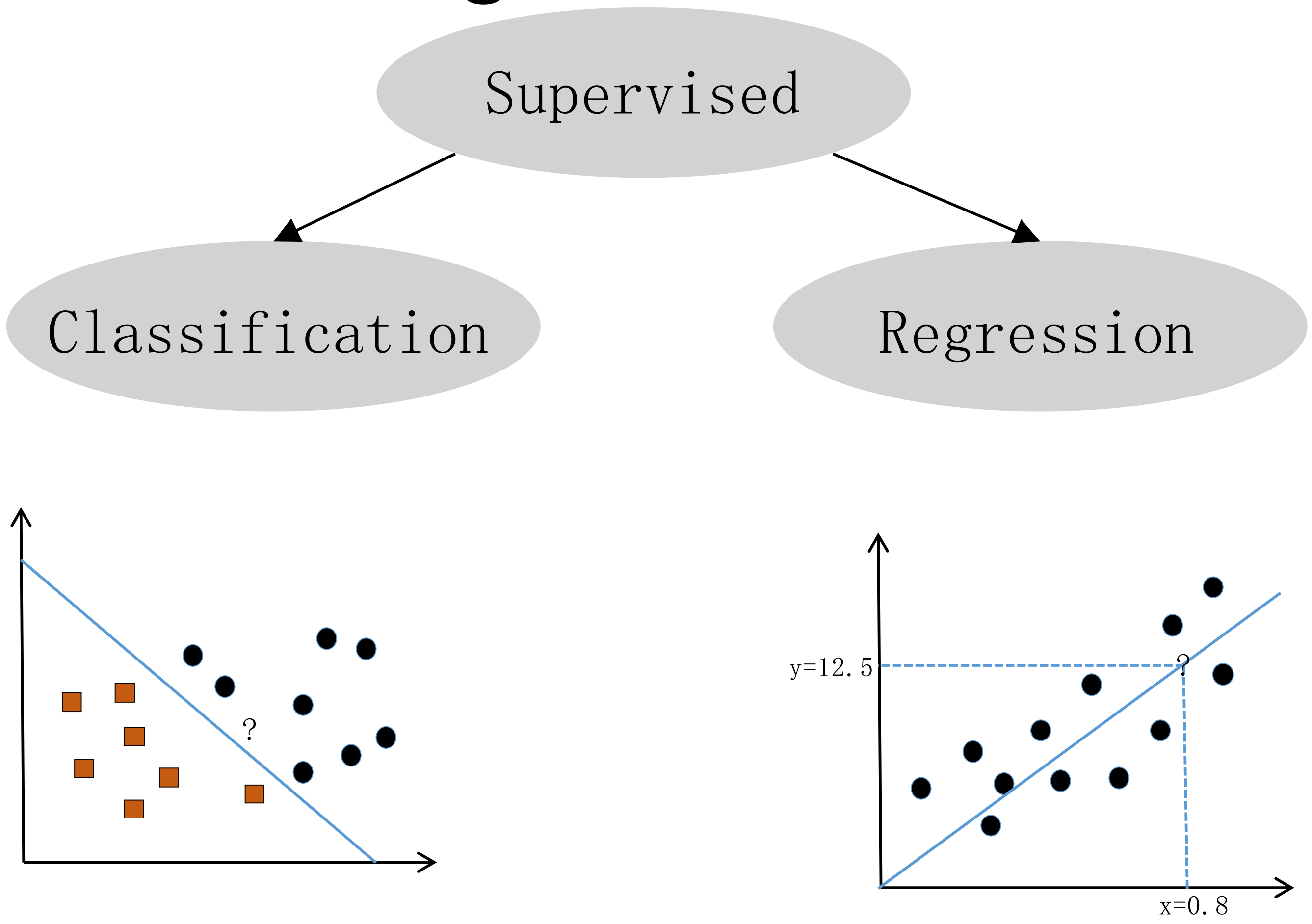


- Learning from labeled data
- E.g., Spam classification

- Discover structure in unlabeled data
- E.g., Document clustering

- Learning by “doing” with delayed reward
- E.g., Chess computer

Supervised Learning

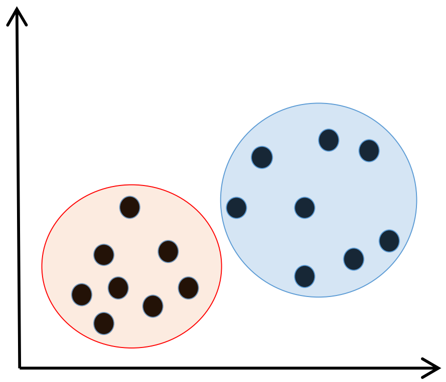


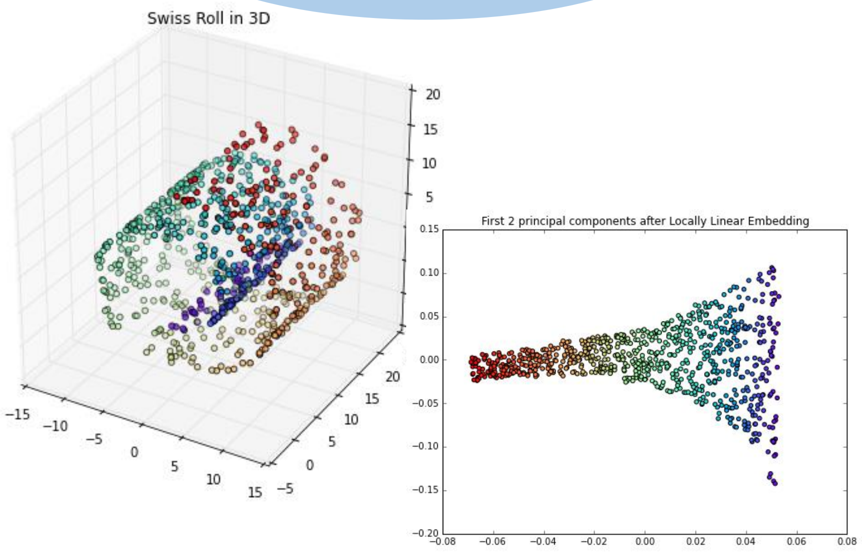
Unsupervised Learning

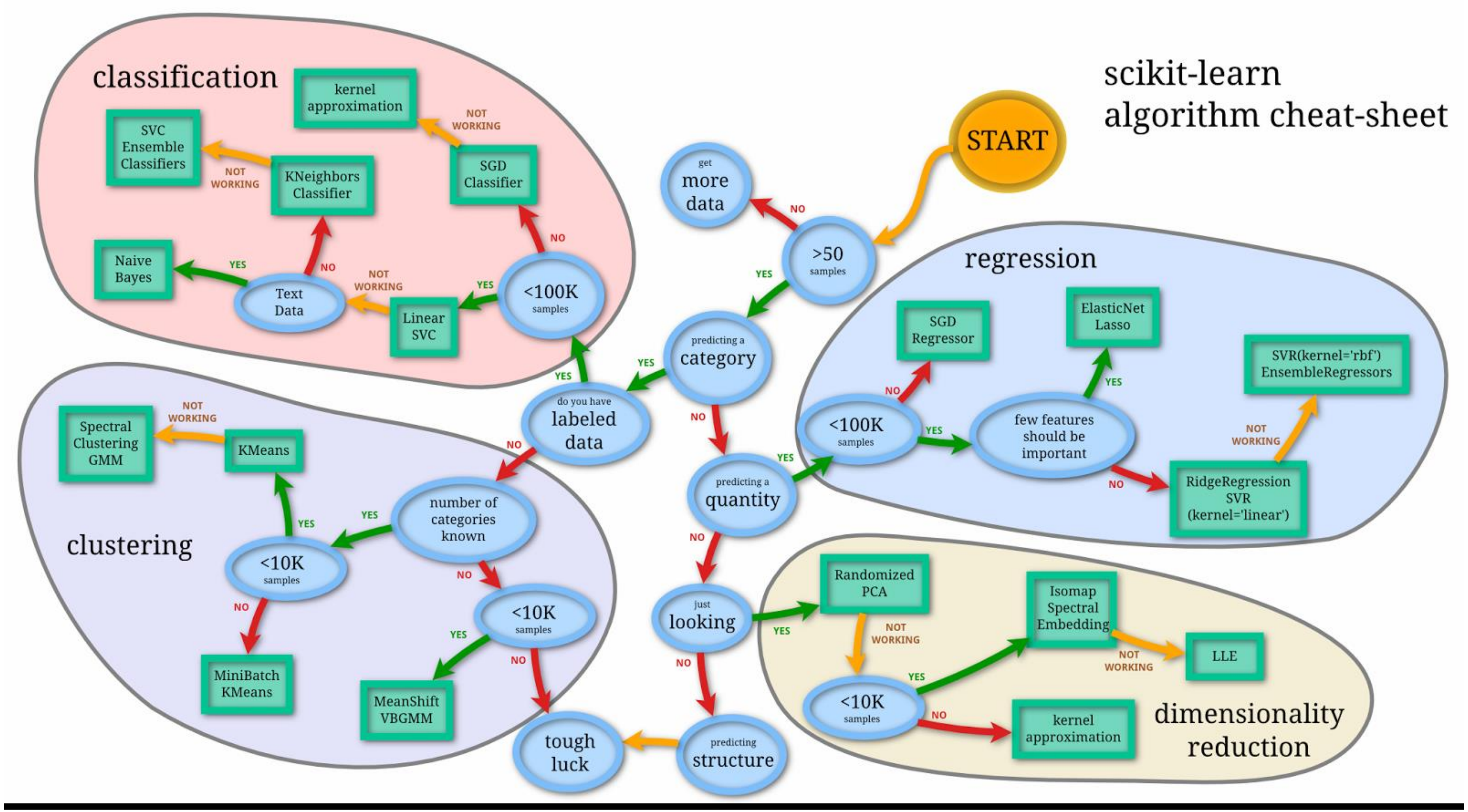
Unsupervised

Clustering

Compression






scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC Ensemble Classifiers
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering GMM
MiniBatch KMeans
MeanShift VBGMM
tough luck
predicting a quantity
regression
SGD Regressor
few features should be important
ElasticNet Lasso
RidgeRegression SVR (kernel='linear')
SVR(kernel='rbf') EnsembleRegressors
just looking
predicting structure
dimensionality reduction
Randomized PCA
Isomap Spectral Embedding
LLE
YES
NO
NOT WORKING

The simplest Sklearn workflow

```
train_x, train_y, test_x, test_y = getData()

model = somemodel()
model.fit(train_x,train_y)
predictions = model.predict(test_x)

score = score_function(test_y, predictions)
```

Flower Classification

Iris-Setosa

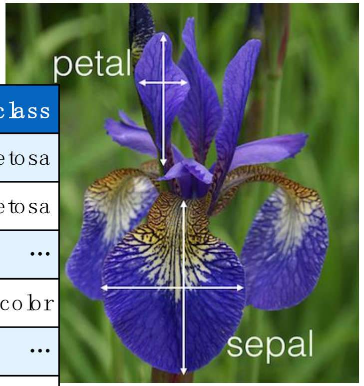
Iris-Setosa

Iris-Versicolor

Data Representation

IRIS

https://archive.ics.uci.edu/ml/datasets/Iris



Instances (samples, observations)

	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
…	…	…	…	…	…
50	6.4	3.2	4.5	1.5	veriscolor
…	…	…	…	…	…
150	5.9	3.0	5.1	1.8	virginica

Features (attributes, dimensions)

Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\mathbf{y} = [y_1, y_2, y_3, \cdots y_N]$$

In [2]:
```python
from sklearn.datasets import load_iris
iris = load_iris()
```

The resulting dataset is a `Bunch` object: you can see what's available using the method `keys()`:

In [3]:
```python
iris.keys()
```

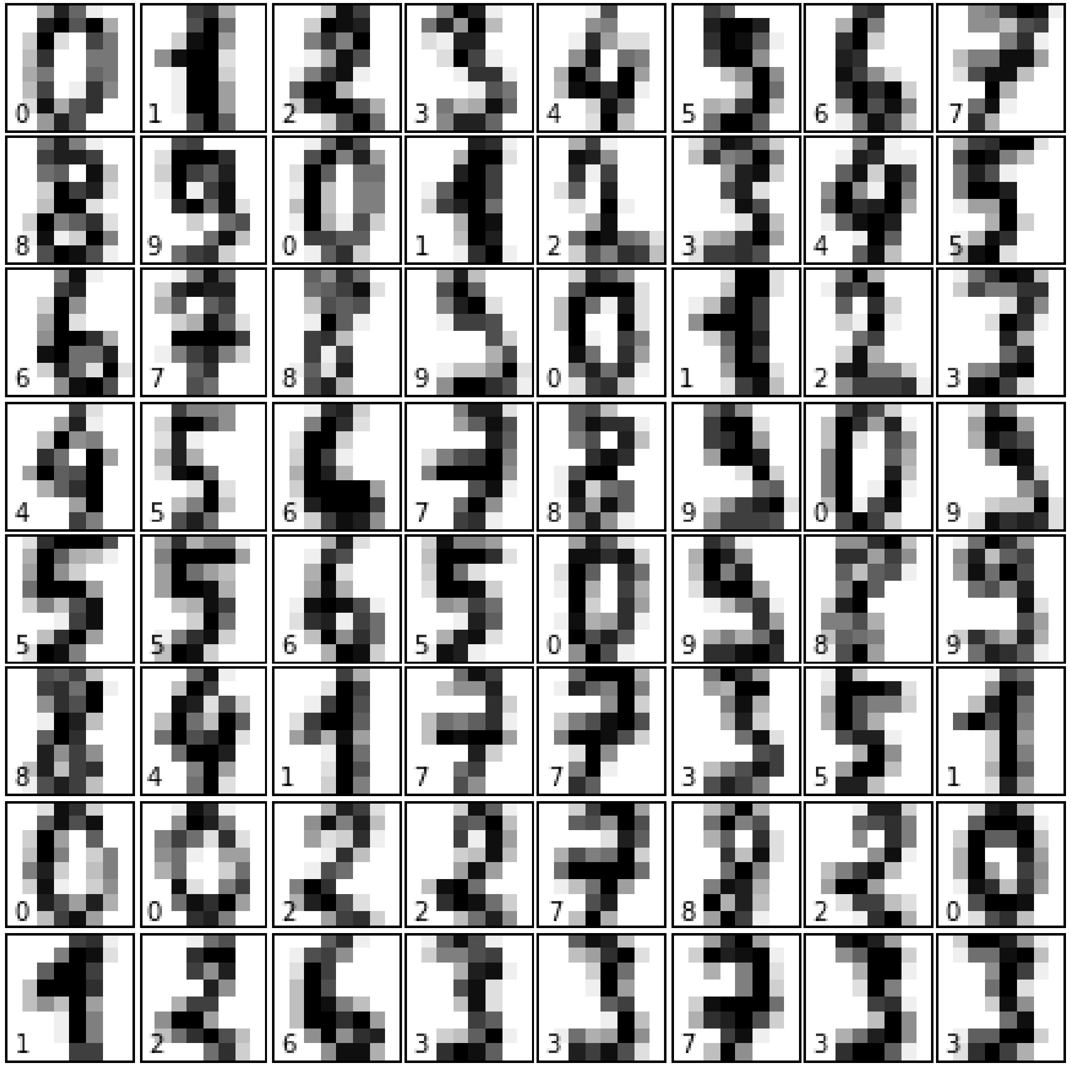
Out[3]:
```
dict_keys(['target_names', 'data', 'feature_names', 'DESCR', 'target'])
```

Iris-Setosa

Iris-Versicolor

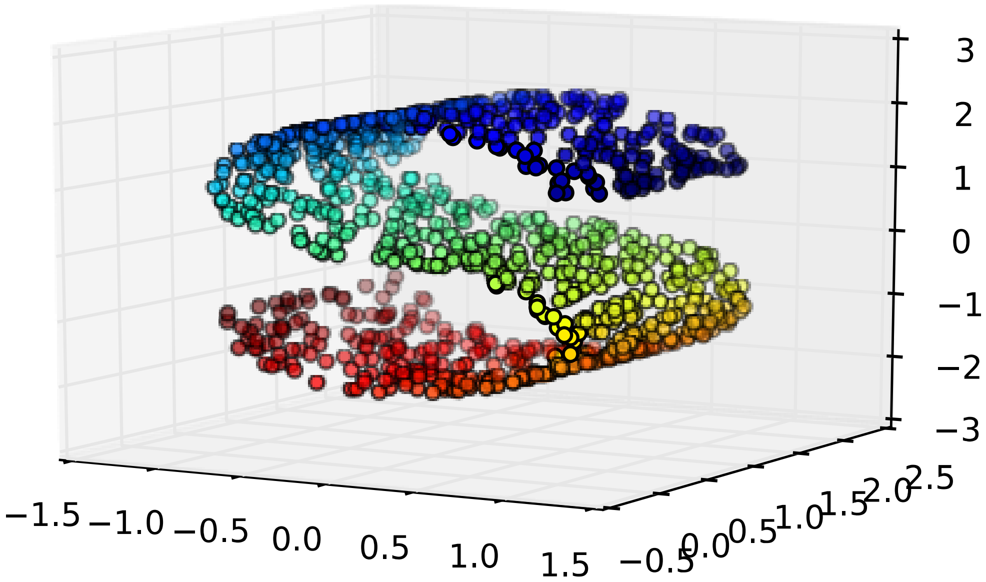
Iris-Setosa

Digits

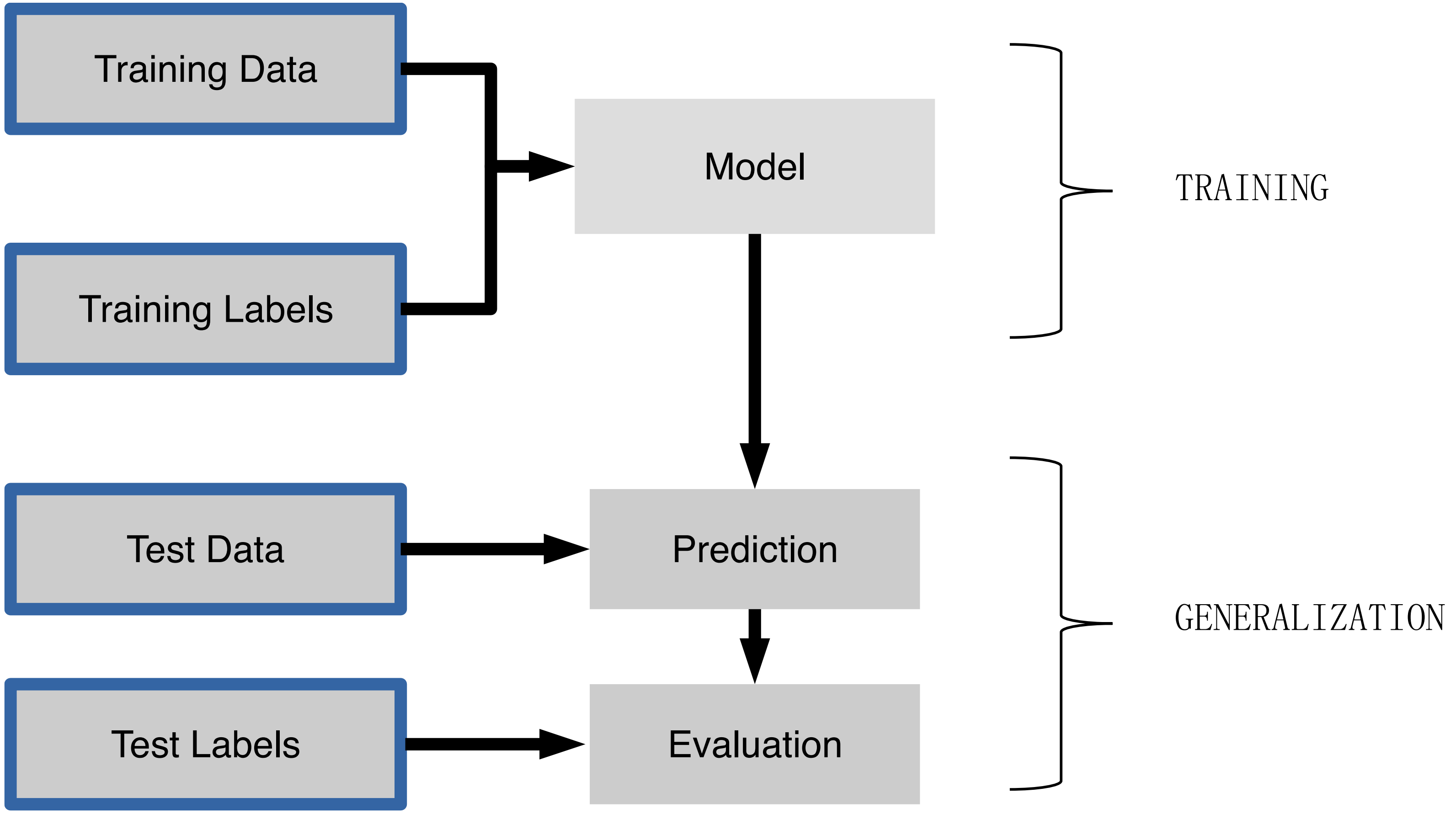

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
9
5
5
6
5
0
9
8
9
8
4
1
7
7
3
5
1
0
0
2
2
7
8
2
0
1
2
6
3
3
7
3
3

Generating Synthetic Data

```
from sklearn.datasets import make_…
```

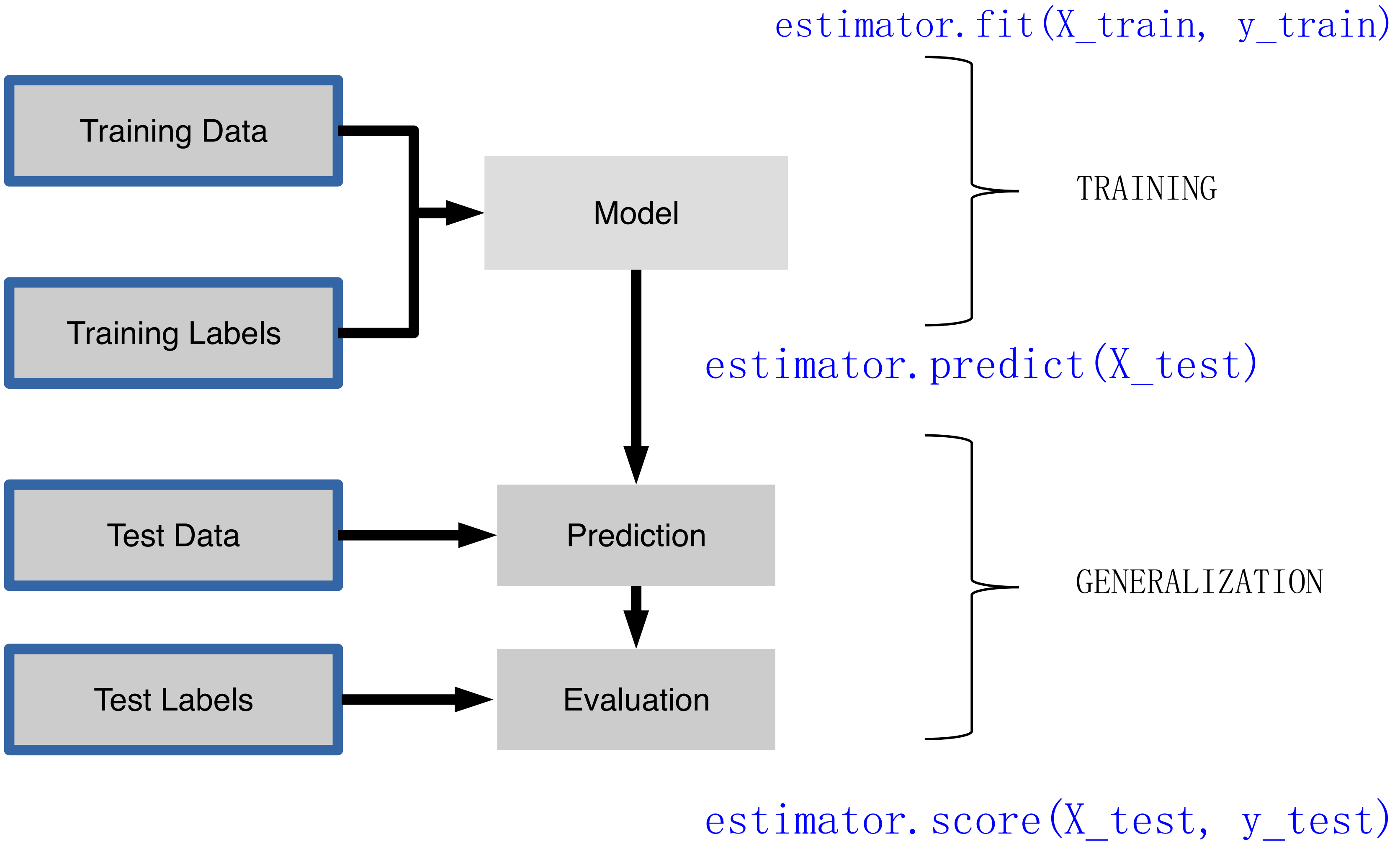


Supervised Workflow



- Fit model on all data after evaluation

Supervised Workflow



Regression Shrinkage and Selection via the Lasso

Regularization

All the answers so far are of the form

$$\widehat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

They require the inversion of $\mathbf{X}^T\mathbf{X}$. This can lead to problems if the system of equations is poorly conditioned. A solution is to add a small element to the diagonal:

$$\widehat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X} + \delta^2 I_d)^{-1}\mathbf{X}^T\mathbf{y}$$

This is the ridge regression estimate. It is the solution to the following **regularised quadratic cost function**

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Derivation

identity matrix

$$\frac{\partial}{\partial\theta} J(\theta) = \frac{\partial}{\partial\theta}\left\{ (y - X\theta)^T (y - X\theta) + \delta^2 \theta I \theta \right\}$$

$$= \frac{\partial}{\partial\theta}\left\{ y^T y - 2y^T X\theta + \theta^T X^T X\theta + \theta^T (\delta^2 I)\theta \right\}$$

$$= -2X^T y + 2X^T X\theta + 2\delta^2 I \theta$$

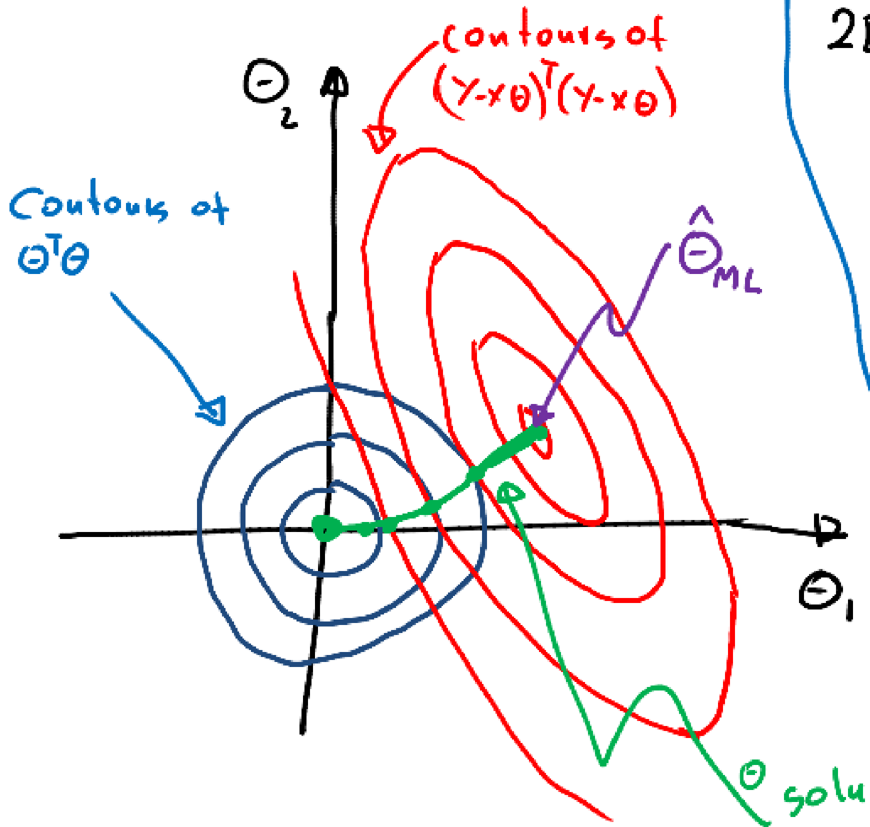
$$= -2X^T y + 2(X^T X + \delta^2 I)\theta$$

Equating to zero, yields

$$\hat{\theta}_{ridge} = (X^T X + \delta^2 I)^{-1} X^T y$$

Ridge regression as constrained optimization

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta} \qquad \min_{\boldsymbol{\theta}\,:\ \boldsymbol{\theta}^T\boldsymbol{\theta} \leq t(\delta)} \{(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})\}$$



θ solutions for different values of δ.

2D Example $\underline{\theta} = (\theta_1, \theta_2)$

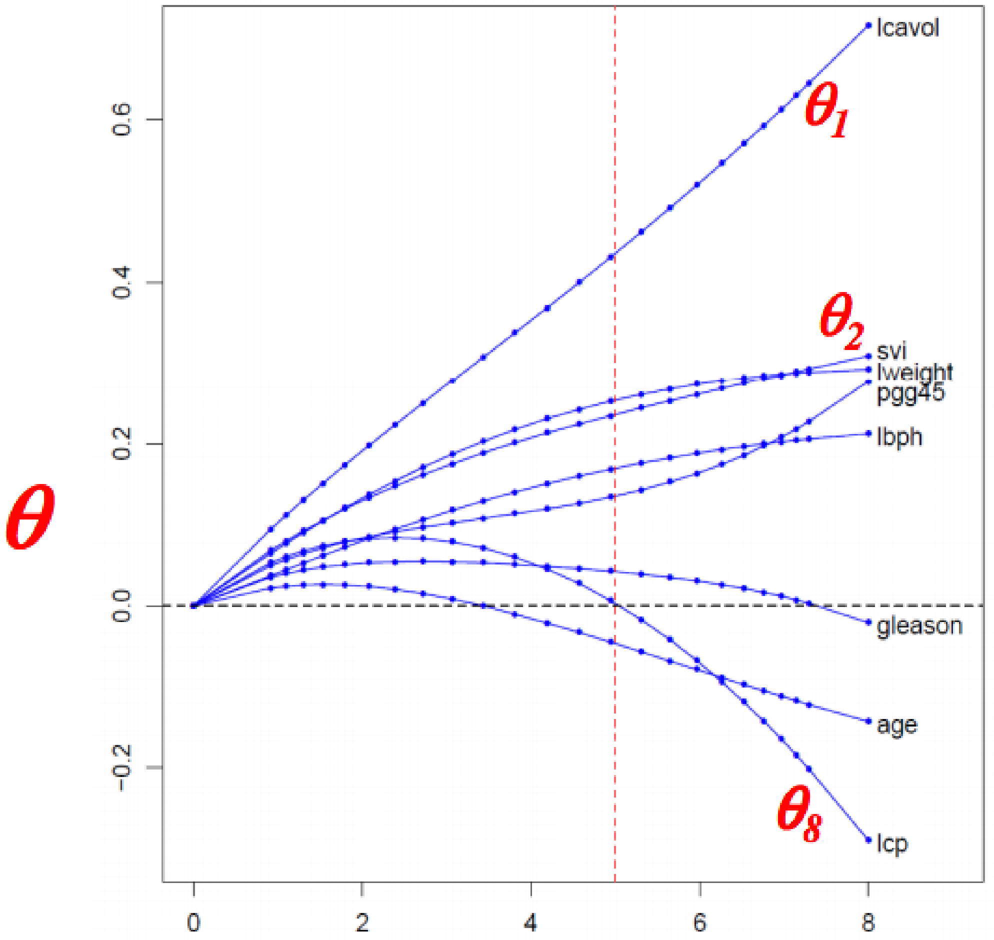
$\underline{\theta}^T\underline{\theta} = t$

$[\theta_1 \ \ \theta_2]\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = t$

$\theta_1^2 + \theta_2^2 = t$

(circles!)

As δ increases, $t(\delta)$ decreases and each θ_i goes to zero.



Ridge, feature selection, shrinkage and weight decay

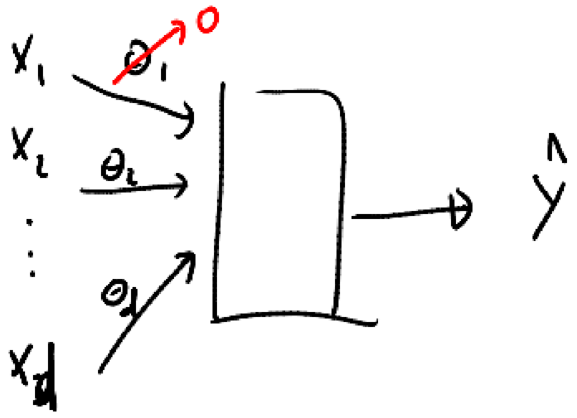
Large values of $\boldsymbol{\theta}$ are penalised. We are *shrinking* $\boldsymbol{\theta}$ towards zero. This can be used to carry out *feature weighting*. An input $x_{i,d}$ weighted by a small θ_d will have less influence on the ouptut y_i. This penalization with a regularizer is also known as weight decay in the neural networks literature.

Note that shrinking the bias term $\boldsymbol{\theta}_1$ is undesirable. To keep the notation simple, we will assume that the mean of $\mathbf{y}$ has been subtracted from $\mathbf{y}$. This mean is indeed our estimate $\widehat{\boldsymbol{\theta}_1}$.

```python
from keras.regularizers import l2, activity_l2

model.add(Dense(64, input_dim=64, W_regularizer=l2(0.01)))
```

Selecting features for prediction



$$\hat{y} = x_1 \overset{\to 0}{\cancel{\theta_1}} + x_2\theta_2 + \dots + x_d\theta_d$$

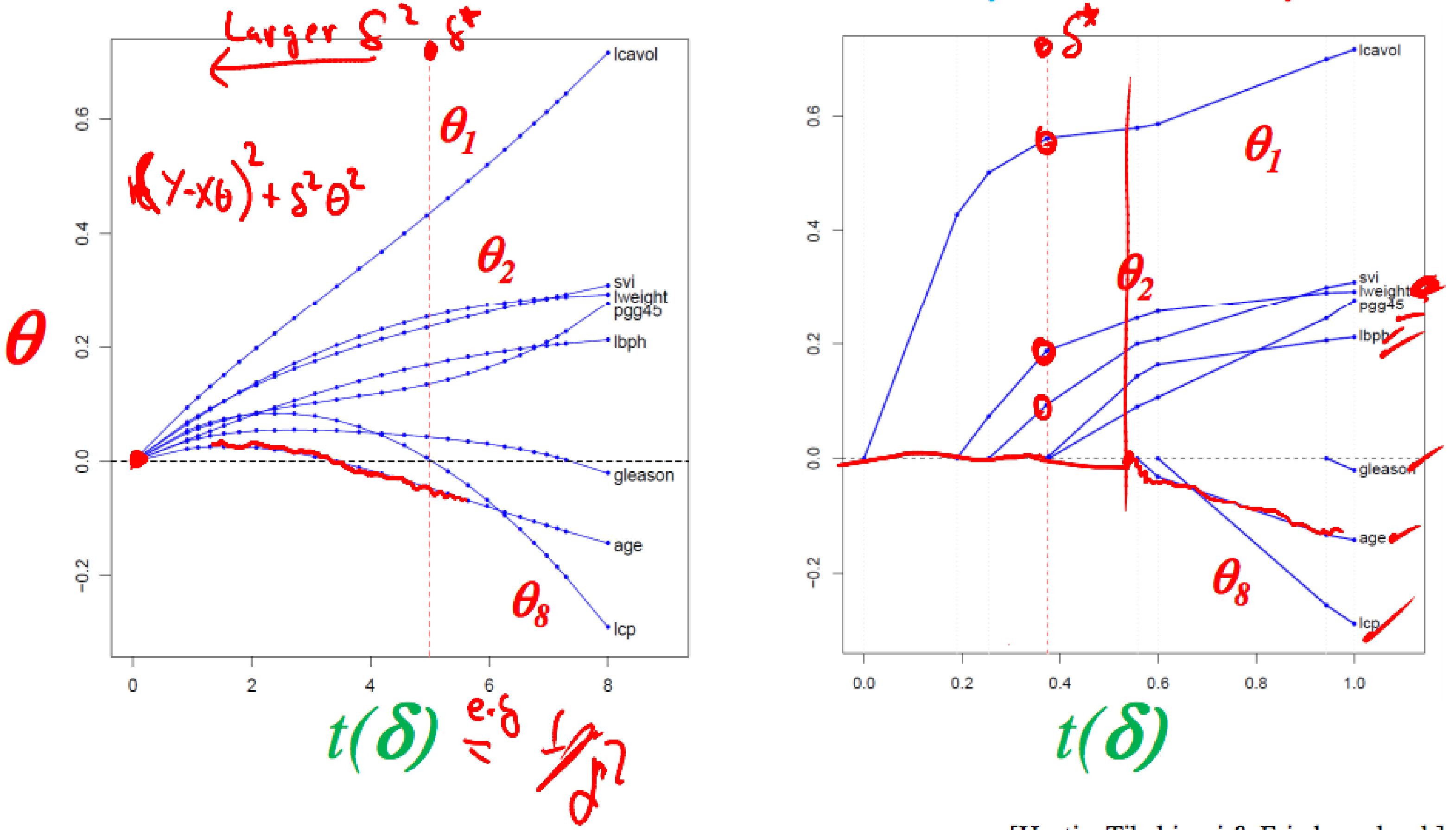
X_1 is expensive

X_1 does not contribute to good predictions $\hat{y}$

Then we want $\theta_1 \to 0$

Selecting features for prediction

As δ increases, $t(\delta)$ decreases and each θ_i goes to zero, but too slowly for ridge. Lasso will ensure that irrelevant features x_i have weight $\theta_i = 0$.

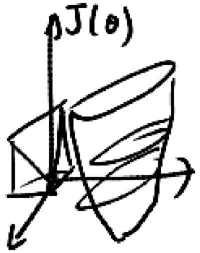


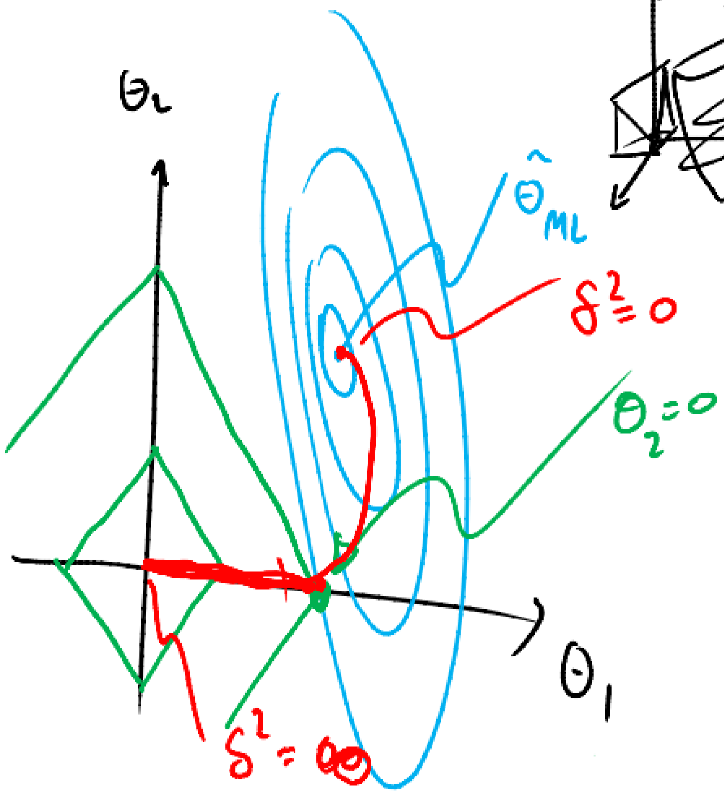
[Hastie, Tibshirani & Friedman book]

The Lasso: least absolute selection and shrinkage operator

$$J(\theta) = \underbrace{(Y - X\theta)^T (Y - X\theta)} + \delta^2 \underbrace{\sum_{j=1}^{d} |\theta_j|}$$

L_1 Norm





in 2D $\quad \theta = (\theta_1, \theta_2)$

$$|\theta_1| + |\theta_2| = \text{const}$$

$$\theta_1 + \theta_2 = \text{const}$$

$$\theta_1 - \theta_2 = \text{const}$$

$$-\theta_1 - \theta_2 = \text{const}$$

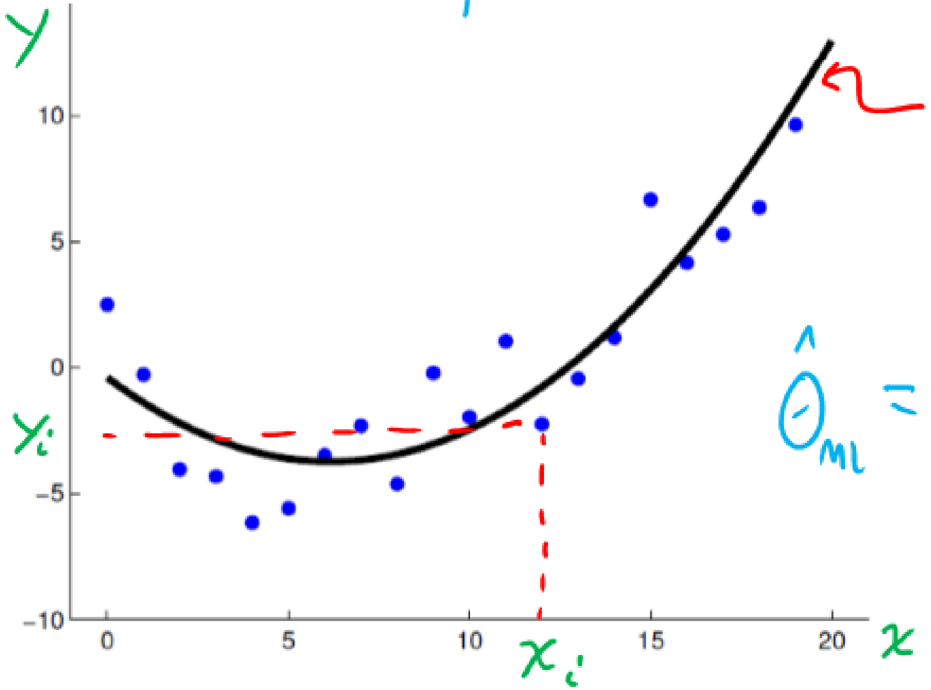
$$-\theta_1 + \theta_2 = \text{const}$$

Going nonlinear via basis functions

We introduce basis functions $\phi(\cdot)$ to deal with nonlinearity:

$$y(\mathbf{x}) = \underbrace{\phi(\mathbf{x})}_{X}\boldsymbol{\theta} + \epsilon$$

For example, $\phi(x) = [1, x, x^2]$



$\hat{y} = \phi(x)\Theta$

$= \Theta_0 + x\Theta_1 + x^2\Theta_2$

$\hat{\Theta}_{ML} = [\phi(x)^T\phi(x)]^{-1}\phi(x)^T y$

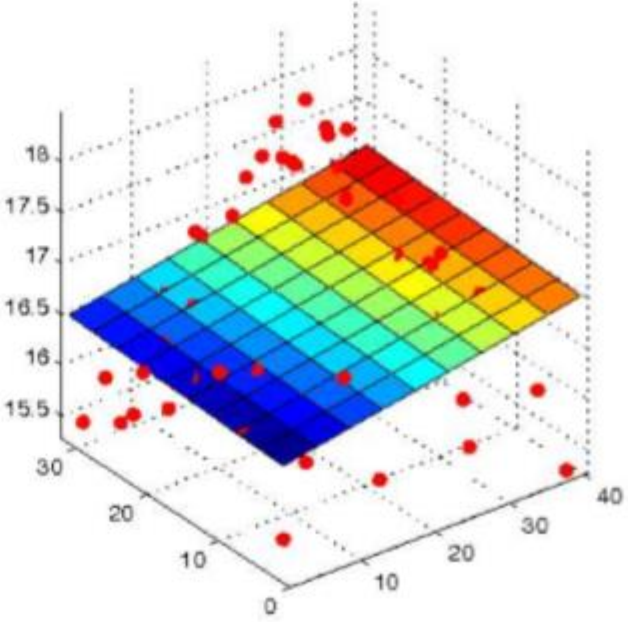
Going nonlinear via basis functions

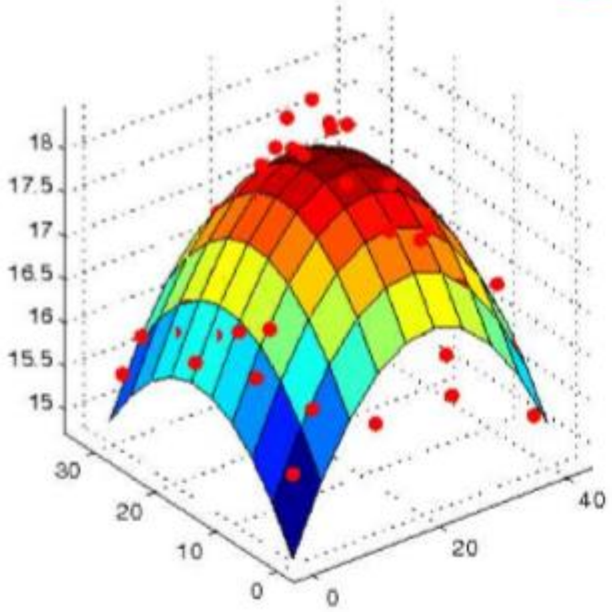
$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\theta} + \epsilon$$

$$\boldsymbol{\phi}(\mathbf{x}) = \underbrace{[1, x_1, x_2]}$$

x_1x_2

$$\boldsymbol{\phi}(\mathbf{x}) = \underbrace{[1, x_1, x_2, x_1^2, x_2^2]}$$



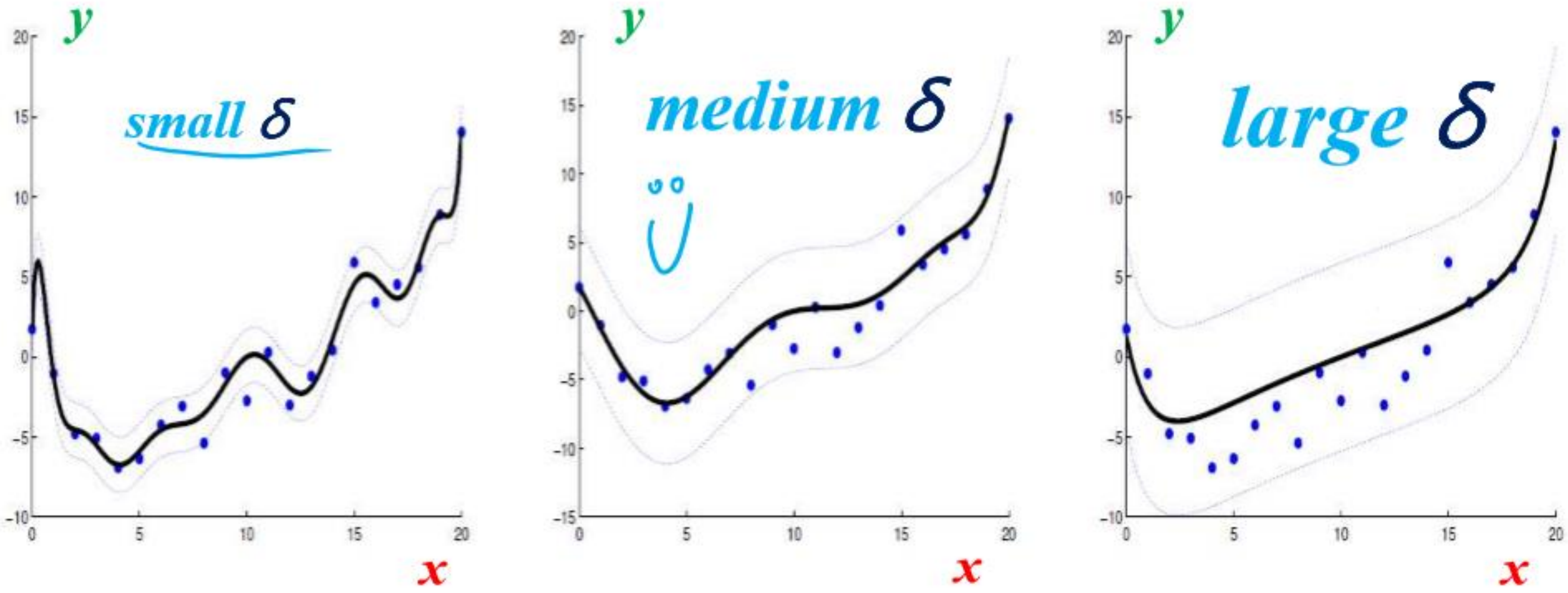


<u>Example</u>: Ridge regression with a polynomial of degree 14

$$\hat{y}(x_i) = 1\,\theta_0 + x_i\,\theta_1 + x_i^2\,\cancel{\theta_2}^{\,0} + \ldots + x_i^{13}\,\theta_{13} + x_i^{14}\,\theta_{14}$$

$$\underline{\Phi} = [\,1 \;\; x_i \;\; x_i^2 \;\; \ldots \;\; x_i^{13} \;\; x_i^{14}\,]\; x_i^{15} \ldots$$

$$J(\theta) = (y - \Phi\theta)^T(y - \Phi\theta) + \underbrace{\delta\,\theta^T\theta}$$



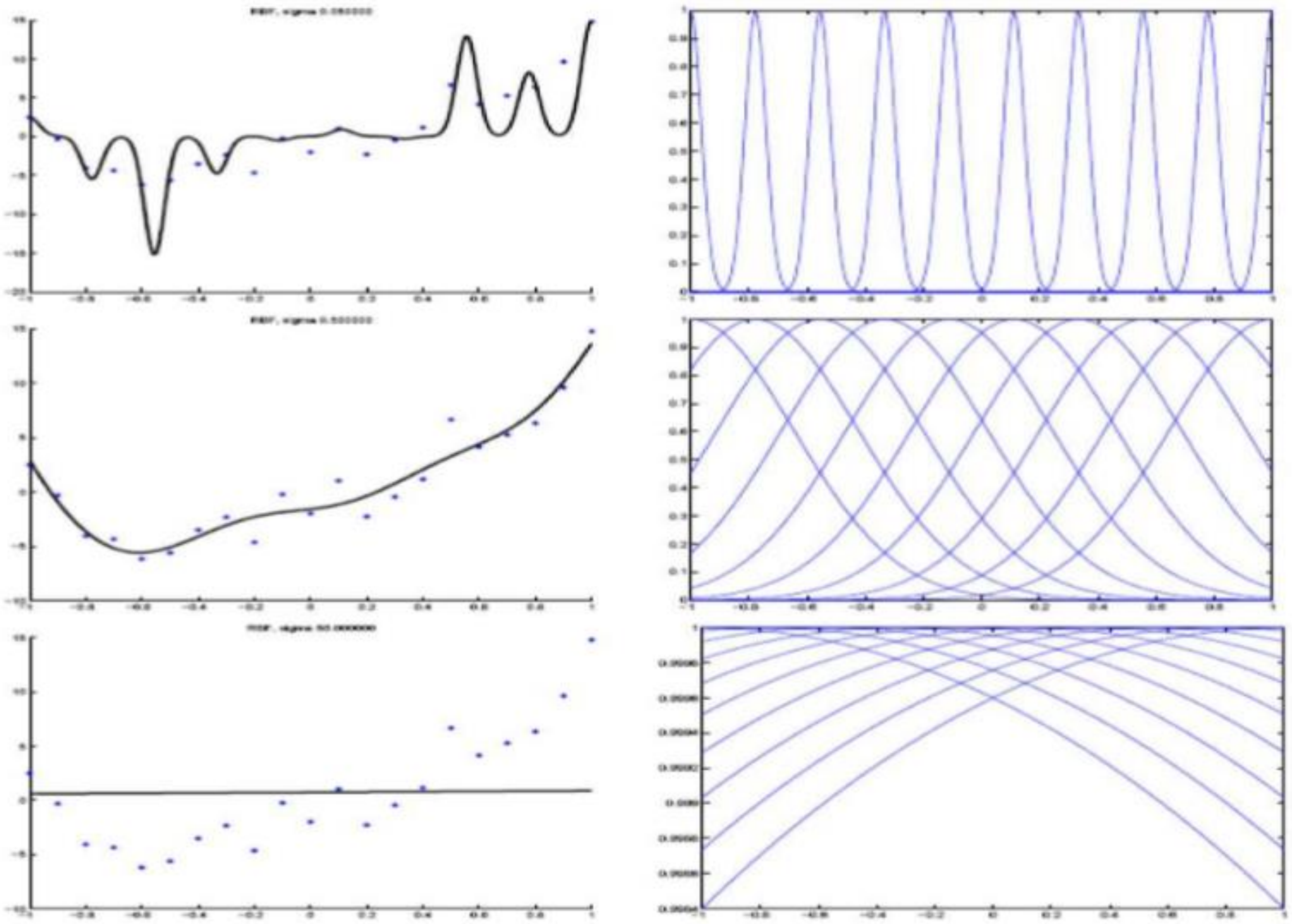
Kernel regression and RBFs

We can use kernels or radial basis functions (RBFs) as features:

$$\boxed{\phi(\mathbf{x})} = [\kappa(\mathbf{x}, \boldsymbol{\mu}_1, \lambda), \ldots, \kappa(\mathbf{x}, \boldsymbol{\mu}_d, \lambda)], \quad e.g.\ \kappa(\mathbf{x}, \boldsymbol{\mu}_i, \lambda) = e^{(-\frac{1}{\lambda}\|\mathbf{x}-\boldsymbol{\mu}_i\|^2)}$$

We can choose the locations μ of the ***basis functions*** *to be the inputs.*
That is, $\mu_i = x_i$. These basis functions are the known as ***kernels****.*
The choice of width λ is tricky, as illustrated below.

kernels



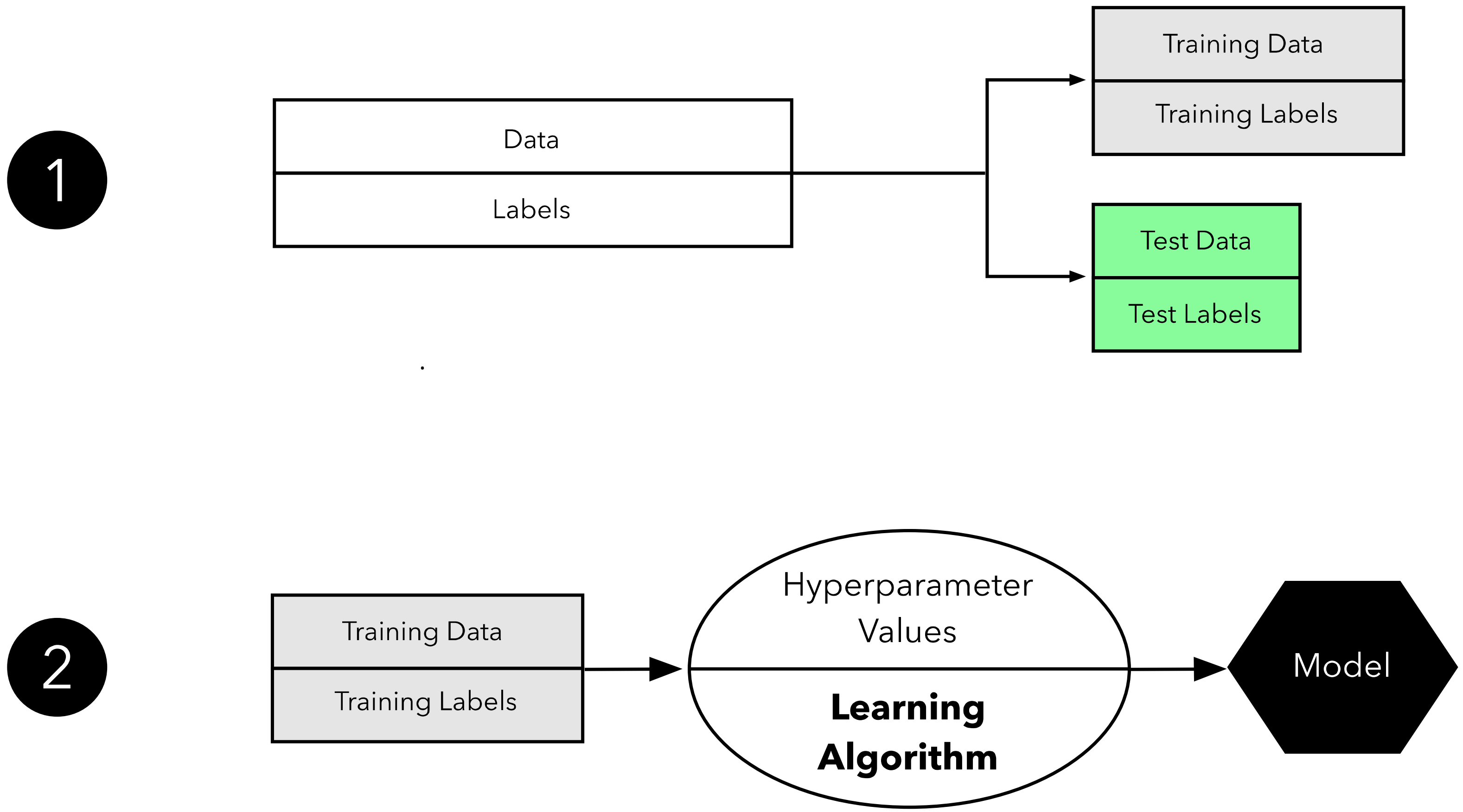
Too small λ

Right λ

Too large λ

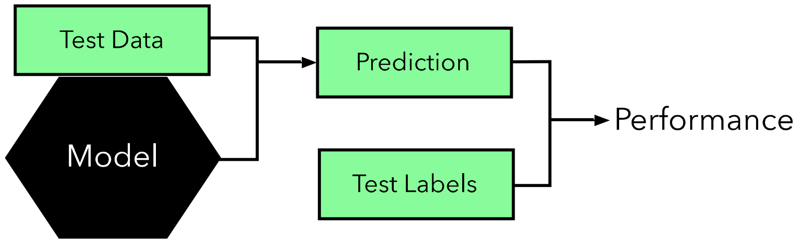
The big question is how do we choose the regularization coefficient, the width of the kernels or the polynomial order?

Holdout Evaluation I

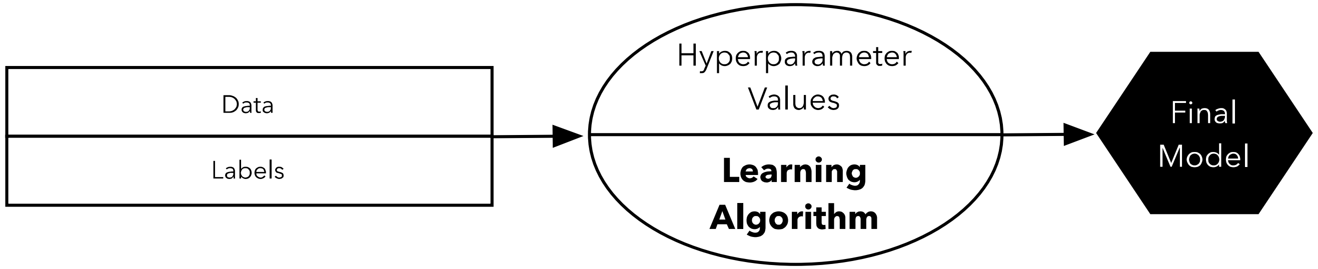


Holdout Evaluation II

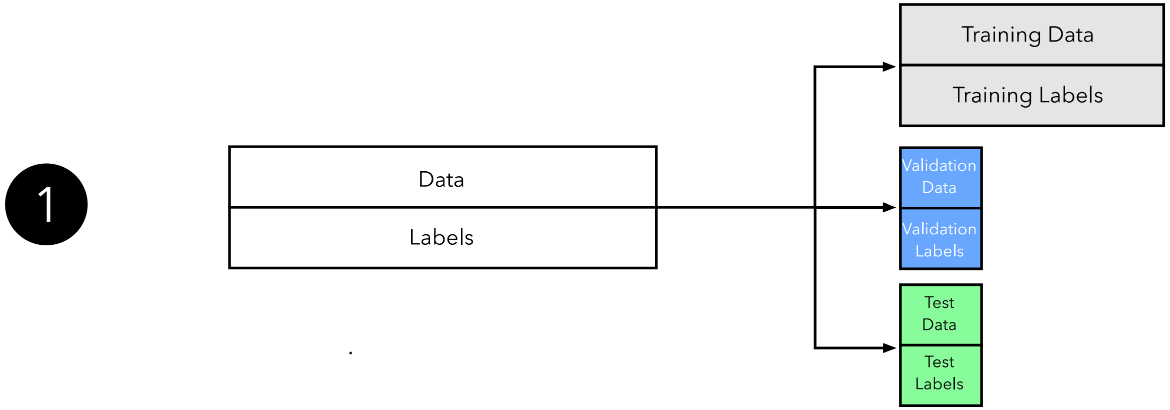
3

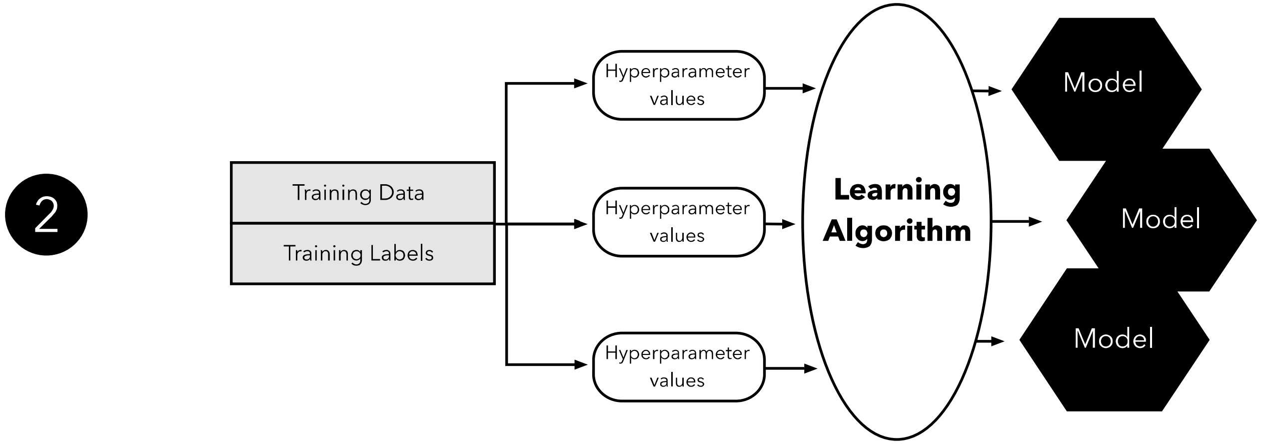


4



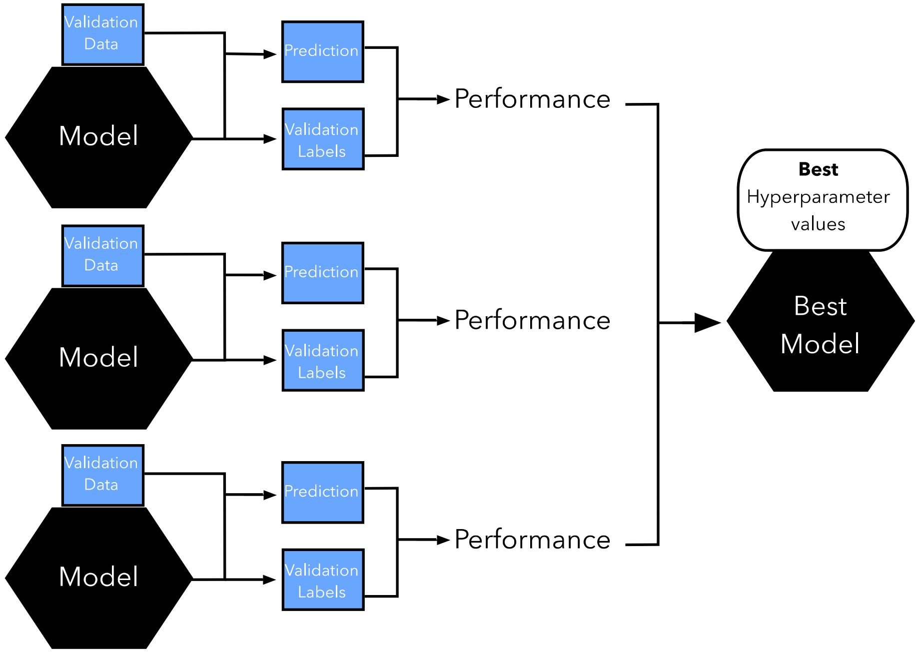
Holdout Validation I



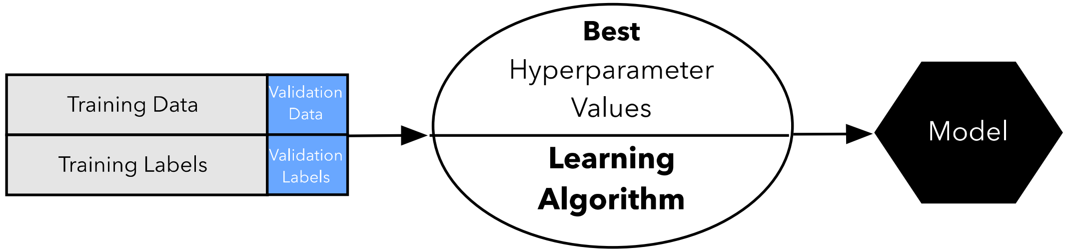


Holdout Validation II

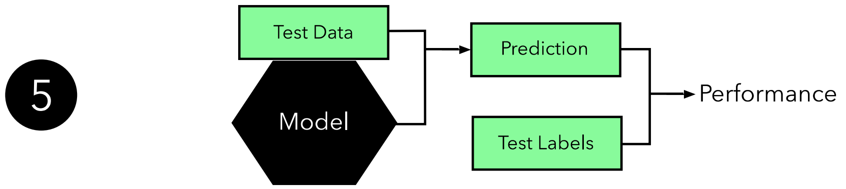
3

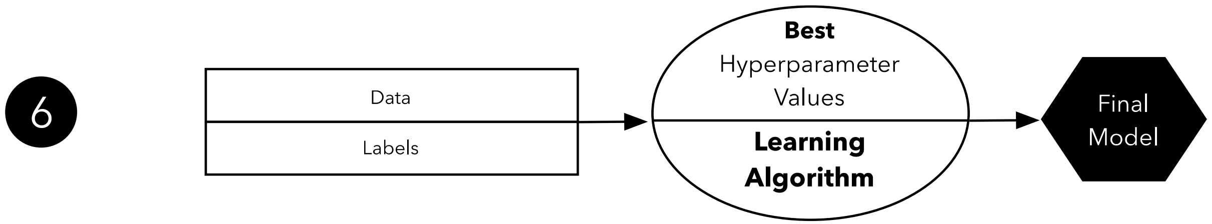


4

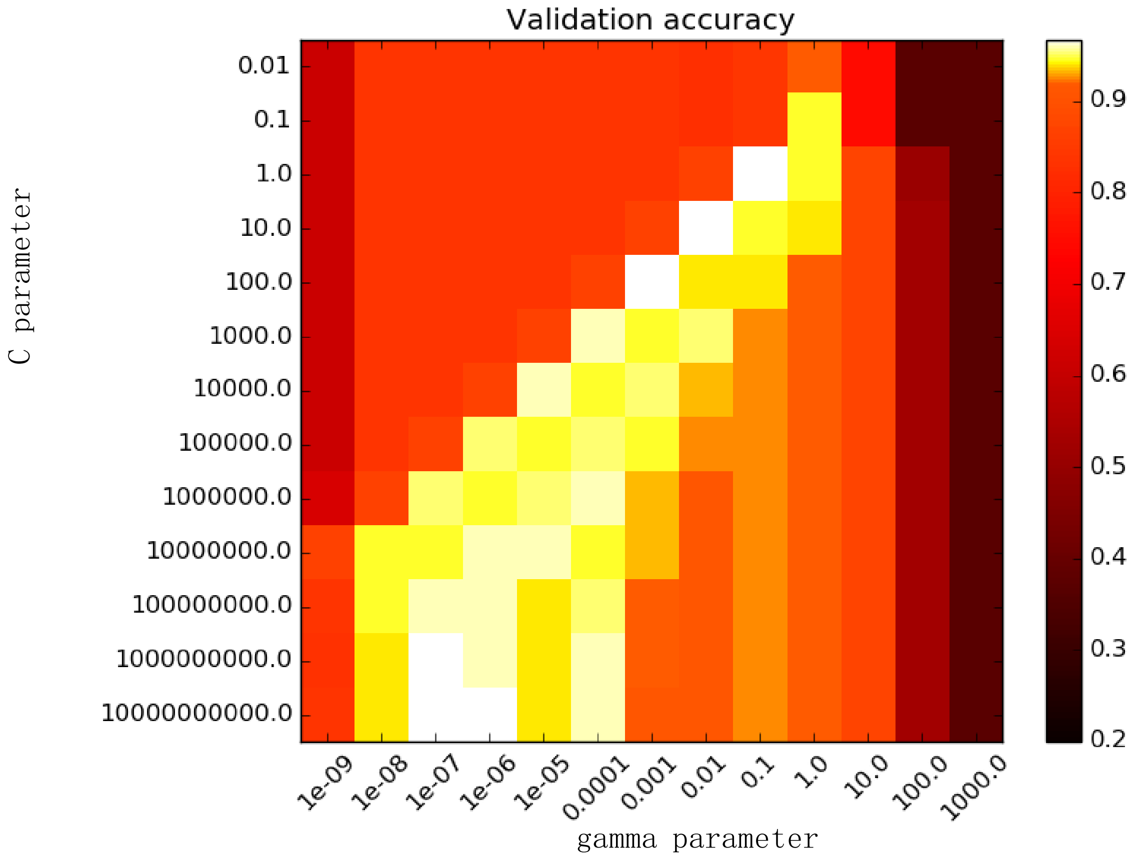


Holdout Validation III





Grid Search



Validation accuracy
C parameter
0.01
0.1
1.0
10.0
100.0
1000.0
10000.0
100000.0
1000000.0
10000000.0
100000000.0
1000000000.0
10000000000.0
1e-09
1e-08
1e-07
1e-06
1e-05
0.0001
0.001
0.01
0.1
1.0
10.0
100.0
1000.0
gamma parameter
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

Now, big question

- How to define input X?

- http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators