Linear, Polynomial, and Logistic Regression

Supervised Learning

Regressions in Machine Learning

Regression problems

- Linear regression with one variable
- Linear regression with multiple variables
- Polynomial regression

Classification problems

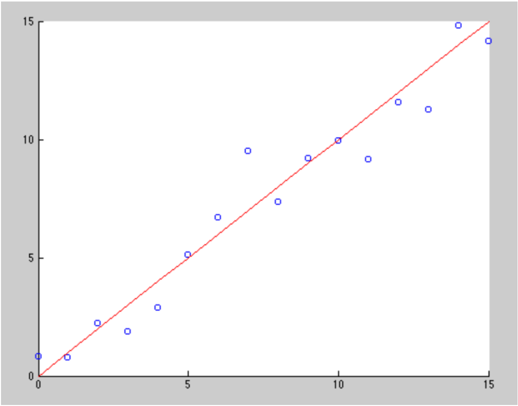
- Logistic regression

Linear Regression with one variable

Supervised Learning

Representation

A straight line with variable intercept and offset

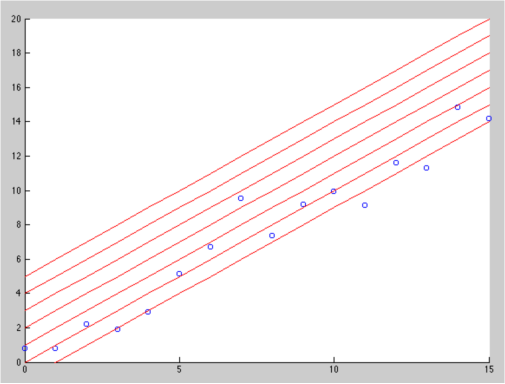


$$h_\theta(x) = \theta_0 + \theta_1 x$$

θ_0 - y intercept
θ_1 - slope

θ_0, θ_1 - parameters

θ_0 up



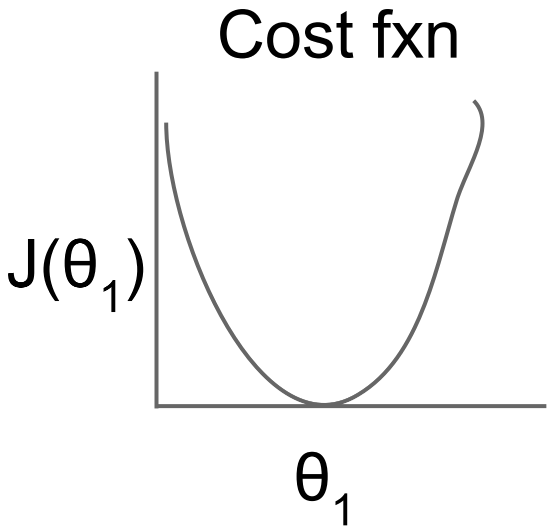
Cost function

Mean squared error

- $$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Cost function with θ_1 only

Let $\theta_0 = 0$ to simplify. (line passes through origin)

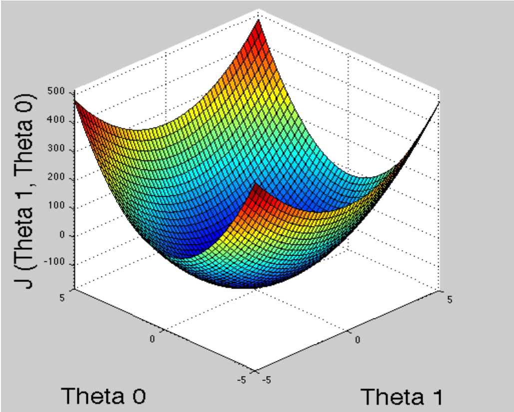


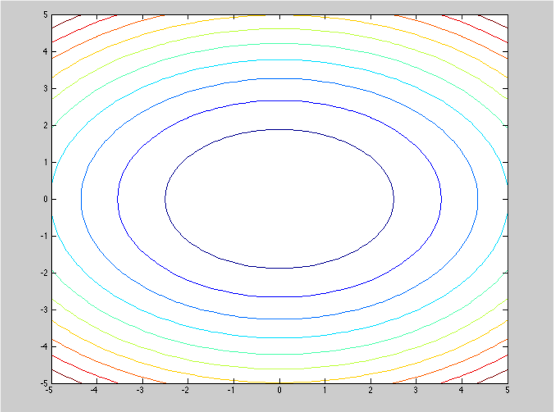
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta x^{(i)} - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

Cost function with θ_0 and θ_1

3 dimensional “bow shaped” cost function

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta x^{(i)} - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$





Gradient Descent

- Can minimize J
- Used widely in machine learning
- Can minimize large number of parameters

General procedure:

1) Start with some θ_0 and θ_1 (typically 0,0)
2) Change θ_0 and θ_1 to reduce $J(\theta_0, \theta_1)$
3) Repeat 2 until at the minimum

Definition of Gradient Descent

Partial derivative (or gradient) of the cost function

$$\theta_j := \theta_j - \alpha \nabla(\theta_j)$$

$$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_0, \theta_1)$$

Applying Gradient Descent to Linear Regression

- Combine the definition of gradient descent and cost function
- Have to do partial derivative of the cost function

Final equations:

$$\theta_0 := \theta_0 - \alpha \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})\, x^{(i)}$$

Properties of gradient descent

- For linear regression always finds global minimum
- Can become unstable if learning rate is too large or too slow if learning rate is too small

Linear Regression with Multiple Variables

Supervised Learning

General form of linear regression

Linear regression can work with more than two parameters

By letting $X_0 = 1$ we can write general form

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \ldots + \theta_n x_n = \theta^T x$$

Applying gradient descent

Algorithm:

Repeat until convergence: $\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_0 \ldots \theta_n)$

Cost function: $J(\theta_0 \ldots \theta_n) = J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})^2$

Final form: $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})\, x_j^{(i)}$

Feature scaling

- If features are different in sizes, gradient descent might take long to converge
- Features can be scaled to be in same range (in approximately $-1<x<1$ range)
- Number of ways to scale such as mean normalization

Learning rate

- Plot number of iterations vs. $J(\theta)$. Should see decrease until convergence
- If learning rate is too large gradient descent might become unstable, and never converge

Polynomial regression

Minimized in the same way as linear regression

For example cubic fit with one feature x:

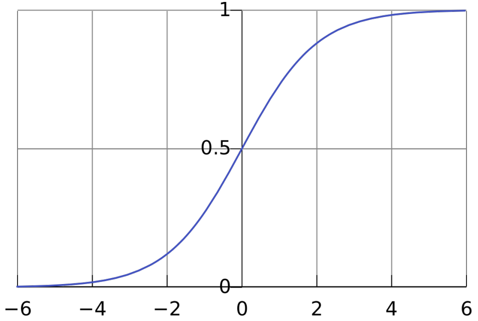
$$h(\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

Generate new feature by squaring cubing the original feature

Logistic regression

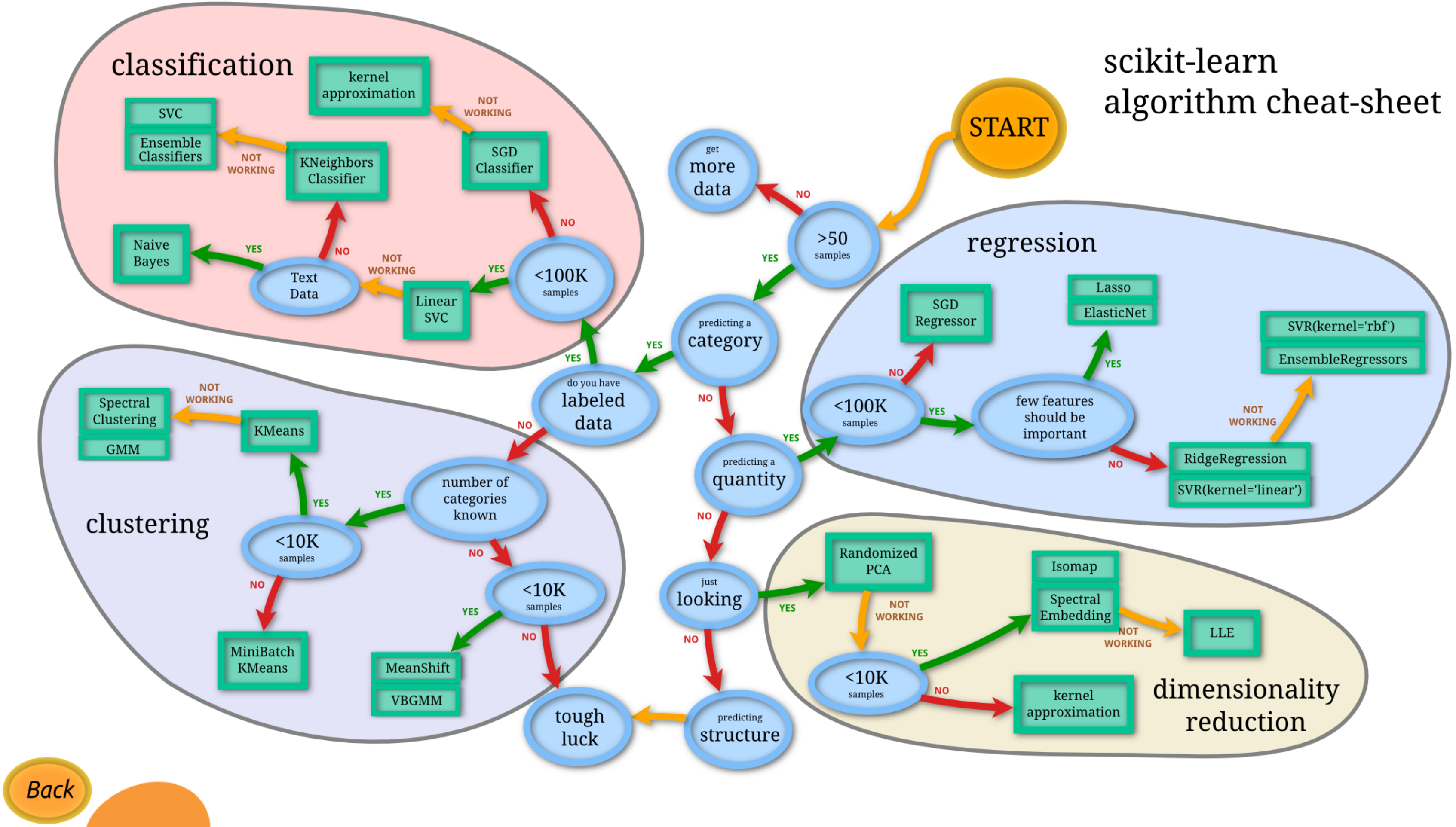
- Binary classification algorithm
- Modify the linear regression to fit logistic function.
- Output is probability of given class

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$



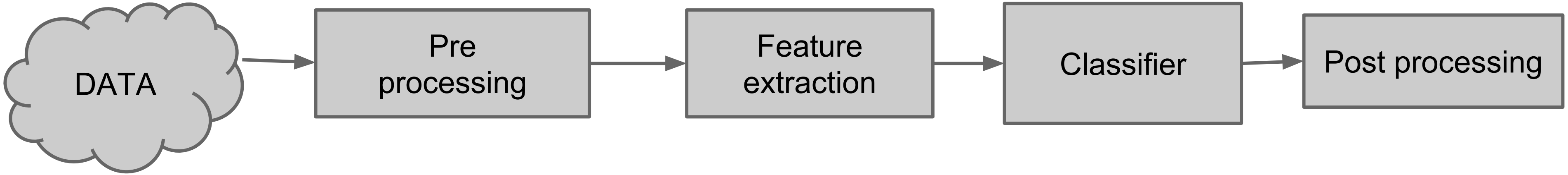
Applying Machine Learning

Supervised and Unsupervised Learning



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

Classifier is only a part of ML system



-What kind of data?
-Understanding the data

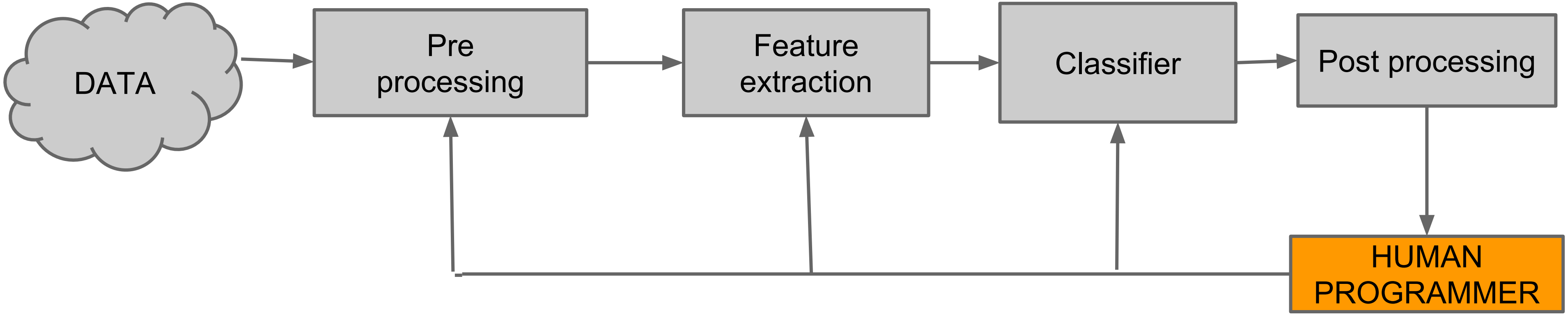
-Normalization
-Scaling
-Filtering
-Outliers
-Missing data

- Spectral analysis
- Statistical features
-

- Type of classifier
- Tuning parameters
- Training data

- Voting
- Human readable form
-Time out

Designing ML system is iterative process



General advices

- It is important to have clean training data
- If human expert can’t classify the data, machine can’t also
- You can’t get something from nothing
- Prototype simple system first

Sources

Great online lectures on linear regression by Andrew Ng : https://class.coursera.org/ml-003/lecture/preview

SKlearn flowchart:

http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html