1. Regresión logística
2. K-means

Meetup Monterrey Data Science & Engineering

CoWo, Monterrey, Nuevo León, México

22 de junio de 2017

Rafael Rodríguez Morales (rafarodrz@gmail.com)

Aprendizaje automático

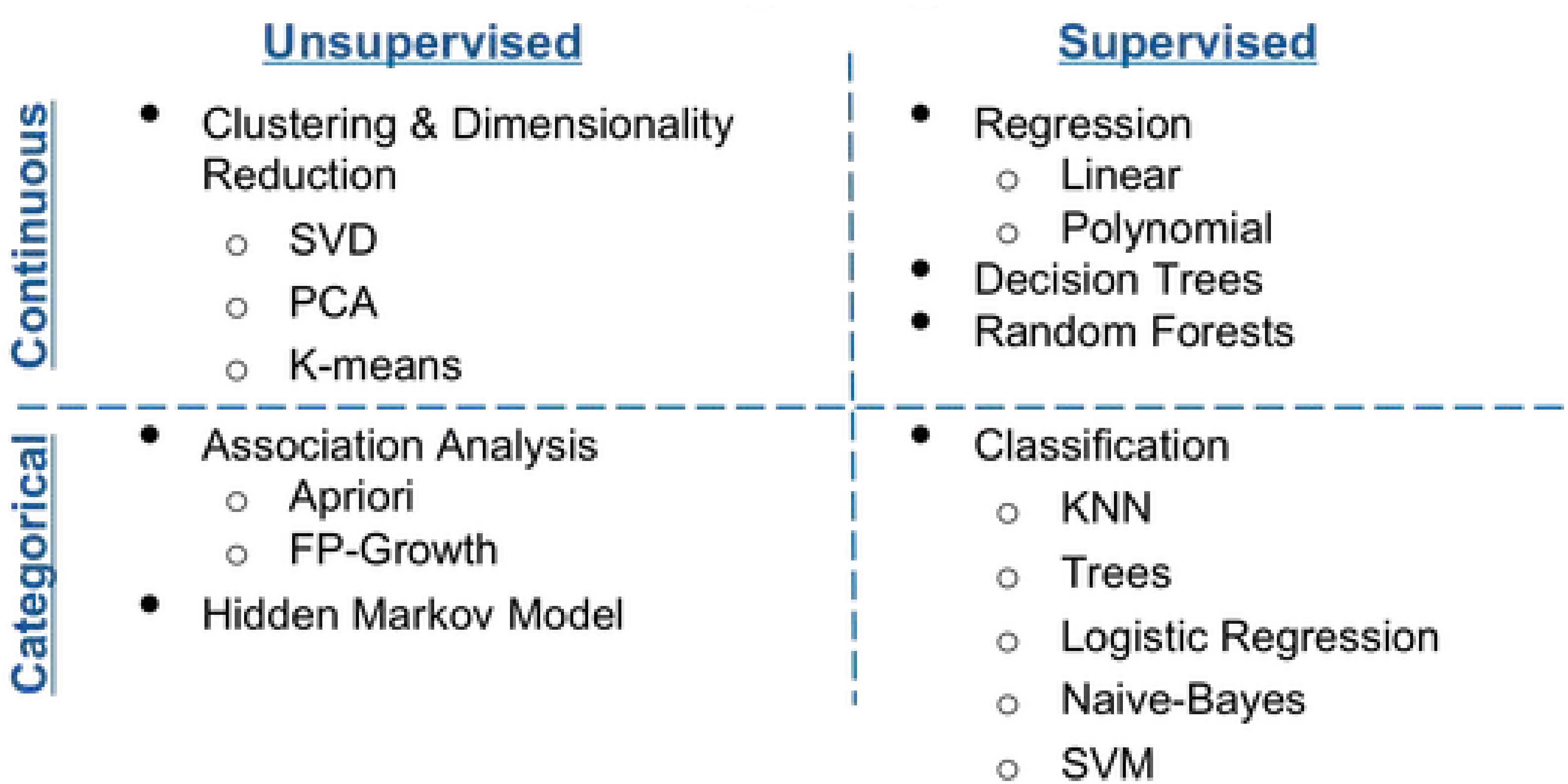


Diagrama de http://mlwithdata.blogspot.mx/2015/04/machine-learning-supervised-vs.html

1.
Regresión logística

0.
Regresión

Regresión

- Tenemos: training set

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

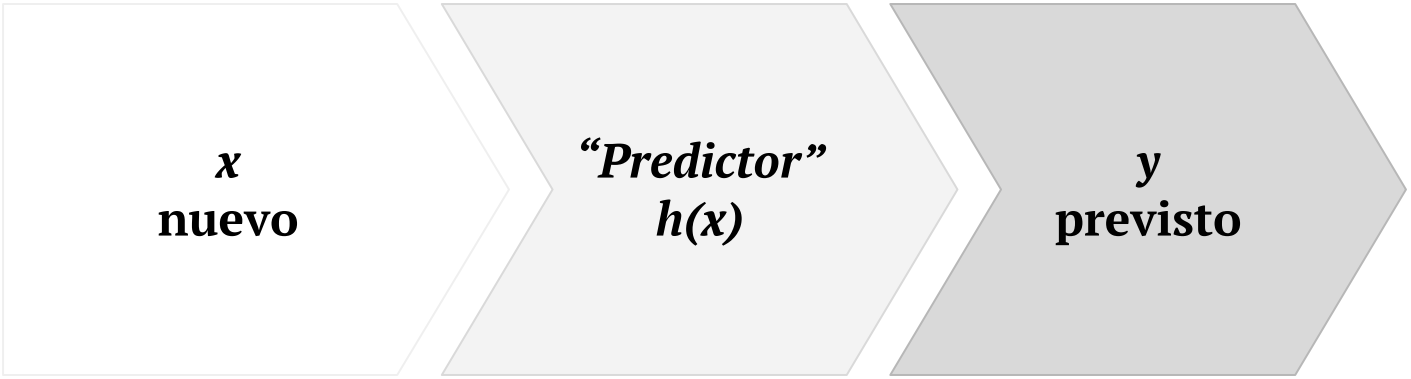
$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

La naturaleza de *y* define si es regresión lineal o regresión logística

- Queremos: los parámetros para la función

$$\mathrm{h}_\theta(x)$$

Regresión



Regresión lineal

- Hipótesis

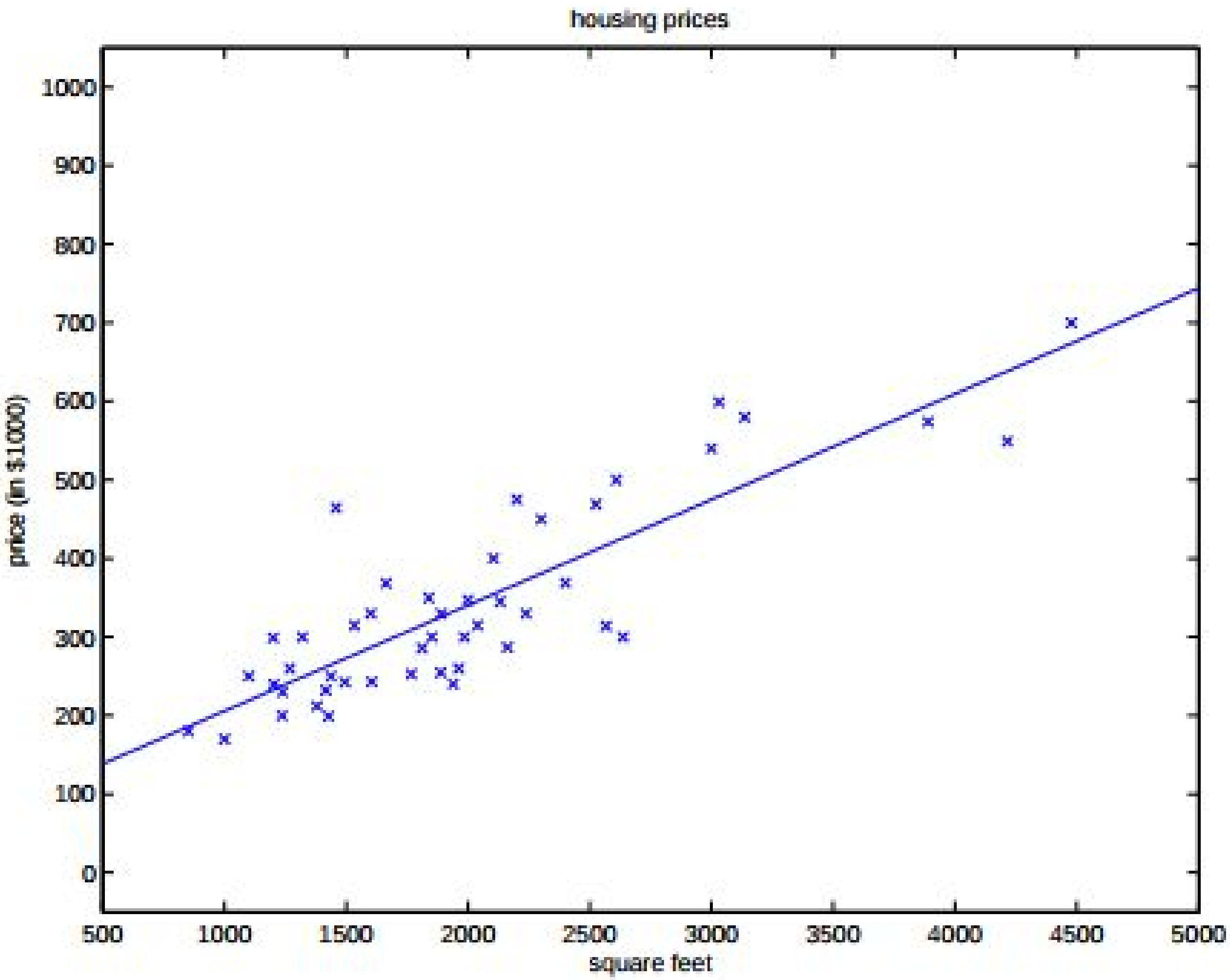
$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

$$\mathrm{h}_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- Función costo (LMS)

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

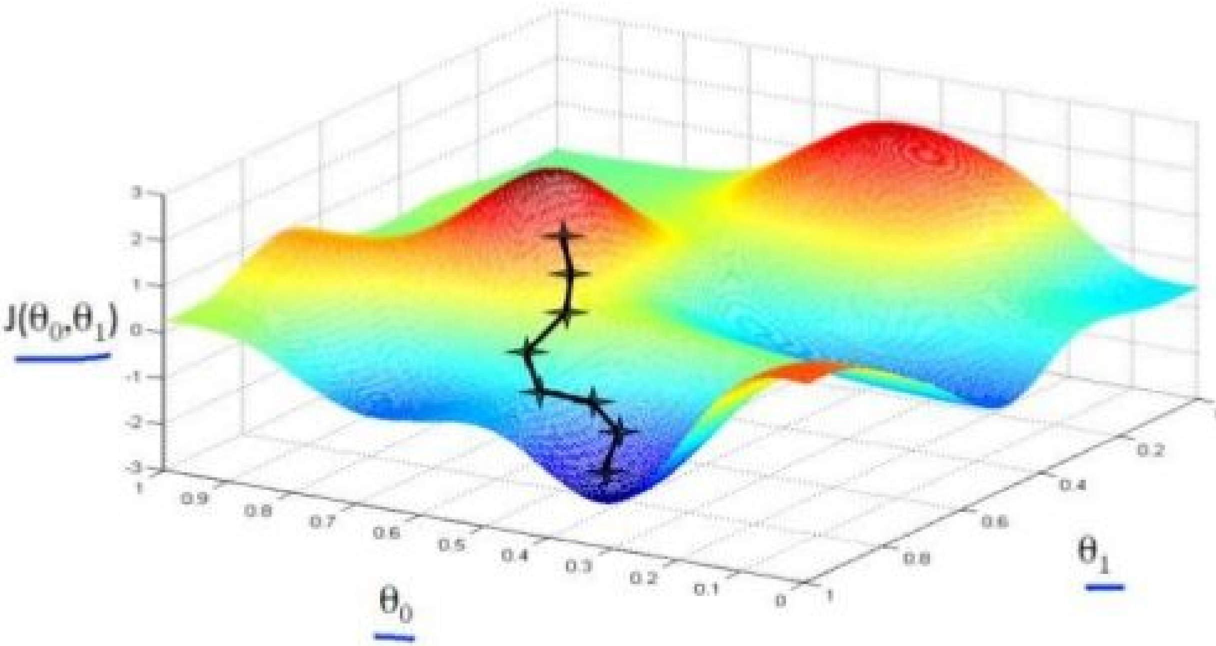
Regresión lineal



Minimizar la función costo

- Gradient descent
- BFGS
- L-BFGS
- Stochastic Average Gradient

Gradient descent idea general



Gradient descent para regresión lineal

- Iterativo
- Parte de vector θ inicial y hace actualizaciones

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

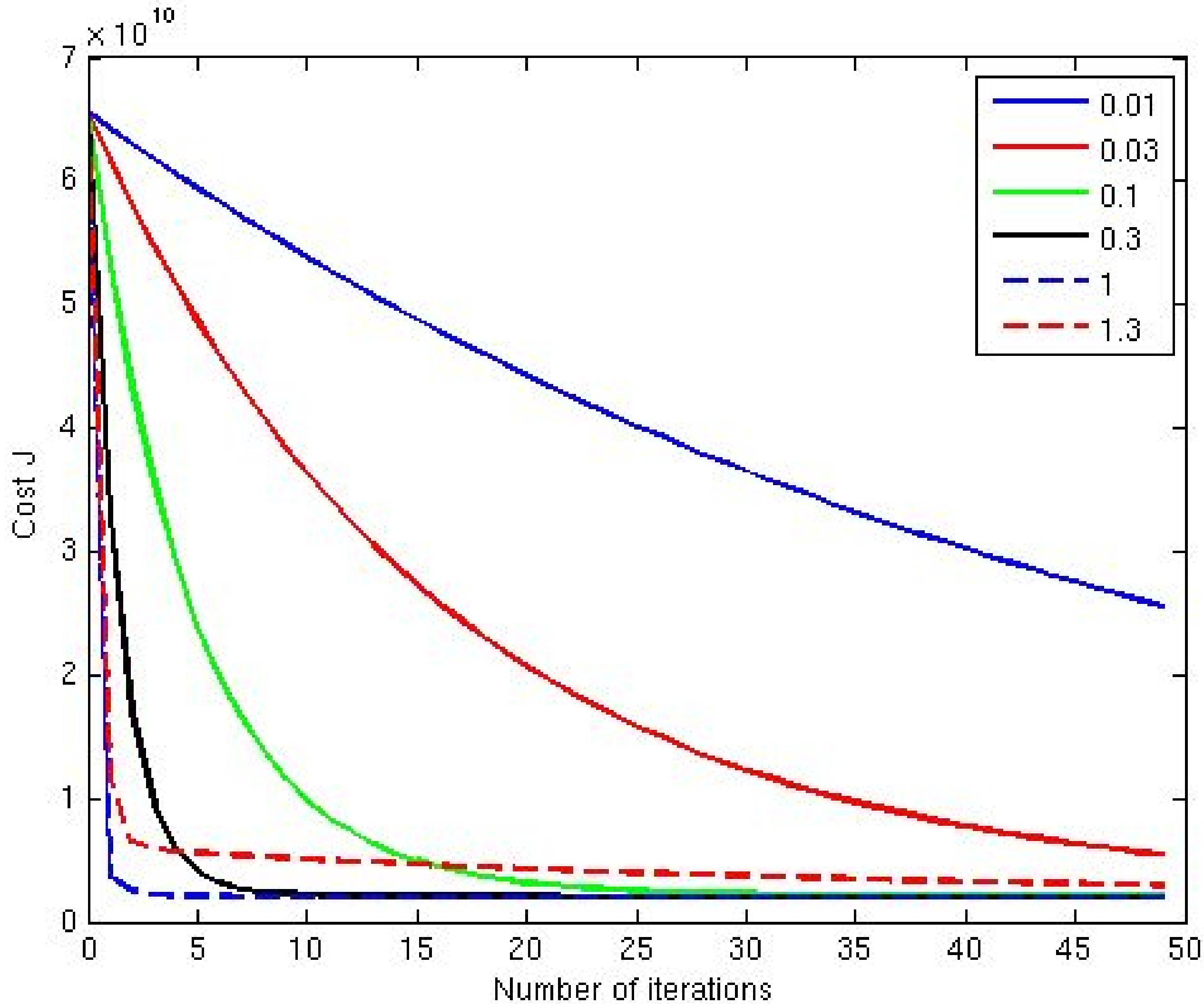
$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}\boxed{J(\theta)}$$

Repeat until convergence {

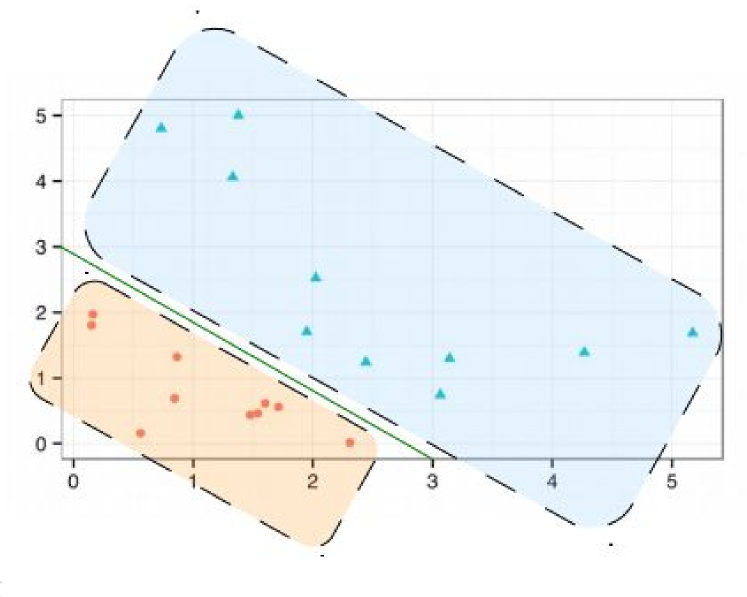
$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}$$

}

Convergencia



1.
Regresión logística



Regresión

- Tenemos: training set

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

La naturaleza de *y* define si es regresión lineal o regresión logística

- Queremos: los parámetros para la función

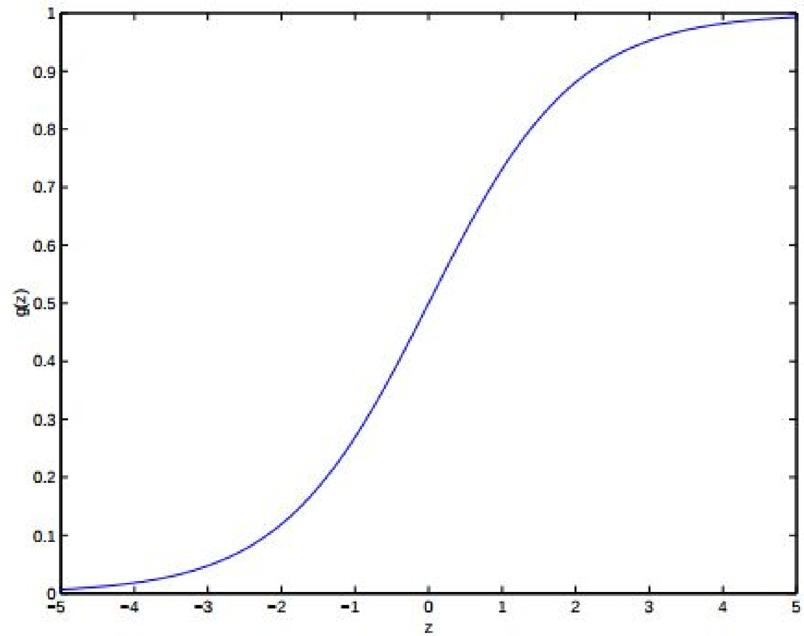
$$h_\theta(x)$$

Regresión logística

- Hipótesis

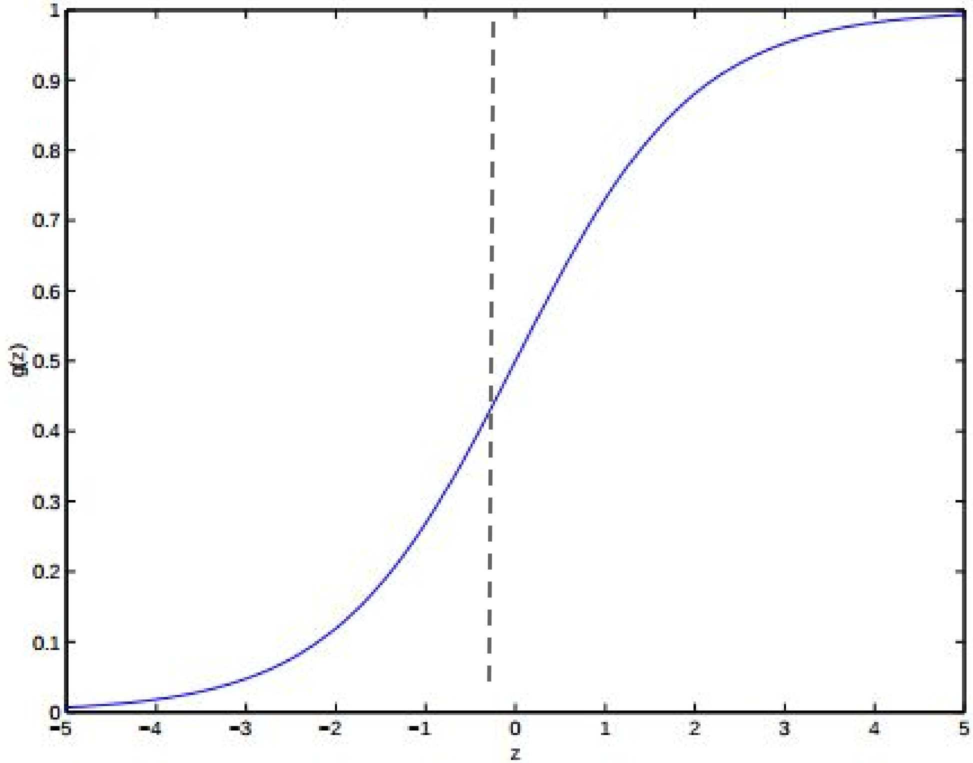
$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

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



Regresión logística

Umbral

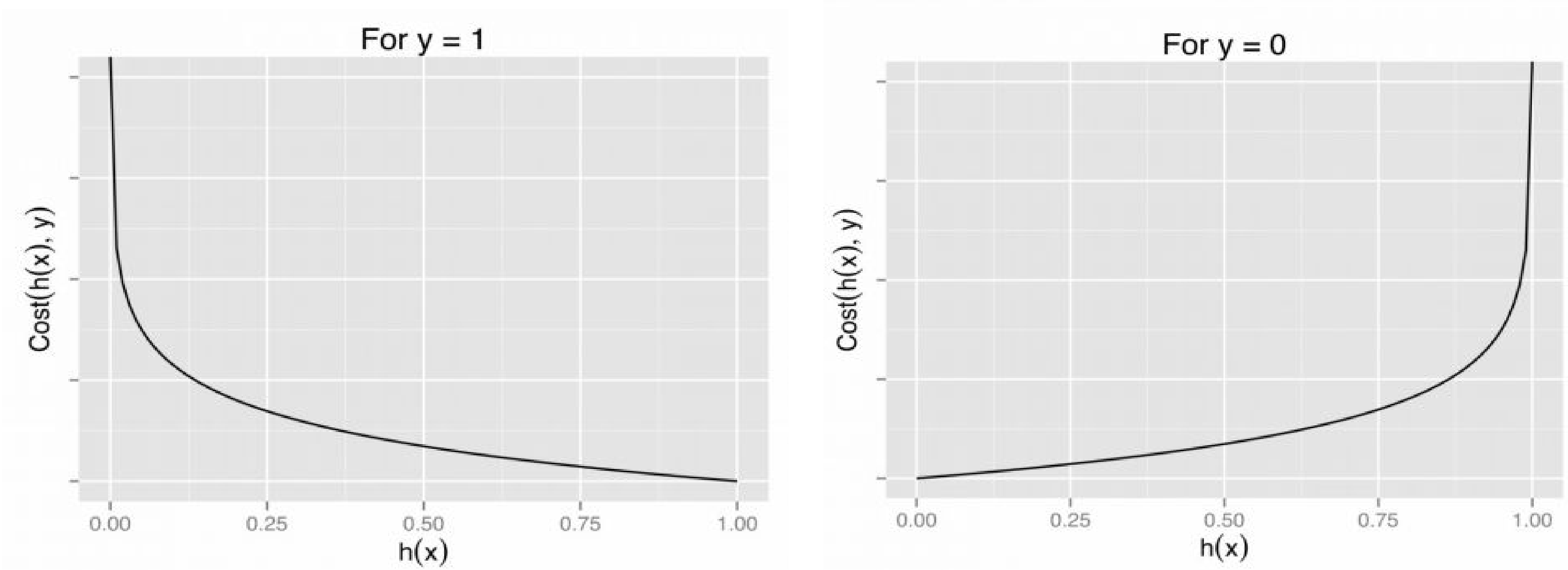


$$P(y=1 \mid x;\theta) \;=\; h_\theta(x)$$
$$P(y=0 \mid x;\theta) \;=\; 1-h_\theta(x)$$

Regresión logística

- Función costo

$$J(\theta) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^{(i)}\ log(h_\theta(x^{(i)})) + (1-y^{(i)})log(1-h_\theta(x^{(i)}))\right)$$



Gradient descent para regresión logística

- Iterativo
- Parte de vector θ inicial y hace actualizaciones

$$\sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \boxed{J(\theta)}$$

- Repetir hasta lograr convergencia

$$\theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

Cantidad de clases

- 2 clases
  - Enfoque ya visto
- Más de 2 clases
  - Enfoque 1 vs. el resto

Ejemplo usando fmin_bfgs de scipy

Ejemplo usando fmin_bfgs de scipy

```
from numpy import loadtxt, where, zeros, e, array, log, ones, append,
linspace
from pylab import scatter, show, legend, xlabel, ylabel, contour, title
from scipy.optimize import fmin_bfgs


def sigmoid(X):
    '''Compute the sigmoid function '''
    #d = zeros(shape=(X.shape))
    den = 1.0 + e ** (-1.0 * X)
    d = 1.0 / den
    return d
```

Ejemplo de http://aimotion.blogspot.mx/2011/11/machine-learning-with-python-logistic.html

Ejemplo usando fmin_bfgs de scipy

```
def cost_function_reg(theta, X, y, l):
    '''Compute the cost and partial derivatives as grads'''
    h = sigmoid(X.dot(theta))
    thetaR = theta[1:, 0]
    J = (1.0 / m) * ((-y.T.dot(log(h))) - ((1 - y.T).dot(log(1.0 - h)))) \
            + (l / (2.0 * m)) * (thetaR.T.dot(thetaR))

    delta = h - y
    sumdelta = delta.T.dot(X[:, 1])
    grad1 = (1.0 / m) * sumdelta

    XR = X[:, 1:X.shape[1]]
    sumdelta = delta.T.dot(XR)

    grad = (1.0 / m) * (sumdelta + l * thetaR)
    out = zeros(shape=(grad.shape[0], grad.shape[1] + 1))

    out[:, 0] = grad1
    out[:, 1:] = grad

    return J.flatten(), out.T.flatten()
```

Ejemplo de http://aimotion.blogspot.mx/2011/11/machine-learning-with-python-logistic.html

Ejemplo usando fmin_bfgs de scipy

```
m, n = X.shape

y.shape = (m, 1)

it = map_feature(X[:, 0], X[:, 1])

#Initialize theta parameters
initial_theta = zeros(shape=(it.shape[1], 1))

#Set regularization parameter lambda to 1
l = 1

# Compute and display initial cost and gradient for regularized logistic
# regression
cost, grad = cost_function_reg(initial_theta, it, y, l)

def decorated_cost(theta):
    return cost_function_reg(theta, it, y, l)

fmin_bfgs(decorated_cost, initial_theta, maxiter=400)
```

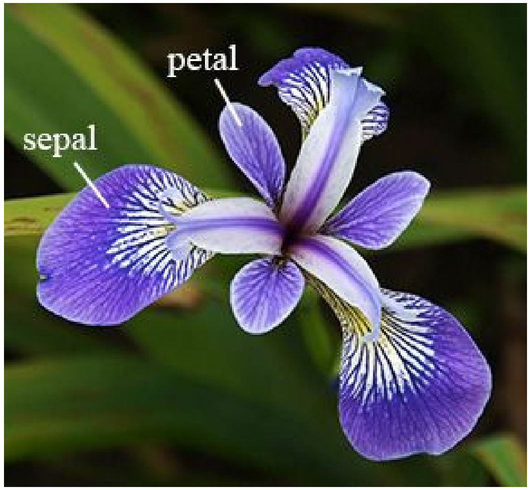
Ejemplo de http://aimotion.blogspot.mx/2011/11/machine-learning-with-python-logistic.html

Ejemplo usando sklearn.linear_model.LogisticRegression

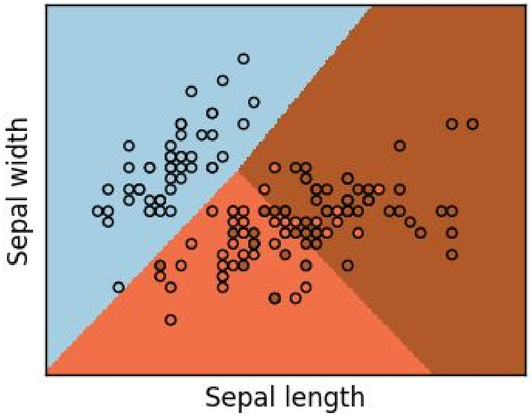
Ejemplo usando sklearn.linear_model.LogisticRegression

Iris versicolor

Iris virginica



Iris setosa



Ejemplo usando sklearn.linear_model.LogisticRegression

```
# Code source: Gaël Varoquaux
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model, datasets

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
Y = iris.target

h = .02  # step size in the mesh

logreg = linear_model.LogisticRegression(C=1e5)

# we create an instance of Neighbours Classifier and fit the data.
logreg.fit(X, Y)
```

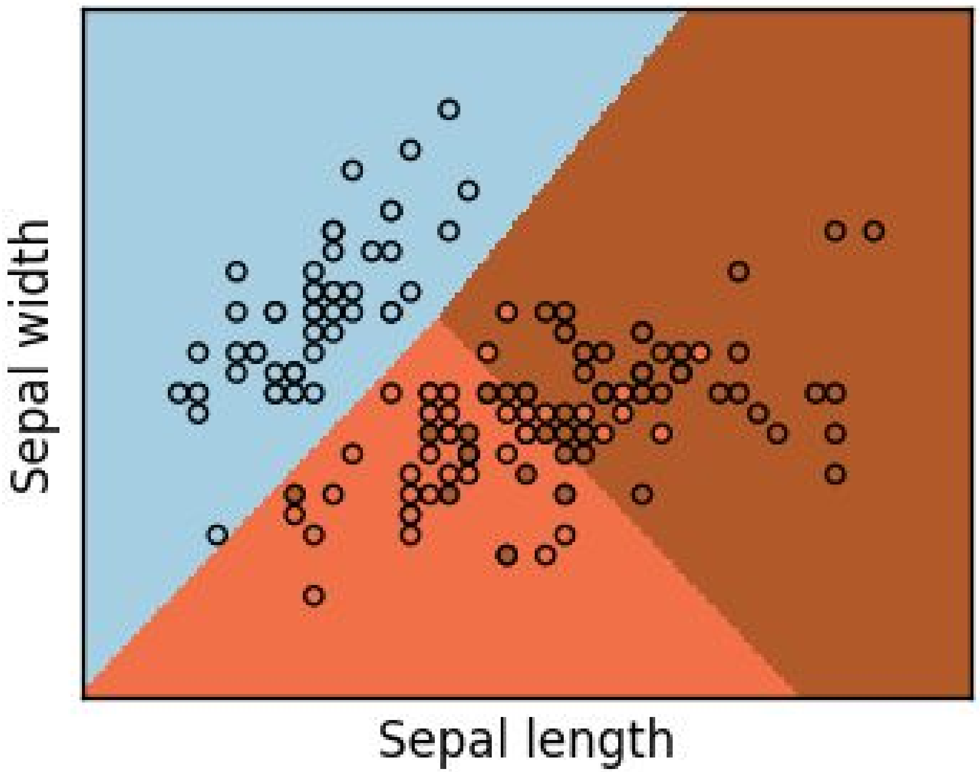
Ejemplo de http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html#sphx-glr-auto-examples-linear-model-plot-iris-logistic-py

Ejemplo usando sklearn.linear_model.LogisticRegression

```
# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = logreg.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=Y, edgecolors='k', cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())
plt.show()
```

Ejemplo de http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html#sphx-glr-auto-examples-linear-model-plot-iris-logistic-py

Ejemplo usando sklearn.linear_model.LogisticRegression



Ejemplo de http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html#sphx-glr-auto-examples-linear-model-plot-iris-logistic-py

2.
K-means

Aprendizaje automático

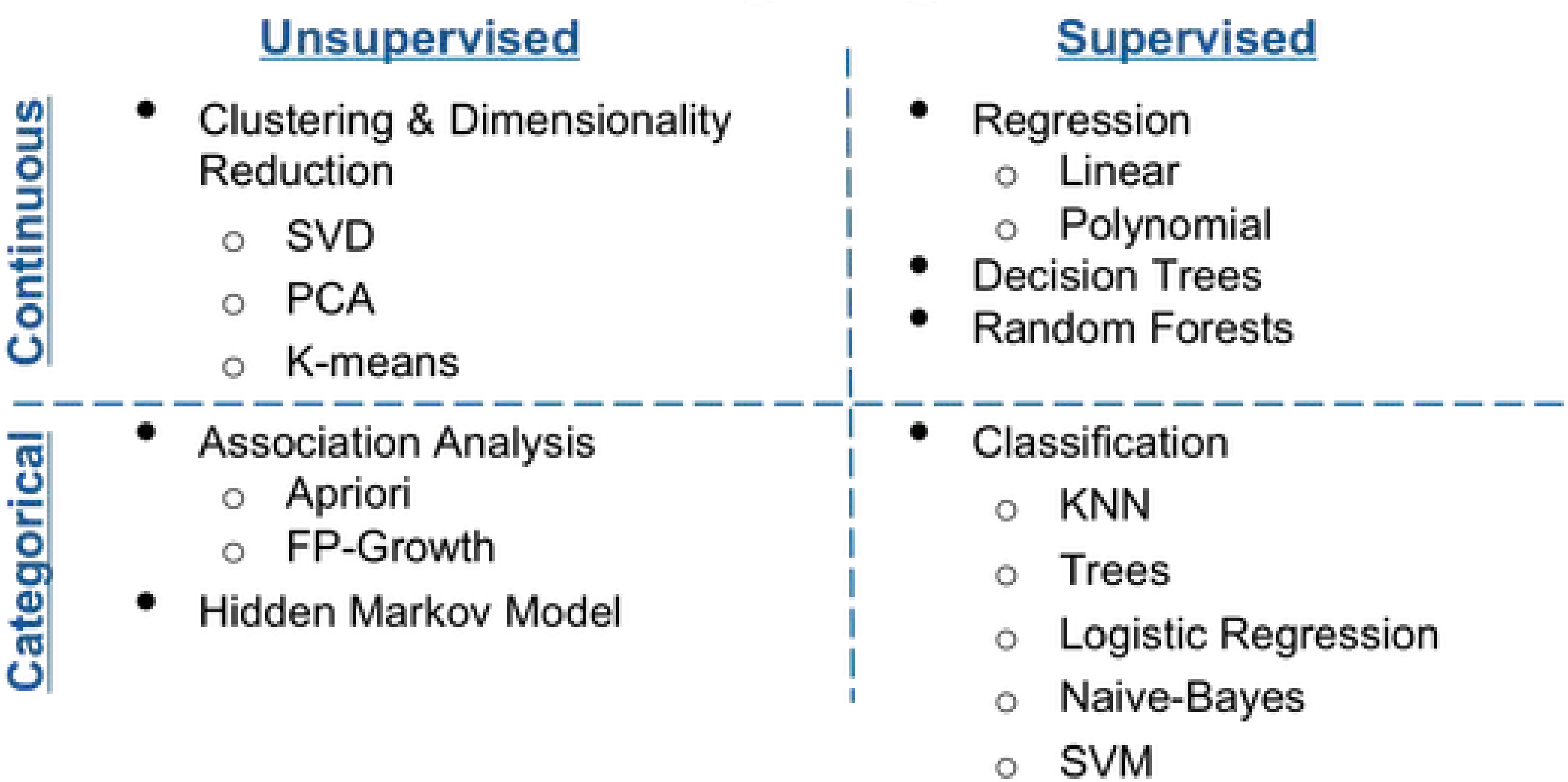


Diagrama de http://mlwithdata.blogspot.mx/2015/04/machine-learning-supervised-vs.html

K-means

- Tenemos

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

- Queremos: hacer una ***partición***

Algoritmos de clustering basados en...

Centroides

Conectividad

Densidad

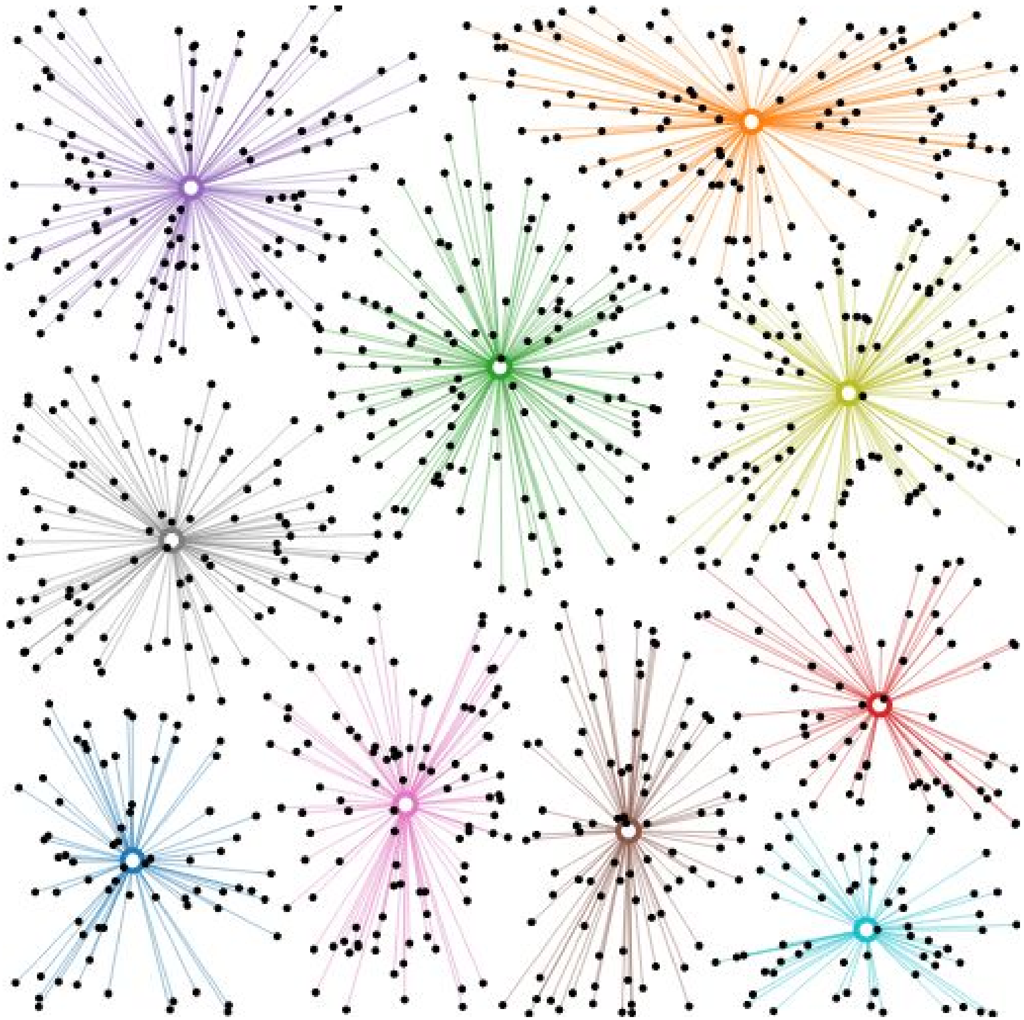
Probabilidad

Reducción de dimensión

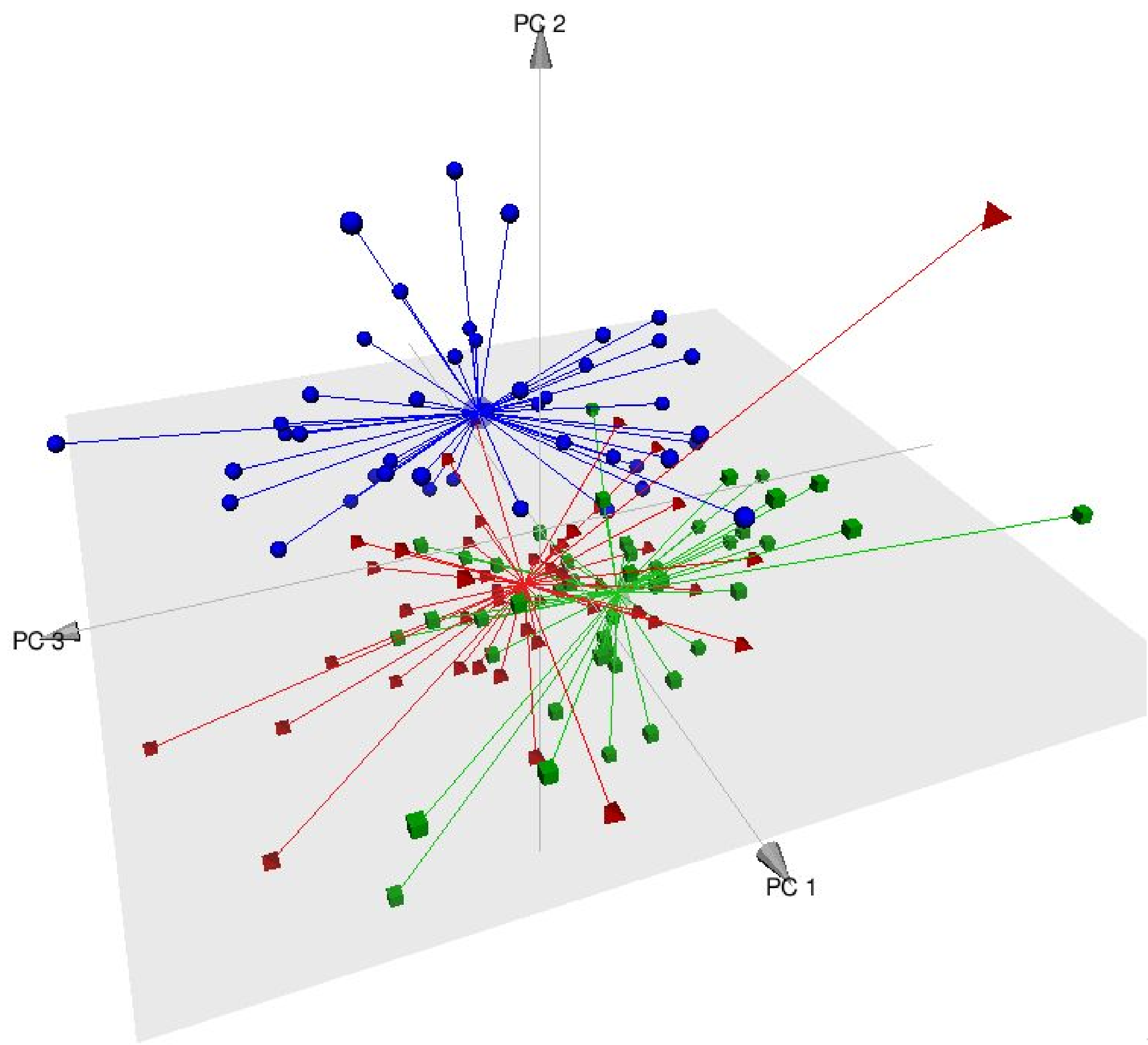
NN/Deep learning

...

Concepto de centroide



Concepto de centroide



Algoritmo k-means

- k centroides

$$\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$$

- m asignaciones (una por elemento)

$$c^{(i)}$$

Algoritmo k-means

1. Inicializar los k centroides
2. Repetir hasta lograr convergencia:
   - Asignar cada elemento al centroide más cercano

     For every i, set

     $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2$$

   - Recalcular los centroides

     For each j, set

     $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}$$

Algoritmo k-means, ilustrado

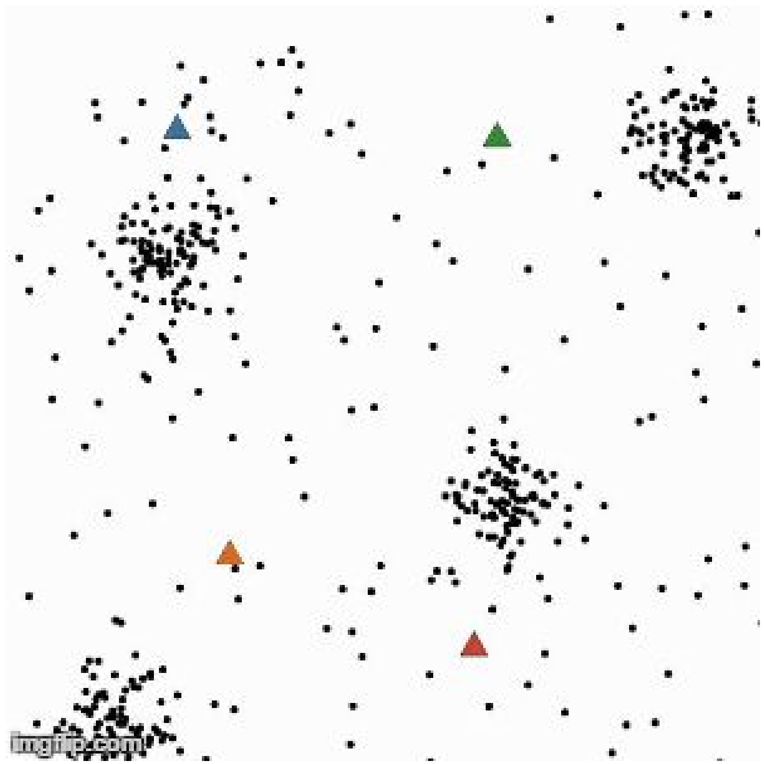
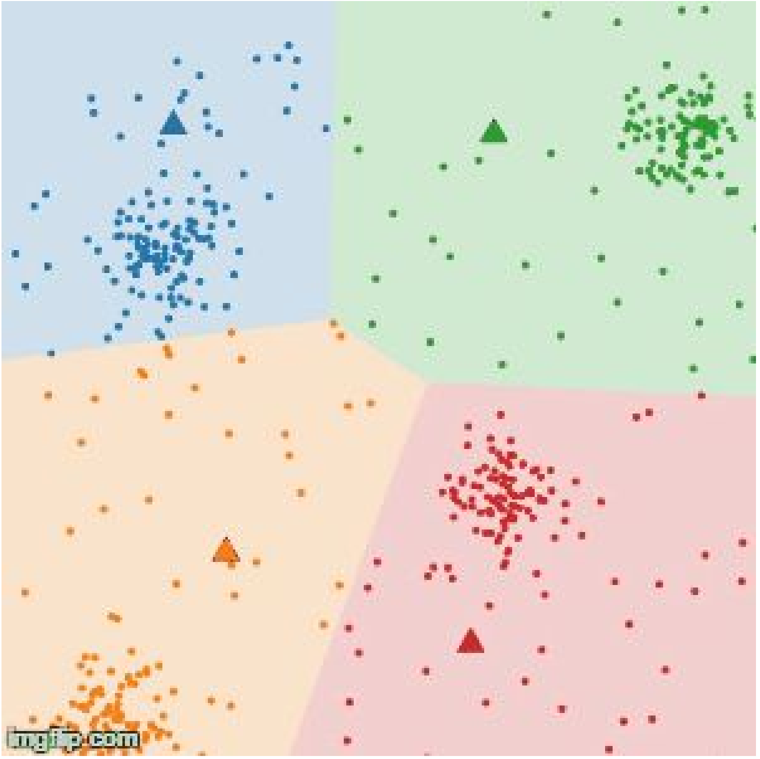
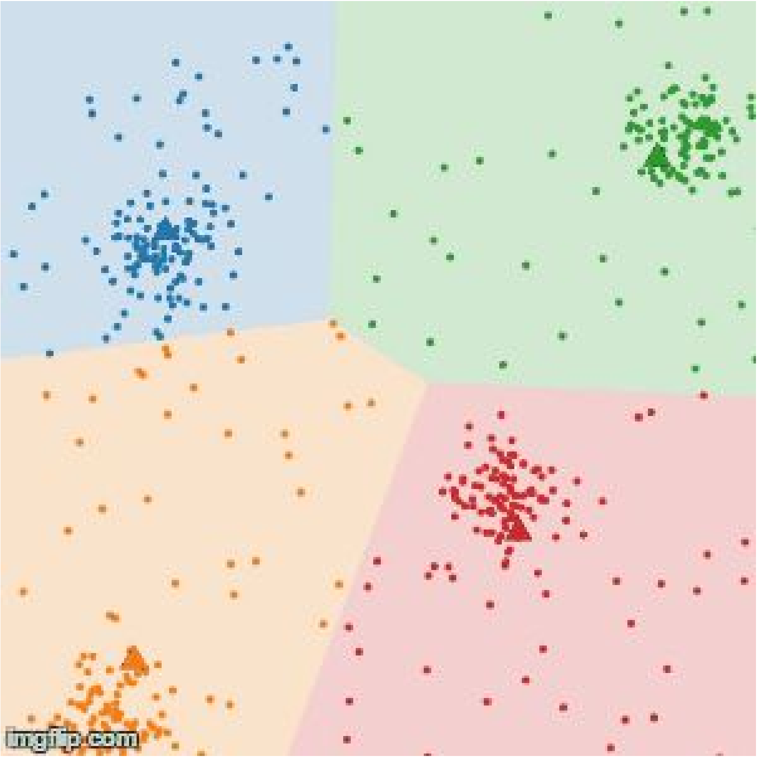


Diagrama de WikiMedia

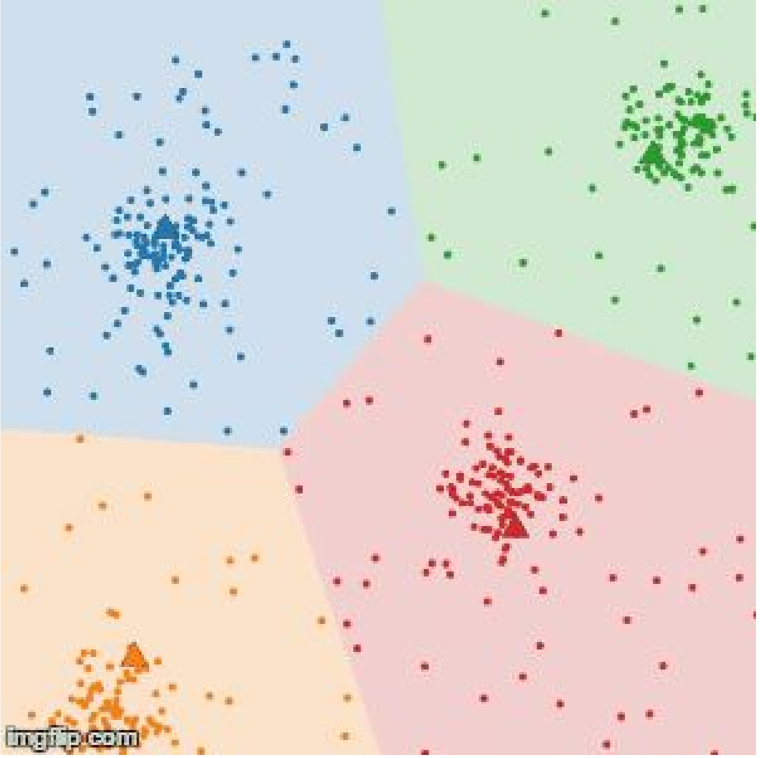
Algoritmo k-means, ilustrado



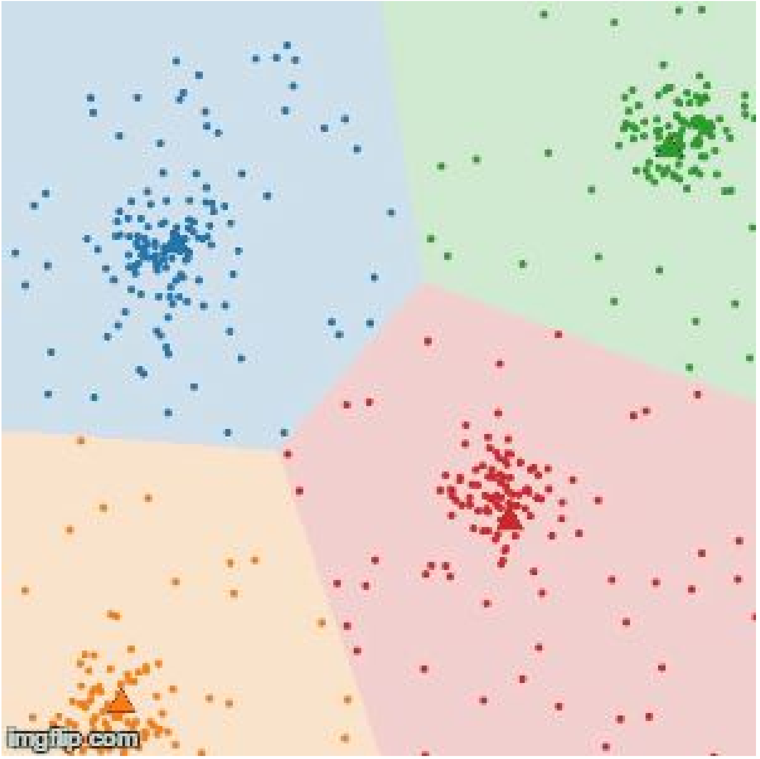
Algoritmo k-means, ilustrado



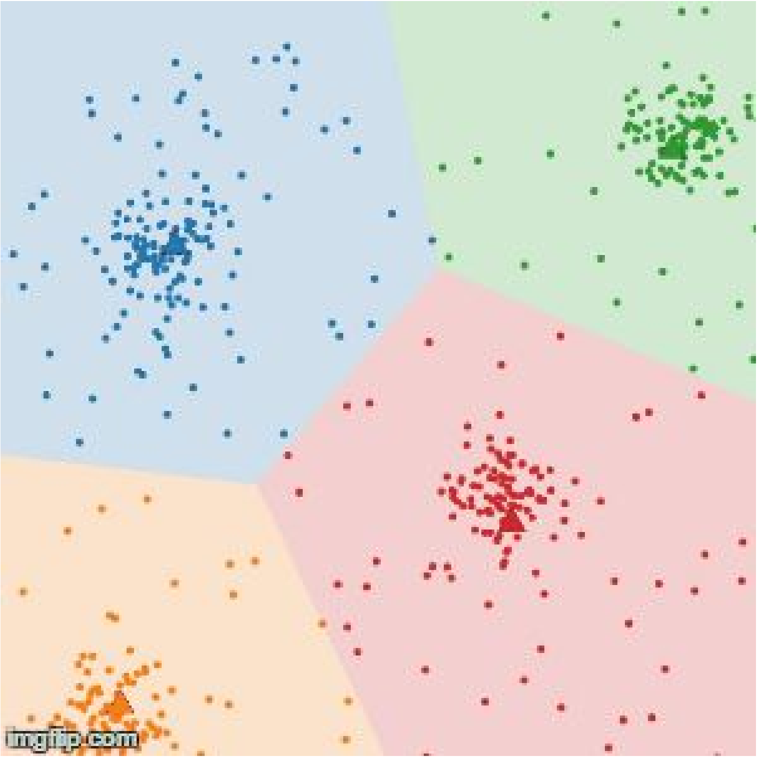
Algoritmo k-means, ilustrado



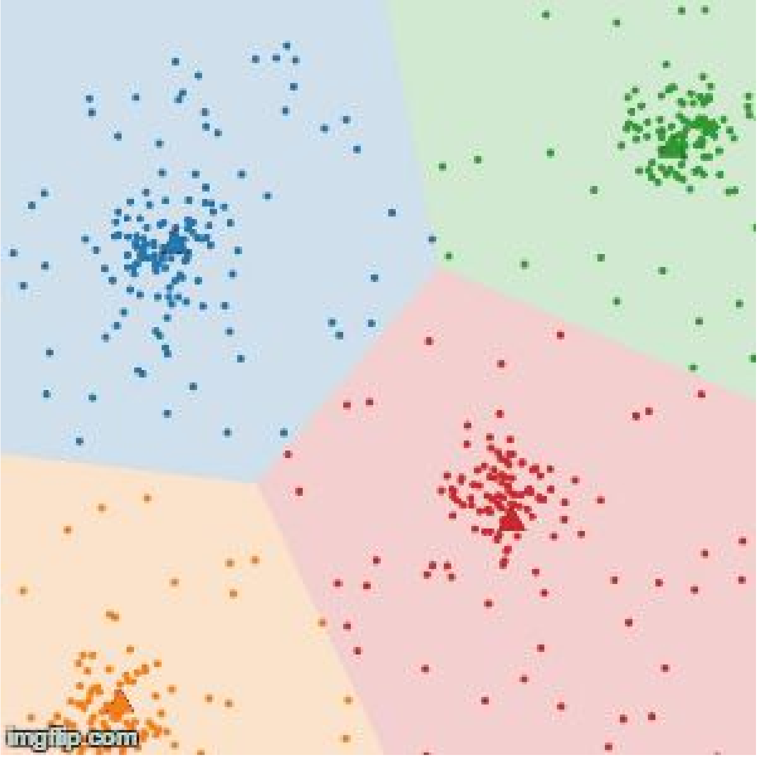
Algoritmo k-means, ilustrado



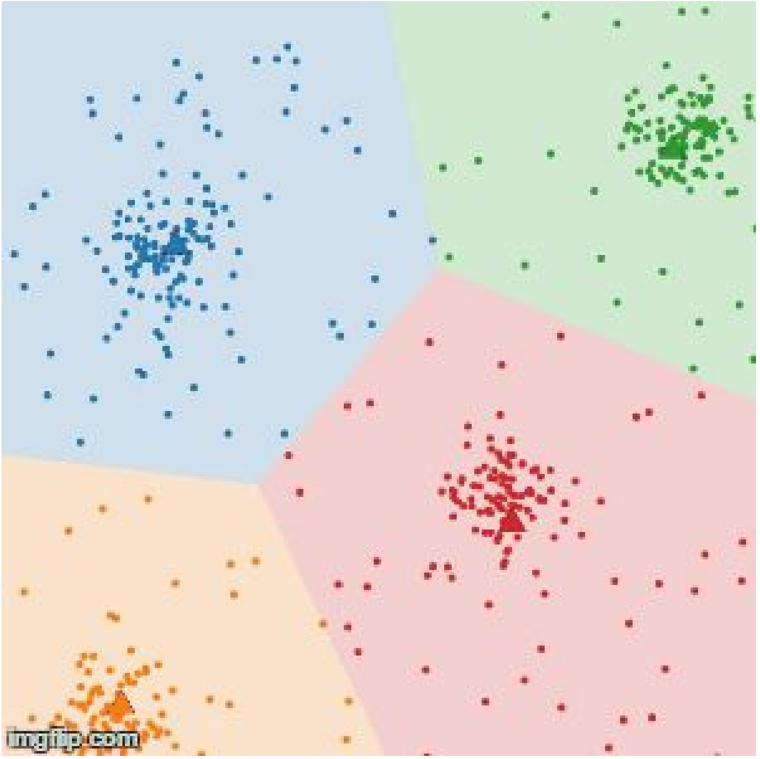
Algoritmo k-means, ilustrado



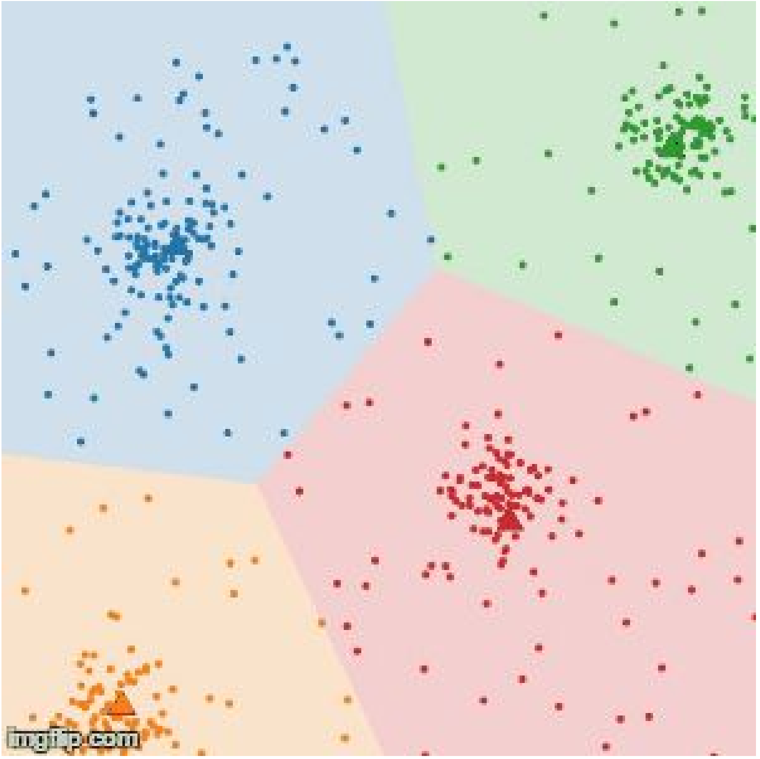
Algoritmo k-means, ilustrado



Algoritmo k-means, ilustrado



Algoritmo k-means, ilustrado



Función distorsión

$$J(c,\mu) = \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

- Se minimiza con **coordinate descent**. Recordar:

Repeat until convergence: {

For every i, set

$$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2$$

For each j, set

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}$$

}

- J es no convexa

Ejemplo k-means con Python

Ejemplo k-means con Python

```
import numpy as np

def cluster_points(X, mu):
    clusters  = {}
    for x in X:
        bestmukey = min([(i[0], np.linalg.norm(x-mu[i[0]])) \
                    for i in enumerate(mu)], key=lambda t:t[1])[0]
        try:
            clusters[bestmukey].append(x)
        except KeyError:
            clusters[bestmukey] = [x]
    return clusters

def reevaluate_centers(mu, clusters):
    newmu = []
    keys = sorted(clusters.keys())
    for k in keys:
        newmu.append(np.mean(clusters[k], axis = 0))
    return newmu
```

Ejemplo de https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/

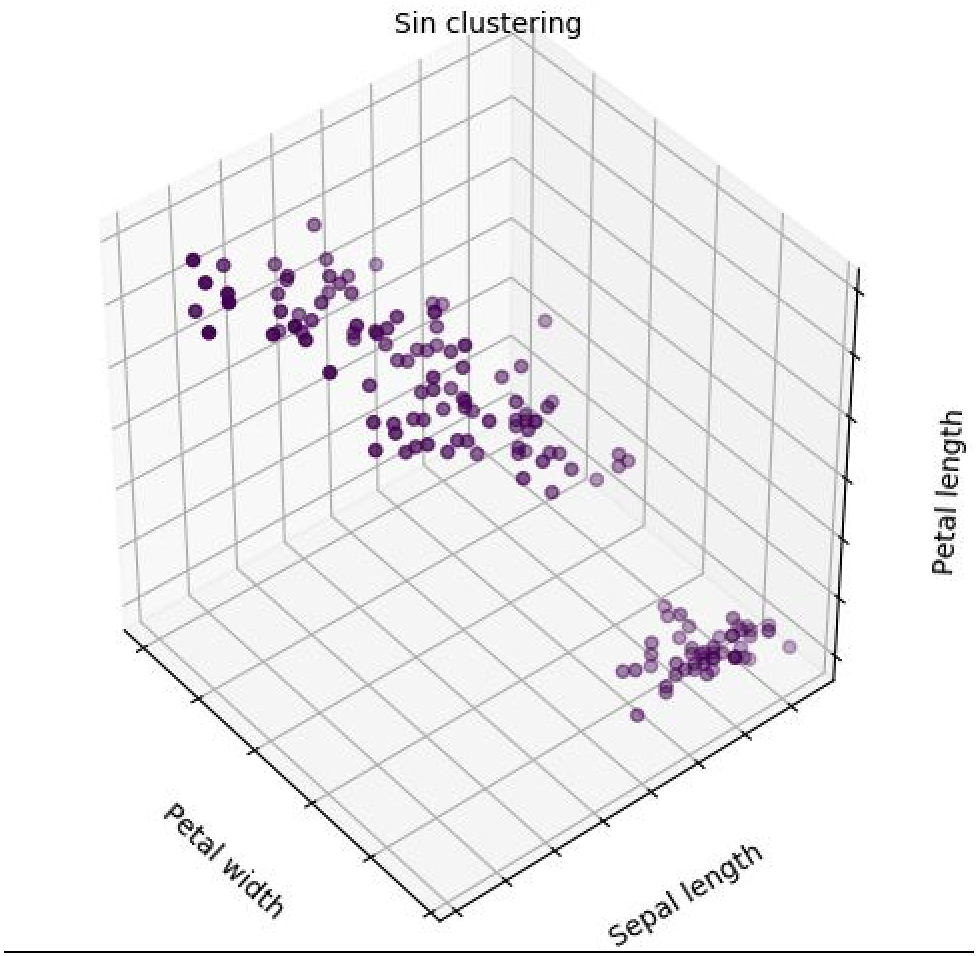
Ejemplo k-means con Python

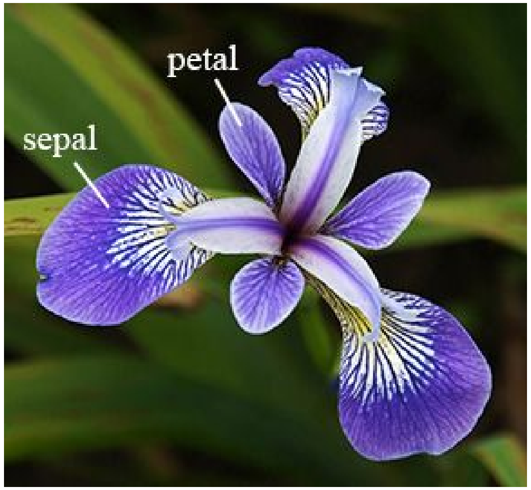
```
def has_converged(mu, oldmu):
    return (set([tuple(a) for a in mu]) == set([tuple(a) for a in oldmu])

def find_centers(X, K):
    # Initialize to K random centers
    oldmu = random.sample(X, K)
    mu = random.sample(X, K)
    while not has_converged(mu, oldmu):
        oldmu = mu
        # Assign all points in X to clusters
        clusters = cluster_points(X, mu)
        # Reevaluate centers
        mu = reevaluate_centers(oldmu, clusters)
    return(mu, clusters)
```

Ejemplo de https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/

Ejemplo usando sklearn.cluster.KMeans

Ejemplo usando sklearn.cluster.KMeans





Ejemplo basado en http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Ejemplo usando sklearn.cluster.KMeans

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


from sklearn.cluster import KMeans
from sklearn import datasets

np.random.seed(5)

centers = [[1, 1], [-1, -1], [1, -1]]
iris = datasets.load_iris()
X = iris.data
y = iris.target

estimators = {'k_means_iris_2': KMeans(n_clusters=2),
              'k_means_iris_3': KMeans(n_clusters=3),
              'k_means_iris_5': KMeans(n_clusters=5)}
```

Ejemplo basado en http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Ejemplo usando sklearn.cluster.KMeans

```
fignum = 1
for name, est in estimators.items():
    fig = plt.figure(fignum, figsize=(4, 3))
    plt.clf()
    #add title
    fig.suptitle(name, fontsize=10)
    ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

    plt.cla()
    est.fit(X)
    labels = est.labels_

    ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=labels.astype(np.float))

    ax.set_xlabel('Petal width')
    ax.set_ylabel('Sepal length')
    ax.set_zlabel('Petal length')
    fignum = fignum + 1
```

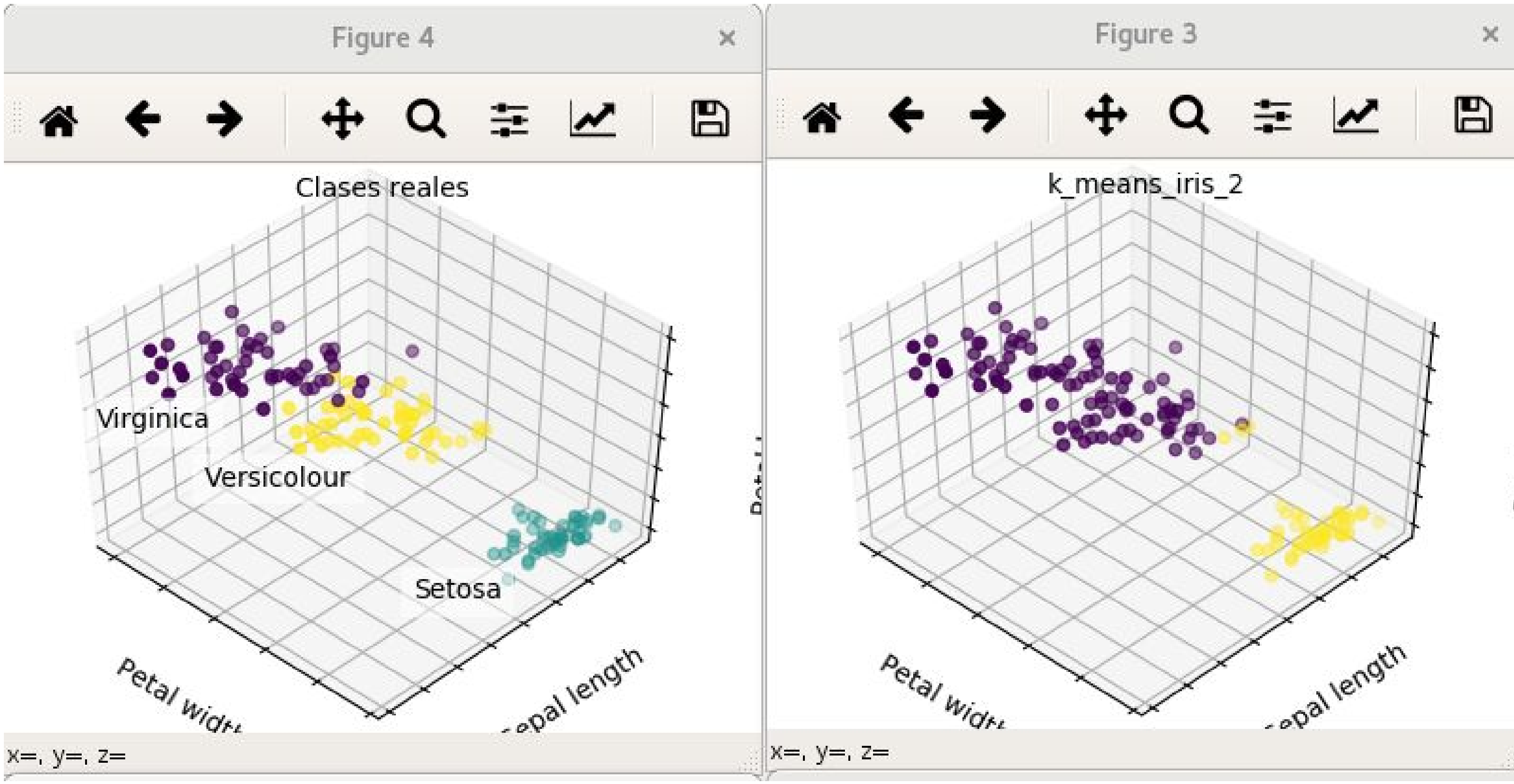
Ejemplo basado en http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Ejemplo usando sklearn.cluster.KMeans

```
# Plot the ground truth
fig = plt.figure(fignum, figsize=(4, 3))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
fig.suptitle('Clases reales',fontsize=10)
plt.cla()

for name, label in [('Setosa', 0),
                    ('Versicolour', 1),
                    ('Virginica', 2)]:
    ax.text3D(X[y == label, 3].mean(),
              X[y == label, 0].mean() + 1.5,
              X[y == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
# Reorder the labels to have colors matching the cluster results
y = np.choose(y, [1, 2, 0]).astype(np.float)
ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=y)
```

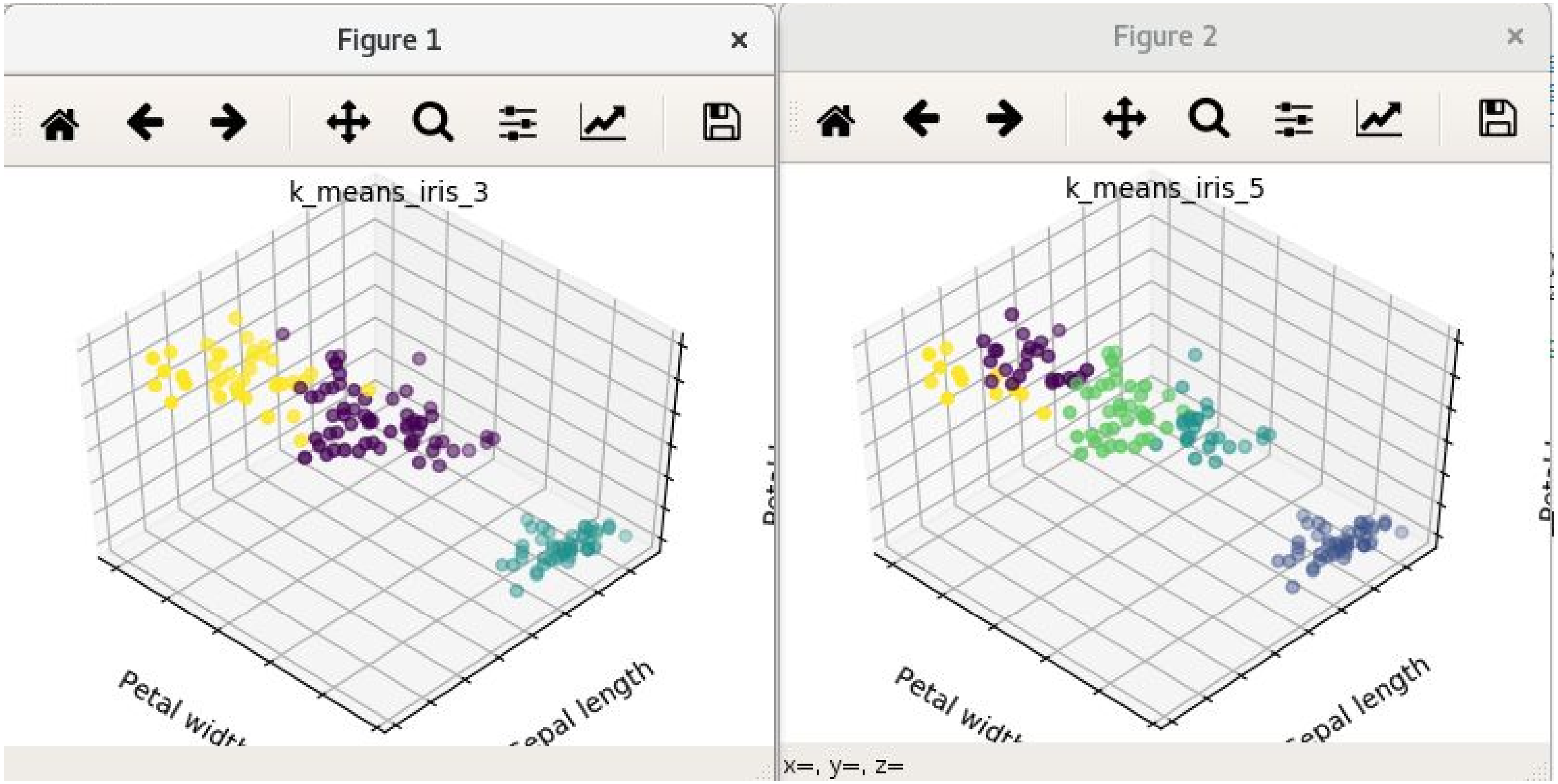
Ejemplo basado en http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Ejemplo usando sklearn.cluster.KMeans



Ejemplo basado en http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Ejemplo usando sklearn.cluster.KMeans



Ejemplo basado en http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Fuentes consultadas

- CS229 Supervised Learning (http://cs229.stanford.edu/notes/cs229-notes1.pdf)
- Slides de Universität Leipzig, Classification using Logistic Regression
- Documentación de scikit LogisticRegression http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- Logistic Regression 3-class classifier (http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html#sphx-glr-auto-examples-linear-model-plot-iris-logistic-py)
- Machine Learning with Python - Logistic Regression (http://aimotion.blogspot.mx/2011/11/machine-learning-with-python-logistic.html)
- Logistic Regression (https://github.com/perborgen/LogisticRegression/blob/master/logistic.py)
- CS229 Lecture notes, The k-means clustering algorithm (http://cs229.stanford.edu/notes/cs229-notes7a.pdf)
- Clustering With K-Means in Python (https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/)
- K-means clustering (http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py)

1. Regresión logística
2. K-means

Meetup Monterrey Data Science & Engineering

CoWo, Monterrey, Nuevo León, México

22 de junio de 2017

Rafael Rodríguez Morales (rafarodrz@gmail.com)