Lab3
EM & GCN

Chen Houshuang

chenhoushuang@sjtu.edu.cn

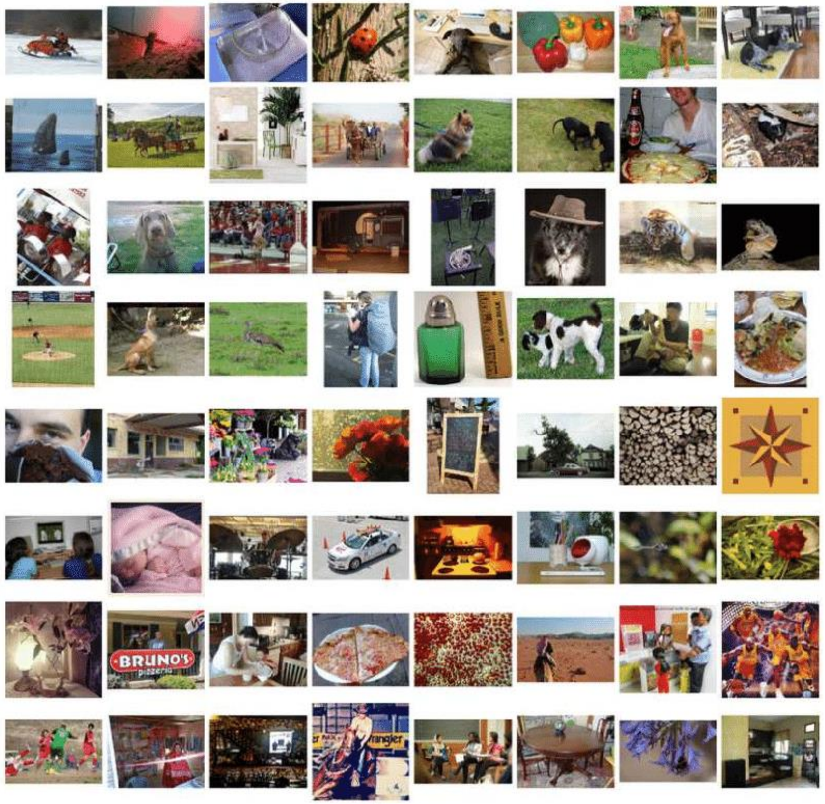
contents

- GCN
- EM

Graph Convolutional Networks

Traditional Deep Learning

- CNN



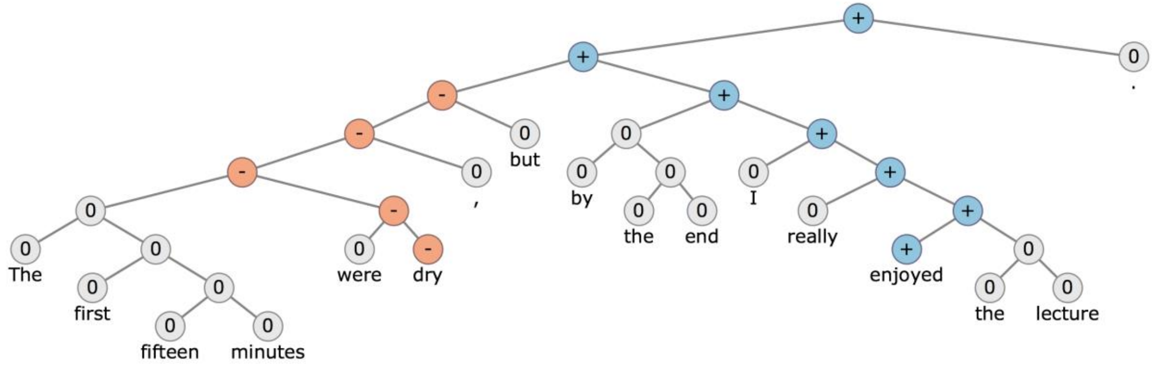
ImageNet

Grid Games

Traditional Deep Learning

- RNN

speech



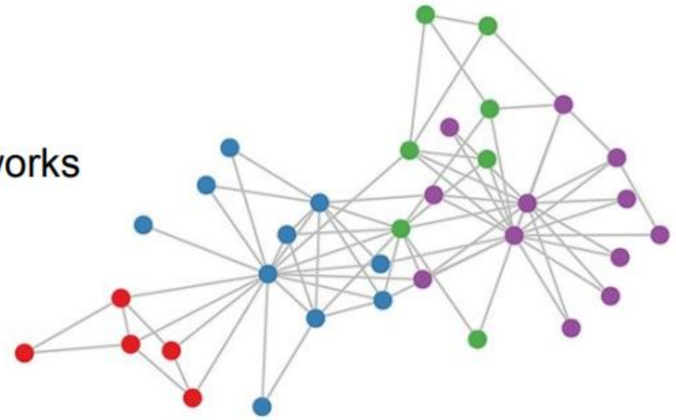
NLP

Graph structured Data

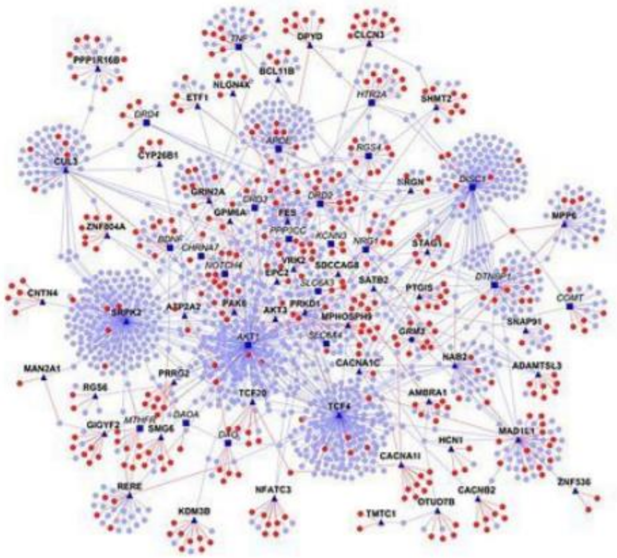
A lot of real-world data does not live on “grids”

Knowledge Graphs

Social Networks
Citation Networks
Communication Networks
Multi-Agent Systems



Protein Interaction Networks

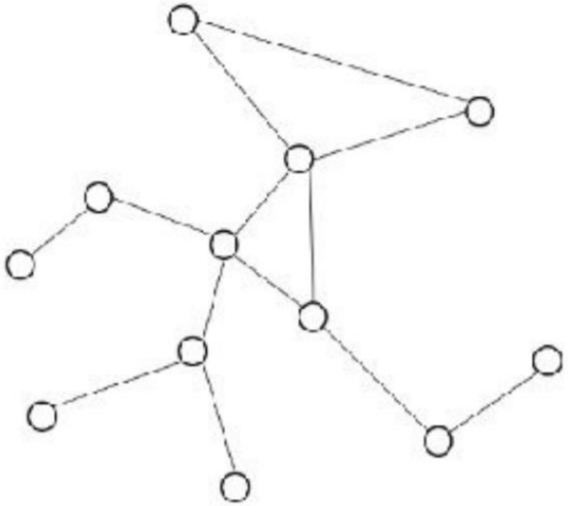


Inspiration from CNN

- Pros of CNNs
  - Local connections
  - Shared weights
  - Use of multiple layers

- Cons of CNNs
  - hard to define convolutional and pooling layers for non-Euclidean data

Image in Euclidean space



Graph in non-Euclidean space

GCN: Definitions

- Input:
  - $N \times D$ feature matrix
    - N: number of nodes
    - D: dimension of input features
  - Adjacent matrix A(with shape $N \times N$)
- Output
  - Node level output Z
    - An $N \times F$ feature matrix, where F is the feature dimension of output per node
- Hidden neural network layer

$$H^{(l+1)} = f\left(H^{(l)}, A\right)$$

$H^{(0)} = X$ and $H^{(L)} = Z$(L: number of layers) and f is a non-linear function

GCN: the intuition

- A simple example

$$f\left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

  $W^{(l)}$: weight matrix for the l-th neural network layer

  $\sigma(\cdot)$: non-linear activation function like ReLU

- Despite its simplicity, this model is already quite powerful

GCN: the intuition

- A simple example

$$f\left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

  $W^{(l)}$: weight matrix for the l-th neural network layer
  $\sigma(\cdot)$: non-linear activation function like ReLU

- Limitations:
  - Sum up all feature vectors of all neighbor nodes except the node itself
  - A is not normalized
- Tricks:
  - Enforce self-loop in graph: add identity matrix to A
  - Normalize A: all rows sum to one
    - i.e. $D^{-1}A$ (D: diagonal node degree matrix)
    - Use symmetrical normalization in practice: $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

GCN

- A simple example

$$f\left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

$W^{(l)}$: weight matrix for the l-th neural network layer

$\sigma(\cdot)$: non-linear activation function like ReLU

- Real used propagate rules

$$f\left(H^{(l)}, A\right) = \sigma(\widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where $\hat{A} = A + I$ and $\widehat{D}$ is the diagonal node degree matrix of $\hat{A}$

- Note:
  - In the forward propagation, $\widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}$ only needs to be calculated once.

Examples

- Construct a two layer GCN

$$Z = \mathrm{softmax}\big(\tilde{A}\ \mathrm{ReLU}\big(\tilde{A}XW^{(0)}\big)W^{(1)}\big)$$

where $\tilde{A} = \widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}$

- Add loss for nodes with labels

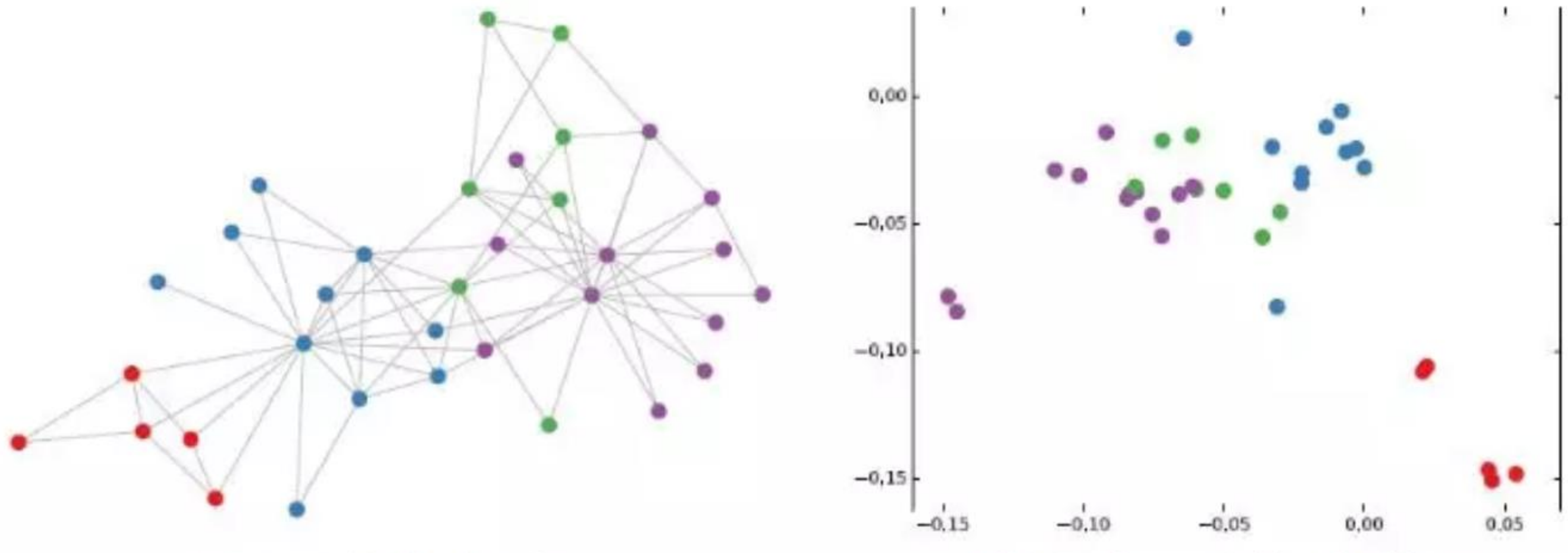
$$\mathcal{L} = -\sum_{l\in y_L}\sum_{f=1}^{F} Y_{lf}\ln Z_{lf}$$

F is the number of classes(output feature dimension)

y_L is the set of node indices that have labels

Examples

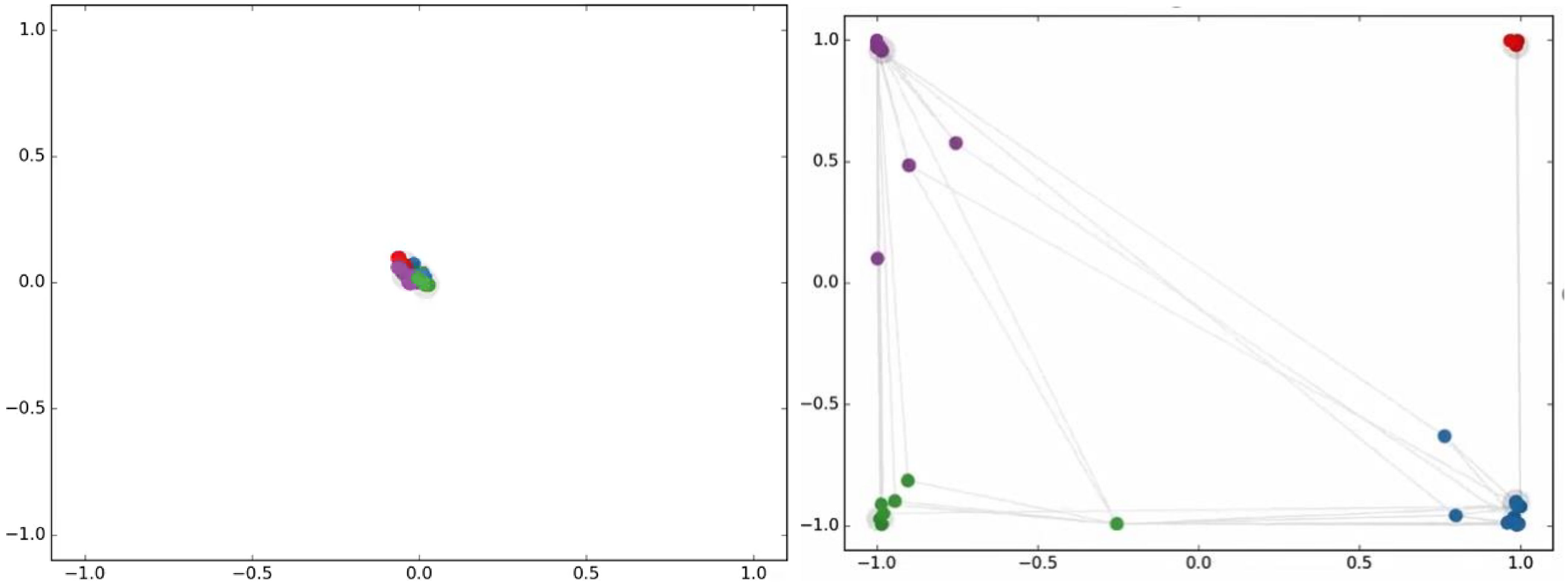
- 3-layer GCN with random initialized weights



(a) Karate club network (b) Random weight embedding

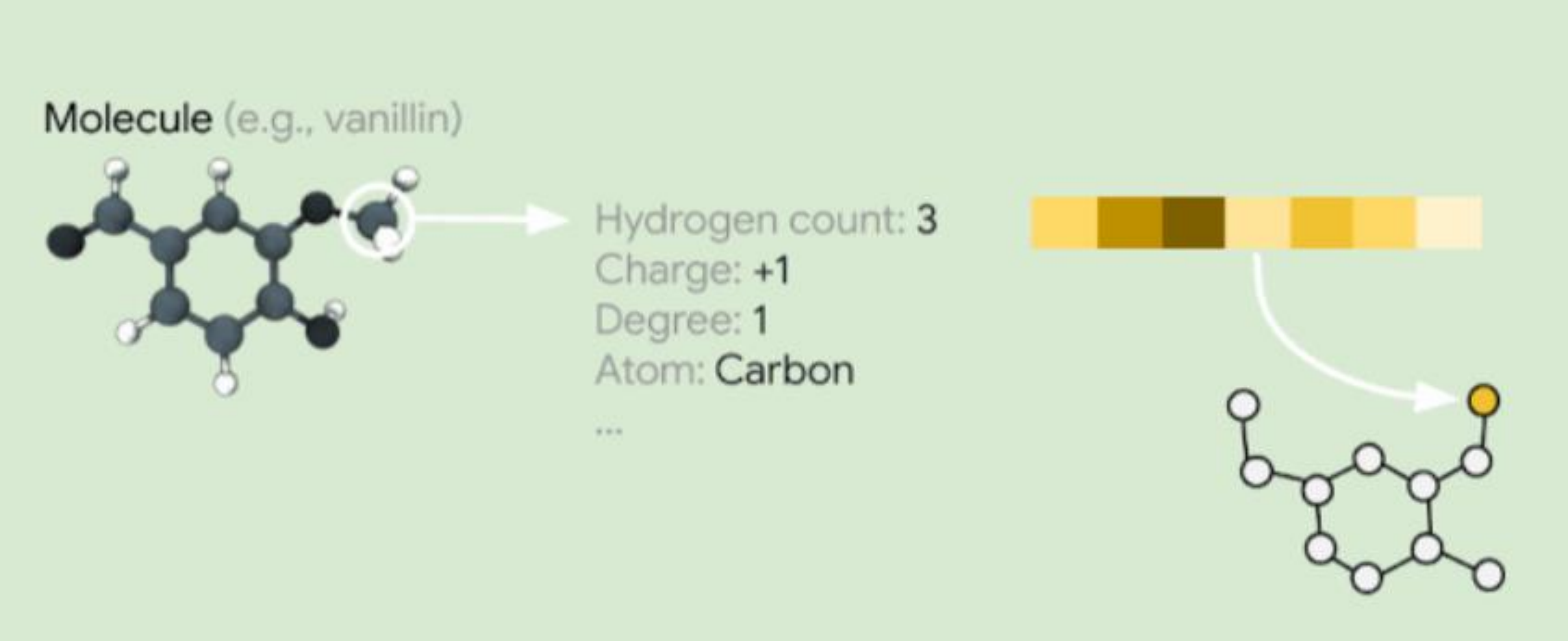
Examples

- Semi-supervised learning
  - label one node per class/community



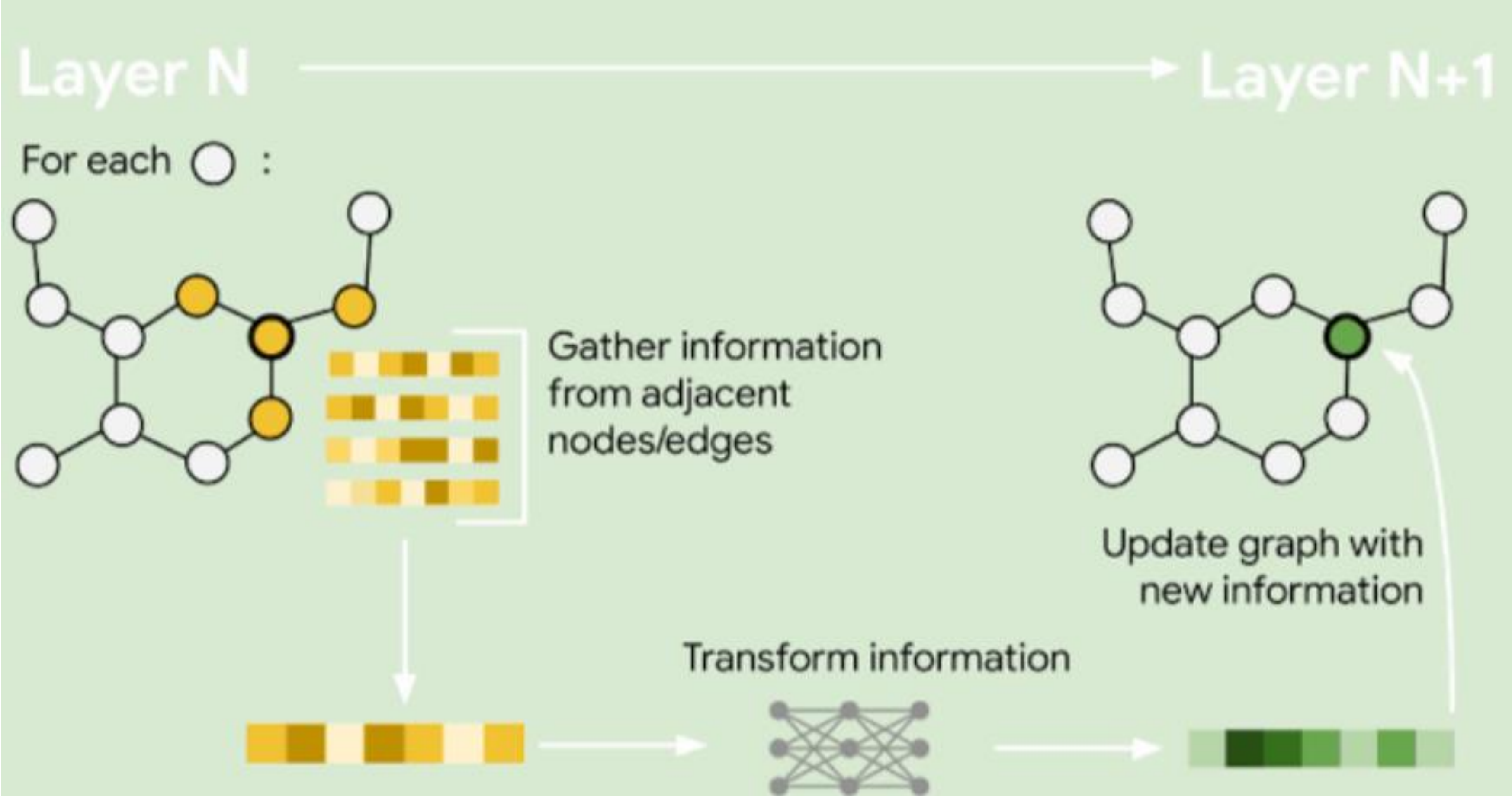
Molecule representation

- for each molecule, extract
  - Adjacent matrix with shape $N \times N$
  - feature vector for each node
    - different feature vectors for the same atom



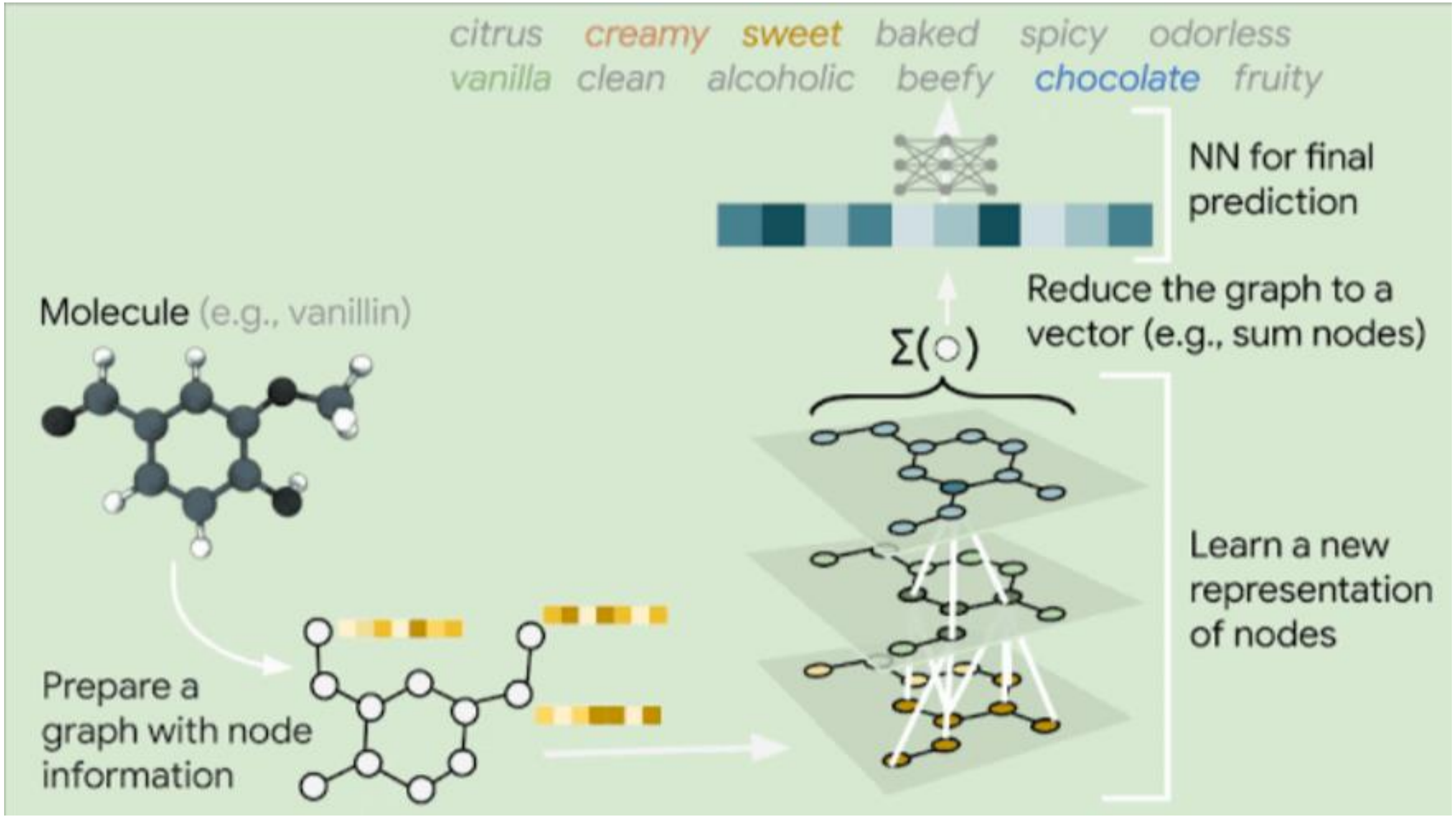
Model: one layer

- internal layer transition



Model training

- molecule⇒feature vectors⇒feature vectors⇒prediction



Expectation Maximization

EM algorithm for GMM

EM: the intuition

First experiment

- We choose 5 times one of the coins.
- We toss the chosen coin 10 times

H T T T H H T H T H

H H H H T H H H H H

H T H H H H H T H H

H T H T T T H H T T

T H H H T H H H T H

$$\theta_1 = \frac{number\ of\ heads\ using\ C1}{total\ number\ of\ flips\ using\ C1}$$

$$\theta_2 = \frac{number\ of\ heads\ using\ C2}{total\ number\ of\ flips\ using\ C2}$$

slide from Stefanos Zafeiriou: Statistical Machine Learning

EM: the intuition

H T T T H H T H T H

H H H H T H H H H H

H T H H H H H T H H

H T H T T T H H T T

T H H H T H H H T H

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

$$\theta_1 = \frac{24}{24+6} = 0.8$$

$$\theta_2 = \frac{9}{9+11} = 0.45$$

slide from Stefanos Zafeiriou: Statistical Machine Learning

EM: the intuition

Assume a more challenging problem

H T T T H H T H T H

H H H H T H H H H H

H T H H H H H T H H

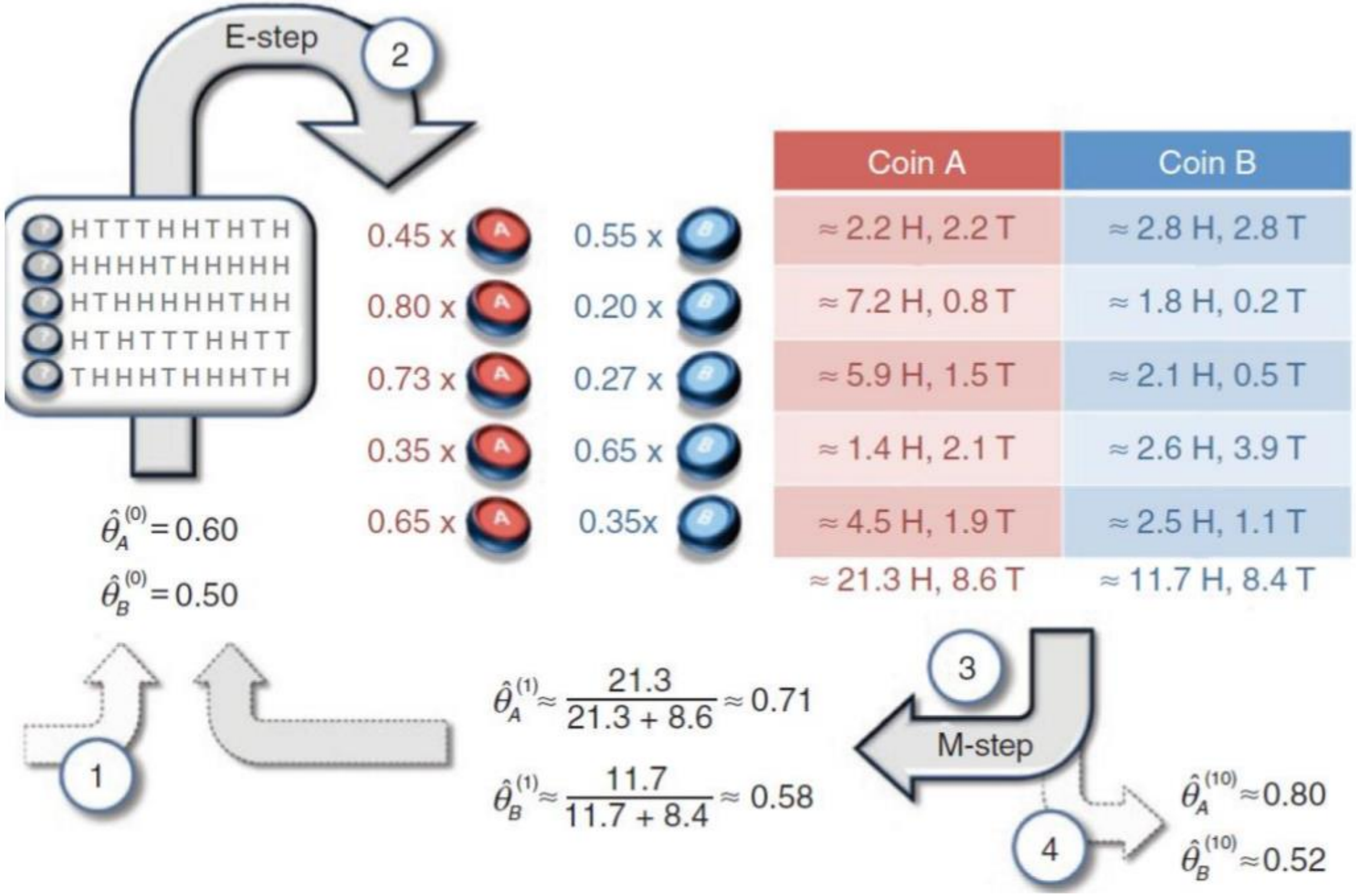
H T H T T T H H T T

T H H H T H H H T H

- We do not know the identities of the coins used for each set of tosses (we treat them as hidden variables).

slide from Stefanos Zafeiriou: Statistical Machine Learning

EM: the intuition



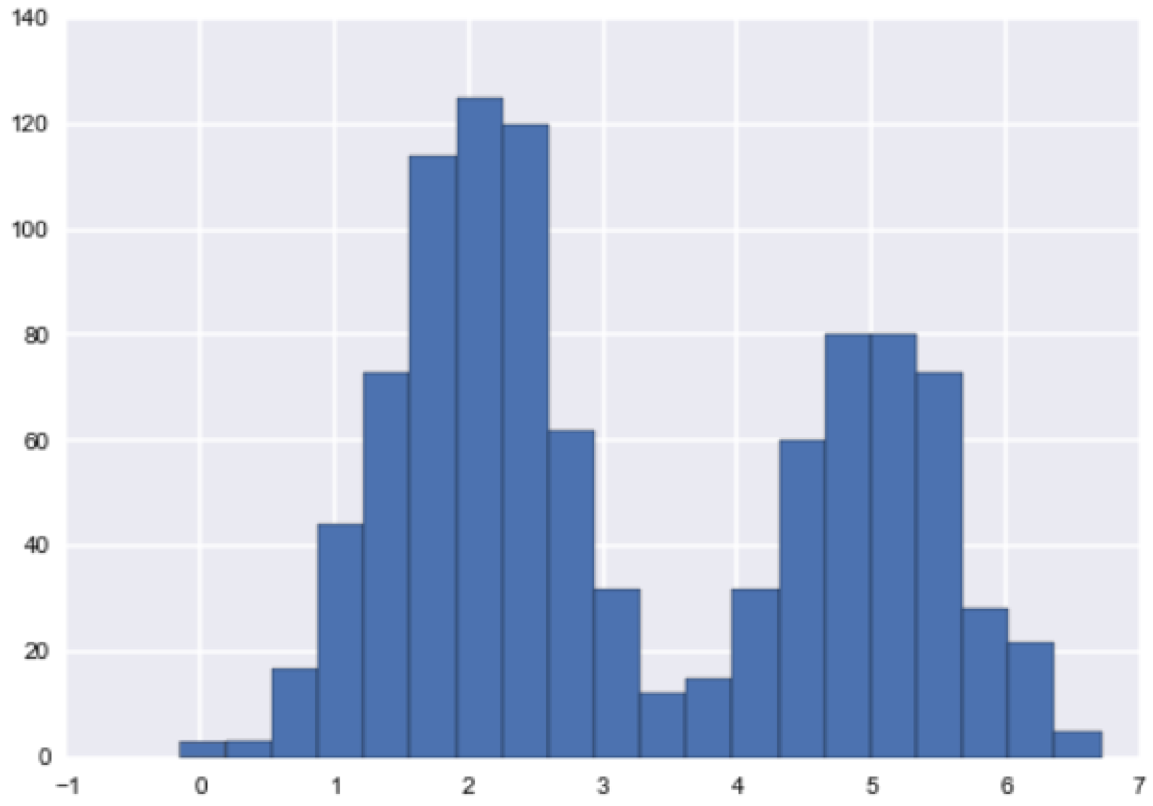
slide from Stefanos Zafeiriou: Statistical Machine Learning

Import library

```
%matplotlib inline
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import pandas as pd
pd.set_option('display.width', 500)
pd.set_option('display.max_columns', 100)
import seaborn as sns
from IPython.display import Image
```

Generate Dataset

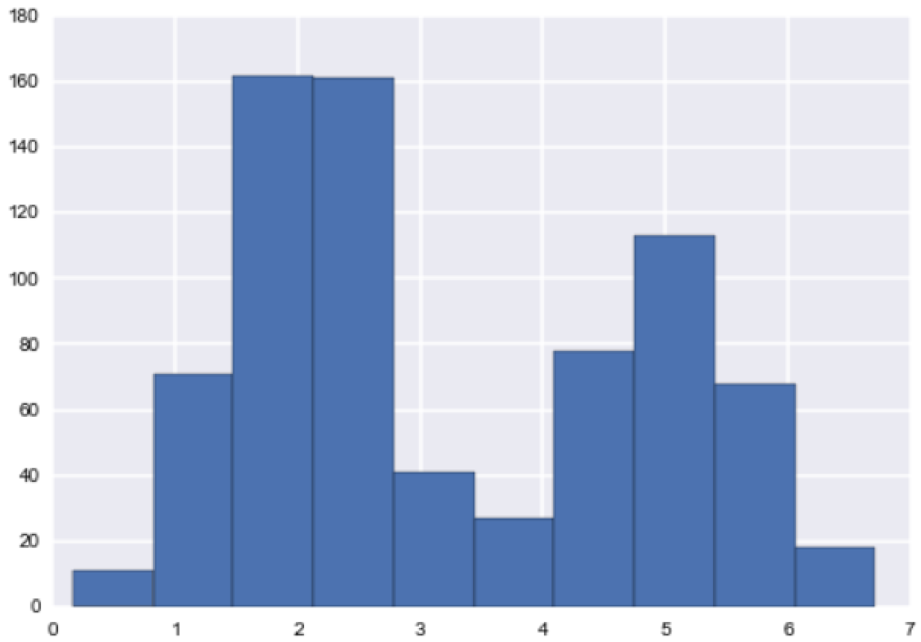
```
#In 1-D
# True parameter values
mu_true = [2, 5]
sigma_true = [0.6, 0.6]
lambda_true = .4
n = 1000

# Simulate from each distribution according to mixing proportion psi
z = np.random.binomial(1, lambda_true, n)
x = np.array([np.random.normal(mu_true[i], sigma_true[i]) for i in z])

plt.hist(x, bins=20);
```



Supervised learning

```python
#the z's are the classes in the supervised learning
#the 'feature' is the x position of the sample
from sklearn.cross_validation import train_test_split
ztrain, ztest, xtrain, xtest = train_test_split(z,x)

plt.hist(xtrain);
```

```python
lambda_train=np.mean(ztrain)
mu0_train = np.sum(xtrain[ztrain==0])/(np.sum(ztrain==0))
mu1_train = np.sum(xtrain[ztrain==1])/(np.sum(ztrain==1))
xmus=np.array([mu0_train if z==0 else mu1_train for z in ztrain])
xdiffs = xtrain - xmus
sigma_train=np.sqrt(np.dot(xdiffs, xdiffs)/xtrain.shape[0])
print lambda_train, mu0_train, mu1_train, sigma_train
```



Supervised learning

- $\theta = \{w_j, \mu_j, \Sigma_j : j\}$
- $p\left(z^{(i)} = j \middle| x^{(i)}; \theta\right) = \dfrac{p\left(z^{(i)} = j, x^{(i)} \middle| \theta\right)}{p(x^{(i)}|\theta)} = \dfrac{p(x^{(i)}|z^{(i)}=j,\theta)p\left(z^{(i)} = j \;\middle|\theta\right)}{p(x^{(i)}|\theta)}$

```
def loglikdiff(x):
    for0= - (x-mu0_train)*(x-mu0_train)/(2.0*sigma_train*sigma_train)
    for0 = for0 + np.log(1.-lambda_train)
    for1 = - (x-mu1_train)*(x-mu1_train)/(2.0*sigma_train*sigma_train)
    for1 = for1 + np.log(lambda_train)
    return 1*(for1 - for0 >= 0)
```

```
pred = np.array([loglikdiff(test_x) for test_x in xtest])
print "correct classification rate", np.mean(ztest == pred)
```

```
correct classification rate 1.0
```

EM

```python
from scipy.stats.distributions import norm

def Estep(x, mu, sigma, lam):
    a = lam * norm.pdf(x, mu[0], sigma[0])
    b = (1. - lam) * norm.pdf(x, mu[1], sigma[1])
    return b / (a + b)
```

```python
def Mstep(x, w):
    lam = np.mean(1-w)

    mu = [np.sum((1-w) * x)/np.sum(1-w), np.sum(w * x)/np.sum(w)]

    sigma = [np.sqrt(np.sum((1-w) * (x - mu[0])**2)/np.sum(1-w)),
             np.sqrt(np.sum(w * (x - mu[1])**2)/np.sum(w))]

    return mu, sigma, lam
```

Average over each cluster

The solution is

- $w_j = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{z^{(i)} = j\}$
- $\mu_j = \frac{\sum_{i=1}^{N} \mathbf{1}\{z^{(i)}=j\}x^{(i)}}{\sum_{i=1}^{N} \mathbf{1}\{z^{(i)}=j\}}$
- $\Sigma_j = \frac{\sum_{i=1}^{N} \mathbf{1}\{z^{(i)}=j\}(x^{(i)}-\mu_j)(x^{(i)}-\mu_j)^{\mathsf{T}}}{\sum_{i=1}^{N} \mathbf{1}\{z^{(i)}=j\}}$

EM

```python
print lambda_true, mu_true, sigma_true
# Initialize values
mu = np.random.normal(0, 10, size=2)
sigma = np.random.uniform(0, 5, size=2)
lam = np.random.random()
print "Initialization", mu, sigma, lam
# Stopping criterion
crit = 1e-15

# Convergence flag
converged = False

# Loop until converged
iterations=1
```

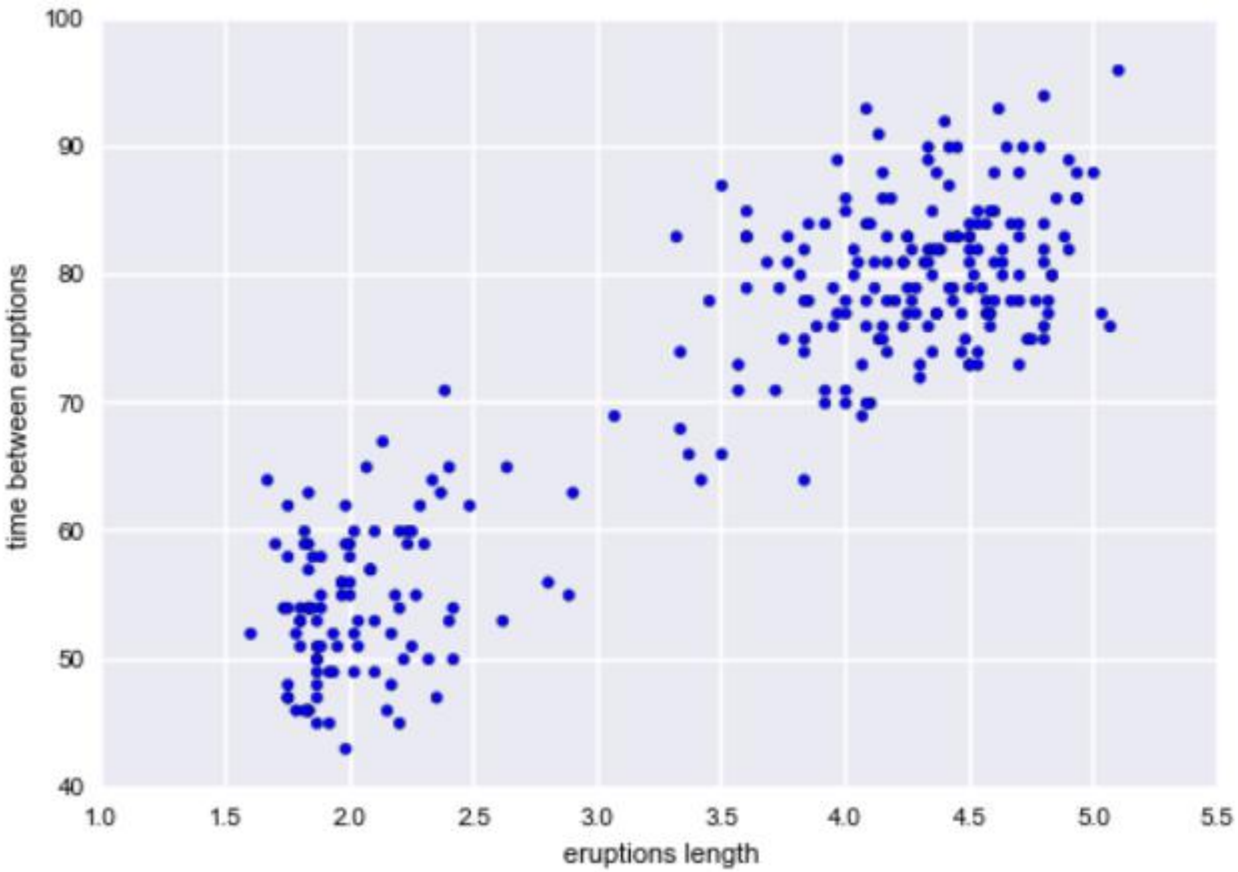
```python
while not converged:
    # E-step
    if np.isnan(mu[0]) or np.isnan(mu[1]) or np.isnan(sigma[0]) or np.isnan(sigma[1]):
        print "Singularity!"
        break

    w = Estep(x, mu, sigma, lam)

    # M-step
    mu_new, sigma_new, lam_new = Mstep(x, w)

    # Check convergence
    converged = ((np.abs(lam_new - lam) < crit)
                 & np.all(np.abs((np.array(mu_new) - np.array(mu)) < crit))
                 & np.all(np.abs((np.array(sigma_new) - np.array(sigma)) < crit)))
    mu, sigma, lam = mu_new, sigma_new, lam_new
    iterations +=1

print "Iterations", iterations
print('A: N({0:.4f}, {1:.4f})\nB: N({2:.4f}, {3:.4f})\nlam: {4:.4f}'.format(
                        mu_new[0], sigma_new[0], mu_new[1], sigma_new[1], lam_new))
```

```
0.4 [2, 5] [0.6, 0.6]
Initialization [ 1.67482614 -5.40579505] [ 2.18150241  2.34191404] 0.914875395098
Iterations 69
A: N(5.0294, 0.6329)
B: N(2.0138, 0.6236)
lam: 0.3985
```

Add Old Faithful 2D dataset

```
ofdata=pd.read_csv("./oldfaithful.csv")
ofdata.head()
```

```
plt.scatter(ofdata.eruptions, ofdata.waiting);
plt.xlabel('eruptions length')
plt.ylabel('time between eruptions')
```

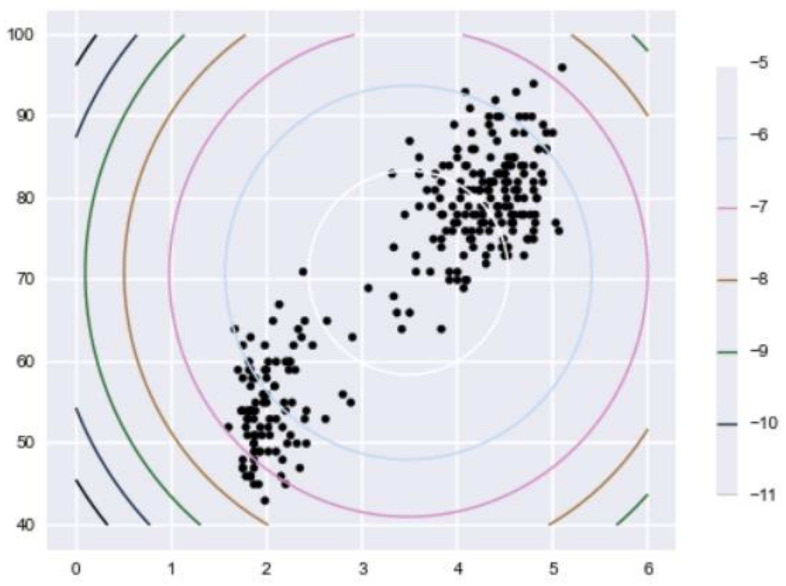


Create GMM with different cluster number

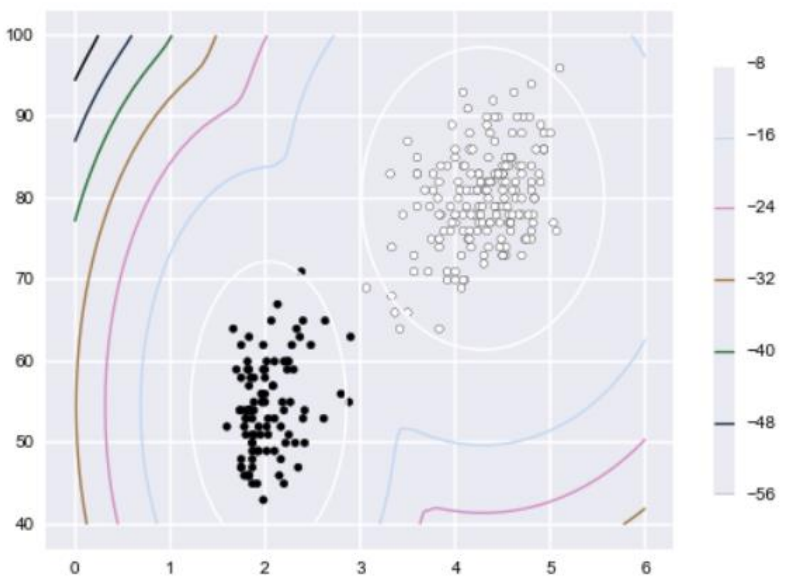
```python
from sklearn import mixture
clf1 = mixture.GMM(n_components=1)
clf1.fit(ofdata.values)
clf2 = mixture.GMM(n_components=2)
clf2.fit(ofdata.values)
clf3 = mixture.GMM(n_components=3)
clf3.fit(ofdata.values)
clf4 = mixture.GMM(n_components=4)
clf4.fit(ofdata.values)
clf5 = mixture.GMM(n_components=5)
clf5.fit(ofdata.values)
clf6 = mixture.GMM(n_components=6)
clf6.fit(ofdata.values)
clf7 = mixture.GMM(n_components=7)
clf7.fit(ofdata.values)
clf8 = mixture.GMM(n_components=8)
clf8.fit(ofdata.values)
```

```python
def plot_clf(clf):
    mask = clf.predict(ofdata.values)
    xx = np.linspace(0, 6)
    yy = np.linspace(40, 100)
    X, Y = np.meshgrid(xx, yy)
    XX = np.c_[X.ravel(), Y.ravel()]
    Z =  clf.score_samples(XX)[0]
    Z = Z.reshape(X.shape)

    CS = plt.contour(X, Y, Z)
    CB = plt.colorbar(CS, shrink=0.8, extend='both')
    plt.scatter(ofdata.eruptions, ofdata.waiting, c=mask);
```

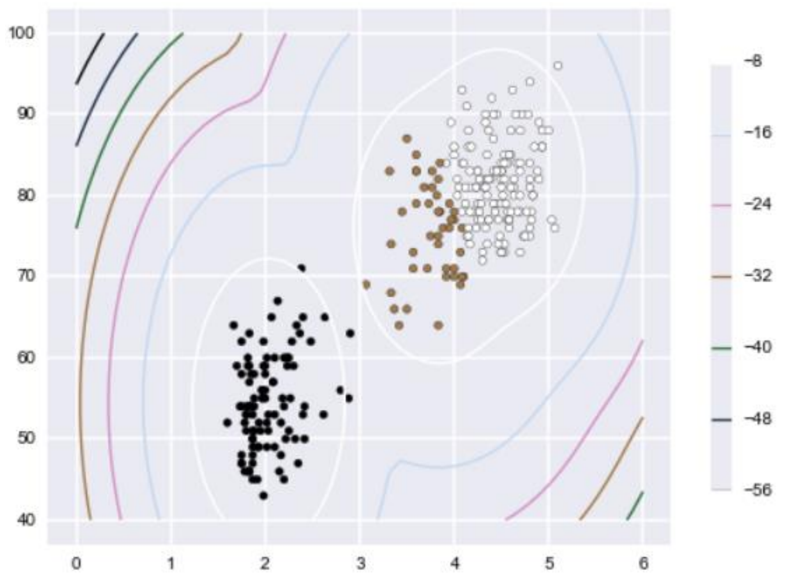
```
plot_clf(clf1)
```



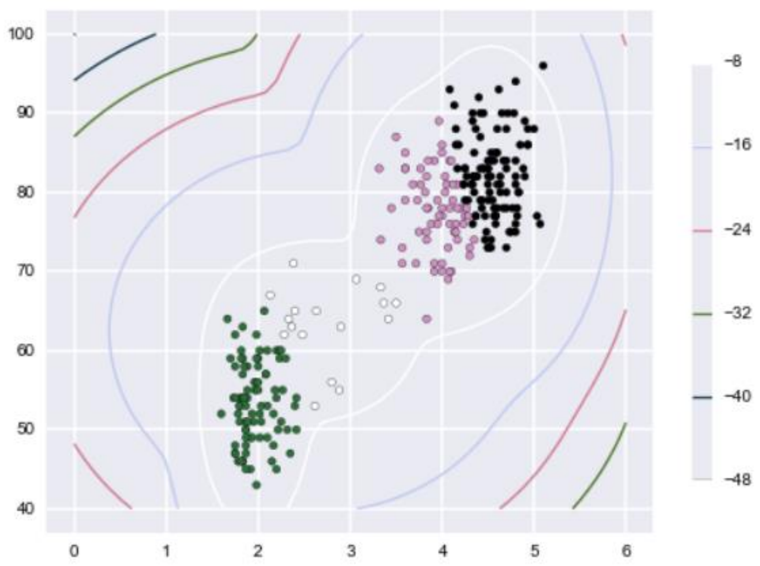

```
plot_clf(clf2)
```



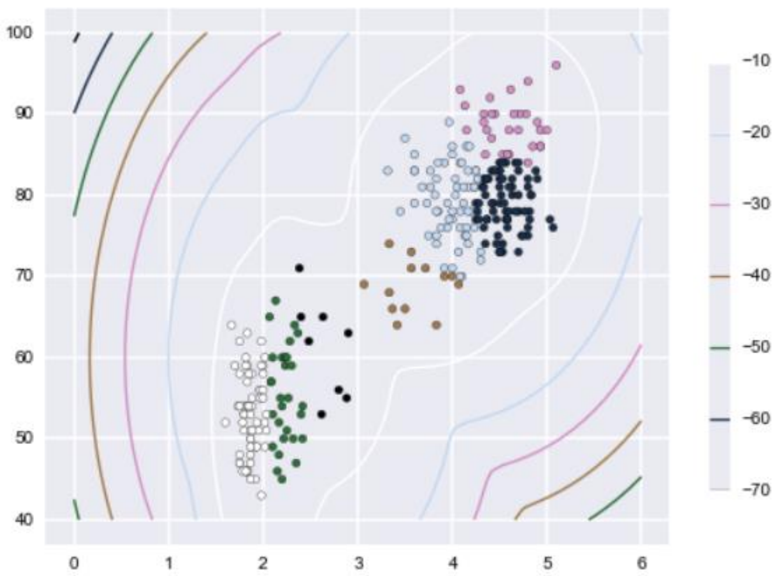

```
plot_clf(clf3)
```




```
plot_clf(clf4)
```



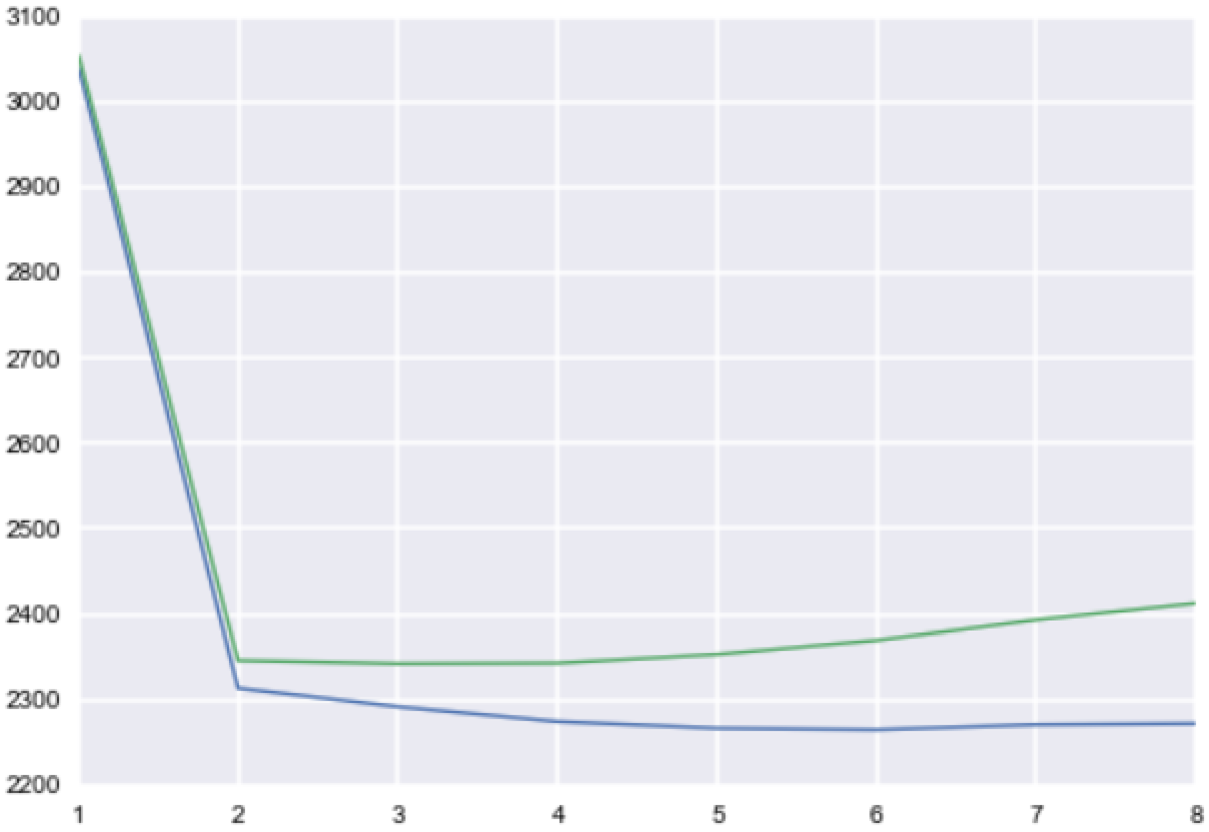

```
plot_clf(clf7)
```



Fit criteria: Choose the number of mixtures

```
aics=[clf1.aic(ofdata.values), clf2.aic(ofdata.values),
      clf3.aic(ofdata.values), clf4.aic(ofdata.values),
      clf5.aic(ofdata.values), clf6.aic(ofdata.values),
      clf7.aic(ofdata.values), clf8.aic(ofdata.values)]
bics=[clf1.bic(ofdata.values), clf2.bic(ofdata.values),
      clf3.bic(ofdata.values), clf4.bic(ofdata.values),
      clf5.bic(ofdata.values), clf6.bic(ofdata.values),
      clf7.bic(ofdata.values), clf8.bic(ofdata.values)]
```

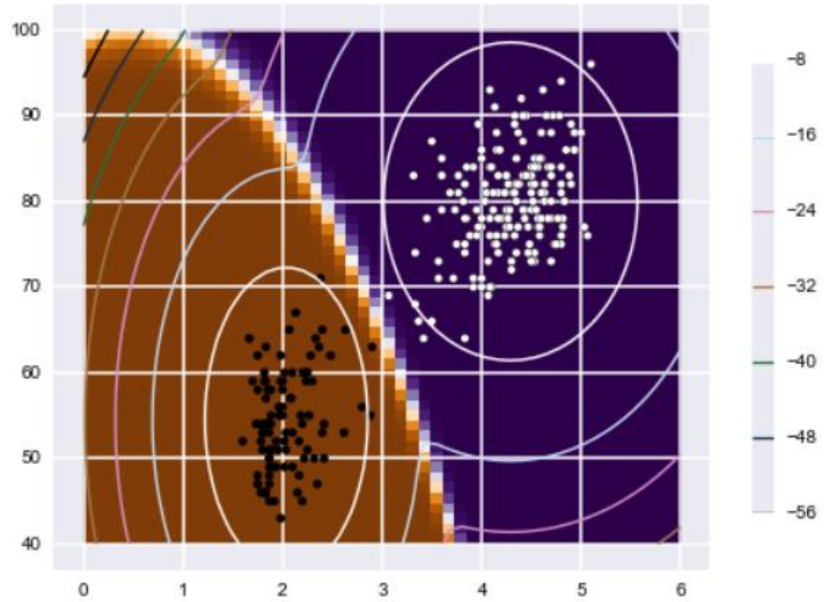
```
plt.plot([1,2,3,4,5,6,7,8],aics)
plt.plot([1,2,3,4,5,6,7,8],bics)
```



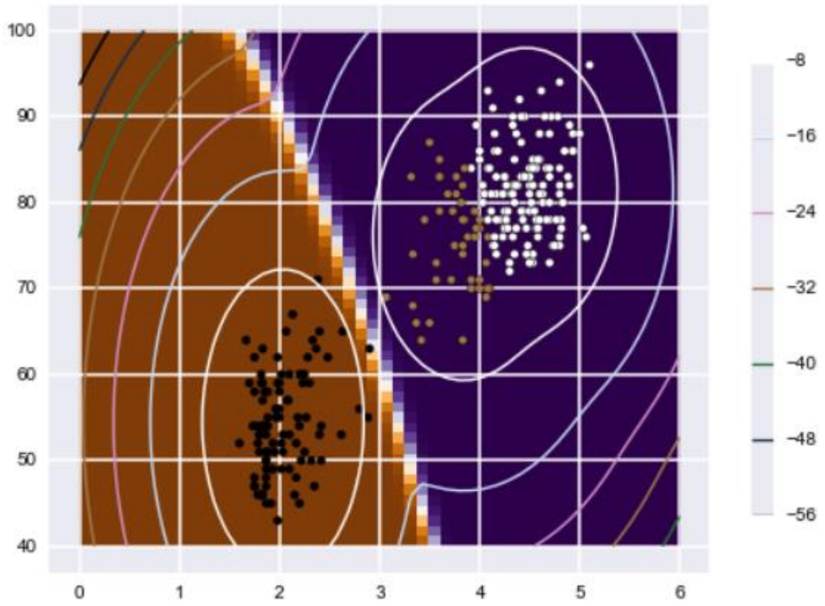
Plot areas of dominance of a component

```
def plot_clf_post(clf):
    mask = clf.predict(ofdata.values)
    xx = np.linspace(0, 6)
    yy = np.linspace(40, 100)
    X, Y = np.meshgrid(xx, yy)
    XX = np.c_[X.ravel(), Y.ravel()]
    cs=clf.score_samples(XX)
    Zline=cs[0]
    Zline=Zline.reshape(X.shape)
    Z =  cs[1][:,0]
    print Z
    Z = Z.reshape(X.shape)
    plt.imshow(Z, interpolation='nearest',
          extent=(X.min(), X.max(), Y.min(), Y.max()), aspect='auto',
          origin='lower', cmap=plt.cm.PuOr_r)
    CS = plt.contour(X, Y, Zline)
    CB = plt.colorbar(CS, shrink=0.8, extend='both')
    plt.scatter(ofdata.eruptions, ofdata.waiting, c=mask);
```

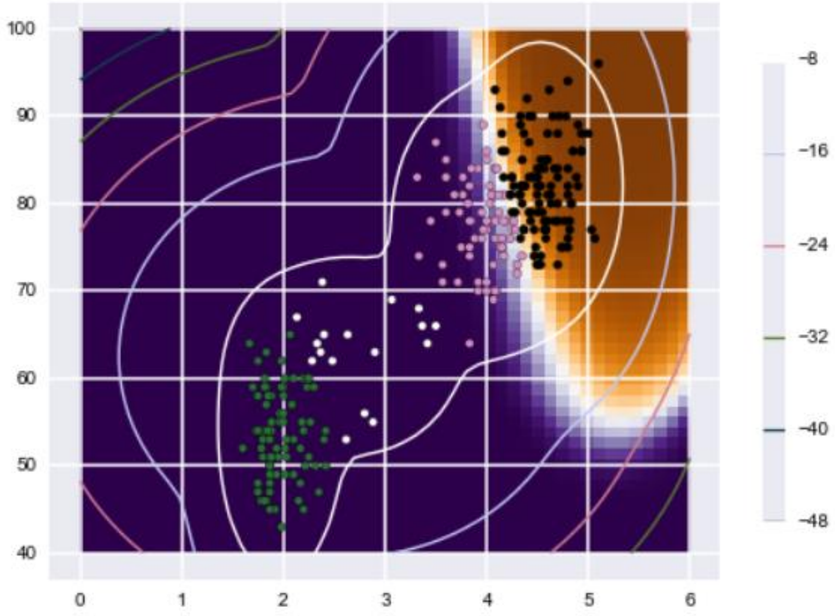
```
plot_clf_post(clf2)
```



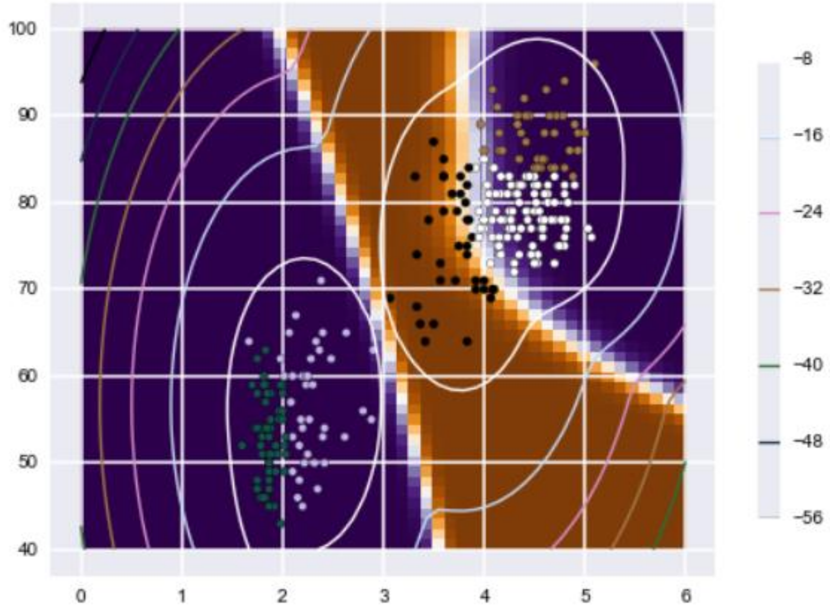

```
plot_clf_post(clf3)
```




```
plot_clf_post(clf4)
```




```
plot_clf_post(clf5)
```



reference

- Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
- https://tkipf.github.io/graph-convolutional-networks/
- code for GCN: https://github.com/tkipf/gcn
- code for EM: http://iacs-courses.seas.harvard.edu/courses/am207/blog/lab-on-em.html
- https://mp.weixin.qq.com/s?__biz=MzI5NTIxNTg0OA==&mid=2247497717&idx=2&sn=0c3ac7100ada7e74ff97c8befd7cda31&chksm=ec544072db23c964bcb19346c9889dd0fb3757b9de6a254dd1cb77492ae69eef7cdfe2de5267&mpshare=1&scene=1&srcid=&sharer_sharetime=1573653749774&sharer_shareid=a74670ea9d0508a5bf81c9a18d234a70#rd
- https://mp.weixin.qq.com/s?__biz=MzIzNjc1NzUzMw==&mid=2247532164&idx=4&sn=f7f33fb633328b429607f4d2636951e0&chksm=e8d0c9f6dfa740e0f67bfa317979fa4288c82c96b26ddacab27fe37557fcac8f65eb7b2229ec&mpshare=1&scene=1&srcid=&sharer_sharetime=1573653677792&sharer_shareid=a74670ea9d0508a5bf81c9a18d234a70#rd