Сокращение размерности

EM алгоритм

Напомним, как выглядит EM алгоритм.

$$\log p(x \mid \theta) = \int q(z) \log p(x \mid \theta) dz = \int q(z) \log \frac{p(x, z \mid \theta)}{p(z \mid x, \theta)} dz = \int q(z) \log \frac{p(x, z \mid \theta)}{q(z)} dz + \int q(z) \log \frac{q(z)}{p(z \mid x, \theta)} dz = \mathcal{L}(q, \theta) + KL(q(z) \mid p(z$$

E-step

$$q(z)^{(n+1)} = p(z \mid x, \theta^{(n)})$$

M-step

$$\theta^{(n+1)} = \max_{\theta} \mathcal{L}(q^{(n+1)}, \theta)$$

PCA как модель с латентными пременными

Процесс генерации данных в модели PCA выглядит следующим образом:

Генеративный процесс

Таким образом, мы получаем модель:

$$P(X, T \mid W, \mu, \sigma^2) = \prod_{n=1}^{N} p(x_n, t_n \mid W, \mu, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(x_n \mid W t_n + \mu, \sigma^2 I) \mathcal{N}(t_n \mid 0, I)$$

Задание 1. (5 баллов) Выпишите E и M шаги для модели PCA.

Необходимо решить задачу максимизации правдоподобия:

$$p(X \mid W, \mu, \sigma^2) = \prod_{n=1}^{N} p(x_n \mid W, \mu, \sigma^2) \rightarrow \max_{W, \mu, \sigma^2}$$

- **E step**

$$p(T \mid X, W, \mu, \sigma^2) = \prod_{n=1}^{N} p(t_n \mid x_n, W, \mu, \sigma^2)$$

$$p(t \mid x, W, \mu, \sigma^2) = \frac{p(t, x \mid W, \mu, \sigma^2)}{p(x \mid W, \mu, \sigma^2)} = \frac{p(x \mid t, W, \mu, \sigma^2) p(t)}{\int p(x \mid t, W, \mu, \sigma^2) p(t) dt} \overset{(1)}{=} \frac{\mathcal{N}(x \mid W t + \mu, \sigma^2 I) \mathcal{N}(t \mid 0, I)}{\mathcal{N}(x \mid \mu, W W^T + \sigma^2 I)} \overset{(2)}{=} \overset{(2)}{=} \mathcal{N}\Big(t \mid (W^T W + \sigma^2 I)^{-1} W$$

1) $\int p(x \mid t, W, \mu, \sigma^2)p(t)dt$ — свёртка плотностей нормального распределения $\Rightarrow$ полученная плотность является плотностью нормального распределения. При этом математическое ожидание и дисперсию полученного распределения можно посчитать. Пусть $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, тогда:

$$\mathrm{E}x = \mathrm{E}[Wt + \mu + \varepsilon] = \mu$$

$$\mathrm{V}x = \mathrm{E}\Big[(x-\mu)(x-\mu)^T\Big] = \mathrm{E}x\Big[(Wt+\varepsilon)(Wt+\varepsilon)^T\Big] = W\mathrm{E}\Big[tt^T\Big]W^T + \mathrm{E}\Big[\varepsilon\varepsilon^T\Big] = WW^T + \sigma^2 I$$

2)

$$\frac{\mathcal{N}(x \mid Wt + \mu, \sigma^2 I)\mathcal{N}(t \mid 0, I)}{\mathcal{N}(x \mid \mu, WW^T + \sigma^2 I)} = C_1 \exp\left(-\frac{1}{2}C_2\right)$$

$$C_1 = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{D}{2}}\left(\frac{1}{2\pi}\right)^{\frac{d}{2}}(2\pi)^{\frac{D}{2}}\left|WW^T + \sigma^2 I\right|^{\frac{1}{2}} = \left(\frac{1}{2\pi}\right)^{\frac{d}{2}}\left|I + \sigma^{-2}WW^T\right|^{\frac{1}{2}}$$

- **M step**

$$\mu_{\text{new}} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$W_{\text{new}} = \left(\sum_{n=1}^{N}(x_n - \mu_{\text{new}})\mathrm{E}t_n^T\right)\left(\sum_{n=1}^{N}\mathrm{E}t_n t_n^T\right)^{-1}$$

$$\sigma^2_{\text{new}} = \frac{1}{ND}\sum_{n=1}^{N}\left((x_n - \mu_{\text{new}})^T(x_n - \mu_{\text{new}}) - 2\mathrm{E}t_n^T W_{\text{new}}^T(x_n - \mu_{\text{new}}) + \mathrm{tr}W_{\text{new}}^T W_{\text{new}}\mathrm{E}t_n t_n^T\right)$$

Здесь $\mathrm{E}t_n$ и $\mathrm{E}t_n t_n^T$, считаются с помощью распределения, полученного в E-step.

Задание 2. (5 баллов) Реализуйте EM алгоритм для модели PCA

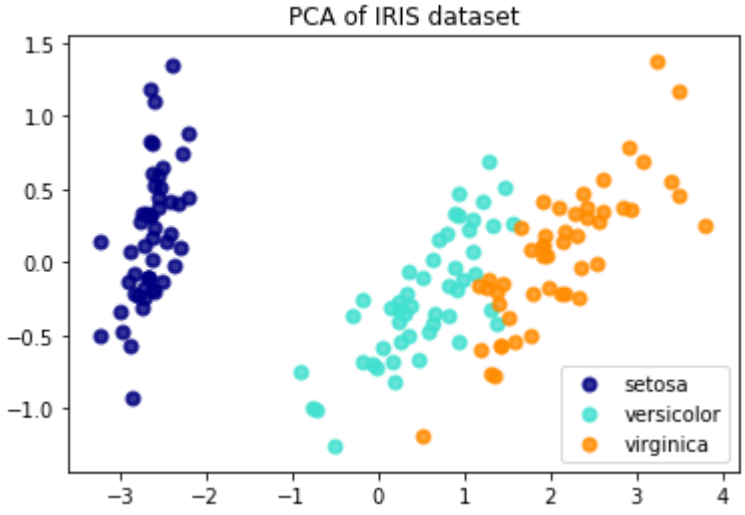
Т.к. μ не изменяется на итерациях, центрируем данные и будет считать $\mu = 0$.

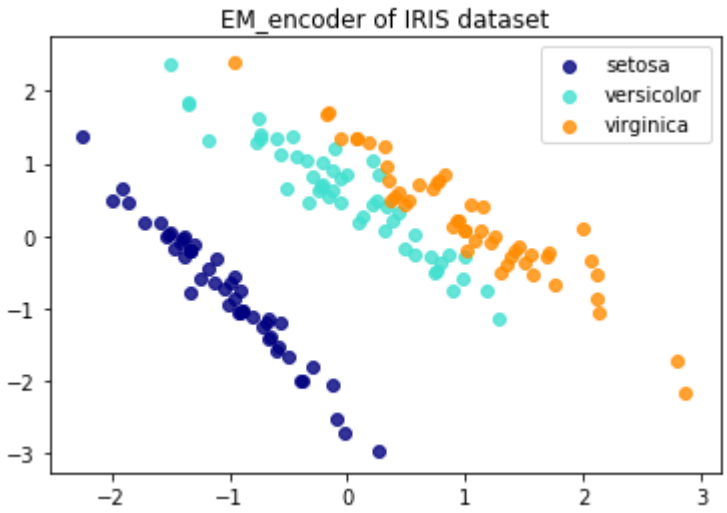
In [4]:

```
from sklearn.preprocessing import StandardScaler
import numpy as np

class EM_encoder:
    def __init__(self, d, max_iter=100):
        self.scaler = StandardScaler(with_std=False)
        self.W = None
        self.sigma2 = None
        self.d = d
        self.max_iter = max_iter

    def make_step(self, X):
        # E-step
        M = np.linalg.inv(self.W.T @ self.W + self.sigma2* np.identity(self.d))
        Sigma = self.sigma2 * M
        means = np.apply_along_axis(lambda x:M @ self.W.T @ x, 1, X)

        # M-step
        self.W = sum(X[i][:,np.newaxis] @ means[i][:,np.newaxis].T for i in range(X
               @ np.linalg.inv(X.shape[0] * Sigma + sum(mean[:,np.newaxis] @ mean[:
                                                    for mean in means))

        self.sigma2 = sum((X[i]**2).sum() - 2 * means[i][:,np.newaxis].T @ self.W.T
                          + np.trace(self.W.T @ self.W @ (Sigma + means[i][:,np.new
                              for i in range(X.shape[0])) / X.size

    def fit(self, X, W=None, sigma2=None):
        if W is None:
            self.W = np.random.uniform(0, 1, (X.shape[1], self.d))
        else:
            self.W = W
        if sigma2 is None:
            self.sigma2 = 1
        else:
            self.sigma2 = sigma2

        X_copy = self.scaler.fit_transform(X)
        for i in range(self.max_iter):
            self.make_step(X_copy)

    def transform(self, X):
        return np.apply_along_axis(lambda x:np.linalg.inv(self.sigma2 + self.W.T @
                                           @ self.W.T @ x,
                                    1, self.scaler.transform(X))
```

In [6]:

```
# check with iris

import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.decomposition import PCA

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

em_encoder = EM_encoder(2, 1000)
em_encoder.fit(X)
X_r2 = em_encoder.transform(X)

# Percentage of variance explained for each components
#print('explained variance ratio (first two components): %s'
#      % str(pca.explained_variance_ratio_))

plt.figure()
colors = ['navy', 'turquoise', 'darkorange']
lw = 2

for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_r[y == i, 0], X_r[y == i, 1], color=color, alpha=.8, lw=lw,
                label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('PCA of IRIS dataset')

plt.figure()
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_r2[y == i, 0], X_r2[y == i, 1], alpha=.8, color=color,
                label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('EM_encoder of IRIS dataset')

plt.show()
```





Задание 3. (10 баллов) Используя свою реализацию метода PCA, решите задачу восстанавления давления по профилю крыла.

Скачать данные со страницы курса. https://yadi.sk/i/065iNGXp3QYv5n (https://yadi.sk/i/065iNGXp3QYv5n)

Данные представляют собой матрицу размера 397×114; в каждой строке находится подвектор, описывающий профиль (первые 57 координат), и подвектор, описывающий соответствующее распределение давления (следующие 57 координат). Задача состоит в том, чтобы на основании метода главных компонент построить алгоритм, позволяющий восстанавливать распределение давления по профилю. Опишем алгоритм обучения и алгоритм восстановления. Пусть $\mathbf{A} \in \mathrm{R}^{57}$ -- вектор, описывающий профиль, а $\mathbf{P} \in \mathrm{R}^{57}$ -- вектор, описывающий распределение давления. Дана обучающая выборка $(\mathbf{A}_i, \mathbf{P}_i)_{i=1}^N$ (взять в качестве обучающей выборки 75% случайно выбранных строк из матрицы данных; остальные 25% использовать в качестве тестового множества данных).

Алгоритм обучения состоит из следующих шагов:

1. По данным $(\mathbf{A}_i, \mathbf{P}_i)_{i=1}^N$ оцениваются первые d (параметр алгоритма) главных компонент $\mathbf{e}_1, \ldots, \mathbf{e}_d$, где $\mathbf{e}_i \in \mathrm{R}^{114}$.
2. Каждый из векторов $\mathbf{e}_i$, $i = 1, \ldots, d$ представляется как объединение двух подвекторов $\mathbf{e}_i = (\mathbf{e}_i^A \in \mathrm{R}^{57}, \mathbf{e}_i^P \in \mathrm{R}^{57})$, $i = 1, \ldots, d$, соответствующих описанию профиля и распределению давления соответственно.

Преобразование произвольного объединенного вектора $\mathbf{Z} = (\mathbf{A}, \mathbf{P})$ в сжатое описание $\lambda = (\lambda_1, \ldots, \lambda_d)$ происходит согласно следующей формуле:

$$\lambda_i = \left(\mathbf{Z} - \mathbf{Z}_{\text{mean}}, \mathbf{e}_i\right), i = 1, \ldots, d,$$

где $\mathbf{Z}_{\text{mean}} = \frac{1}{N}\sum_{k=1}^N \mathbf{Z}_k$, $\mathbf{Z}_k = (\mathbf{A}_k, \mathbf{P}_k)$ (выборочное среднее; подсчитывается по обучающей выборке). Восстановления подвекторов $\mathbf{A}$ и $\mathbf{P}$ объединенного вектора $\mathbf{Z} = (\mathbf{A}, \mathbf{P})$ по сжатому описанию $\lambda = (\lambda_1, \ldots, \lambda_d)$ происходит согласно формулам

$$\mathbf{A}^*(\lambda) = \mathbf{A}_{\text{mean}} + \sum_{i=1}^d \lambda_i \mathbf{e}_i^A, \; \mathbf{P}^*(\lambda) = \mathbf{P}_{\text{mean}} + \sum_{i=1}^d \lambda_i \mathbf{e}_i^P.$$

Алгоритм восстановления давления $\mathbf{P}$ по профилю $\mathbf{A}$ может быть описан следующим образом:

1. Для заданного профиля $\mathbf{A}$ определяются такие $\lambda = (\lambda_1, \ldots, \lambda_d)$, что $\|\mathbf{A}^*(\lambda) - \mathbf{A}\|_2^2$ принимает минимальное значение (такое значение λ подсчитывается с помощью псевдообращения матрицы $E = [\mathbf{e}_1^A, \ldots, \mathbf{e}_d^A]$, то есть $\lambda = E^+ \left(\mathbf{A} - \mathbf{A}_{\text{mean}}\right)$, где E^+ -- псевдообратная матрица).
2. По вычисленному вектору λ оценивается давление согласно формуле $\mathbf{P}^*(\lambda) = \mathbf{P}_{\text{mean}} + \sum_{i=1}^{d} \lambda_i \mathbf{e}_i^P$.

Задача состоит в том, чтобы реализовать предложенный алгоритм; подсчитать график зависимости средней ошибки восстановления векторов давлений из тестовой выборки от размерности сжатия d; нарисовать несколько восстановленных графиков давления (для нескольких разных профилей) с наложением истинных значений давления. Ошибка между набором векторов $(\mathbf{P}_i)_{i=1}^M$ и восстановленными векторами $(\mathbf{P}_i^*)_{i=1}^M$ оценивается согласно формуле $\sqrt{\frac{1}{M}\sum_{i=1}^{M} \| \mathbf{P}_i - \mathbf{P}_i^* \|^2}$, где $\| \cdot \|$ -- евклидово расстояние

In [121]:

```
from sklearn.model_selection import train_test_split
```

In [122]:

```
data = np.fromfile('A2P.txt', sep=' ').reshape(397, 114)
A = data[:,57]
P = data[:,58:]
A_train, P_train, A_test, P_test = train_test_split(A, P, test_size=0.25)
```

In [ ]:

```

```