COMP755-Lect14

October 9, 2018

1 COMP 755

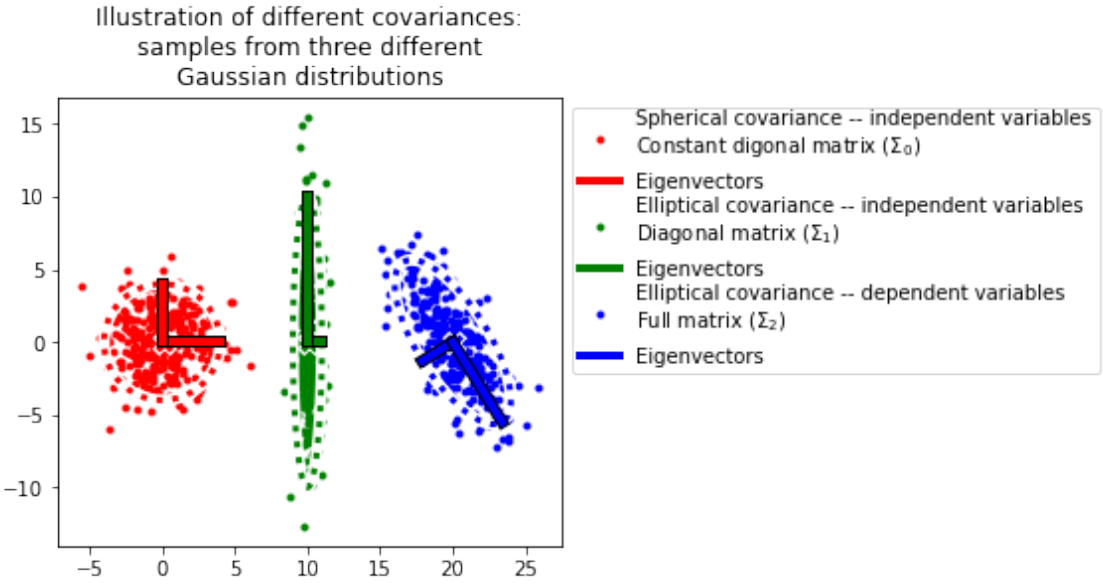
Plan for today

1. Latent Linear Models
2. Factor Analysis
3. Principal Component Analysis

```
In [1]: import numpy
        import matplotlib.pyplot as plt
        %matplotlib inline
        def generate_data(N,d,K,proby,mus,As=None):
            if As is None:
                As = numpy.zeros((d,d,K))
                for k in range(K):
                    As[:,:,k] = numpy.eye(d)
            ys = numpy.zeros(N,dtype='int')
            xs = numpy.zeros((d,N))
            for i in range(N):
                # Sample class according to the prior p(y)
                # in this case it is uniform
                ys[i] = numpy.random.choice(K,1)[0]
                # Sample feature values according to p(x|y)
                # In this case, x ~ N(mmu[y[i]],sigma2*I)
                # To accomplish this, draw z1,z2 ~ N(0,I)
                z = numpy.random.randn(2,1)
                # transform by matrix A and shift by class mean
                A = As[:,:,ys[i]].squeeze()
                mu = mus[:,ys[i]]
                Az = numpy.dot(A,z)
                x = Az + mu[:,numpy.newaxis]
                xs[:,i] = x[:,0]
            return xs,ys

        def plot_samples(xs,ys,mus=None,Sigmas=None,colors=['r','g','b','k','c','m'],labels=None
            N = xs.shape[1]
            if not ys is None:
                K = numpy.max(ys)+1
```

```
            for c in range(K):
            # indices of samples assigned to class c
                ind = [i for i in range(N) if ys[i]==c]
                if labels is None:
                    label = "Samples in cluster " + str(c)
                else:
                    label = labels[c]
                plt.plot(xs[0,ind],xs[1,ind],colors[c]+'.',label=label)
                if not mus is None:
                    plt.plot(mus[0,c],mus[1,c],'wx',markersize=9,markeredgewidth=5)
                    plt.plot(mus[0,c],mus[1,c],colors[c]+'x',markersize=7,markeredgewidth=3)
                if not Sigmas is None:
                    plot_covariance(mus[:,c],Sigmas[:,:,c],2.0,colors[c])
            plt.legend(loc=2, bbox_to_anchor=(1,1))
        else:
            plt.plot(xs[0,:],xs[1,:],'.')

    def plot_covariance(mu,Sigma,std_devs,color):
        N = 50
        alphas = numpy.linspace(0,2*numpy.pi,N)
        x = numpy.cos(alphas)
        y = numpy.sin(alphas)
        xy = numpy.vstack((x,y))
        d,v = numpy.linalg.eig(Sigma)
        d = numpy.sqrt(d)


        xy = std_devs*numpy.dot(numpy.dot(v,numpy.diag(d)),xy) + mu[:,numpy.newaxis]
        plt.plot(xy[0,:],xy[1,:],'w-',linewidth=6)
        plt.plot(xy[0,:],xy[1,:],color+':',linewidth=3)
        label = 'Eigenvectors'
        for j in range(2):
            plt.plot([mu[0],mu[0]+2.0*d[j]*v[0,j]],[mu[1],mu[1]+2.0*d[j]*v[1,j]],'k-',linewi

            plt.plot([mu[0],mu[0]+2.0*d[j]*v[0,j]],[mu[1],mu[1]+2.0*d[j]*v[1,j]],color+'-',l
            label = None


In [2]: import matplotlib.pyplot as plt

    plt.figure(figsize=(10,10))
    plt.subplot(2,2,1)
    K = 3
    d=2
    mus = 10*numpy.asarray([[0.0,1.0,2.0],[0.0,0.0,0.0]])
    As = numpy.asarray([[[2.0,0.0],[0.0,2.0]],
                        [[0.5,0.0],[0.0,5.0]],
                        [[2.0,-2.0],[0.0,2.0]]])
```

```
As = numpy.swapaxes(As,0,2)
Sigmas = numpy.zeros((d,d,K))
for c in range(K):
    A = As[:,:,c]
    Sigmas[:,:,c] = numpy.dot(A,A.transpose())

proby = [1./K]*K
numpy.random.seed(1)
xs,ys = generate_data(1000,2,K,proby,mus,As)

plot_samples(xs,ys,mus=mus,Sigmas=Sigmas,
             labels=['Spherical covariance -- independent variables\nConstant digonal ma
                     'Elliptical covariance -- independent variables\nDiagonal matrix
                     'Elliptical covariance -- dependent variables\nFull matrix ($\Sig
plt.axis('image')
plt.title('Illustration of different covariances:\nsamples from three different\nGaussia
```



2 Difference between regression, mixure models, and subspace models

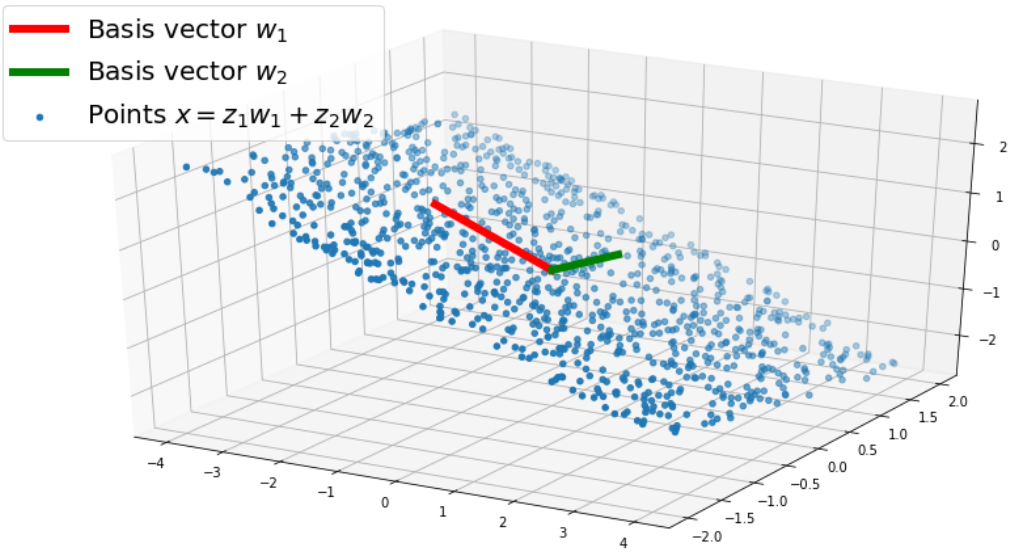
Regression models describe each sample using the same small number of weights, assuming predictors are given.

Mixture models describe each sample by its cluster membership and learned cluster parameters.

Subspace models describe a sample by the same learned basis, weighted by **different** small number of **coordinates** for each sample.

```
In [3]: import numpy as np
        from mpl_toolkits.mplot3d import Axes3D
```

```
import matplotlib.pyplot as plt

%matplotlib inline
np.random.seed(3)


# basis vectors
v1 = np.asarray([-2.0,0.0,1.0])
v2 = np.asarray([0.0,1.0,-0.3])
# coordinates in subspace
z1 = 4.0*np.random.rand(1000,1) - 2.0
z2 = 4.0*np.random.rand(1000,1) - 2.0

xs =  z1*v1 + z2*v2
fig = plt.figure(figsize=(15,8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(xs[:,0],xs[:,1],xs[:,2],label='Points $x = z_1 w_1 + z_2 w_2$')
ax.plot([0,v1[0]],[0,v1[1]],[0,v1[2]],color='r',linewidth=6,label='Basis vector $w_1$')
ax.plot([0,v2[0]],[0,v2[1]],[0,v2[2]],color='g',linewidth=6,label='Basis vector $w_2$')
ax.legend(loc=2,fontsize=20);
```



3 Factor analysis model

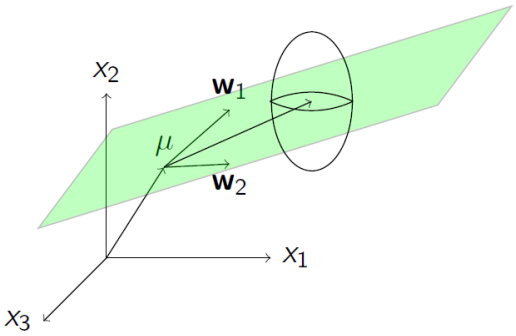
Data is composed of samples $\mathbf{x}_t$ of dimension d.

We assume that there is a basis composed of L vectors ($\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_L]$),

Generative model for a point $\mathbf{x}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0},\mathbf{I})$$
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}+\mathbf{W}\mathbf{z},\boldsymbol{\Psi})$$

$\mathbf{z}$ Factors
$\mathbf{W}$ Factor loading matrix



$$\mathbf{x} = \boldsymbol{\mu} + z_1\mathbf{w}_1 + z_2\mathbf{w}_2 + \epsilon$$

$$\begin{aligned}\mathbf{z} &\sim \mathcal{N}(0, I)\\ \mathbf{x} &\sim \mathcal{N}(\bar{} + \mathbf{W}\mathbf{z}, \Psi)\end{aligned}$$

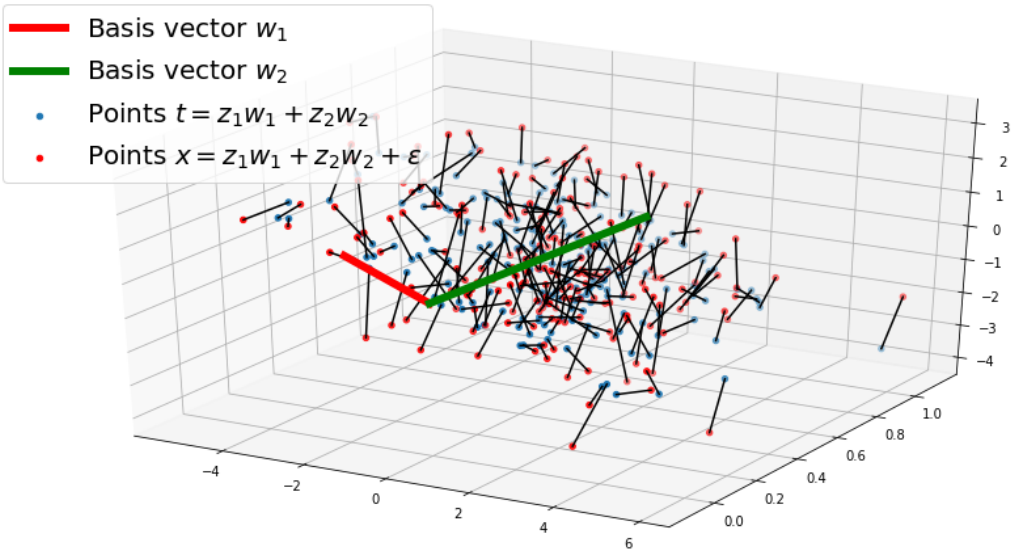
where $\mathbf{W}, \bar{}, \Psi$ are parameters.

Note that we assume that Ψ is diagonal -- any dependencies in $\mathbf{x}$ must be accounted for by the $\mathbf{W}$ and $\mathbf{z}$.

4 Factor analysis

```
In [4]: import numpy as np
        from mpl_toolkits.mplot3d import Axes3D
        import matplotlib.pyplot as plt

        %matplotlib inline
        np.random.seed(3)

        T = 200
        # basis vectors
        v1 = np.asarray([-2.0,0.0,1.0])
        v2 = np.asarray([0.0,1.0,-0.3])
        # coordinates in subspace
        z1 = np.random.randn(T,1)
        z2 = np.random.rand(T,1)

        ts =  z1*v1 + z2*v2
        epsilon = np.random.randn(T,3)*np.asarray([0.1,0.1,1.0])
        xs = ts + epsilon
        fig = plt.figure(figsize=(15,8))
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(ts[:,0],ts[:,1],ts[:,2],label='Points $t = z_1 w_1 + z_2 w_2$')
```

```
ax.scatter(xs[:,0],xs[:,1],xs[:,2],label='Points $x = z_1 w_1 + z_2 w_2 + \epsilon$',c='
for (t,x) in zip(xs,ts):
    ax.plot([x[0],t[0]],[x[1],t[1]],[x[2],t[2]],color='k')
ax.plot([0,v1[0]],[0,v1[1]],[0,v1[2]],color='r',linewidth=6,label='Basis vector $w_1$')
ax.plot([0,v2[0]],[0,v2[1]],[0,v2[2]],color='g',linewidth=6,label='Basis vector $w_2$')
ax.legend(loc=2,fontsize=20);
```



5 Multivariate Gaussian distribution

Next few slides will deal with multivariate Gaussian distributions.

Key observations: 1. if $p(x)$ and $p(y|x)$ are Gaussian then $p(y,x)$ is Gaussian 2. if $p(y,x)$ is Gaussian then $p(x)$ and $p(y|x)$ are Gaussian 3. if $p(x)$ is Gaussian and $p(y)$ is Gaussian then $p(y+x)$ is Gaussian

6 Gaussian equalities

Forming a joint from a marginal and a conditional

$$
\begin{aligned}
\mathbf{x} &\sim \mathcal{N}(\mathbf{a}, \mathbf{A}) \\
\mathbf{y} \mid \mathbf{x} &\sim \mathcal{N}(\mathbf{b} + \mathbf{C}\mathbf{x}, \mathbf{B}) \\
\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} + \mathbf{C}\mathbf{a} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{A}^{\mathsf{T}}\mathbf{C}^{\mathsf{T}} \\ \mathbf{C}\mathbf{A} & \mathbf{B} + \mathbf{C}\mathbf{A}\mathbf{C}^{\mathsf{T}} \end{bmatrix} \right)
\end{aligned}
$$

7 Gaussian equalities

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^{\mathbf{T}} & \mathbf{B} \end{bmatrix} \right)$$

Marginals

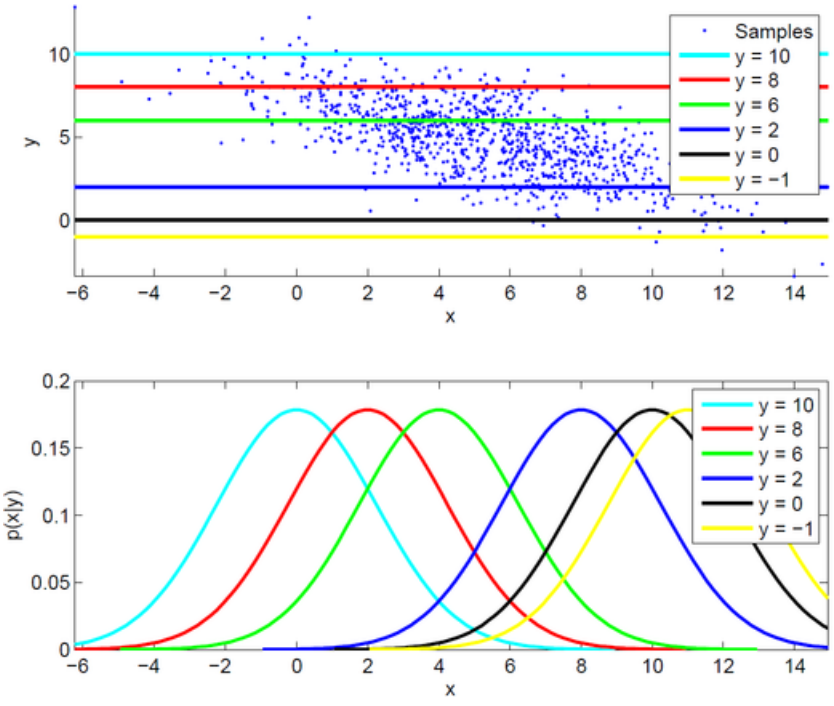
$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\mathbf{a}, \mathbf{A}) \\ \mathbf{y} &\sim \mathcal{N}(\mathbf{b}, \mathbf{B}) \end{aligned}$$

From a joint to conditionals

$$\begin{aligned} \mathbf{x}|\mathbf{y} &\sim \mathcal{N}\left(\mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^{T}\right) \\ \mathbf{y}|\mathbf{x} &\sim \mathcal{N}\left(\mathbf{b} + \mathbf{C}^{T}\mathbf{A}^{-1}(\mathbf{x} - \mathbf{a}), \mathbf{B} - \mathbf{C}^{T}\mathbf{A}^{-1}\mathbf{C}\right) \end{aligned}$$

8 Variance of conditional p(x | y)

$$\mathbf{y}|\mathbf{x} \sim \mathcal{N}\left(\mathbf{b} + \mathbf{C}^{T}\mathbf{A}^{-1}(\mathbf{x} - \mathbf{a}), \mathbf{B} - \mathbf{C}^{T}\mathbf{A}^{-1}\mathbf{C}\right)$$



9 Sum of Gaussian random variables is a Gaussian random variable

Given two Gaussian distributed random variables **x** and **y**

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\bar{}_1, \Sigma_1) \\ \mathbf{y} &\sim \mathcal{N}(\bar{}_2, \Sigma_2) \end{aligned}$$

their sum $\mathbf{z} = \mathbf{x} + \mathbf{y}$ is also Gaussian distributed

$$\mathbf{z} \sim \mathcal{N}(\bar{}_1 + \bar{}_2, \Sigma_1 + \Sigma_2)$$

Note that variances and means are added together. Sum is more uncertain than each of the variables on their own.

10 Inferring coordinates

One of the tasks we want to accomplish is to infer coordinates $\mathbf{z}$ from data $\mathbf{x}$ and parameters.

$$\begin{aligned} p(\mathbf{z} \mid \mathbf{x}, \mu, \Sigma) &= \mathcal{N}(\mathbf{z} \mid \mathbf{m}, S) \\ S &= (I + \mathbf{W}^T \Psi^{-1} \mathbf{W})^{-1} \\ \mathbf{m} &= S(\mathbf{W}^T \Psi^{-1}(\mathbf{x} - \mu)) \end{aligned}$$

Derivation of this distribution requires use of conditional Gaussian equalities and Sherman-Morrison-Woodbury/inversion lemma.

11 EM algorithm for factor analysis:

$$\bar{} = \frac{1}{T} \sum_t \mathbf{x}_t$$

E step:

$$\begin{aligned} \mathbf{m}_t &= \mathbf{W}^T (\mathbf{W}\mathbf{W}^T + \Psi)^{-1} (\mathbf{x}_t - \bar{}) \\ \mathbf{V} &= I - \mathbf{W}^T (\mathbf{W}\mathbf{W}^T + \Psi)^{-1} \mathbf{W} \end{aligned}$$

M step:

$$\begin{aligned} \mathbf{W}^{\text{new}} &= \left(\sum_t \mathbf{x}_t \mathbf{m}_t^T \right) \left(\sum_t \mathbf{V} \right)^{-1} \\ \Psi^{\text{new}} &= \frac{1}{T} \text{diag} \left(\sum_t \mathbf{x}_t \mathbf{x}_t^T + \mathbf{W}^{\text{new}} \sum_t \mathbf{m}_t \mathbf{x}_t^T \right) \end{aligned}$$

12 Factor Analysis is unindentifiable

Unindentifability refers to situation where parameters of a distribution cannot be uniquely determined.

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(0, I) \\ \mathbf{x} &\sim \mathcal{N}(\bar{} + \mathbf{W}\mathbf{z}, \Psi) \end{aligned}$$

where R is a rotation matrix ($RR^T = I$) achieve the same log-likelihood.
To show this, let R is a rotation matrix ($RR^T = I$), and compte we marginal distribution of $\mathbf{x}$:

$$\begin{aligned} p(\mathbf{x} \mid \bar{}, \Psi, \mathbf{W}) &= \mathcal{N}(\bar{} + \mathbf{W}0, \Psi + \mathbf{W}I\mathbf{W}^T) \\ &= \mathcal{N}(\bar{}, \Psi + \mathbf{W}\mathbf{W}^T) \\ p(\mathbf{x} \mid \bar{}, \Psi, \mathbf{W}R) &= \mathcal{N}(\bar{} + \mathbf{W}0, \Psi + \mathbf{W}RIR^T\mathbf{W}^T) \\ &= \mathcal{N}(\bar{}, \Psi + \mathbf{W}\mathbf{W}^T) \end{aligned}$$

Hence, the data can be represented using factor analysis, but interpretation of matrix $\mathbf{W}$ can be problematic.

13 Principal Component Analysis

If we use the same model as Factor Analysis but make additional assumptions: 1. Subspace basis vectors are orthogonal ($W^TW = I$) 2. Variance $\Psi = \sigma^2 I$ with very small σ^2

we arrive at a variant of Principal Component Analysis.

14 Principal Component Analysis

For full statement see Theorem 12.2.1 in your book. Here we digest it a little bit.

Let: 1. each sample $\mathbf{x}_t$ be of dimension d 2. $\mathbf{W}$ be a matrix of size $d \times L$ such that $\mathbf{W}^T\mathbf{W} = I$ 3. and $\mathbf{z}_t, t = 1, ...T$, vectors of length L.

Minimum of

$$J(\mathbf{W}, \mathbf{Z}) = \frac{1}{T}\sum_{t=1}^{T} \underbrace{\|\mathbf{x}_t - \mathbf{W}\mathbf{z}_t\|^2}_{\text{reconstruction error}}$$

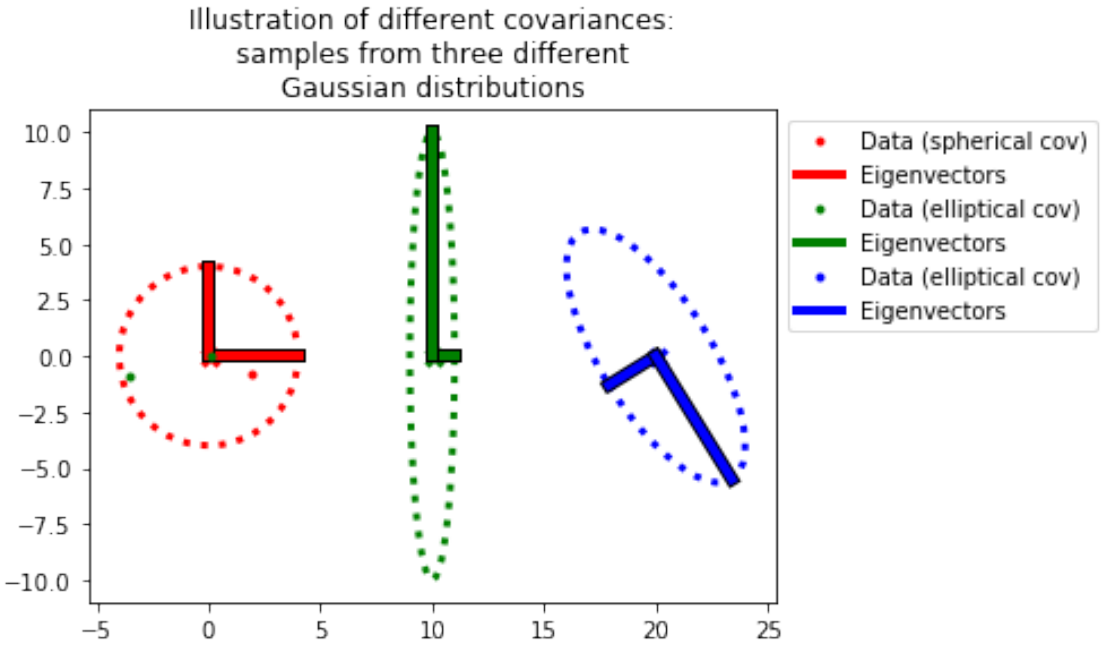
is achieved for 1. $\mathbf{W}^*$ composed of top L eigenvectors of covariance matrix $\hat{\Sigma} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{x}_t\mathbf{x}_t^T$ 2. $\mathbf{z}_t^* = (\mathbf{W}^*)^T\mathbf{x}_t$

15 Principal Component Analysis -- interpretation

Given Data $= \{\mathbf{x}_t : t = 1, ...T\}$, where samples $\mathbf{x}_t$ are d-long vectors, if you want to learn how to compress these vectors into shorter L long representations $\mathbf{z}_t$ 1. compute covariance of the dataset $\hat{\Sigma} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{x}_t\mathbf{x}_t^T$ 2. extract top eigenvectors $\mathbf{W}^*$ 3. project data to the space spanned by those eigenvectors $\mathbf{z}_t^* = (\mathbf{W}^*)^T\mathbf{x}_t$

Top eigenvectors of the covariance matrix are directions along which the data varies the most.

```
In [5]: plot_samples(xs,ys,mus=mus,Sigmas=Sigmas,
                     labels=['Data (spherical cov)',
                             'Data (elliptical cov)',
                             'Data (elliptical cov)'])
        plt.axis('image')
        plt.title('Illustration of different covariances:\nsamples from three different\nGaussia
```



16 Applying PCA

1. Data is converted into a vector -- image of size $W \times H \times 3$ becomes a vector of $W * H * 3$ values
2. Meaning of j^{th} entry in a data vectors -- assume data is aligned
3. Eigenvector of covariance matrix is of same dimensionality as a data vector -- you can transform it back to the original shape
4. Make sure that you are computing the correct covariance

$$\mathbf{X}\mathbf{X}^T \neq \mathbf{X}^T\mathbf{X}$$

   You could be looking at eigensample or eigenfeature.
5. You can visualize
6. $\mathbf{z}_t$ scores, one per sample
7. $\mathbf{w}_i$ eigenvectors or loadings
8. $\mathbf{W}\mathbf{z}_t$ reconstructions, one per sample

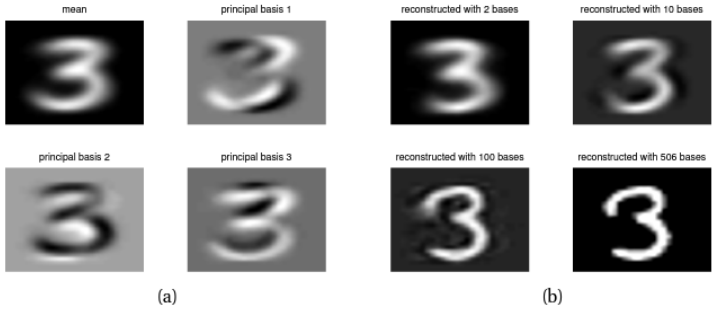
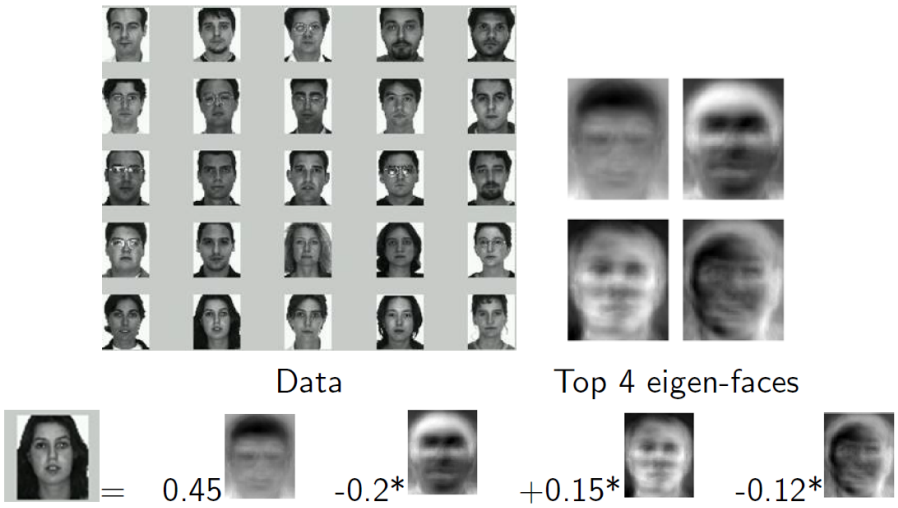


Figure 12.6 (a) The mean and the first three PC basis vectors (eigendigits) based on 25 images of the digit 3 (from the MNIST dataset). (b) Reconstruction of an image based on 2, 10, 100 and all the basis vectors. Figure generated by `pcaImageDemo`.



17 PCA on digit data

18 PCA on faces

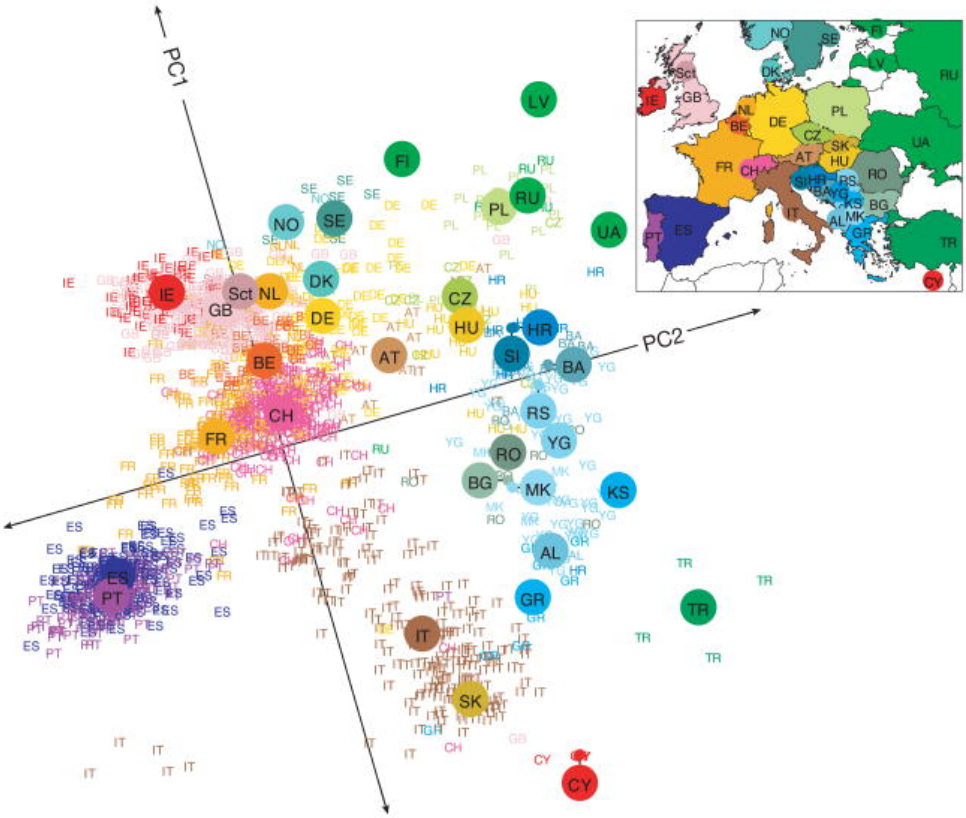
19 PCA on genetic data

Take genome of people in Europe. Compute PCA to obtain eigen-genomes. Project people on top two eigen-genomes.

20 PCA

Principal component analysis is a commonly used tool; you should become familar with it. Sometimes it is misunderstood

Jeff Porter, Cooking for Geeks:



Eigen Pancakes: The Hello, World! of Recipes

No one's ever wrong on the Internet, so the average of a whole bunch of right things must be righter, right? The quantities here are based on the average of the eight different pancake recipes from an online search. For each ingredient, I converted the measurement to grams and then calculated that ingredient's percentage of the total weight of the recipe. (This is somewhat like a "baker's percentage," in which ingredients are given as a percentage of the weight of flour in a recipe.)

In a mixing bowl, measure out and whisk together:

1½ cups (190g) flour
2 tablespoons (25g) sugar
2 teaspoons (10g) baking powder
½ teaspoon (3g) salt

In a separate, microwave-safe bowl, melt:

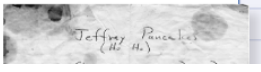
2 tablespoons (25g) butter

By default, assume that the order of ingredients in a recipe indicates the order in which you should add the ingredients into your bowl. It doesn't always matter, of course, but in this case you should add the milk before the eggs to prevent the eggs from cooking in the hot butter.

from two proteins, glutenin and gliadin, present in flour (they crosslink and bond together to create a stretchy, net-like matrix—think French bread).

Place a nonstick frying pan on a burner set to medium-high. Wait until the pan is hot. The standard test is to toss a few drops of water into the pan and see if they sizzle; the geek test is to take an IR thermometer and check that the pan is around 400°F / 200°C. Use a ladle, measuring cup, or ice cream scoop to pour about half a cup of batter into the pan. As the first side cooks, you'll see bubbles forming on the top surface of the pancake. Flip the pancake after those bubbles have started to form, but before they pop (about two minutes).

Wolfram|Alpha (*http://www.wolframalpha.com*) is a great resource for converting standard measurements to metric. Enter 1T sugar and it'll tell you 13g; enter the entire Eigen Pancake recipe—use "+" between individual ingredients—and it'll tell you 38 grams of fat, 189 grams of carbs, and 46 grams of protein.



21 COMP 755

Today we covered:

1. Latent Linear Models
2. Factor Analysis
3. Principal Component Analysis