COMP 790-125: Goals for today

- Mixture of Position Weight Matrices
- Another look at EM
- K-means as approximate EM
- Covariance matrices and Principal Component Analysis

Position Weight Matrix, Position Specific Scoring Matrix

A position specific probabilistic model of a sequence

	1	2	3	4	5	6
A	0.00	0.98	0.00	0.00	0.28	0.00
C	0.00	0.00	0.00	1.00	0.00	0.10
G	0.27	0.02	0.96	0.00	0.68	0.21
T	0.73	0.00	0.04	0.00	0.04	0.69

Probability of a sequence under PWM

Probability of a sequence $s =$ TAGCAT using PWM specified by this matrix θ

	1	2	3	4	5	6
A	0.00	0.98	0.00	0.00	0.28	0.00
C	0.00	0.00	0.00	1.00	0.00	0.10
G	0.27	0.02	0.96	0.00	0.68	0.21
T	0.73	0.00	0.04	0.00	0.04	0.69

$$\begin{aligned} f(\mathbf{s}|\theta) &= \prod_i \theta_{i,s_i} = \theta_{1,T}\theta_{2,A}\theta_{3,G}\theta_{4,C}\theta_{5,A}\theta_{6,T} \\ &= \prod_i \prod_v \theta_{i,v}^{[s_i=v]} \end{aligned}$$

where

$$[s_i = v] = \begin{cases} 1, & \text{if } s_i = v \\ 0, & \text{otherwise} \end{cases}$$

Mixture of PWMs

A PWM mixture component can now be specified as

$$f(\mathbf{x}|\theta_c) = \prod_i \prod_{v \in \{\mathrm{A,C,G,T}\}} \theta_{c,i,v}^{[x_i = v]}$$

So θ is just a large table of component, position, letter specific probabilities.

Note that $\sum_v \theta_{c,i,v} = 1$

Log-Likelihood

And if we have a matrix $\mathbf{X}$ with each row a sequences, for example $\mathbf{x}_i = \mathrm{ACGATA}$, then we can write a log-likelihood

$$\begin{aligned}
\mathrm{LL}(\pi, \theta; \mathbf{X}) &= \log\left\{\prod_i p(\mathbf{x}_i)\right\} \\
&= \log\left\{\prod_i \sum_{h_i} p(\mathbf{x}_i, h_i)\right\} \\
&= \log\left\{\prod_i \sum_{h_i} p(\mathbf{x}_i|h_i)p(h_i)\right\} \\
&= \sum_i \log\left\{\sum_{h_i} p(\mathbf{x}_i|h_i)p(h_i)\right\} \\
&= \sum_i \log\left\{\sum_{h_i=c} f_c(\mathbf{x}_i)\pi_c\right\}
\end{aligned}$$

Mixture of PWMs - E-step

Computation of f_c is different than in the Gaussian case (table lookups in this case) but computation of q is essentially the same

$$q(h_i) = \frac{p(\mathbf{x}_i|h_i)p(h_i)}{\sum_c p(\mathbf{x}_i|h_i = c)p(h_i = c)} = \frac{f(\mathbf{x}_i|\theta_{h_i})\pi_{h_i}}{\sum_c f(\mathbf{x}_i|\theta_{h_i=c})\pi_c}$$

Mixture of PWMs - M-step for π

Unsurpisingly, it is the same as in the case of Gaussians

$$\pi_c = \frac{\sum_i q(h_i = c)}{\sum_i \sum_{h_i} q(h_i)} = \frac{1}{N} \sum_i q(h_i = c)$$

and it will remain the same even if you change the mixture component distributions.

Mixture of PWMs - M-step for θ

Terms in lower bound for log likelihood involving θ are

$$\underbrace{\sum_i}_{\text{instances}} \underbrace{\sum_{h_i}}_{\text{classes}} q(h_i) \underbrace{\sum_j}_{\text{sequence positions}} \underbrace{\sum_v}_{\text{letters}} \log \theta_{c,j,v}^{[x_{i,j}=v]}$$

and the partial derivative is

$$\sum_i \sum_{h_i} q(h_i) \frac{\partial \log p(\mathbf{x}, h_i)}{\partial \theta_{c,j,v}} = \sum_i q(h_i = c)[x_{i,j} = v] \frac{1}{\theta_{c,j,v}}$$

Note that there is a constraint $\sum_v \theta_{c,j,v} = 1$ so we have to use the same approach we used mixing proportions (i.e. introduce a constrained optimization problem)

Mixture of PWMs - M-step for θ

Our constrained optimization problem for updating the vector $\theta_{c,j}$ (probabilities of letters in position j and class c)

$$\begin{array}{ll} \underset{\theta_{c,j} \in \mathbf{R}_4^+}{\text{maximize}} & \sum_i \sum_v q(h_i = c)[x_{i,j} = v] \log \theta_{c,j,v} \\ \text{subject to} & \sum_v \theta_{c,j,v} = 1 \end{array}$$

and its Lagrangian is

$$L(\theta_{c,j}, \lambda) = \sum_i \sum_v q(h_i = c)[x_{i,j} = v] \log \theta_{c,j,v} \; + \lambda(\sum_v \theta_{c,j,v} - 1)$$

Mixture of PWMs - M-step for θ

Again, conditions at optimum obtained by equating partial derivatives of Lagrangian to 0

$$\begin{aligned}\frac{\partial L(\theta_{c,j},\lambda)}{\partial \theta_{c,j,v}} &= \sum_i \frac{q(h_i=c)[x_{i,j}=v]}{\theta_{c,j,v}} + \lambda = 0 \\ \frac{\partial L(\theta_{c,j},\lambda)}{\partial \lambda} &= \sum_v \theta_{c,j,v} - 1 = 0\end{aligned}$$

lead to update

$$\theta^{\text{new}}_{c,j,v} = \frac{\sum_i q(h_i=c)[x_{i,j}=v]}{\sum q(h_i=c)}$$

Numerical considerations

Generally all of the probabilities ought to be kept in log domain.

But if we keep them in log domain and need to normalize them or take some average we run a risk of numerical problems

Suppose that $\log p(\mathbf{x}, h_i = 1) = \log p(\mathbf{x}, h_i = 2) = -800$, then after normalization $p(h_i = 1|\mathbf{x}) = p(h_i = 2|\mathbf{x}) = 0.5$.

Computing directly

$$\frac{\exp \log p(\mathbf{x}, h_i = 1)}{\exp \log p(\mathbf{x}, h_i = 1) + \exp \log p(\mathbf{x}, h_i = 2)} = \frac{\exp\{-800\}}{\exp\{-800\} + \exp\{-800\}}$$

will give a NaN due to divison by 0.

Log-Sum-Exp

Instead of naively exponentiating we can divide both numerator and denominator by a constant (for example $\exp\{-800\}$).
The constant should be the max of the log of values that you are summing.

$$\log \sum_c \exp v_c = \left[\log \sum_c \exp\{v_c - M\} \right] + M$$

where $M = \max\{v\}$

Initializing EM

As a rule of thumb, initial parameters should lead to mildly discernible $q(h)$ in the first iterations.

For example, initializing Gaussian means at the center of all data with minor perturbation and with covariances somewhat larger than the whole data covariance.

For PWM nearly uniform probabilities for each letter in the PWM (e.g. near 0.25 for nucleotide alphabet).

Initialize with a robust algorithm that makes fewer assumptions about the structure of the data and possibly has local minima.

Finally, always try multiple random initializations, just to make sure that the local minima that you get to see are not wildly different.

Demo Mixture of PWMs

Another look at EM

Last time we started with a log likelihood function

$$\mathrm{LL}(\theta) = \log p(x) = \log \sum_h p(x, h|\theta)$$

we lower bounded the log likelihood with

$$\mathcal{F}(\theta, q) = \sum_h q(h) \log \frac{p(x, h|\theta)}{q(h)}$$

using Jensen's inequality.

Another look at the bound

The bound

$$\mathcal{F}(\theta, q) = \sum_h q(h) \log \frac{p(x,h)}{q(h)}$$

can be expanded a bit to yield some of the familiar players

$$\begin{aligned}
\mathcal{F}(\theta, q) &= \sum_h q(h) \log \frac{p(h|x)p(x)}{q(h)} \\
&= \underbrace{\log p(x)}_{\mathrm{LL}(\theta)} + \underbrace{\overbrace{\sum_h q(h) \log p(h|x)}^{-\mathbf{H}(p,q)} \overbrace{- \sum_h q(h) \log q(h)}^{\mathbf{H}(q)}}_{-\mathrm{KL}(q||p(h|x))}
\end{aligned}$$

Interpeting the bound[1]

$$\mathcal{F}(\theta, q) = \underbrace{\log p(x)}_{\mathrm{LL}(\theta)} + \underbrace{\overbrace{\sum_h q(h) \log p(h|x)}^{-H(q,p)} \overbrace{- \sum_h q(h) \log q(h)}^{H(q)}}_{-KL(q||p(h|x))}$$

Looking at this expression you can make a couple of statements

- the bound prefers low $KL(q||p(h|x))$ in the extreme $KL(q||p(h|x)) = 0$ bound is tight ($\mathcal{F} = LL$)
- The bound prefers higher entropy $H(q)$ but low cross entropy $H(q, p)$

[1] remember we are maximizing

Another look at EM

The EM algorithm can be seen as a sequence of maximization steps[2]

$$\begin{aligned} E : q^{\text{new}} &= \operatorname*{argmax}_q \sum_h q(h) \log p(x, h|\theta) - \sum_h q(h) \log q(h) \\ M : \theta^{\text{new}} &= \operatorname*{argmax}_\theta \sum_h q^{\text{new}}(h) \log p(x, h|\theta) - \sum_h q^{\text{new}}(h) \log q^{\text{new}}(h) \end{aligned}$$

We made a shortcut earlier by asserting that optimal q is $p(h|x, \theta)$, the exact posterior for h.

$$\begin{aligned} E : q^{\text{new}} &= p(h|x, \theta) \\ M : \theta^{\text{new}} &= \operatorname*{argmax}_\theta \sum_h q^{\text{new}}(h) \log p(x, h|\theta) \end{aligned}$$

This algorithm is called exact EM.

[2]E-step is loosely referred to as inference step and M-step a learning step

How about inexact EM?

Let's go back to our mixture models for an illustration of a famous algorithm that turns out to be an inexact EM.

Instead of using $q(h_i) = p(h_i|x_i)$ let's just put all probability mass on a class that has the highest posterior probability.[3]

$$q_{\mathrm{ICM}}(h_i = c) = \begin{cases} 1, & p(h_i = c, x_i) > p(h_i = l, x_i), l \neq c \\ 0, & \textit{otherwise} \end{cases}$$

Unsurprisingly $H(q_{\mathrm{ICM}}) = 0$ so our bound would not willingly give us this solution.

[3] Of course, you may have to break ties

Getting to K-means

We've placed a constraint on $q(h_i)$ that it is a highly peaked distribution (it puts all of its mass on a single class).

We further assume that $\pi_c = \frac{1}{k}$ and each $\Sigma_c = I$

After making all of these additional assumptions about q, π, Σ we get the famous K-means algorithm.

K-means

Initialize cluster means $\mu_1, \ldots \mu_k$

1. for each datapoint find closest mean and assign the datapoint to that cluster
2. update means of each class to mean of points associated with that cluster

K-means in our language

$$
\begin{aligned}
VE \quad &: \quad q^{\text{new}}(h_i = c|x_i) \quad = \quad \begin{cases} 1, & f_c(x_i) \geq f_l(x_i), l \neq c \\ 0, & \text{otherwise} \end{cases} \\
M \quad &: \quad \mu_c^{\text{new}} \quad = \quad \frac{\sum_i q(h_i=c)x_i}{\sum_i q(h_i=c)}
\end{aligned}
$$

VE because this is, as astounding as that may sound, a variational E-step.

We will cover variational methods in general and iterated conditional modes in particular in a later lecture.

Why bother?

Upshot: by placing restrictions on the posterior and parameterizations you can build new algorithms or reinterpret old ones.

Knowing the assumptions that an algorithm makes enables you to construct examples to test or trip it up.

Alternatively, we can construct our own approximate algorithms from EM by judiciously making such assumptions.

Care to guess how we can trip up K-means? Recall it makes assumptions on q, π, Σ.

K-medians

Hard question for you[4]

K-means is to mixture of Gaussians as
K-medians is to mixture of ....

[4]not so hard if you know of a relationship between medians and a particular distance and consequently a distribution.

Rotation matrix

In 2D

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

for example

$$R\left(-\frac{\pi}{4}\right) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

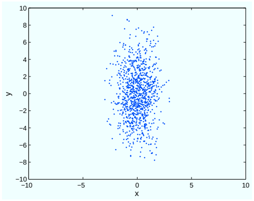
of course this generalizes to higher dimensions.

Relevant observations to us are

- $R^{-1} = R^T$ (orthogonal matrix)
- $|R| = 1$ (no stretching)

Covariances and rotation matrices

Suppose we have two independent Gaussian distributions



$$
\begin{aligned}
p(x) &= \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2}\right\} \\
p(y) &= \frac{1}{\sqrt{2\pi 9}} \exp\left\{-\frac{y^2}{18}\right\} \\
p(x,y) &= \frac{1}{2\pi\sqrt{9}} \exp\left\{-\frac{1}{2}\left(x^2 + \frac{y^2}{9}\right)\right\}
\end{aligned}
$$

or in matrix form

$$
p(x,y) = \frac{1}{2\pi\sqrt{|\mathbf{D}|}} \exp\left\{-(1/2)\begin{bmatrix} x \\ y \end{bmatrix}^T \mathbf{D}^{-1} \begin{bmatrix} x \\ y \end{bmatrix}\right\}
$$

where

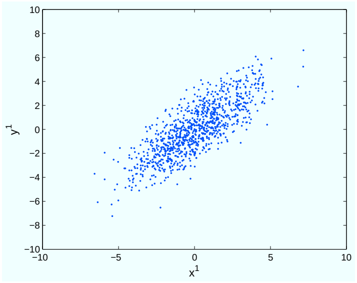
$$
D = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}
$$

Covariances and rotation

Suppose I sample $\begin{bmatrix} x \\ y \end{bmatrix}$ again independently from the same distribution. But then I use a rotation matrix to produce a different sample

$$\begin{bmatrix} x^1 \\ y^1 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}}_{A} \begin{bmatrix} x \\ y \end{bmatrix}$$

and we plot (x^1, y^1)



Covariances and rotation

We performed a change of variables[5] so what is

$$p(x^1, y^1)$$

mapping back to original coordinate system is done by inverse

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix}$$

and we can write

$$p(x^1, y^1) \propto \exp\left\{ -\frac{1}{2} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} \left(\mathbf{ADA}^T\right)^{-1} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} \right\}$$

and after computing normalization constant

$$p(x^1, y^1) = \frac{1}{2\pi\sqrt{|\mathbf{ADA}^T|}} \exp\left\{ -\frac{1}{2} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix}^T (\mathbf{ADA}^T)^{-1} \begin{bmatrix} x^1 \\ y^1 \end{bmatrix} \right\}$$

[5]if you remember Jacobians from your probability class, it is a multiplicative constant that will fall out of $\int_{x^1, y^1} p(x^1, y^1) dxdy = 1$

Covariances in the context of multivariate Gaussian

A Gaussian distribution

$$p(\mathbf{x}|\mu, \Sigma) \propto \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right\}$$

can be thought of in terms of some underlying Gaussian with diagonal covariance and a rotation matrix A

$$p(\mathbf{z}|\mathbf{d}) \propto \prod \exp\left\{-\frac{\mathbf{z}_i^2}{2\mathbf{d}_i}\right\}$$

where

$$\mathbf{x} = \mathbf{A}\mathbf{z} + \mu$$

Taking covariance apart

Suppose someone gives us a covariance matrix Σ.

Can we get rotation matrix A and variances $\mathbf{d}$?

Of course: we perform eigen-decomposition of the matrix (eig in matlab)

$$\Sigma = \mathbf{VDV}^T$$

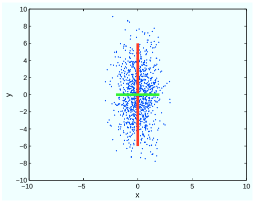
where V contains eigenvectors of Σ and D is a diagonal matrix of eigenvalues of Σ.

And hence our A is a matrix of eigenvectors of Σ and $\mathbf{d}$ are eigenvalues of Σ.

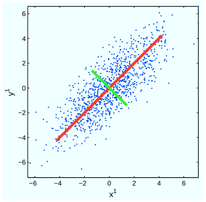
Principal components

Now that we have a list of variances **d** we can say which of the **z** coordinates are the most variable.

Consequently we can translate observations about the most variable **z** coordinates into observations about the most variable *directions* in the space of the original data **x**.



space of **z**



space of **x**

You order components based on their variability: the first component is the most variable.

About plotting PCA

Suppose you reordered coordinates[6] of **z** according to descending order of **d**

For PC1 (largest variance) in **x** space: you would draw a line from

$$A \begin{bmatrix} -2\sqrt{d_1} \\ 0 \\ 0 \\ \vdots \end{bmatrix} \text{ to } A \begin{bmatrix} 2\sqrt{d_1} \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

For PC2: from $A \begin{bmatrix} 0 \\ -2\sqrt{d_2} \\ 0 \\ 0 \\ \vdots \end{bmatrix}$ to $A \begin{bmatrix} 0 \\ 2\sqrt{d_2} \\ 0 \\ 0 \\ \vdots \end{bmatrix}$

[6]of course when you reorder coordinates of **z** you have to reorder A as well

Covariances in general

Covariance is defined regardless of the underlying distribution

$$\mathrm{Cov}[X] = E[(X - E[X])(X - E[X])^T]$$

we can estimate it from the empirical distribution of the data (i.e. sample that we have in hand) as

$$\frac{1}{n-1}(\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T$$

note that this is an outer product, more explicitly

$$\frac{1}{n-1}\begin{bmatrix} (\mathbf{x}_1 - \mu)^T(\mathbf{x}_1 - \mu) & (\mathbf{x}_1 - \mu)^T(\mathbf{x}_2 - \mu) & \dots & (\mathbf{x}_1 - \mu)^T(\mathbf{x}_n - \mu) \\ (\mathbf{x}_2 - \mu)^T(\mathbf{x}_1 - \mu) & (\mathbf{x}_2 - \mu)^T(\mathbf{x}_2 - \mu) & \dots & (\mathbf{x}_2 - \mu)^T(\mathbf{x}_n - \mu) \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{x}_n - \mu)^T(\mathbf{x}_1 - \mu) & (\mathbf{x}_n - \mu)^T(\mathbf{x}_2 - \mu) & \dots & (\mathbf{x}_n - \mu)^T(\mathbf{x}_n - \mu) \end{bmatrix}$$

Fast data inspection

If you can put your data in a matrix of real values you can compute PCA.

It is a very quick way to see the top 2 or 3 sources of variability.

You achieve this by projecting your data onto the top k PC components.

$$\hat{\mathbf{z}} = (\mathbf{A}_{:,1:k})^T \hat{\mathbf{x}}$$

Do not get carried away in attaching interpretation to PCs.

PCs are orthogonal by design. Eventually (e.g. after a handful of components) this starts to bear on the composition of an eigenvector.

Dimensionality reduction and distortion

Expressing your data in terms of the top k principal components is dimensionality reduction.

Suppose $\mathbf{x}$ was in the space of 1000 features (dimensions).

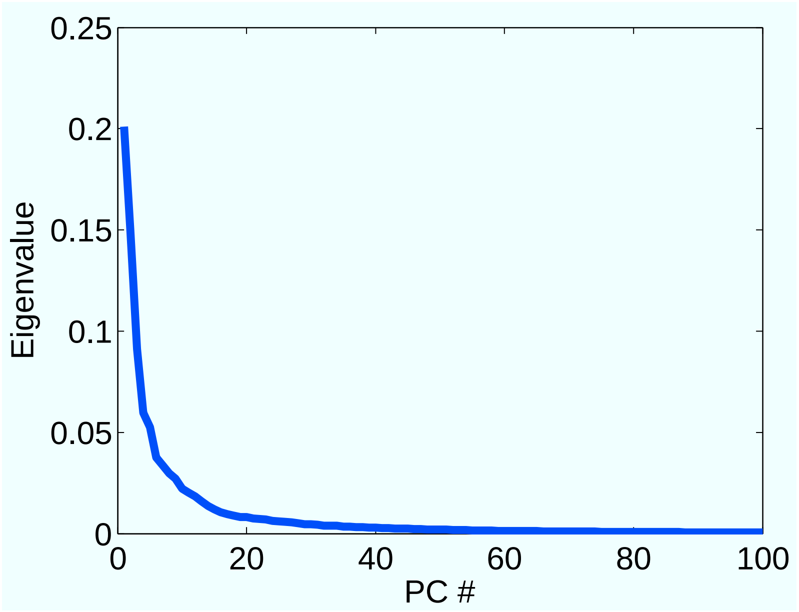
After computing the top $k = 10$ PC components you represent the data in this new lower dimensional space $\hat{\mathbf{z}}$.

Instead of multiplying 1000-long vectors you multiply 10-long vectors – immediate computational savings.

Representing your data in space spanned by the top k principle components yields a distortion that can be explicitly computed.

$$\frac{\|\mathbf{x} - \mathbf{A}_{:,1:k}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} \leq \frac{\sum_{i=k+1}^{n} d_i}{\sum_{i=1}^{n} d_i}$$

Typical spectrum of covariance matrix



A more intelligent way to compute PCA

If the number of features is large you do not want to compute the whole covariance matrix.

Assume that you centered the data, then the covariance estimate is

$$\frac{1}{n-1}XX^T$$

There is another matrix decomposition that operates on rectangular matrices: singular vector decomposition (SVD)

PCA and SVD

Using SVD we can compute the decomposition of X

$$X = USV^T$$

where U and V are again orthonormal matrices and S is diagonal

$$XX^T = USV^T VS^T U^T = US^2 U^T.$$

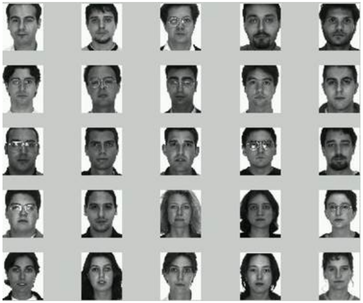
So computing SVD on a potentially skinny X (esp. if $p \gg n$) can be used to obtain the U and S^2 matrices that correspond to our A and D at a fraction of the cost.

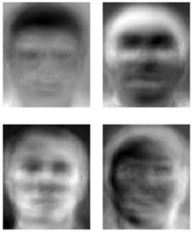
Further efficient algorithms for computing SVDs based on Lanzos method (a variant of power method) can compute the top k singular vectors/values.

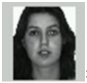
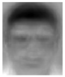
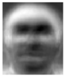
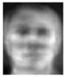
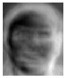
EigenFace, EigenGene, EigenSNP, EigenEverything

Once you get a hold of PCA it is hard to let go, recent examples

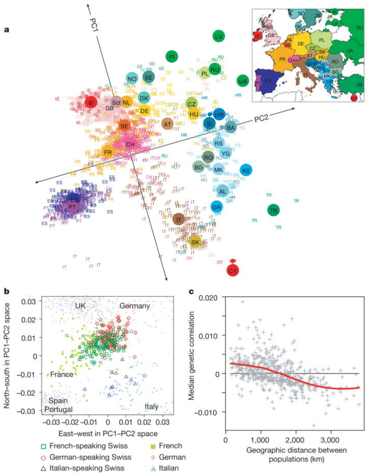
One well known application is Eigen-faces method (remember images get unrolled into vectors):


Data


Top 4 eigen-faces

 = 0.45  -0.2*  +0.15*  -0.12* 

Top 2 principal components of genome covariance recapitulate geography



European geography shaped distribution of European genomes.

PCA on ... pancake recipes [7]

Every pancake recipe, up to a slight approximation, is a scaled version of this eigen-pancake recipe.
Data: pancake recipes on the internet
Top Eigen-pancake-recipe:

- 1 1/2 cups (190g) flour
- 2 tablespoons (25g) sugar
- 2 teaspoons (10g) baking powder
- 1/2 teaspoon (3g) salt
- 2 tablespoons (25g) butter
- 11/4 cups (330g) milk
- 2 small or 1 extra-large (80g) eggs

[7] Cooking for Geeks: Real Science, Great Hacks, and Good Food by Jeff Potter