Contents

1 Images and Basis Functions 4
1.1 Basis Functions, and Linear Algebra 5
1.2 Principle Component Analysis 7
1.3 Sparsity and Over-Complete Bases 13
1.4 Dictionaries and Matched Filter Interpretation. 16
1.5 K-means and Dictionaries 17
1.6 Deeper Understanding; Soft k-means and Mixtures of Gaussians 18

1 Images and Basis Functions

In Section (1.1) we discuss how images can be represented in terms of basis functions. This builds on the Fourier theory introduced in the previous lecture. SP!! Get material from LinearFilteringKokkinos.Pages 24 to 32. Need the mathematics (not images). and Pages 39-43.

In Section (1.2) we describe PCA and SVD. This is a way to learn basis functions from natural images. Shift-invariance means that these are often like sinusoids. A problem is that the number of basis functions is foxed (dimension of the image).

In Section (1.3) we describe how we can get a sparse basis – and the miracle of L_1 sparsity. Material from the SeyounPowerpoints – pages 20-24. SP!!

In Section (1.4) we use k-means (and other methods) to get dictionaries of ultra-sparse basis functions. This gives matched filters. We also discuss mini-epitomes.

Question – what about Independent Component Analysis??

1.1 Basis Functions, and Linear Algebra

This is a quick reminder about orthogonal basis functions. It needs to be fleshed out with some words.

Let $\mathbf{u} \in \mathbb{R}^N$
Basis: N linearly independent vectors $\{\mathbf{v}_i\}, i = 1, ..., N$
Expansion on basis: $\mathbf{u} = \sum_i c_i \mathbf{v}_i$
Orthonormal basis: $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases}$
Expansion coefficients: $\langle \mathbf{v}_i, \mathbf{u} \rangle = c_i$
Expansion: $\mathbf{u} = \sum_i \langle \mathbf{v}_i, \mathbf{u} \rangle \mathbf{v}_i$
Canonical basis (an example is the three-dimensional coordinate system of Euclidean space).

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{1}$$

$$\mathbf{u} = \sum_i u_i \mathbf{e}_i = \sum_i \langle \mathbf{e}_i, \mathbf{u} \rangle \mathbf{e}_i \tag{2}$$

Canonical basis for 2D signals
Canonical basis for signals: expansion
Signal expansion: $g[n] = \sum_k c_k d_k[n]$
Identify terms: $g[k] = c_k$
Rewrite: $d_k[n] = d[n-k]$
$d[n] =$
Sifting property: $g[n] = \sum_k g[k] d[n-k]$
Canonical basis for signals and LTI filters $d[n] \rightarrow h[n]$ $d[n-k] \rightarrow h[n-k]$
Any signal: $g[n] = \sum_k g[k] d[n-k]$
By linearity: $\psi(g) = \sum_k g[k] h[n-k] = g[n] * h[n]$
Output of any LSI filter for any input: convolution of input with filter's impulse response
Convolution - discrete and continuous
2D convolution sum:

$$\begin{aligned} f[n_1, n_2] &= \sum_{k_1, k_2} g[k_1, k_2] h[n_1 - k_1, n_2 - k_2] \\ &= g[n_1, n_2] * h[n_1, n_2] \end{aligned} \tag{3}$$

2D convolutional integral:

$$\begin{aligned} f(x, y) &= \int \int g(a, b) h(x - a, y - b) da db \\ &= g(x, y) * h(x, y) \end{aligned} \tag{4}$$

Page 32
page 39: Linear algebra reminder: eigenvectors $\mathbf{M} : N \times N$
Eigenvectors: $\mathbf{M}\mathbf{v}_i = \lambda_i \mathbf{v}_i, i = 1, ..., N$
Full-rank, real and symmetric: eigenbasis
$\mathbf{u} = \sum_k \underbrace{\langle \mathbf{v}_k, \mathbf{u} \rangle}_{c_k} \mathbf{v}_k$

$$\mathbf{M}\mathbf{u} = \sum_k c_k \mathbf{M}\mathbf{v}_k = \sum_k \underbrace{c_k \lambda_k}_{c'_k} \mathbf{v}_k \tag{5}$$

$$\mathbf{M}(\mathbf{M}\mathbf{u}) = \sum_k \underbrace{c_k \lambda_k^2}_{c''_k} \mathbf{v}_k \tag{6}$$

Page 40: Eigenvectors and eigenfuctions
Eigenvector: $\mathbf{Mv} = \lambda \mathbf{v}$
Eigenfunction: $\psi(b) = \lambda b$
Input: $f = \sum_k a_k b_k$
Output: $\psi(f) = \sum_k a_k \psi(b_k)$
$f \leftrightarrow \{a_k\}$ $\psi(f) \leftrightarrow \{a_k \lambda_k\}$
Page 41: Eigenfunctions for LTI filters
LTI filter: $\psi(g)[n] = \sum_k h[k] g[n-k]$
Let's guess: $b_\omega[n] = \exp(j\omega n) = \cos(\omega n) + j \sin(\omega n)$ It works

$$\begin{aligned} \psi(b_\omega)[n] &= \sum_k h[k] b_{omega}[n-k] \\ &= \sum_k h[k] \exp(j\omega[n-k]) \\ &= \sum_k h[k] \exp(-j\omega k) \exp(j\omega n) \\ &= H(\omega) b_\omega[n] \end{aligned} \tag{7}$$

Frequency response: $H(\omega) = \sum_k h[k] \exp(-j\omega k)$
Page 42: Expansion on harmonic basis
From orthonormality: $\mathbf{u} = \sum_k \langle \mathbf{u}, \mathbf{v}_k \rangle \mathbf{v}_k$
Inner product for complex functions: $\langle f, g \rangle = \sum_n f[n] g^*[n]$
Discrete-time: $F(\omega) = \langle f, b_\omega \rangle = \sum_n f[n] e^{-j\omega n}$
Continuous-time: $F(\omega) = \int_{-\infty}^{\infty} f(t) \exp(-j\omega t) dt$
Page 43: Change of basis
Canonical expansion: $\mathbf{u} = \sum_k u_k \mathbf{e}_k$
Eigenbasis expansion: $\mathbf{u} = \sum_k \underbrace{\langle \mathbf{u}, \mathbf{v}_k \rangle}_{c_k} \mathbf{v}_k$ Rotation matrix from eigenbasis:

1.2 Principle Component Analysis

Learning the basis functions. Mathematics of PCA and SVD. If images are shift-invariant, then the eigenvectors are sinusoids.If the images are aligned – e.g., faces – then we get eigenfaces.

Probably shorten and put some material in an Appendix.

Principal Component Analysis (PCA)

One way to deal with the curse of dimensionality is to project data down onto a space of low dimensions, see figure (1). There are a number of different techniques for doing this. The most basic method is Principal Component Analysis (PCA) .

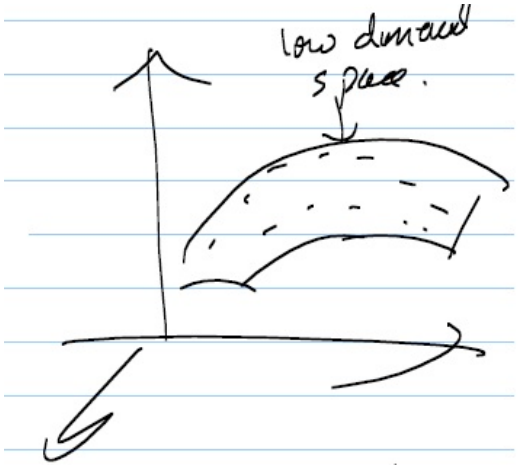

Figure 1:

We will use the following convention:

$\vec{\mu}^T\vec{\mu}$ is a scalar $\mu_1^2+\mu_2^2+\cdots+\mu_D^2$

$\vec{\mu}\,\vec{\mu}^T$ is a matrix $\begin{pmatrix} \mu_1^2 & \mu_1\mu_2 & \mu_1\mu_3 & \cdots \\ \vdots & \mu_2^2 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$

The data samples are $\vec{x}_1,\ldots,\vec{x}_N$ in a D-dimension space. First, compute their mean

$$\vec{\mu}=\frac{1}{N}\sum_{i=1}^{N}\vec{x}_i$$

and their covariance

$$\mathbf{K}=\frac{1}{N}\sum_{i=1}^{N}(\vec{x}_i-\vec{\mu})(\vec{x}_i-\vec{\mu})^T$$

Next, compute the eigenvalues and eigenvector of $\mathbf{K}$:

Solve $\mathbf{K}\vec{e}=\lambda\vec{e}$

$\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_N$

Note: $\mathbf{K}$ is a symmetric matrix- so eigenvalues are real, eigenvectors are orthogonal. $\vec{e}_{nu}\cdot\vec{e}_{\mu}=1$ if $\mu=\nu$, and $=0$ otherwise. Also, by construction, the matrix $\mathbf{K}$ is positive semi-definite, so $\lambda_N\geq 0$ (i.e. no eigenvalues are negative).

PCA reduces the dimension by projection the data onto a space spanned by the eigenvectors $\vec{e}_i$ with $\lambda_i > T$, where T is a threshold. Let M eigenvectors be kept. Then, project data $\vec{x}$ onto the subspace spanned by the first M eigenvectors, after subtracting out the mean. Formally:

$$\vec{x} - \vec{\mu} = \sum_{\nu=1}^{D} a_\nu \vec{e}_\nu,$$

where the coefficients $\{a_\nu\}$ are given by

$$a_\nu = (\vec{x} - \vec{\mu}) \cdot \vec{e}_\nu$$

Note: orthogonality means $\vec{e}_\nu \cdot \vec{e}_\mu = \delta_{\nu\mu}$, which denotes the Kronecker delta.

Hence:

$$\vec{x} = \vec{\mu} + \sum_{\nu=1}^{D} \langle (\vec{x} - \vec{\mu}) \cdot \vec{e}_\nu \rangle \vec{e}_\nu$$

and there is no dimension reduction (no compression).

Then, approximate

$$\vec{x} \approx \vec{\mu} + \sum_{\nu=1}^{M} \langle (\vec{x} - \vec{\mu}) \cdot \vec{e}_\nu \rangle \vec{e}_\nu$$

This Projects the data into the M-dimension subspace of form:

$$\vec{\mu} + \sum_{\nu=1}^{M} b_\nu \vec{e}_\nu$$

See a 2-dimensions example in figure (2). The eigenvector of $\mathbf{K}$ corresponds to the second order movements of the data.Section3ImagePatches/figures/

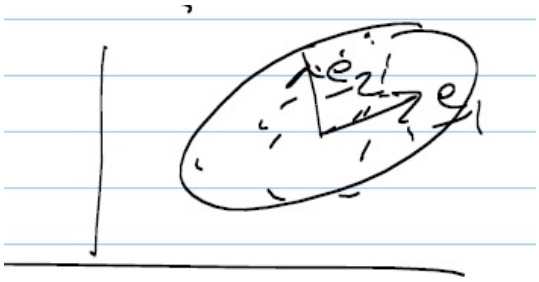

Figure 2: In two-dimensions, the eigenvectors give the principal axes of the data.

If the data lies (almost) on a straight line, then $\lambda_1 \gg 0, \lambda_2 \approx 0$, see figure (3).

PCA and Gaussian Distribution

PCA is equivalent to performing ML estimation of the parameters of a Gaussian distribution

$$p(\vec{x} \mid \vec{\mu}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{D/2}\sqrt{\det \mathbf{\Sigma}}} e^{-\frac{1}{z}(\vec{x}-\vec{\mu})^T \sum^{-1} (\vec{x}-\vec{\mu})}$$

to get $\hat{\vec{\mu}}, \hat{\mathbf{\Sigma}}$ by performing ML on $\prod_i p(\vec{x}_i \mid \vec{\mu}, \mathbf{\Sigma})$, and then throw away the directions where the standard deviation is small. ML gives $\vec{\mu} = \frac{1}{N}\sum_{i=1}^{N} \vec{x}_i$ and $\Sigma = \frac{1}{N}\sum_{i=1}^{N} (\vec{x}_i - \vec{m}u)(\vec{x}_i - \vec{\mu})^T$. See Bishop's book for probabilistic PCA.

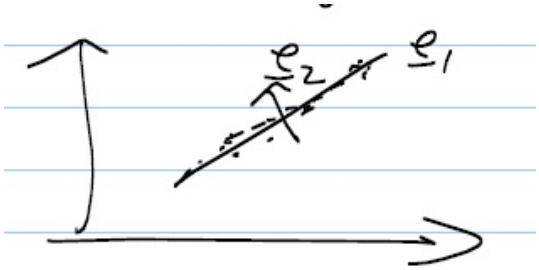

Figure 3: In two-dimensions, if the data lies along a line then $\lambda_1 > 0$ and $\lambda_2 \approx 0$.

When is PCA appropriate?

PCA is almost always a good technique to try, because it is so simple. Obtain the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ and plot $f(M) = \sum_{i=1}^{M} \lambda_i / \sum_{i=1}^{N} \lambda_i$, to see how $f(M)$ increases with M and takes maximum value 1 at $M = D$. PCA is good if $f(M)$ asymptotes rapidly to 1. This happens if the first eigenvalues are big and the remainder are small. PCA is bad if all the eigenvalues are roughly equal. See examples of both cases in figure (4).

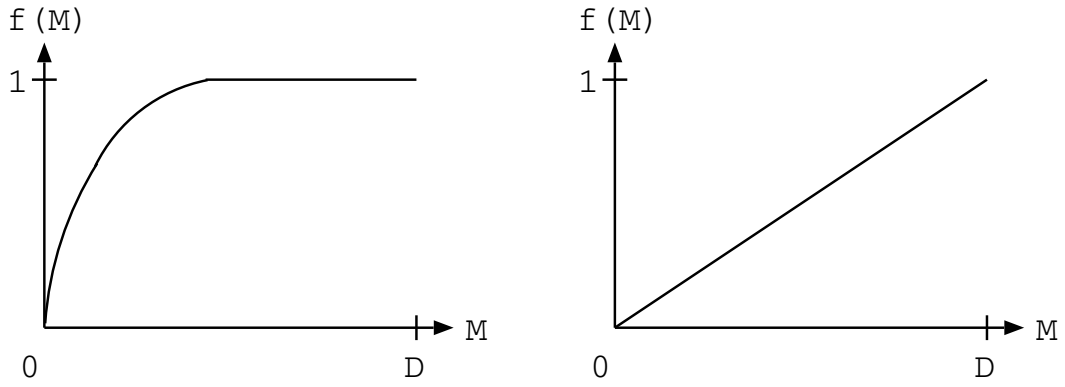

Figure 4: Left: eigenvalues asymptote rapidly to 1, this is good. Right: all eigenvalues are equally important, PCA is not appropriate here.

PCA would be bad in an example in which the data is a set of strings
$(1, 0, 0, 0, \dots) = \vec{x}_1$
$(0, 1, 0, 0, \dots) = \vec{x}_2$
$(0, 0, 0, 0, \dots, 0, 1) = \vec{x}_N$

Then, it can be computed that there is one zero eigenvalue of PCA. But all the other eigenvalues are not small. In general, PCA works best if there is a linear structure to the data. It works poorly if the data lies on a curved surface and not on a flat surface.

Interpretation of PCA

What is PCA doing? There are two equivalent ways to interpret PCA: (i) minimize the projection error, and (ii) maximize the variance of the projection.

Consider the variance of the data $\frac{1}{N} \sum_{i=1}^{N} (\vec{x}_i - \vec{\mu})^2$. It is independent of the projection. We can express

$(\vec{x}_i - \vec{\mu})^2 = \sum_{i=1}^{D}\{(\vec{x}_i - \vec{\mu}) \cdot \vec{e}_\nu\}^2$, where the $\vec{e}_\nu$ are the eigenvectors of the correlation. Hence,

$$\frac{1}{N}\sum_{i=1}^{N}(\vec{x}_i - \vec{\mu})^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{\nu=1}^{M}\{(\vec{x}_i - \vec{\mu}) \cdot \vec{e}_\nu\}^2 + \frac{1}{N}\sum_{i=1}^{N}\sum_{\nu=M+1}^{D}\{(\vec{x}_i - \vec{\mu}) \cdot \vec{e}_\nu\}^2,$$

where the left-hand side is the variance of the data, the first term of the right-hand side is the variance of the data within the plane $\vec{e}_1, \cdots, \vec{e}_M$, and the last term is the projection error.

When $\vec{x}_i$ is projected to a point $\vec{x}_{i,p} = \vec{\mu} + \sum_{\nu=1}^{M}\{(\vec{x}_i - \vec{\mu}) \cdot \vec{e}_\nu\}\vec{e}_\nu$, it has a projection error $\sum_{\nu=M+1}^{D}\{(\vec{x}_i - \vec{\mu}) \cdot \vec{e}_\nu\}^2$. The sum of the projection error and the variance of projection are constant. So maximizing on is equivalent to minimizing the other.

Also, this relationship can be expressed in terms of eigenvalues. It reduces to

$$\sum_{\nu=1}^{N}\lambda_\nu = \sum_{\nu=1}^{M}\lambda_\nu + \sum_{\nu=M+1}^{D}\lambda_\nu$$

To see this, $\frac{1}{N}\sum_{i=1}^{N}(\vec{x}_i - \vec{\mu}) \cdot (\vec{x}_i - \vec{\mu}) = \text{Trace}(\mathbf{C}) = \sum_{\nu=1}^{D}\lambda_\nu$. The variance of the projection is $\sum_{\nu=1}^{M}\lambda_\nu$, by similar reasoning. Hence, the projection error is $\sum_{\nu=M+1}^{D}\lambda_\nu$.

Cost Function for PCA

The cost function for PCA can be defined as

$$J(\vec{M}, \{a\}, \{e\}) = \sum_{k=1}^{N}||(\vec{\mu} + \sum_{i=1}^{M}a_{ki}\vec{e}_i) - \vec{x}_k||^2),$$

where The $\{a_{ki}\}$ are projection coefficients.

Minimize J w.r.t. $\vec{M}, \{a\}, \{e\}$ Data $\{\vec{x}_k : k = 1 \text{ to N}\}$
Intuition: find the M-dimensional subspace s.t. the projections of the data onto this subspace have minimal error, see figure (5).

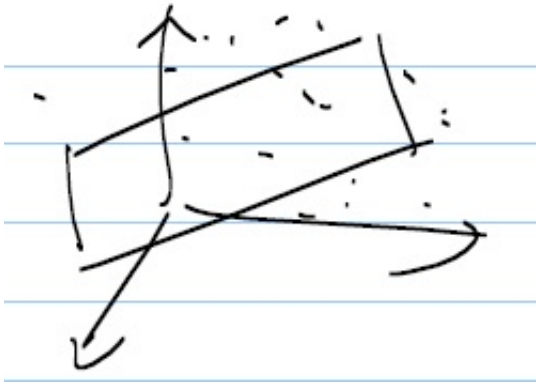

Figure 5: PCA can be obtained as the projection which minimizes the least square error of the residuals.

Minimizing J, gives the $\{\hat{\vec{e}}_i\}$'s to be the eigenvectors of the covariance matrix
$\vec{K} = \frac{1}{N}\sum_{k=1}^{N}(\vec{x}_k - \vec{\mu})(\vec{x}_k - \vec{\mu})^T$
$\vec{\mu} = \frac{1}{N}\sum_{k=1^N}\vec{x}_k$
$\hat{a}_{ki} = (\vec{x}_k - \hat{\vec{\mu}}) \cdot \vec{e}_i$ the projection coefficients.

To fully understand why PCA minimizes or maximizes these terms we must express the criterion slightly differently. Then we use Singular Value Decomposition (SVD), which is advanced material of this lecture.

We can re-express the criteria as

$$J[\vec{\mu},\{a\},\{e\}] = \sum_{k=1}^{N}\sum_{b=1}^{D}\{(\mu_b - x_{bk}) + \sum_{i=1}^{M} a_{ki}e_{ib})\}^2,$$

where b denotes the vector components.

This is an example of a general class of problem.

Let $E[\Psi, e] = \sum_{a=1,k=1}^{a=D,k=N}(\tilde{x}_{ak} - \sum_{\nu=1}^{M}\Psi_{a\nu}\Phi_{\nu k})^2$.

Goal: minimize $E[\Psi, e]$ w.r.t. Ψ, e.

This is a bilinear problem, that can be solved by SVD.

Note: $\tilde{x}_{ak} = x_{ak} - \mu_a$ is the position of the point, relative to the mean.

Singular Value Decomposition SVD

We can express any $N \times D$ matrix $\vec{X}$, x_{ak} in form

$$\mathbf{X} = \mathbf{E}\,\mathbf{D}\,\mathbf{F}$$

$$x_{ak} = \sum_{\mu,\nu=1}^{M} e_{a\mu}d_{\mu\nu}f_{\nu k}$$

where $\mathbf{D} = \{d_{\mu\nu}\}$ is a diagonal matrix ($d_{\mu\nu} = 0, \mu \neq \nu$). Note: $\mathbf{X}$ is not a square matrix (unless $D = N$). So it has no eigenvalues or eigenvectors.

$\mathbf{D} = \begin{pmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda_N} \end{pmatrix}$, where the $\{\lambda_i\}$ are eigenvalues of $\mathbf{X}\,\mathbf{X}^T$(equivalently of $\mathbf{X}^T\,\mathbf{X}$)

$\mathbf{E} = \{e_{a\mu}\}$ are eigenvectors of $(\mathbf{X}\,\mathbf{X}^T)_{ab}$,

$\mathbf{F} = \{f_{\nu k}\}$ are eigenvectors of $(\mathbf{X}^T\,\mathbf{X})_{kl}$,

μ, ν label the eigenvectors.

Note: For $\bar{\mathbf{X}}$ defined on previous page, we get that $(\tilde{\mathbf{X}}\,\bar{\mathbf{X}}^T) = \sum_{k=1}^{N}(\vec{x}_k - \underline{\mu})(\vec{x}_k - \underline{\mu})^T$. Also note that if $(\mathbf{X}\,\mathbf{X}^T)\vec{e} = \lambda\vec{e}$, then $(\mathbf{X}^T\,\mathbf{X})(\mathbf{X}^T\vec{e}) = \lambda(\mathbf{X}^T\vec{e})$.

This relates the eigenvectors of $\mathbf{X}\,\mathbf{X}^T$ and of $\mathbf{X}^T\,\mathbf{X}$ (calculate the eigenvectors for the smallest matrix, then deduce those of the bigger matrix – usually $D < N$).

Minimize:

$$E[\psi, e] = \sum_{a=1,k=1}^{a=D,k=N}(\tilde{x}_{ak} - \sum_{\nu=1}^{M}\psi_{a\nu}\phi_{\nu k})^2$$

We set $\begin{cases} \psi_{a\nu} = \sqrt{\delta_{\nu\nu}}e_a^\nu \\ \phi_{\nu k} = \sqrt{\delta_{\nu\nu}}f_k^\nu \end{cases}$

Take M biggest terms in the SVD expansion of $\mathbf{x}$.

But there is an ambiguity.

$$\sum_{\nu=1}^{M}\psi_{a\nu}\phi_{\nu k} = (\psi\phi)_{ak} = (\psi\mathbf{A}\mathbf{A}^{-1}\phi)_{ak}$$

for any M $\times$ M invertible matrix $\mathbf{A}$

$\psi \to \psi\mathbf{A}$

$\phi \to \mathbf{A}^{-1}\phi$

For the PCA problem, we have constants that the projection directions one orthogonal unit eigenvectors. This gets rid of the ambiguity.

Relate SVD to PCA

Linear algebra can be used to relate SVD to PCA. Start with an $n \times m$ matrix $\mathbf{X}$.
$\mathbf{X}\mathbf{X}^T$ is a symmetric $n \times n$ matrix
$\mathbf{X}^T\mathbf{X}$ is a symmetric $m \times m$ matrix
Note that $(\mathbf{X}\mathbf{X}^T)^T = \mathbf{X}\mathbf{X}^T$.

By standard linear algebra,

$$\mathbf{X}\mathbf{X}^T\vec{e}^{\mu} = \lambda^{\mu}\vec{e}^{\mu},$$

with n eigenvalues λ^{μ} and eigenvectors $\vec{e}^{\mu}$. The eigenvectors are orthogonal $\vec{e}^{\mu} \cdot \vec{e}^{\nu} = \delta^{\mu\nu}$ ($= 1$ if $\mu = \nu$, $= 0$ if $\mu \neq \nu$).

Similarly,

$$\mathbf{X}\mathbf{X}^T\vec{f}^{\nu} = \tau^{\nu}\vec{f}^{\nu},$$

with m eigenvalues τ^{ν} and eigenvectors $\vec{f}^{\nu}$, where $\vec{f}^{\mu} \cdot \vec{f}^{\nu} = \delta^{\mu\nu}$.

The $\{\vec{e}^{\mu}\}$ and $\{\vec{f}^{\nu}\}$ are related because

$$(\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\vec{e}^{\mu}) = \lambda^{\mu}(\mathbf{X}^T\vec{e}^{\mu})$$

$$(\mathbf{X}\mathbf{X}^T)(\mathbf{X}\vec{f}^{\mu}) = \tau^{\mu}(\mathbf{X}\vec{f}^{\mu})$$

Hence, $\mathbf{X}^T\vec{e}^{\mu} \propto \vec{f}^{\mu}$, $\mathbf{X}\vec{f}^{\mu} \propto \vec{e}^{\mu}$ and $\lambda^{\mu} = \tau^{\mu}$. If n >m , then there are n eigenvectors $\{\vec{e}_{\mu}\}$ and m eigenvectors $\{\vec{f}_{\mu}\}$. So, several $\{\vec{e}_{\mu}\}$ relate to the same $\vec{f}_{\mu}$.

Claim: we can express

$$\mathbf{X} = \sum_{\mu} \alpha^{\mu}\vec{e}^{\mu}\vec{f}_{\mu}^{T}$$

$$\mathbf{X}^T = \sum_{\mu} \alpha^{\mu}\vec{f}^{\mu}\vec{e}_{\mu}^{T}$$

(For some α^{μ}. We will solve for all α^{μ} later.)

Verify the claim:

$$\mathbf{X}\vec{f}^{\nu} = \sum_{\mu} \alpha^{\mu}\vec{e}^{\mu}\vec{f}_{\mu}^{T}\vec{f}_{\nu} = \sum_{\mu} \alpha^{\mu}\delta_{\mu\nu}\vec{e}^{\mu} = \alpha^{\nu}\vec{e}^{\nu}$$

$$\mathbf{X}\mathbf{X}^T = \sum_{\mu,\nu} \alpha^{\nu}\vec{e}^{\nu}\vec{f}_{\nu}^{T}\alpha^{\mu}\vec{f}^{\mu}\vec{e}_{\mu}^{T} = \sum_{\mu,\nu} \alpha^{\nu}\alpha^{\mu}\vec{e}^{\nu}\delta_{\mu\nu}\vec{e}_{\mu}^{T} = \sum_{\mu} (\alpha^{\mu})^2\vec{e}^{\mu}\vec{e}_{\mu}^{T}$$

Similarly, $\mathbf{x}^T\mathbf{x} = \sum_{\mu}(\alpha^{\mu})^2\vec{f}^{\mu}\vec{f}_{\mu}^{T}$. So, $(\alpha^{\mu})^2 = \lambda^{\mu}$. (Because we can express any symmetric matrix in form $\sum_{\mu} \lambda_{\mu}\vec{e}^{\mu}\vec{e}_{\mu}^{\ T}$, where λ^{μ} are the eigenvalues and $\vec{e}^{\mu}$ are eigenvectors.)

$\mathbf{X} = \sum_{\mu} \alpha^{\mu}\vec{e}^{\mu}\vec{f}_{\mu}^{\ T}$ is the SVD of $\mathbf{X}$

In coordinates:
$x_{ai} = \sum_{\mu} \alpha^{\mu}e_a^{\mu}f_i^{\mu}$
$x_{ai} = \sum_{\mu,\nu} e_a^{\mu}\alpha^{\mu}\delta_{\mu\nu}f_i^{\nu}$
$\mathbf{x} = \mathbf{EDF}$
$E_{a\mu} = e_a^{\mu}, \mathbf{D}_{\mu\nu} = \alpha^{\mu}\delta_{\mu\nu}, F_{\nu i} = f_i^{\nu}$.

1.3 Sparsity and Over-Complete Bases

From Arbib Chapter

This section considers receptive field models from different perspectives. This includes the use of *sparsity* to suggest receptive field properties based on the statistics of natural images and also the idea of *matched filters* which revert to an older idea of receptive fields as feature detectors [**?**]. Sparsity was proposed by Barlow [**?**] as a general principle for modeling the brain based on the observation that typically only a small number of neurons are active. It was developed as a way to predict receptive field properties by Olshausen and Field [**?**]. It is natural to ask whether the receptive fields of cells encode basis functions which somehow capture the typical structure of images and represent it in a form which is suitable for later processing.

Our starting point is the idea that images, and particularly local regions of images. can be represented as a linear combination of basis functions $I(\vec{x}) = \sum_i \alpha_i \vec{b_i}(\vec{x})$, see equation (**??**).

Over-Complete Bases and Sparsity.

This section introduces the idea of *over-complete* basis functions and *sparsity*. To motivate this idea, consider an image which consists partly of regions where the intensity varies spatially smoothly and others where the intensity is more jagged and consists of a number of bright spots, or *impulses*. The smoothly varying regions of the image can be represented by fourier analysis efficiently, in the sense that we can approximate the intensity by only a small number of weighted sinusoids (in other words, the fourier transform of the image is peaked at a limited number of frequencies). By contrast, the impulses are not well described by fourier analysis because the fourier transform is not zero for all frequencies (the fourier transform of an impulse at $\vec{x}_0$ is $\exp\{i\vec{\omega} \cdot \vec{x}_0\}$, so the amplitude spectrum is constant at all frequencies). Instead it would be better to represent the spikes in terms of a basis of impulse functions, but this representation would be very inefficient for the smoothly varying parts of the image. In short, different types of basis functions are suitable for different regions of the image. This suggests a strategy where we seek a representation in terms of an over-complete set of basis functions, in this case sinusoids and impulse functions, and a criterion which selects an efficient representation so that only a small number of basis functions are activated for each image. This requirement is called *sparsity*.

More formally, we represent an image, or local image region, by:

$$I(\vec{x}) = \sum_{i=1}^{N} \alpha_i b_i(\vec{x}),$$

where the $\{b_i\}$ are the basis functions (which are the same for all images, and could include sinusoids and impulse functions) and the $\{\alpha_i\}$ are the coefficients of the bases (which depend on the image). The number N of bases is much bigger than the dimension of the image, and hence the bases are *over-complete*. This differs from fourier analysis where the data (e.g., an image) is expressed in terms of a set of basis functions which are mutually orthogonal, which enables the coefficients α to for each image to be estimated by $\alpha_i = \sum_{\vec{x}} b_i(\vec{x}) \cdot I(\vec{x})$. Over-completeness implies that there are many ways to represent the image in terms of these basis functions (by different choices of the α's) and we need an additional criterion to select the α's. The *sparsity* criterion proposes that we favor representations which make $\sum_{i=1}^{N} |\alpha_i|$ small, which penalize the weights of the basis functions and encourages most coefficients to be 0.

More precisely, we represent an image $\vec{I}$ by the approximation $\sum_{i=1}^{N} \hat{\alpha}_i \vec{b}_i$, where the $\{\hat{\alpha}_i\}$ are chosen to minimize the function:

$$E(\alpha) = \sum_{\vec{x}} (I(\vec{x}) - \sum_{i=1}^{N} \alpha_i b_i(\vec{x}))^2 + \lambda \sum_{i=1}^{N} |\alpha_i|. \tag{8}$$

The first penalizes the error of the approximation and the second term, whose strength is weighted by a parameter λ, penalizes the coefficients $\{\alpha_i\}$. The solution $\hat{\alpha} = \arg\min_\alpha E(\alpha)$ can not not be specified in closed form (unlike the case for orthogonal basis function), but $E(\alpha)$ is a *convex* function of α and efficient algorithms exist for minimizing it to estimate $\hat{\alpha}$. The results of these algorithms can, for example, decompose an image into a sum of sinusoids and a sum of impulse functions.

These ideas give an alternative way to think about the receptive fields of cells in V1. Firstly, observe that V1 has far more cells than the retina or the LGN and so it is has enough neural machinery to implement over-complete bases. Secondly, over-complete bases can be designed for specific image structures of interest (e.g., impulse functions or edges) which enables us to start interpreting the image instead of simply representing it. Thirdly, it relates to the observation that cells in V1 fire *sparsely*, which suggests [**?**] that they are tuned to specific stimuli and may relate to metabolic processes(firing a neuron takes energy which needs to be replenished). Hence the idea that the visual cortex seeks to obtain sparse, and hence presumably more easily interpretable representations, has intuitive appeal.

How does this discussion of over-completeness and sparsity relate to our previous description of V1 cells in terms of Gabor filters? Gabor filters have some of the properties that this approach requires. Families of Gabor filters are built by taking a basic functions and performing transformations on it which give an over-complete basis. Hence they do not specify a unique representation of an image (i.e. any image can be represented many different ways in terms of Gabor functions). These issues, and the relations of Gabors to wavelets, are discussed in more detail in [**?**].

Sparsity and Natural Images.

Sparsity can also be used to derive the properties of receptive fields of cell in V1 if we assume that these cells are designed to be able to represent properties of natural images [**?**], see figure (6)(Left). Hence instead of hypothesizing models of receptive fields (e.g., Gabor filters) we can try to predict these receptive fields from studying images. These predictions do give some justification for Gabor functions but they also suggest other receptive field models which have also been experimentally observed.

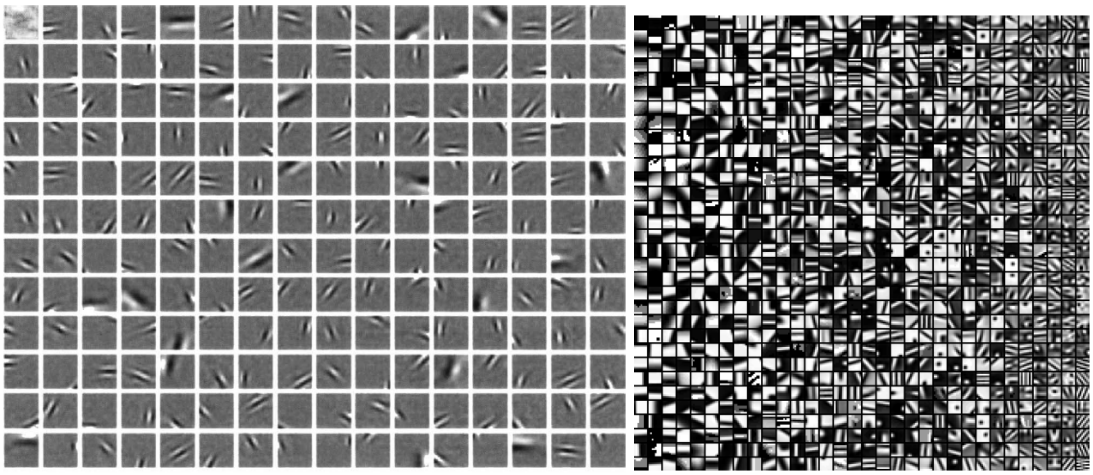

Figure 6: Left: The receptive fields learnt using sparsity [**?**]. Right: receptive fields learnt by matched filters.

This requires learning the basis functions $\{\vec{b}_i\}$ from a set of natural images $\{\vec{I}^\mu : \mu \in \Lambda\}$. This can be found be extending equation (8) to obtain a criteria $E(b, \alpha)$ for fitting basis functions b and coefficients α to the set of images:

$$E(b, \alpha) = \sum_{\mu \in \Lambda} (I^\mu(\vec{x}) - \sum_{i=1}^{N} \alpha_i^\mu b_i(\vec{x}))^2 + \lambda \sum_{\mu \in \Lambda} \sum_{i=1}^{N} |\alpha_i|.$$

We estimate the basis functions $\hat{b}$ and the coefficients $\hat{\alpha}$ by minimizing $E(b, \alpha)$ to obtain:

$$(\hat{b}, \hat{\alpha}) = \arg\min_{(b,\alpha)} E(b, \alpha).$$

Note that the basis functions are the same for all images but the coefficients vary for each image (hence they are indexed by the image μ as well as the basis coefficient i). This minimization is non-convex but there are efficient algorithms to perform it.

This criterion has been applied to natural images (where the $\vec{I}$ represent small image regions) and the resulting basis functions, see figure (6)(left), include filters which look like Gabor functions but they also include other types of filters which are also observed in experiments [**?**].

We note that there are other methods for predicting receptive field properties from natural images using a similar image model, $I(\vec{x}) = \sum_{i=1}^{N} \alpha_i b_i(\vec{x})$, but imposing different assumptions on the form of the bases. In particular, independent component analysis (ICA) gives similar receptive field models [**?**]. Hyvarinen [**?**] explains this by showing that both types of models – L1 sparsity and ICA – both encourage the α_i to be strongly peaked at 0, but can occasionally have large non-zero values.

What happens if we remove the sparsity requirement and instead find the basis functions that minimize $\sum_{\mu \in \Lambda} (I^{\mu}(\vec{x}) - \sum_{i=1}^{N} \alpha_i^{\mu} b_i(\vec{x}))^2$? The basis functions will be the eigenvectors of the correlation matrix of the images and can be found by principal component analysis (PCA). Code for performing PCA is supplied in interactive demo (2c). It can be shown that the principal components of images will typically be sinusoids (provided the images are sufficiently representative of natural images). We return to this issue in section (**??**) when we describe unsupervised ways to learn receptive fields of neurons.

Sparsity and Faces

Grab material from Ethan Gao – or form the Wright, Yi Ma, sparsity paper!!

Sparsity: Interpretation

The miracle of sparsity is illustrated by the following example (here we replace $|\vec{J}|$ by J to simplify the argument).

$$f(\omega; I) = (\omega - I)^2 + \lambda |\omega| \tag{9}$$

What is $\hat{\omega}(I) = \arg\min_{\omega} f(\omega; I)$?

$$f_+(\omega; I) = (\omega - I)^2 + \lambda\omega. \quad \text{For } \omega \geq 0 \tag{10}$$

$$f_-(\omega; I) = (\omega - I)^2 - \lambda\omega. \quad \text{For } \omega \leq 0 \tag{11}$$

$$\frac{df_+}{d\omega} = 2(\omega - I) + \lambda = 0 \Rightarrow \hat{\omega}(I) = \frac{2I - \lambda}{2}, \forall \omega \geq 0 \tag{12}$$

$$\frac{df_-}{d\omega} = 2(\omega - I) - \lambda = 0 \Rightarrow \hat{\omega}(I) = \frac{2I + \lambda}{2}, \forall \omega \leq 0 \tag{13}$$

Hence, we have

$$\hat{w}(I) = \frac{2I - \lambda}{2}, \quad I \geq \tfrac{\lambda}{2} \tag{14}$$

$$\hat{w}(I) = 0, \quad |I| \leq \tfrac{\lambda}{2} \tag{15}$$

$$\hat{w}(I) = \frac{2I + \lambda}{2}, \quad I \leq \tfrac{-\lambda}{2} \tag{16}$$

As we show in Figure 1.3, the use of the $L1$ norm $|\omega|$ biases the solution to $\hat{\omega}(I) = 0$ for small I.

By contrast, $f_2(\omega; I) = (w - I)^2 + \lambda\omega^2$ has minimum at $\omega - I + \lambda\omega = 0$. Therefore,

$$\hat{\omega}(I) = \frac{I}{1 + \lambda} \tag{17}$$

which always smooths I, but doesn't force it to 0.

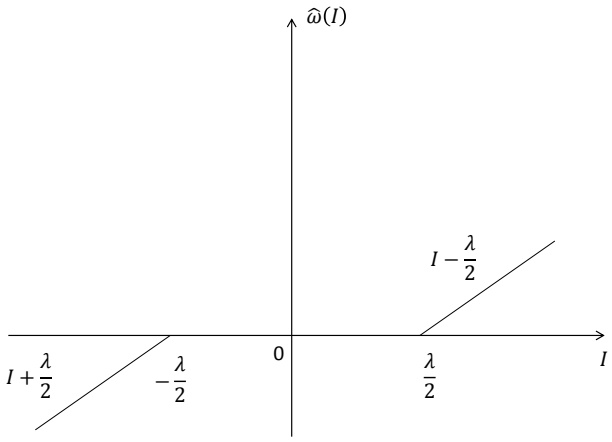

Figure 7: L1 norm is better

1.4 Dictionaries and Matched Filter Interpretation.

This section describes a way to represent each image in terms of a single basis element, or *matched filter*. Examples of matched filters are shown in figure (6)(right). We now describe the details of this approach.

Suppose we have a filter $\vec{B}$ and an input image patch $\vec{I}_p$. We want to find the best fit of the filter to the image by allowing us to transform the filter by $\vec{B} \mapsto a\vec{B} + b\vec{e}$, where $\vec{e} = (1/\sqrt{N})(1, ..., 1)$. This corresponds to scaling the filter by a and adding a constant vector b. If $\vec{B}$ is a derivative filter then, by definition, $\vec{B} \cdot \vec{e} = 0$. We normalize $\vec{B}$ and $\vec{e}$ so that $\vec{B} \cdot \vec{B} = \vec{e} \cdot \vec{e} = 1$.

The goal is to find the best scaling/contrast a and background b to minimize the match:

$$E(a, b) = |\vec{I}_p - a\vec{B} - b\vec{e}|^2.$$

The solution $\hat{a}, \hat{b}$ are given by (take derivatives of E with respect to a and b, recalling that $\vec{B}$ and $\vec{e}$ are normalized):

$$\hat{a} = \vec{B} \cdot I_p, \;\; \hat{b} = \vec{e} \cdot \vec{I}_p.$$

In this interpretation, the filter response is just the best estimate of the contrast a. The estimate of the background b is just the mean value of the image. Finally, the energy $E(\hat{a}, \hat{b})$ is a measure of how well the filter "matches" the input image. Receptive fields learnt by matched filters are shown in figure (6)(right).

The idea of a matched filter leads naturally to the idea of having a"dictionary" of filters $\{\vec{B}^\mu : \mu \in \Lambda\}$, where different filters $\vec{B}^\mu$ are tuned to different types of image patches. In other words, the input image patch is encoded by the filter that best matches it. The magnitude of the dot product $\vec{B} \cdot \vec{I}$ is less important than deciding which filter best matches the input $\vec{I}_p$. Matched filters can be thought of an extreme case of sparsity. In the previous sections, an image was represented by a linear combination of basis functions whose weights were penalizes by the *L1-norm*, $\sum_i |\alpha_i|$. By comparison, matched filters represent an image by a single basis function. This gives an ever sparser representation of the image, but at the possible cost of a much larger image dictionary. Matched filters can be thought of as *feature detectors* because they respond only to very specific inputs.

1.5 K-means and Dictionaries

One way to learn a dictionary of basis functions, for matched filters, is by using the K-means algorithm. This is a classic clustering algorithm but there are many others. As we will show, it related to mixtures of Gaussians and the EM algorithm.

The K-means algorithm

The input to K-means is a set f unlabeled data: $D = \{x_1, ..., x_n\}$. The goal is to decompose it into disjoint classes $w_1, ..., w_k$ where k is known.

The basic assumption is that the data D is clustered round (unknown) mean values $m_1, ..., m_k$, see figure (8).

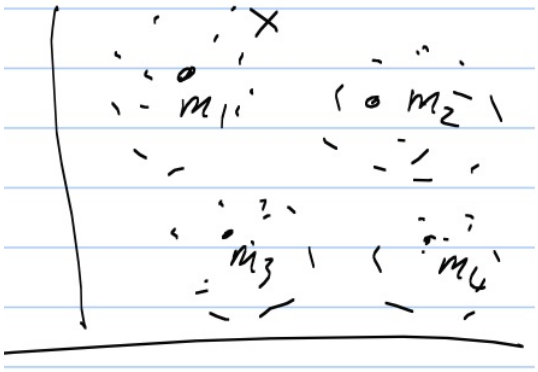

Figure 8: The k-means algorithm assumes that the data can be clustered round (unknown) mean values $m_1, ..., m_k$. Equivalently, that the data is generated by a mixture of Gaussians with fixed covariance and with these means.

We defines an association variable V_{ia}. $V_{ia} = 1$ if datapoint x_i is associated to mean m_a and $V_{ia} = 0$ otherwise. we have the constraint $\sum_a V_{ia} = 1$ for all i (i.e. eqach datapoint is assigned to a single mean). This gives a decomposition of the data. $D_a = \{i : V_{ia} = 1\}$ is the set of datapoints associated to mean m_a. The set $D = \bigcup_a D_a$ is the set of all datapoints. $D_a \bigcap D_b = \phi$ for all $a \neq b$ TODO: ϕ is the empty set.

We defines a goodness of fit:

$$E(\{V\}, \{m\}) = \sum_{i=1}^{n} \sum_{a=1}^{k} V_{ia}(x_i - m_a)^2 = \sum_{a=1}^{k} \sum_{x \in D_a} (x - m_a)^2 \tag{18}$$

The goal of the k-means algorithm is to minimize $E(\{V\}, \{m\})$ with respect to $\{V\}$ and $\{m\}$. $E(.,.)$ is a non-convex function and no known algorithm can find its global miminum. But k-means converges to a local minimum. TODO: It can be given a set of random initialization, obtain a local minima for each, then select the solution which has lowest energy. Or can use K++ as an initialization.

The k-means algorithm

1. Initialize a partition $\{D_a^0 : a = 1 \text{ to k}\}$ of the data. (I.e. randomly partition the datapoints – ir use K++).

2. Compute the mean of each cluster D_a, $m_a = \frac{1}{|D_a|} \sum_{x \in D_a} x$

3. For i=1 to n, compute $d_a(x_i) = |x_i - m_a|^2$
Assign x_i to cluster D_{a^*} s.t. $a^* = \arg\min\{d_a(x_i), ..., d_k(x_i)\}$

4. Repeat steps 2 & 3 until convergence.

This will converge to a minimum of the energy function because steps 2 and 3 each decrease the energy function (or stop if the algorithm is at a local minimum). This will divide the space into disjoint regions TODO: sketch this.

k-means can be formulated in terms of the assignment variable. At step 2, $m_a = \frac{1}{\sum_i V_{ia}} \sum_i V_{ia} x_i$. At step 3. $V_{ia} = 1$ if $|x_i - m_a|^2 = \min_b |x_i - m_b|^2$ and $V_{ia} = 0$ otherwise.

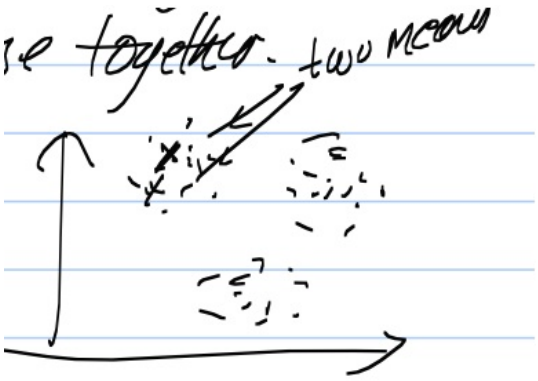

Figure 9: There are three clusters and four 'means'. In practice, two 'means' will usually be assigned to one cluster.

Soft k-means

A "softer" version of k-means – the Expectation-Maximization (EM) algorithm. Assign datapoint $\underline{x}_i$ to each cluster with probability $(P_1, \ldots, P_k)$

1. Initialize a partition of the datapoints.

2. For j=1 to n. Compute the probability that x_j belongs to ω_a
$P(\omega_a|x_j) = \frac{\exp -\frac{1}{2\sigma^2}(x_j - m_a)^2}{\sum_b \exp -\frac{1}{2\sigma^2}(x_j - m_b)^2}$.

3. Compute the mean for each cluster:
$m_a = \sum_j x_j P(\omega_a|x_j)$

Repeat steps 2 & 3 until convergence.

Note: in this version the *hard-assign* variable V_{ia} is replaced by a *soft-assign* variable $P(\omega_a|x_j)$. Observe that $\sum_a P(\omega_a|x_j) = 1$. Also observe that the softness is controlled by σ^2. In the limit, as $\sigma^2 \mapsto 0$, the distribution $P(\omega_a|x_j)$ will become binary valued, and soft k-means will be the same as k-means.

1.6 Deeper Understanding; Soft k-means and Mixtures of Gaussians

Soft k-means can be reformulated in terms of mixtures of Gaussians and the Expectation-Maximization (EM) algorithm. This assumes that the data is generated by a mixture of Gaussian distributions with means

$\{m\}$ and variance
$sigma^2\mathbf{I}$.

$$P(x|\{V\},\{m\}) = \frac{1}{Z}\exp\{-\sum_{ia} V_{ia}\frac{||x_i - m_a||^2}{\sigma^2}\}. \quad (19)$$

This is equivalent to a mixture of Gaussians:

$$P(x|V,m) = \mathcal{N}(x : \sum_a V_{ia} m_a, \sigma^2), \quad (20)$$

where the variable V identifies the mixture component (i.e. $V_{ia} = 1$ if datapoint x_i was generated by mixture a).

We need to impose a prior $P(\{V\})$ on the assignment variable V. It is natural to choose a uniform distribution $P(V) = 1/Z$, where Z is the number of possible assignments of the datapoints to the means.

This gives distributions $P(x,\{V\}|\{m\}) = P(x|\{V\},\{m\})P(\{V\})$. This form enables us to use the EM algorithm TODO: see later section to estimate the mean variables $\{m\}$ despite the presence of unknown/missing/latent variables $\{V\}$. The EM algorithm can be applied to problems like this where there are quantities to be estimated but also missing/latent variables. The EM algorithm can be formulated in terms of minimizing an energy function, but this energy function is non-convex and EM can be only guaranteed to converge to a minimum of the energy function and not to a global minimum. Deriving the soft k-means algorithm by applying the EM algorithm to $P(x|V,m)$.is left as an exercise for the reader.

We can extend soft k-means in several ways. The simplest, which we will do next, is to allow the covariances of the Gaussians to differ and to estimate them as well. More generally, we can have a process $P(x,h|\theta)$ where x is the observed data, h is a hidden/missing/latent variable, and θ are the model parameters. TODO: Add Gaussian material from other notes.

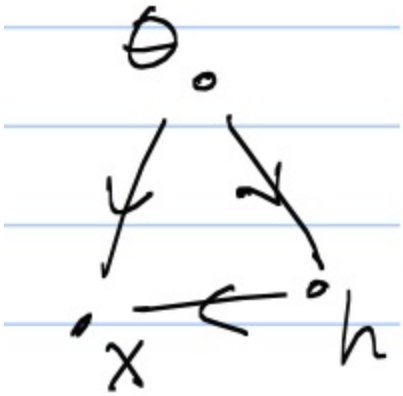

Figure 10: The data x is generated by hidden variable h by a probability model with unknown parameters θ.

Mini-Epitomes

This is another way to learn dictionaries which uses a complicated variant of mixtures of Gaussians by introducing extra variables. It is motivated by deal with the shift-invariance of images and introduces additional variables to deal with this. TODO: get figures from the powerpoints to show why we care about shofts.

This is another way to learn a dictionary with a more complicated generative model with more hidden variables.. It is motivated by the fact that images are shift-invariant (unless they are carefully aligned).

Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of possibly overlapping patches of size $h \times w$ pixels cropped from a large collection of images.

Our dictionary comprises K mini-epitomes $\{\mu_k\}_{k=1}^K$ of size $H \times W$, with $H \geq h$ and $W \geq w$. The length of the vectorized patches and epitomes is then $d = h \cdot w$ and $D = H \cdot W$, respectively.

We approximate each image patch $\mathbf{x}_i$ with its best match in the dictionary by searching over the $N_p = h_p \times w_p$ (with $h_p = H - h + 1$, $w_p = W - w + 1$) distinct sub-patches of size $h \times w$ fully contained in each mini-epitome. Typical sizes we employ are 8×8 for patches and 16×16 for mini-epitomes, implying that each mini-epitome can generate $N_p = 9 \cdot 9 = 81$ patches of size $\mathbf{y}88$.

We model the appearance of image patches using a Gaussian mixture model (GMM). We employ a generative model in which we activate one of the image epitomes μ_k with probability $P(l_i = k) = \pi_k$, then crop an $h \times w$ sub-patch from it by selecting the position $p_i = (x_i, y_i)$ of its top-left corner uniformly at random from any of the N_p valid positions. We assume that an image patch $\mathbf{x}_i$ is then conditionally generated from a multivariate Gaussian distribution

$$P(\mathbf{x}_i|\mathbf{z}_i, \theta) = \mathcal{N}(\mathbf{x}_i; \alpha_i \mathbf{T}_{p_i} \mu_{l_i} + \beta_i \mathbf{1}, c_i^2 \mathbf{\Sigma}_0) \tag{21}$$

The label/position latent variable vector $\mathbf{z}_i = (l_i, x_i, y_i)$ controls the Gaussian mean via $\nu_{\mathbf{z}_i} = \mathbf{T}_{p_i} \mu_{l_i}$. Here $\mathbf{T}_{p_i}$ is a $d \times D$ projection matrix of zeros and ones which crops the sub-patch at position $p_i = (x_i, y_i)$ of a mini-epitome. The scalars α_i and β_i determine an affine mapping on the appearance vector and account for some photometric variability and $\mathbf{1}$ is the all-ones $d \times 1$ vector. Here $\bar{x}$ denotes the patch mean value and λ is a small regularization constant (we use $\lambda = d$ for image values between 0 and 255).

We choose $\pi_k = 1/K$ and fix the $d \times d$ covariance matrix $\mathbf{\Sigma}_0^{-1} = \mathbf{D}^T \mathbf{D} + \epsilon \mathbf{I}$, where $\mathbf{D}$ is the gradient operator computing the $x-$ and $y-$ derivatives of the $h \times w$ patch and ϵ is a small constant.

To match a patch $\mathbf{x}_i$ to the dictionary, we seek the mini-epitome label and position $\mathbf{z}_i = (l_i, x_i, y_i)$, as well as the photometric correction parameters (α_i, β_i) that maximize the probability in Eq. (21), or equivalently minimize the squared reconstruction error (note that $\mathbf{D1} = \mathbf{0}$)

$$R^2(\mathbf{x}_i; k, p) = \frac{1}{c_i^2} \left(\|\mathbf{D}\left(\mathbf{x}_i - \alpha_i \mathbf{T}_p \mu_k\right)\|^2 + \lambda(|\alpha_i| - 1)^2 \right), \tag{22}$$

where the last regularization term discourages matches between patches and mini-epitomes whose contrast widely differs. We can compute in closed form for each candidate match $\nu_{\mathbf{z}_i} = \mathbf{T}_{p_i} \mu_{l_i}$ in the dictionary the optimal $\hat{\beta}_i = \bar{x}_i - \hat{\alpha}_i \bar{\nu}_{\mathbf{z}_i}$ and $\hat{\alpha}_i = \frac{\tilde{\mathbf{x}}_i^T \tilde{\nu}_{\mathbf{z}_i} \pm \lambda}{\tilde{\nu}_{\mathbf{z}_i}^T \tilde{\nu}_{\mathbf{z}_i} + \lambda}$, where $\tilde{\mathbf{x}}_i = \mathbf{D}\mathbf{x}_i$ and $\tilde{\nu}_{\mathbf{z}_i} = \mathbf{D}\nu_{\mathbf{z}_i}$ are the whitened patches. The sign in the nominator is positive if $\tilde{\mathbf{x}}_i^T \tilde{\nu}_{\mathbf{z}_i} \geq 0$ and negative otherwise. Having computed the best photometric correction parameters, we can substitute back in Eq. (22) and evaluate the reconstruction error $R^2(\mathbf{x}_i; k, p)$.

In order to learn the parameters we use the EM algorithm.Given a large training set of unlabeled image patches $\{\mathbf{x}_i\}_{i=1}^N$, our goal is to learn the maximum likelihood model parameters $\theta = (\{\pi_k, \mu_k\}_{k=1}^K)$ for the epitomic GMM model in Eq. (21). As is standard with Gaussian mixture model learning, we employ the EM algorithm [**?**] and maximize the expected complete log-likelihood

$$L(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{p \in \mathcal{P}} \gamma_i(k,p) \cdot \log\left(\pi_k \mathcal{N}\left(\mathbf{x}_i; \alpha_i \mathbf{T}_p \mu_k + \beta_{i1} c_i^2 \Sigma_0\right)\right), \tag{23}$$

where $\mathcal{P}$ is the set of valid positions in the epitome. In the E-step, we compute the assignment of each patch to the dictionary, given the current model parameter values. We use the hard assignment version of EM and set $\gamma_i(k,p) = 1$ if the i-th patch best matches in the p-th position in the k-th mini-epitome and 0 otherwise. In the M-step, we update each of the K mini-epitomes μ_k by

$$\left(\sum_{i,p} \gamma_i(k,p) \frac{\alpha_i^2}{c_i^2} \mathbf{T}_p^T \Sigma_0^{-1} \mathbf{T}_p \right) \mu_k = \sum_{i,p} \gamma_i(k,p) \frac{\alpha_i}{c_i^2} \mathbf{T}_p^T \Sigma_0^{-1} (\mathbf{x}_i - \bar{x}_i \mathbf{1}). \tag{24}$$

The EM Algorithm

Suppose we have data x which is generated by a model $P(x|h,\theta)$ with a prior $p(h)$ for the hidden variables h. This gives a distribution $p(x,h|\theta)$ from which we can compute $p(x|\theta) = \sum_h p(x,h|\theta)$.

The goal is to estimate $\hat{\theta} = \arg\max P(x|\theta)$ (i.e. the maximum likelihood estimate of θ). This can be formulated in terms of minimizing $-\log p(x|\theta)$. To obtain EM we introduce a new variable $q(h)$ which is a distribution over the hidden variables. We define an energy function:

$$F(\theta, q) = -\log p(x|\theta) + \sum hq(h) \log \frac{q(h)}{p(h|x,\theta)}. \tag{25}$$

The second term is the Kullback-Leibler divergence TODO: state relations to Information Theory which has the property that it is non-negative and is zero only if $q(h) = p(h|x,\theta)$. This implies that minimizing $F(\theta, q)$ with respect to θ and q is equivalent to minimizing $-\log p(x|\theta)$ with respect to θ (by setting $q(h) = p(h|x,\theta)$).

The EM algorithm corresponds to minimizing $F(\theta, q)$ with respect to θ and $q(.)$ alternatively. These correspond to the two steps of the k-means algorithm. The algorithm is specified most simply by re-expressing $F(\theta, q) = \sum_h q(h) \log q(h) - \sum q(h) \log p(h, x|\theta)$ (which exploits $p(h, x|\theta) = p(h|x,\theta)p(x|\theta$ and $\sum hq(h) \log p(x|theta) = \log p(x|theta)$.

The algorithm starts with an initialization. Then follows by repeating the two staps.

Step 1. Fix θ and estimate $\hat{q}(.)$ by $p(h|x,\theta)$. This requires computing $P(h, x|theta)/p(x|theta)$.

Step 2. Fix $q(.)$ and estimate $\hat{\theta} = \arg\min | - \sum_h q(h) \log p(h, x|\theta)$.

Step 1 minimizes $F(\theta, q)$ with respect to q and Step 2 minimizes $F(\theta, q)$ with respect to θ. Hence each step is guaranteed to reduce the energy (to stop if the energy cannot be reduced0. There are many variants because convergence does not require minimizing $F(.,.)$ at each step, but only reducing it.

TODO: we can derive soft k-means from this.

A variant of EM

TODO: Older material – Another variant. Instead we have $p(\theta|D) = \sum_h p(\theta, h|D)$

Define a new distribution $q(h)$

Minimize $F(\theta, q) = -\log p(\theta|D) + \sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)}$

* Kullback-Leibler divergence: $\sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)} \geq 0$

Note, the minimum occurs at $\hat{\theta} = \arg\min_\theta\{-\log p(\theta|D)\} = \arg\max_\theta p(\theta|w)$
and at $\hat{q}(h) = p(h|\hat{\theta}, D)$
(Because the Kullback-Liebler divergence attains its minimum at $\hat{q}(h) = p(h|\hat{\theta}, w)$).

We can re-express the Free Energy

$F(\theta, q) = -\log p(\theta|D) + \sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)}$
$= -\sum_h q(h) \log p(\theta|D) + \sum_h q(h) \log q(h) - \sum_h q(h) \log p(h|\theta, D)$
$= \sum_h q(h) \log\{p(\theta|D)p(h|\theta, D)\} + \sum_h q(h) \log q(h)$

$$F(\theta, q) = -\sum_h q(h) \log p(h, \theta|D) + \sum_h q(h) \log q(h)$$

EM minimize $F(\theta, q)$ w.r.t. $\theta \& q$ alternatively. This is called 'coordinate descent', see figure (11). Here is an intuition for this algorithm. You live in a city with hills, like Seoul, and the streets are arranged in a horizontal grid of blocks. You start on a hill (at high altitude) and want to decrease your altitude. You can

either walk in the North-South direction or in the East-West direction. You look in all directions and must choose to walk North, South, East, or West. You see that North and East are uphill, so you do not choose them. You see that you can walk South and decrease your altitude by only 10 meters before the street starts going uphill. You look West and see that you can decrease your altitude by 100 meters by walking in that direction (until that street also starts going uphill). So you – i.e. coordinate descent – chooses to walk West and stop when the street starts going uphill (i.e. you have lost 100 meters). The you stop, look again in the directions North, South, East, West and walk in the direction to maximize your decrease in altitude. The you repeat until you are at a place where all directions are uphill.

Note: coordinate descent will be important when the dimension of the space is very big. In two-dimensions, like the city example, the number of choices is small – so if you have 'moved' East-West last time then you have to move North-South the next time. In high-dimensions, there are an enormous number of choices. So the algorithm will often choose only to 'move' in a small number of possible directions, and the other directions will be irrelevant. This will be important for AdaBoost learning.

Note: you may not be able to calculate how much you will decrease altitude by walking in one direction. This will depend on the specific problem.

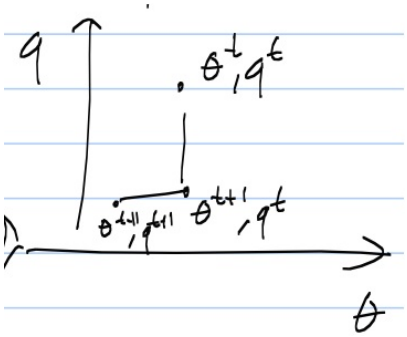

Figure 11: The coordinate descent algorithm. You start at θ^t, q^t. You fix q^t and decrease $F(\theta, q)$ by changing θ until you reach θ^{t+1}, where $\frac{\partial}{\partial \theta} F(\theta, q^t) = 0$. Then fix θ^{t+1} and decrease $F(\theta, q)$ by changing q until you reach q^{t+1}, where $\frac{\partial}{\partial q} F(\theta^{t+1}, q) = 0$.

Step 1: Fix q^t, set $\theta^{t+1} = \arg\min_\theta \{-\sum_h q(h) \log p(h, \theta | D)\}$
Step 2: Fix θ^t, set $q^{t+1}(h) = p(h|\theta^t, D)$

Iterate steps 1 & 2 until convergence.

Note: this algorithm is guaranteed to converge to a local minimum of $F(\theta, q)$, but there is no guarantee that it will converge to the global minimum. You can use multiple starting points – and pick the solution which has lowest value of $F(.,)$ – or you can use extra knowledge about the problem to have a good starting point, or starting points. Or you can use a stochastic algorithm.