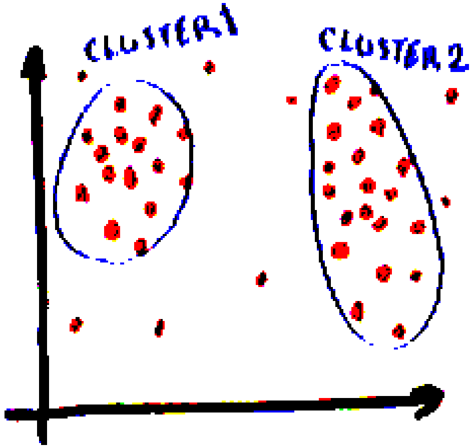
MACHINE LEARNING AND PATTERN RECOGNITION

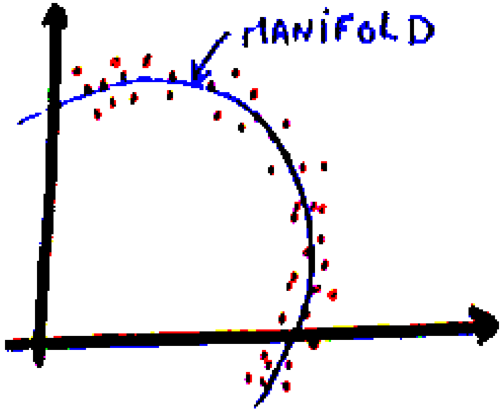
Fall 2004, Lecture 8: Unsupervised Learning, Density Estimation, EM

Yann LeCun
The Courant Institute,
New York University
http://yann.lecun.com

Unsupervised Learning

The basics idea of unsupervised learning: Learn an energy function $E(Y)$ such that $E(Y)$ is small if Y is “similar” to the training samples, and $E(Y)$ is large if Y is “different” from the training samples. What we mean by “similar” and “different” is somewhat arbitrary and must be defined for each problem.





- Probabilistic unsupervised learning: Density Estimation. Find a function f such $f(Y)$ approximates the empirical probability density of Y, $p(Y)$, as well as possible.
- Clustering: discover “clumps” of points
- Embedding: discover low-dimensional manifold or surface that is as close as possible to all the samples.
- Compression/Quantization: discover a function that for each input computes a compact “code” from which the input can be reconstructed.

Parametric Density Estimation

Use Maximum Likelihood: Given a model $P(Y|W)$, find the parameter W that best "explains" the training samples, i.e. the W that maximizes the likelihood of the training samples $Y^1, Y^2, ...Y^P$. Assuming that the total data likelihood factorizes into individual sample likelihoods:

$$P(Y^1, Y^2, ...Y^P|W) = \prod_i P(Y^i|W)$$

Equivalently, find the W that minimizes the negative log likelihood.

$$L(W) = -log \prod_i P(Y^i|W) = \sum_i -log P(Y^i|W)$$

This is called *parametric* estimation because we assume that the family of possible densities is parameterized by W.

Parametric Density Estimation

Assuming $P(Y|W)$ is the normalized exponential of an energy function:

$$P(Y|W) = \frac{\exp(-\beta E(Y,W))}{\int \exp(-\beta E(Y,W))dY}$$

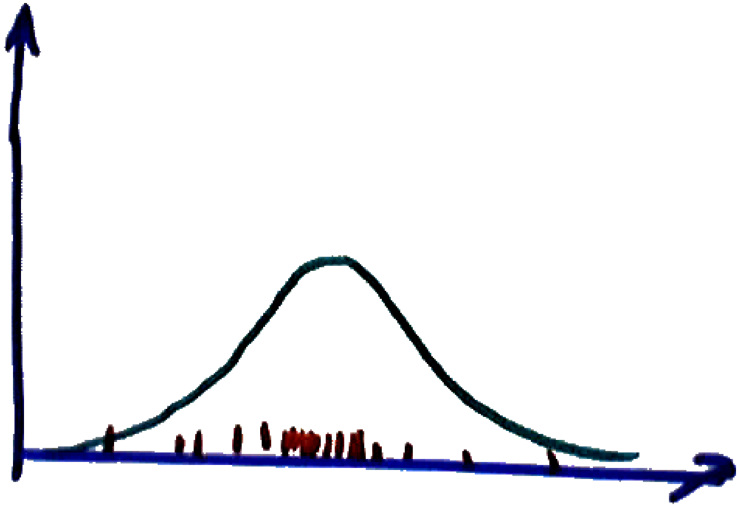
and after an irrelevant division by β, we get the loss function:

$$L(W) = \sum_i \left(E(Y^i, W) + \frac{1}{\beta} \log \int \exp(-\beta E(Y,W))dY \right)$$

The Maximum A Posteriori Estimate is similar but includes a penalty on W:

$$L(W) = \sum_i \left(E(Y^i, W) + \frac{1}{\beta} \log \int \exp(-\beta E(Y,W))dY \right) + H(W)$$

Example: Univariate Gaussian



- Maximum Likelihood: find the parameters of a Gaussian that best “explains” the training samples $y^1, y^2, ...y^P$.
- negative log-likelihood of the data (one dimension): $L(m, v) = -\sum_i \log \frac{1}{\sqrt{2\pi v}} \exp(-\frac{1}{2v}(y^i - m)^2)$

$$L(m, v) = \frac{1}{2}\sum_i \frac{1}{v}(y^i - m)^2 + \log 2\pi v$$

Minimize $L(m, v)$ with respect to m and v.

Example: Univariate Gaussian

- Minimize $L(m, v)$ with respect to m

$$\frac{\partial L(m,v)}{\partial m} = \frac{1}{2}\sum_i \frac{1}{v}(y^i - m) = 0$$

Hence, $m = \frac{1}{P}\sum_i y^i$

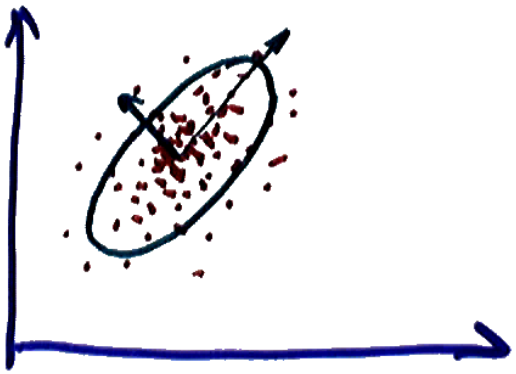
- Now minimize $L(m, v)$ with respect to v

$$\frac{\partial L(m,v)}{\partial v} = \frac{1}{2}\sum_i \left(-\frac{1}{v^2}(y^i - m)^2 + \frac{1}{v}\right) = 0$$

Hence $v = \frac{1}{P}\sum_i (y^i - m)^2$

- surprise-surprise: The maximum likelihood estimates of the mean and variance of a Gaussian are the mean and variance of the samples.

Example: Multi-variate Gaussian



Maximum Likelihood: find the parameters of a Gaussian that best "explains" the training samples $Y^1, Y^2, ...Y^P$.
The negative log-likelihood of the data (M is a vector, V is a matrix):

$$L(M,V) = -\sum_i \log\left(|2\pi V|^{-1/2}\exp(-1/2(Y^i - M)'V^{-1}(Y^i - M))\right)$$

$$L(M,V) = \frac{1}{2}\sum_i (Y^i - M)'V^{-1}(Y^i - M) - \log|V^{-1}| + \log(2\pi)$$

Multi-variate Gaussian (continued)

$$L(M,V) = \frac{1}{2}\sum_i (Y^i - M)'V^{-1}(Y^i - M) - \log|V^{-1}| + \log(2\pi)$$

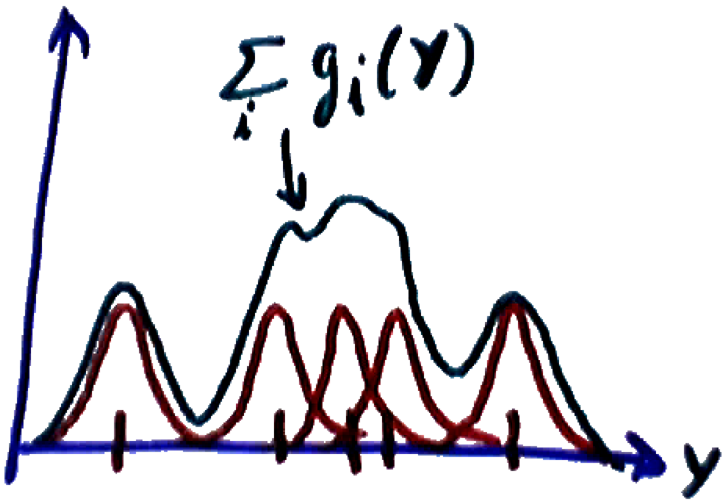
$$\frac{\partial L(M,V)}{\partial M} = \frac{1}{2}\sum_i V^{-1}(Y^i - M) = 0$$

Hence, $M = \frac{1}{P}\sum_i Y^i$ Now minimize $L(M,V)$ with respect to V^{-1}

$$\frac{\partial L(M,V)}{\partial V^{-1}} = \frac{1}{2}\sum_i \left((Y^i - M)(Y^i - M)' - V\right)$$

(using the fact $\frac{\partial \log|V^{-1}|}{\partial V^{-1}} = V'$).
Hence $V = \frac{1}{P}\sum_i (Y^i - M)(Y^i - M)'$

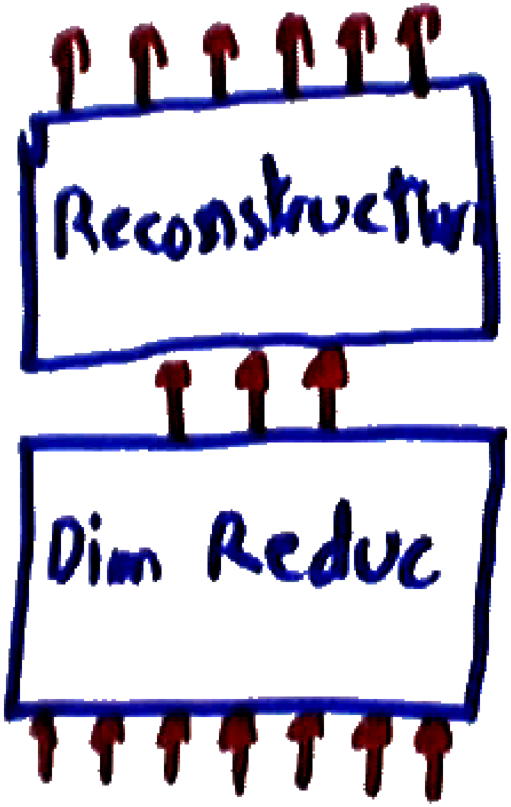
Non-Parametric Methods: Parzen Windows



- The sample distribution can be seen as a bunch of delta functions. Idea: make it smooth.
- Place a “bump” around each training sample Y^i.
- example: Gaussian bump $g_i(Y) = \frac{1}{Z}\exp(-K||Y - Y^i||^2)$ where Z is the Gaussian normalization constant.
- The density is $P(Y) = \frac{1}{P} \sim_{i=1}^{P} g_i(Y)$
- It’s simple, but it’s expensive.

Dimensionality Reduction

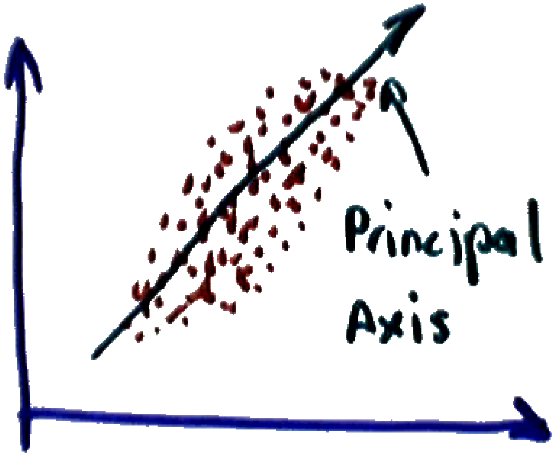
A slightly simpler problem than full-fledged density estimation: Find a low-dimensional surface (a manifold) that is as close as possible to the training samples.



- Example 1: reducing the number of input variables (features) to a classifier so as to reduce the over-parameterization problem.
- Example 2: images of human faces can be seen as vectors in a very high dimensional space. Actual faces reside in a small subspace of that large space. If we had a parameterization of the manifold of all possible faces, we could generate new faces or interpolate between faces by moving around that surface. (this has been done, see Blanz and Vetter "Face recognition based on fitting a 3D morphable model" IEEE Trans. PAMI 25:1063-1074, 2003).
- Example 3: Parameterizing the possible shapes of a mouth so we can make a simulated human speak (see http://www.vir2elle.com).

Linear Subspace: Principal Component Analysis

Problem: find a *linear* manifold that best approximates the samples. In other words, find a linear projection P such that the projection of the samples are as close as possible to the originals.



- We have a training set $Y^1 ... Y^P$. We assume all the components have zero mean. If not we center the vectors by subtracting the mean from each component.
- Question: what is the direction that we can remove (project out) while minimally affecting the training set.
- Let U be a unit vector in that dimension
- Removing the dimension in the direction of U will cost us $C = \sum_{i=1}^{P} (Y^{i\prime} U)^2$ (the square length of the projections of Y^i on U).

Principal Component Analysis

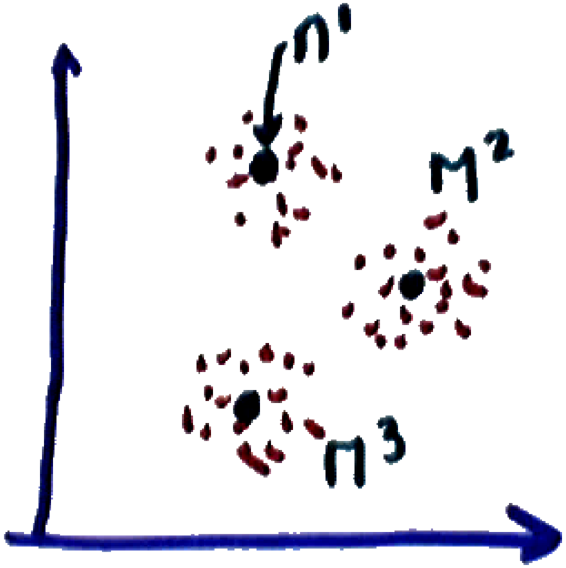
- Removing the dimension in the direction of U will cost us $C = \sum_{i=1}^{P}(Y^{i\prime}U)^2$ (the square length of the projections of Y^i on U).
- $C = \sum_{i=1}^{P} U'Y^iY^{i\prime}U = [U' \sum_{i=1}^{P} Y^iY^{i\prime}]U$ Q: How do we pick U so as to minimize the quantity in the bracket?
- The covariance matrix $A = \sum_{i=1}^{P} Y^iY^{i\prime}$ can be diagonalized: $A = Q\Lambda Q'$, where Q is a rotation matrix, whose lines Q_i are the normalized (and mutually orthogonal) eigenvectors of A, and Λ a diagonal matrix that contain the (positive) eigenvalues of A.
- It is easy to see that the unit vector U that minimizes $U'Q\Lambda Q'$ is aligned with the eigenvector of smallest eigenvalue of A.
- To eliminate more directions, we can repeat the process while remaining in the orthogonal space of the previously found directions.
- Practically: we simply find first K eigenvectors of A (associated with the K largest eigenvalues) and keep those.

Principal Component Analysis (PCA)

- step 1: We have a training set $Y^1 ... Y^P$ whose component variables have zero mean (or have been centered).
- step 2: compute the covariance matrix $A = \frac{1}{P}\sum_{i=1}^{P} Y^i Y^{i\prime}$
- step 3: diagonalize the covariance matrix: $A = Q' \Lambda Q$,
- step 4: Construct the matrix Q^k whose rows are the the eigenvectors of largest eigenvalues of A (a subset of rows of Q).

Multiplying a vector by Q^k gives the projections of the vector onto the principal eigenvectors of A. We can Now compute the k PCA features of any vector Y as $\mathrm{PCA^k}(Y) = Q^k Y$.

K-Means Clustering



Idea: find K prototype vectors that "best represent" the training samples $Y^1...Y^P$. More precisely, find K vectors $M^1, ...M^K$, such that

$$L = \sum_{i=1}^{P} \min_{k=1}^{K} ||Y^i - M^k||^2$$

is minimized. In other words, the M^k are chosen such that the error caused by replacing any Y^i by its closest prototype is minimized.

Application 1: Discovering hidden categories.

Application 2: Lossy data compression: to code a vector, find the prototype M^k that is closest to it, and transmit k. This process is called *Vector Quantization*.

Algorithm for K-Means Clustering

- Minimizing L: $\frac{\partial L}{\partial M^k} = 2\sum_{i \in S^k}(M^k - Y^i) = 0$ where S^k is the set of i for which M^k is the closest prototype to Y^i. We get:

  $$M^k = \frac{1}{|S^k|} \sum_{i \in S^k} Y^i$$

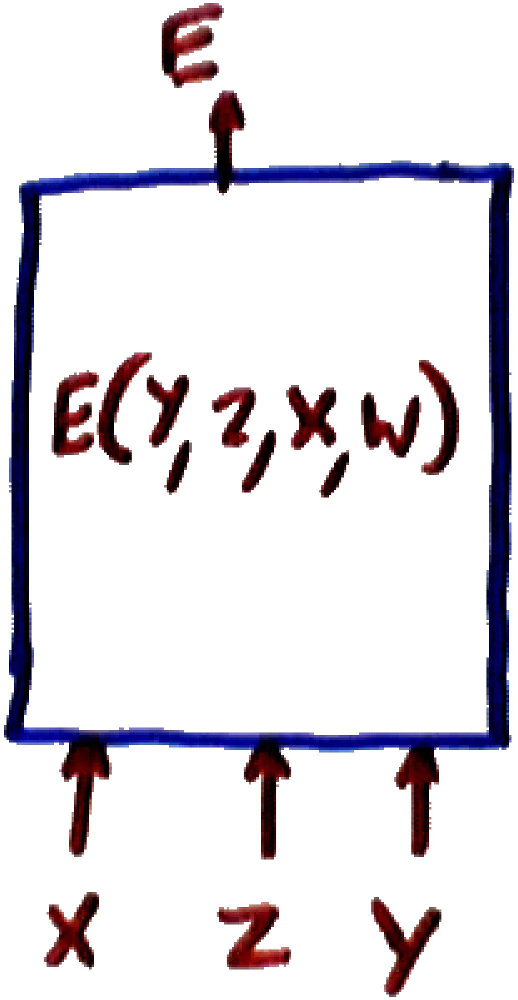
  where $|S^k|$ is the number of elements in S^k.
- Algorithm:
  - initialize the M^k (e.g. randomly).
  - repeat until convergence:
  - for each k compute the set S^k, the set of all i for which $||M^k - Y^i||^2$ is smaller than all other $||M^j - Y^i||^2$.
  - compute $M^k = \frac{1}{|S^k|}\sum_{i \in S^k} Y^i$
  - iterate
- Naturally, this algorithm works with any distance measure.

Hierarchical K-Means

Problem: Sometimes, K-Means may get stuck in very bad solutions (e.g. some prototypes have no samples assigned to them).
This is often caused by inappropriate initialization of the prototypes.
Cure: Hierarchical K-Means.
Main Idea:: run K-Means with $K = 2$, then run again K-Means with $K = 2$ on each of the two subsets of samples (those assigned to prototype 1, and those assigned to prototype 2).
What do we use K-Means for?: data compression (vector quantization) initialization of RBF nets of Mixtures of Gaussian.

Latent Variables

Latent variables are unobserved random variables Z that enter into the energy function $E(Y, Z, X, W)$.



The X variable (input) is always observed, the Y must be predicted. The Z variable is *latent*: it is not observed. We need to *marginalize* the joint probability $P(Y, Z|X, W)$ over Z to get $P(Y|X, W)$:

$$P(Y|X, W) = \int P(Y, Z|X, W)dZ$$

We can also write this as:

$$P(Y|X, W) = \int P(Y|Z, X, W)P(Z|X, W)dZ$$

Latent Variables

If we assume that $P(Y,Z|W)$ derives from an energy function $E(Y,Z,X,W)$ as $P(Y,Z|W) = \frac{\exp(-\beta E(Y,Z,X,W)}{\int\int \exp(-\beta E(y,z,X,W) dydz}$, the likelihood for one sample, Y^i is:

$$P(Y^i|W) = \frac{\int \exp(-\beta E(Y^i,Z,W))dZ}{\int\int \exp(-\beta E(Y,Z,W))dZdY}$$

We can also decompose this using $P(Y|X,W) = \int P(Y|Z,W)P(Z|W)dZ$:

$$P(Y^i|W) = \int \frac{\exp(-\beta E(Y^i,Z,W))}{\int \exp(-\beta E(Y,Z,W))dY} \frac{\exp(-\beta' E(Y^i,Z,W))}{\int \exp(-\beta' E(Y^i,Z,W))dZ} dZ$$

The parameters β and β' do not have to have the same value.

Latent Variables: Zero Temperature Limit

If we make β' go to infinity in the expression for the likeliood, then $P(Z|X,W)$ reduces to a Dirac delta function around its mode (the minimum of $E(Y,Z,W)$ as a function of Z).

Then, the likelihood becomes:

$$P(Y^i|W) = \frac{\exp(-\beta E(Y^i, Z^*, W))}{\int \exp(-\beta E(Y, Z^*, W))dY}$$

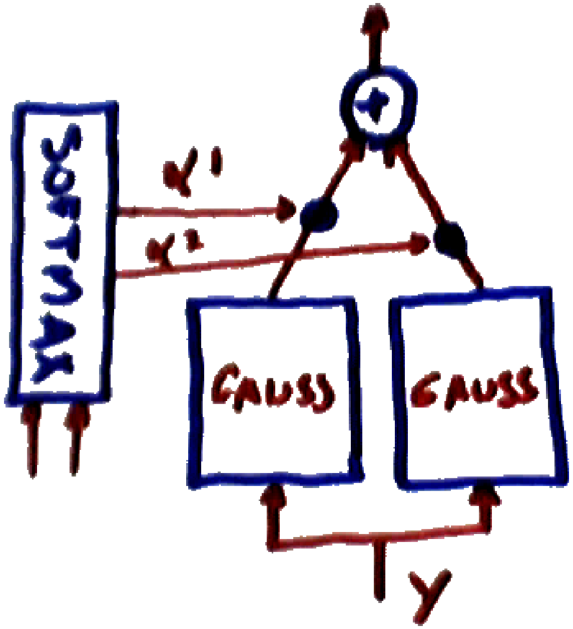
where Z^* is:

$$Z^* = \text{argmin}_Z E(Y^i, Z, W)$$

Example: Mixture Models

We have K normalized densities $P^k(Y|W^k)$, each of which has a positive coefficient α^k (whose sum over k is 1), and a switch controlled by a discrete latent variable Z that picks one of the comonent density. There is no input X, only an "output" Y (whose distribution is to be modeled) and a latent variable Z.

The likelihood for one sample Y^i:



$$P(Y^i, Z|W) = \sum_k \alpha^k P_k(Y^i|W^k)$$

with $\sum_k \alpha^k = 1$. Using Bayes' rule, we can compute the posterior prob of the mixture components for each data point Y^i:

$$r_k(Y^i) = P(Z = k|Y^i, W) = \frac{\alpha^k P_k(Y^i|W^k)}{\sum_j \alpha^j P_j(Y^i|W^j)}$$

These quantities are called "responsabilities".

Learning a Mixture Model with Gradient

We can learn a mixture with gradient descent, but there are much better methods as we will see later. The negative log-likelihood of the data is:

$$L = -\log \prod_i P(Y^i|W) = \sum_i -logP(Y^i|W)$$

Let us consider the likelihood of one data point Y^i:

$$L^i = -logP(Y^i|W) = -log \sum_k \alpha_k P_k(Y^i|W)$$

$$\frac{\partial L^i}{\partial W} = \frac{1}{P(Y^i|W)} \sum_k \alpha_k \frac{\partial P_k(Y^i|W)}{\partial W}$$

Learning a Mixture Model with Gradient (cont)

$$\frac{\partial L^i}{\partial W} = \frac{1}{P(Y^i|W)} \sum_k \alpha_k \frac{\partial P_k(Y^i|W)}{\partial W}$$

$$= \sum_k \alpha_k \frac{1}{P(Y^i|W)} P_k(Y^i|W) \frac{\partial \log P_k(Y^i|W)}{\partial W}$$

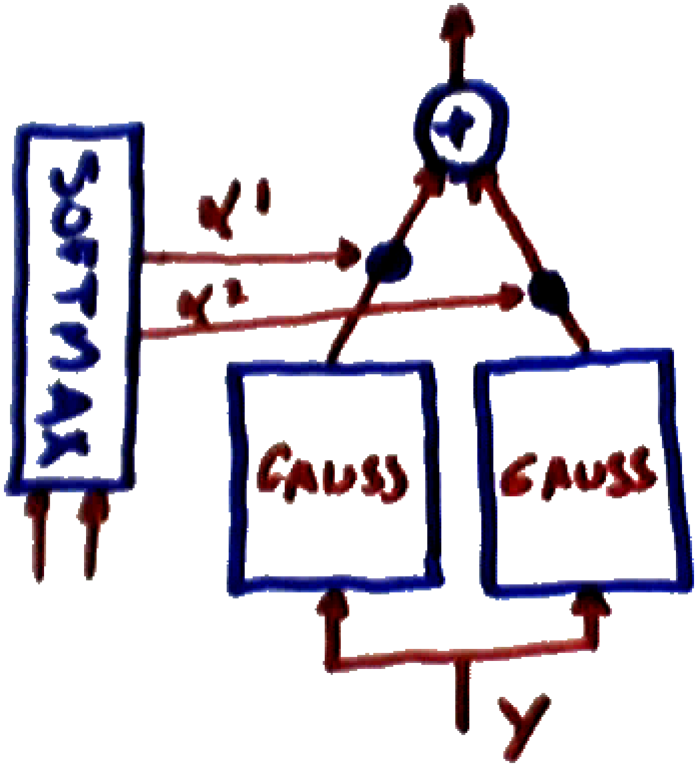
$$= \sum_k \alpha_k \frac{P_k(Y^i|W)}{P(Y^i|W)} \frac{\partial \log P_k(Y^i|W)}{\partial W} == \sum_k r_k(Y^i) \alpha_k \frac{\partial \log P_k(Y^i|W)}{\partial W}$$

The gradient is the weighted sum of gradients of the individual components weighted by the responsabilities.

Example: Gaussian Mixture

$$P(Y|W) = \sum_k \alpha_k |2\pi V^k|^{-1/2} \exp(-1/2(Y - M^k)'(V^k)^{-1}(Y - M^k))$$

This is used a lot in speech recognition.



The Expectation-Maximization Algorithm

Optimizing likelihoods with gradient is the only option in some cases, but there is a considerably more efficient procedure known as EM.

Every time we update the parameters W, the distribution over latent variables Z must be updated as well (because it depends on W.

The basic idea of EM is to keep the distribution over Z constant while we find the optimal W, then we recompute the new distribution over Z that result from the new W, and we iterate. This process is sometimes called *coordinate descent*.

EM: The Trick

The negative log likelihood for a sample Y^i is:

$$L^i = -\log P(Y^i|W) = -\log \int P(Y^i, Z|W) dZ$$

For any distribution $q(Z)$ we can write:

$$L^i = -\log \int q(Z) \frac{P(Y^i, Z|W)}{q(Z)} dZ$$

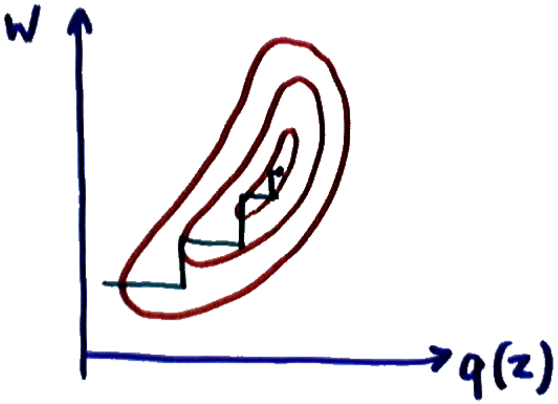
We now use Jensen's inequality, which says that for any concave function G (such as log)

$$-G(\int p(z) f(z) dz) \leq -\int p(z) G(f(z)) dz$$

We get:

$$L^i \leq F^i = -\int q(Z) \log \frac{P(Y^i, Z|W)}{q(Z)} dZ$$

EM



$$L^i \leq F^i = -\int q(Z)\log\frac{P(Y^i, Z|W)}{q(Z)}dZ$$

EM minimizes F^i by alternately
finding the $q(Z)$ that mininizes F **(E-step)**
then finding the W that minimizes F **M-step)**
E-step: $q(Z)^{t+1} \leftarrow \text{argmin}_q F^i(q(Z)^t, W^t)$
M-step: $W(Z)^{t+1} \leftarrow \text{argmin}_W F^i(q(Z)^{t+1}, W^t)$

M Step

We can decompose the free energy:

$$F^i(q(Z), W) = -\int q(Z) \log \frac{P(Y^i, Z|W)}{q(Z)} dZ$$

$$= -\int q(Z) \log P(Y^i, Z|W) dZ + \int q(Z) \log q(Z) dZ$$

The first term is the expected energy with distribution $q(Z)$, the second is the entropy of $q(Z)$, and does not depend on W.
So in the M-step, we only need to consider the first term when minimizing with respect to $q(Z)$.

$$W(Z)^{t+1} \leftarrow \text{argmin}_W - \int q(Z) \log P(Y^i, Z|W) dZ$$

E Step

Proposition: the value of $q(Z)$ that minimizes the free energy is $q(Z) = P(Z|Y^i, W)$
This is the posterior distrib over the latent variabled given teh sample and the current parameter.
Proof:

$$F^i(P(Z|Y^i, W), W) = -\int P(Z|Y^i, W) \log \frac{P(Y^i, Z|W)}{P(Z|Y^i, W)} dZ$$

$$= -\int P(Z|Y^i, W) \log P(Y^i|W) dZ =$$

$$-\log P(Y^i|W) \int_z P(Z|Y^i, W) = -\log P(Y^i|W).1$$