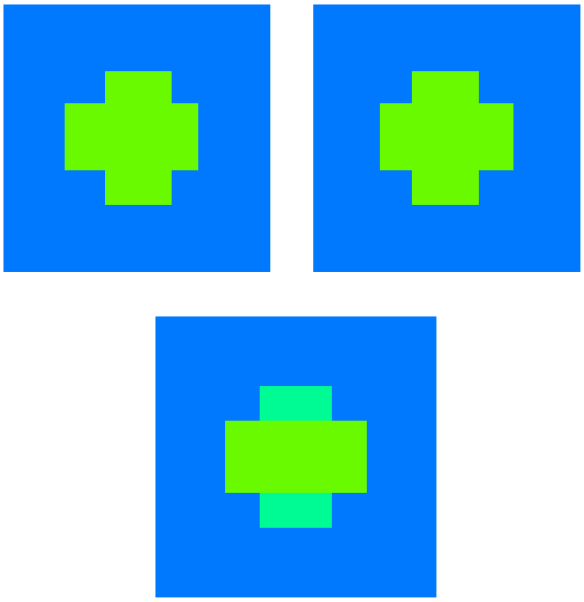
Introduction to Neural Networks
U. Minn. Psy 5038
Daniel Kersten
Expectation-Maximization (EM)

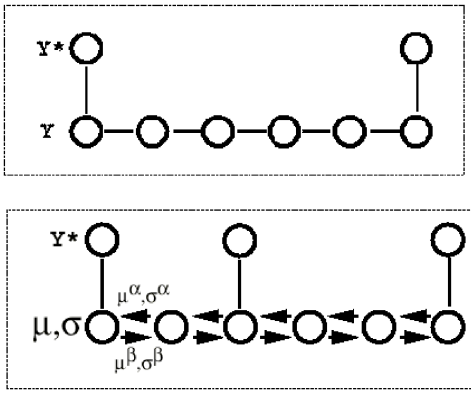
Initialize

■ **Read in Statistical and Graphics Add-in packages:**

In[1]:=
```
Off[General::spell1];
Needs["ErrorBarPlots`"];
Needs["MultivariateStatistics`"];
Needs["Histograms`"];
```

Last time





A simulation: Belief propagation for interpolation with missing data

■ Initialization

In[5]:=
```
size = 32; xs = Table[0, {i, 1, size}]; xs[[1]] = 1; xs[[size]] = 1;
data = Table[N[j] xs[[j]], {j, 1, size}];
g3 = ListPlot[Table[N[j], {j, 1, size}], Joined → True,
  PlotStyle → {RGBColor[0, 0.5, 0]}];
g2 = ListPlot[data, Joined → False,
  PlotStyle → {Opacity[0.35], RGBColor[0.75, 0., 0], PointSize[Large]}];
```

In[7]:=
```
size = 32;
μ0 = 1; μα = 1; σα = 100 000; (*large uncertainty *) μβ = 1;
σβ = 100 000;(*large*)
σR = 4.0; σD = 1.0;
μ = Table[μ0, {i, 1, size}]; σ = Table[σα, {i, 1, size}];
μα = Table[μ0, {i, 1, size}]; σα = Table[σα, {i, 1, size}];
μβ = Table[μ0, {i, 1, size}]; σβ = Table[σβ, {i, 1, size}];
iter = 0; i = 1;
j = size;
```

In[68]:=

$$\mu[[i]] = \frac{\frac{xs[[i]]\,data[[i]]}{\sigma D} + \frac{\mu\alpha[[i]]}{\sigma\alpha[[i]]} + \frac{1.\,\mu\beta[[i]]}{\sigma\beta[[i]]}}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\alpha[[i]]} + \frac{1}{\sigma\beta[[i]]}};\ \sigma[[i]] = \frac{1.}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\alpha[[i]]} + \frac{1}{\sigma\beta[[i]]}};$$

$$\mu[[j]] = \frac{\frac{xs[[j]]\,data[[j]]}{\sigma D} + \frac{\mu\alpha[[j]]}{\sigma\alpha[[j]]} + \frac{1.\,\mu\beta[[j]]}{\sigma\beta[[j]]}}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]} + \frac{1}{\sigma\beta[[j]]}};\ \sigma[[j]] = \frac{1.}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]} + \frac{1}{\sigma\beta[[j]]}};$$

```
nextj = j - 1;
```

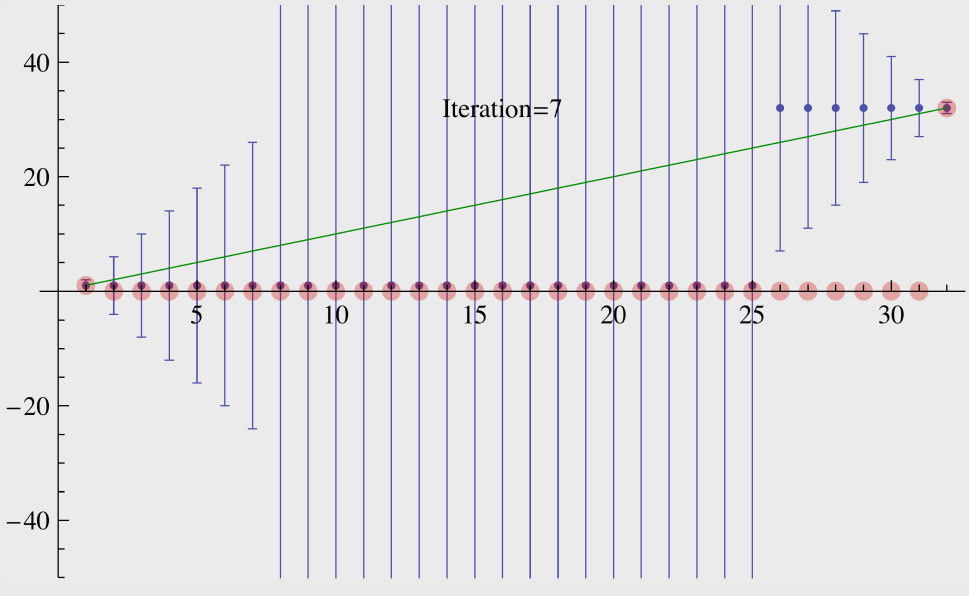
$$\mu\alpha[[nextj]] = \frac{\frac{xs[[j]]\,data[[j]]}{\sigma D} + \frac{1.\,\mu\alpha[[j]]}{\sigma\alpha[[j]]}}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]}};\ \sigma\alpha[[nextj]] = \sigma R + \frac{1.}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]}};$$

```
nexti = i + 1;
```

$$\mu\beta[[nexti]] = \frac{\frac{xs[[i]]\,data[[i]]}{\sigma D} + \frac{1.\,\mu\beta[[i]]}{\sigma\beta[[i]]}}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\beta[[i]]}};\ \sigma\beta[[nexti]] = \sigma R + \frac{1.}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\beta[[i]]}};$$

```
j--; i++; iter++;
yfit = Table[{μ[[i1]], σ[[i1]]}, {i1, 1, size}]; g1b = ErrorListPlot[{yfit}];
Show[
 {g1b, g2, g3,
  Graphics[{Text["Iteration=" <> ToString[iter], {size/2, size}]}]},
 PlotRange → {-50, 50}, ImageSize → Medium]
```

Out[76]=



The code below implements the above iterative equations, taking care near the boundaries. The plot shows the estimates of $y_i = \mu$, and the error bars show $\pm\sigma_i$.

Introduction

The Expectation Maximization algorithm can be viewed as an example of *unsupervised learning* and crops up in a wide range of applications, including probability density estimation, clustering, and discovering prototypes from data. In principle, it is very general and is guaranteed to converge to a local likelihood maximum. The EM algorithm is a common procedure for integrating out "hidden variables"--i.e. for marginalizing.

Integrating out secondary variables

Suppose we have 5 (hidden) processes each of which can contribute to our data.

■ Generating samples from a Gaussian Mixture Distribution

In[123]:=
```
Clear[ndist];
ndist[μ_, Σ_] := MultinormalDistribution[μ, Σ];
```

In[147]:=
```
r = .05;
σ = 1.0;
μ = Table[3 * N[{Cos[2 Pi i], Sin[2 Pi i]}], {i, 0, 2 Pi, Pi / 4}];
Σ = Table[{{σ, r}, {r, σ}}, {i, 0, 2 Pi, Pi / 4}];

Det[{{σ, r}, {r, σ}}]
```

Out[151]= 0.9975

The generator: randomindex := Random[Integer,{1,5}]; would give uniformly distributed mix labels.

The following gives mixing mix probabilities of 1/7, 1/7, 3/7, 1/7 and 1/7 for subdistributions 1,2,3,4,5 respectively.

In[153]:=
```
randomindex := {1, 2, 3, 3, 3, 4, 5}[[RandomInteger[{1, 7}]]]
```

Thus our stochastic generative process can be written:

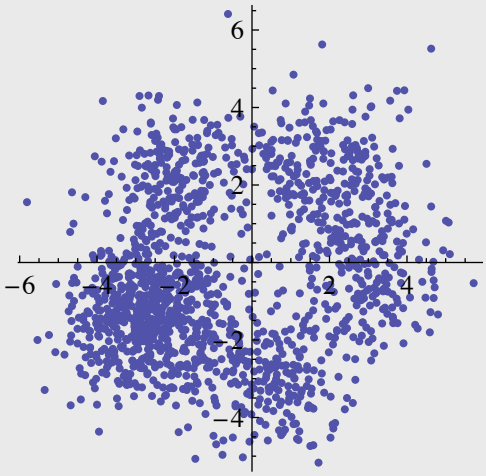
In[154]:=
```
RandomReal[ndist[μ[[t = randomindex]], Σ[[t]]]]
```

Out[154]=
```
{1.55142, −2.23543}
```

In[157]:=
```
scatterdata = Table[Random[ndist[μ[[t = randomindex ]], Σ[[t]]]],
   {i, 1, 1500}];
```

In[162]:=
```
ListPlot[scatterdata, AspectRatio → Automatic, ImageSize → Small]
```

Out[162]=



For mixture distributions, we may not be directly interested in the mixing probabilities--but are interested in the parameters of the distributions, e.g. the 5 means. These means, in an N-dimensional space for example, could represent prototypes in memory.

■ Simple case & intuition

(Example from chapter by: Yuille, Coughlan, Kersten & Schrater)

Suppose there is some agent that randomly chooses whether to flash a bright (a=1) or a dim (a=2) light, and for each flash we make a measurement x . Further suppose that the light measurements x are Gaussian distributed, and the standard deviations are known (we will drop these assumptions later). We could use Bayes rule to decide for a given measurement, which flash setting is the most probable, using an expression for the posterior probability: p(Flash = bright vs. dim | x).

However, rather than making decisions about whether a given flash was caused by the high-level (bright) or low-level (dim) switch , we want to estimate the two means from a series of measurements (e.g. photon counts), {x_i: i =1,...,N } , without knowing which measurement came from which light setting. Of course, if we knew which measurement came from which switch setting, our job would be easy. Let V_{ia}=1 if data x_i is generated by model P(x| mu _a, sigma ^2) and V_{ia}=0 otherwise, for a=1,2 . Then using the usual formula for the estimate of the mean:

$$\mu_a \approx \frac{\sum_{i=1}^{N} V_{ia} x_i}{\sum_{i=1}^{N} V_{ia}}$$

The problem is that the values of V_{ia} are unknown secondary (hidden) variables which influence the observations. However, if we could somehow estimate the probability of which switch generated each measurement x_i , then the means could be approximated as:

$$\mu_a \approx \frac{\sum_{i=1}^{M} \bar{V}_{ia} x_i}{\sum_{i=1}^{M} \bar{V}_{ia}}, \ \forall \ a$$

where $\bar{V}${ia} is the average value of V{ia} and is given by $\bar{V}${ia} = p(V{ia}=1 | x_i) = p(a |x_i) . This brings us a step closer to a solution, but raises another problem--to calculate $\bar{V}${ia} we will need to know the means--the very parameters we were trying to estimate in the first place! This is because p(a |x_i) depends on the means for switch setting a=1 or a=2. In other words, we need the formula for p(a |x_i, μ_a) (=p(a |x_i)).

Let's see how to justify our intuitive estimate of the means, and in the process solve the dilemma of how to determine the probability of which switch generated each measurement. Our goal will be to find the maximum likelihood estimates of the means (and eventually other state variables) conditional on the observations. We will derive the EM rules for a mixture of multi-variate gaussians.Then we will derive a more general rule for arbitrary discrete distributions.

Mixtures of multivariate gaussians

It is almost as easy to see the derivation for a random vector (i.e. multivariate) as a random scalar, so we'll derive the results for the general vector case.

Let x represent a d-dimensional vector generated from a mixture of M Gaussian densities, with s= {mu_a,C_a } and h= {a} , where mu_a , C_a , and p(a) are the vector means, covariance matrices, and mixture probs, respectively.

The notation s, and h is used later for an unknown parameter s (or a set s)-- a "primary variable", that we want to estimate, and another parameter h (or set h) that is hidden--a "secondary" variable. (But we'll also see how to estimate p(a)). So the mixture probability distribution of x is:

$$p(x) = p(x|\mu_a, C_a) = \sum_a p(x|a)p(a)$$

where a = 1,...,M , and $\sum_a$p(a) = 1 . The p(x |a) are the class-conditional probability densities:

$$p(x|a) = \frac{1}{\sqrt{(2\pi)^d |C_a|}} exp\{-(x-\mu_a)^T \cdot C_a^{-1} \cdot (x-\mu_a)/2\}$$

We are given a sequence of vector measurements {x_i: i =1,...,N } and wish to estimate the means, the covariances and mixing parameters. Let the probability that x_i came from mixture component a be p(x_ila) .

$$p(x_1, x_2, \ldots, x_N \mid \mu_a, C_a) = \prod_i \sum_a p(x_i \mid a)\, p(a)$$

The log-likelihood of the data is given by:

$$E(\mu_a, C_a) = \log(p(x_1, x_2, \ldots, x_N)) = \sum_i \log\{\sum_a p(x_i|a)p(a)\}$$

For the moment, suppose the mixing parameters and covariance matrices are known, and we want to estimate the means. The values of $\mu_a$which maximize E can be found analytically by solving:

$$\partial E/\partial \mu_j = \sum_i \frac{p(x_i|j)p(j)}{\sum_a p(x_i|a)p(a)} \frac{(\mu_j - x_i)}{\sigma_j^2} = \sum_i p(j|x_i) \frac{(\mu_j - x_i)}{\sigma_j^2} = 0$$

where we've used:

$$p(a|x_i) = \frac{p(x_i, a)}{p(x_i)} = \frac{p(x_i|a)p(a)}{\sum_a p(x_i|a)p(a)}$$

Solving for the μ_j's (= μ_a's), we have

$$\mu_a = \frac{\sum_i x_i p(a|x_i)}{\sum_i p(a|x_i)}.$$

This corresponds to the intuition we had above for the bright/dim example. And recall, the rub, of course, is that p(alx_i) depends on μ_a..., so we guess.

E&M iteration steps

■ Estimate conditional mixing probabilities: E-step

Let $\mu_{a,t}$ be an initial guess at time step t. Then in the E-step (E for "expectation") we let:

$$p(a|x_i, \mu_{a,t}) = \frac{p(x_i|a, \mu_{a,t})p(a)}{\sum_a p(x_i|a, \mu_{a,t})p(a)}$$

We can think of this as estimating the probability of which process the data belongs to (e.g. the bright vs. dim switch setting) based on our current guess of the mean.

■ And then find the mean that maximizes the likelihood of the data: M-step

Now that we have an estimate of the current conditional mixing probability, we can proceed to update our estimates of the mean.

This is the M-step ("maximization" step).

$$\mu_{a,t+1} = \frac{\sum_i x_i p(a|x_i, \mu_{a,t})}{\sum_i p(a|x_i, \mu_{a,t})}.$$

■ How about the covariance?

Suppose we don't know the covariance matrix or the mixing probabilities? We can use the updated values of p_t(alx_i) ,

$$p_{t+1}(a|x_i,) = p_t(a|x_i, \mu_a(t), C_a(t), p_t(a))$$

to progressively estimate the state parameters C_a and p(a) , as well as mu_a . The M-step for the covariance is:

■ **M-step for covariance & p(a)**

$$C_a(t+1) = \frac{\sum_i (x_i - \mu_a)(x_i - \mu_a)^T p_t(a|x_i)}{\sum_i p_t(a|x_i)}$$

(where **ab^T** is the outer product between vectors **a** and **b**). The mixing probabilities are estimated by:

$$p_{t+1}(a) = \frac{1}{N} \sum_i p_t(a|x_i)$$

Demo: Two gaussians, unknown μ's σ's , and mixing probabilities (revised)

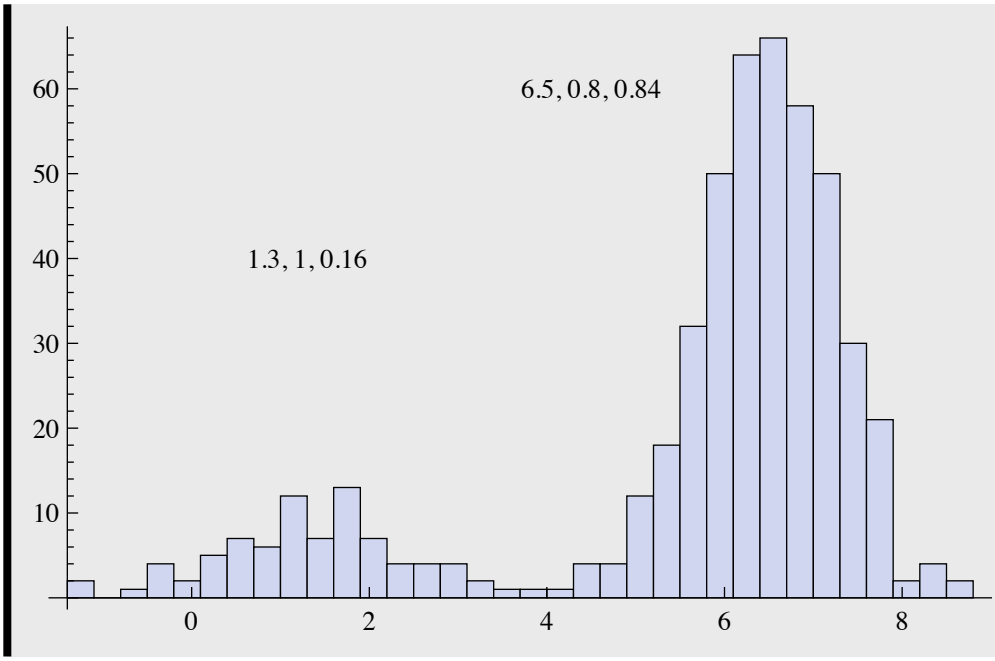
Generating samples from a Gaussian Mixture Distribution

■ **Generative model**

In[80]:=
```
μ1 = 1.3; σ1 = 1; μ2 = 6.5; σ2 = 0.8; a1 = 0.16; a2 = 1 - a1;
p1dist = NormalDistribution[μ1, σ1]; p2dist = NormalDistribution[μ2, σ2];
p1[x_] := PDF[p1dist, x]; p2[x_] := PDF[p2dist, x];
samplemix := If[RandomReal[] < a1, Random[p1dist], Random[p2dist]];
```

In[81]:=
```
data = Table[samplemix, {i, 1, 500}];
Histogram[data, HistogramCategories → Range[-20, 20, 0.3],
 Epilog →
  {Text[ToString[μ1] <> ", " <> ToString[σ1] <> ", " <> ToString[a1],
    {μ1, 40}],
   Text[ToString[μ2] <> ", " <> ToString[σ2] <> ", " <> ToString[a2],
    {μ2 - 2, 60}]}]
```

Out[82]=



EM algorithm--Now learn the parameters--means, standard deviations, and mixing probabilities

In[83]:=
```
pxdm[x_, mu_, σ_] := PDF[NormalDistribution[mu, σ], x];
```

E-step: The prob of mixing labels conditional on the data x is:

$$p(a|x_i, \mu_{a,t}) = \frac{p(x_i|a, \mu_{a,t})p(a)}{\sum_a p(x_i|a, \mu_{a,t})p(a)}$$

In *Mathematica* code,

In[84]:=
```
pmcx[a_, x_] := pxdm[x, μ[[a]], σ[[a]]] *
   pa[[a]] /
    (pxdm[x, μ[[1]], σ[[1]]] * pa[[1]] + pxdm[x, μ[[2]], σ[[2]]] * pa[[2]]);
```

M-step: The maximum likelihood estimate of the means is:

$$\mu_a = \frac{\sum_i x_i p(a|x_i)}{\sum_i p(a|x_i)}.$$

or in *Mathematica*:

μ[[1]]=pmcx[1,data].data/Plus@@(pmcx[1,#]&/@data)

μ[[2]]=pmcx[2,data].data/Plus@@(pmcx[2,#]&/@data)

Similarly, the variances and mixing probabilities are also weighted averages:

σ[[1]]=Sqrt[pmcx[1,data].(data-μ[[1]])^2/Plus@@(pmcx[1,#]&/@data)]

σ[[2]]=Sqrt[pmcx[2,data].(data-μ[[2]])^2/Plus@@(pmcx[2,#]&/@data)]

pa[[1]]=Plus@@(pmcx[1,#]&/@data)/Length[data]

pa[[2]]=Plus@@(pmcx[2,#]&/@data)/Length[data]

To estimate the unknown parameters from the data, we first initialize to random values:

In[85]:=
```
μ = RandomReal[{-10, 10}, 2]; σ = RandomReal[{0, 10}, 2];
pa = {temp = RandomReal[], 1 - temp};
```

The interate for i=1 to iter. Note that the E-step occurs on the right-hand side, where pmcx[] is a function of the most recent value of the mean, std, and mixing parameters.

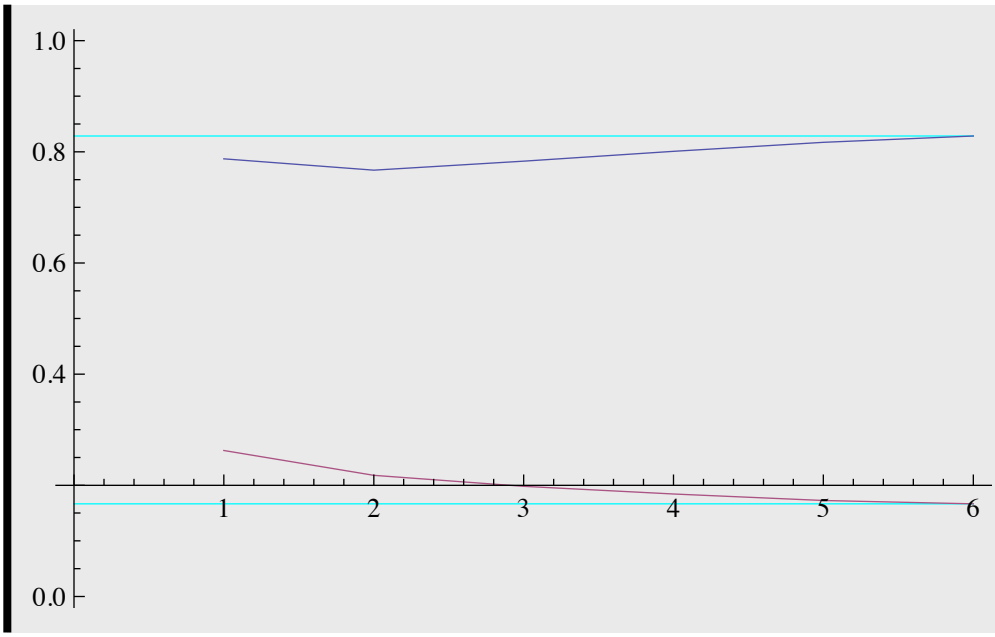
In[86]:=
```
iter = 6;
μparameterList = Table[0, {a, 1, 2}, {k, 1, iter}];
σparameterList = μparameterList; paparameterList = μparameterList;
For[k = 1, k ≤ iter, k++,
  For[a = 1, a ≤ 2, a++,
    μ[[a]] = pmcx[a, data].data / Plus @@ (pmcx[a, #] & /@ data);
    σ[[a]] = Sqrt[pmcx[a, data].(data - μ[[a]]) ^ 2 /
       Plus @@ (pmcx[a, #] & /@ data)];
    pa[[a]] = Plus @@ (pmcx[a, #] & /@ data) / Length[data];
    μparameterList[[a, k]] = μ[[a]];
    σparameterList[[a, k]] = σ[[a]];
    paparameterList[[a, k]] = pa[[a]];
   ];
 ];
```

■ Plot the evolution of the mixing parameters

Blue line shows true values from the generative model.

In[90]:=
```
Show[{Plot[{pa[[1]], pa[[2]]}, {x, 0, iter}, PlotStyle → Hue[0.5]],
  ListPlot[{paparameterList[[1]], paparameterList[[2]]}, Joined → True]},
 PlotRange → {0, 1}]
```

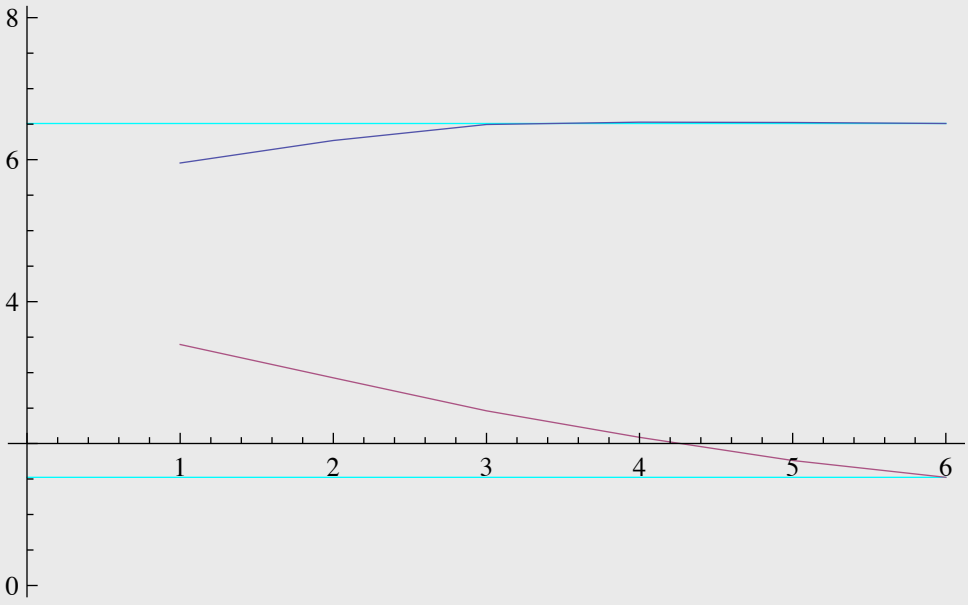
Out[90]=



■ Plot the evolution of the means

In[91]:=
```
Show[{Plot[{μ[[1]], μ[[2]]}, {x, 0, iter}, PlotStyle → Hue[0.5]],
  ListPlot[{μparameterList[[1]], μparameterList[[2]]}, Joined → True]},
 PlotRange → {0, 8}]
```

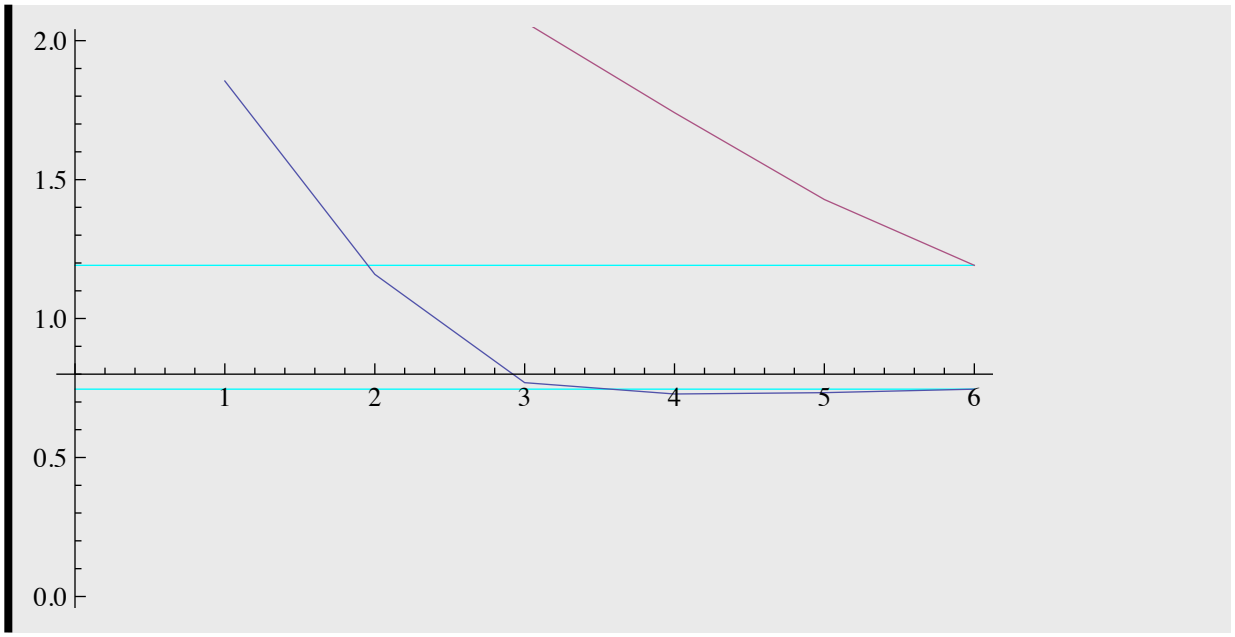
Out[91]=



■ **Plot the evolution of the standard deviations**

In[92]:=
```
Show[{Plot[{σ[[1]], σ[[2]]}, {x, 0, iter}, PlotStyle → Hue[0.5]],
  ListPlot[{σparameterList[[1]], σparameterList[[2]]}, Joined → True]},
 PlotRange → {0, 2}]
```

Out[92]=



EM: Theory for arbitrary probability distributions

We'd like to generalize the theory, and also make clearer the function of EM in integrating out hidden or secondary variables. The notation s, and h is used for an unknown parameter s (or a set s), that we want to estimate (e.g. the means and covariances), and another parameter h (or set {hi}) that is hidden (e.g. an indicator variable representing the probability of mixing, p(a)).)

We'd like to find the value of s that maximizes the likelihood of the data {xi}, given other intervening variables hi.

The log-likelihood of the observations is given by:

$$\log p(x_1, \ldots, x_N|s) = \log \prod_{i=1}^{N+1} p(x_i|s) = \sum_i \log p(x_i|s) = \sum_i \log \sum_{h_i} p(x_i, h_i|s)$$

To find the maximum of the likelihood, we calculate the derivative of the log-likelihood with respect to s , and then set the derivative equal to zero:

$$\frac{\partial \log p(x_1, \ldots, x_N|s)}{\partial s} = \sum_i \frac{1}{\sum_{h_i} p(x_i, h_i|s)} \frac{\partial}{\partial s} \{\sum_{h_i} p(x_i, h_i|s)\}$$

Where we have used the relation (that you can prove from basic calculus):

$$\frac{\partial \log \phi(s)}{\partial s} = \frac{1}{\phi(s)} \frac{\partial \phi(s)}{\partial s}$$

Bringing the summation over h_i towards the front, we have:

$$\sum_i \sum_{h_i} \frac{1}{\sum_{h_i} p(x_i, h_i|s)} \frac{\partial}{\partial s} p(x_i, h_i|s)$$

Again using the above derivative of a log relation, we have

$$\sum_i \sum_{h_i} \frac{p(x_i, h_i|s)}{\sum_{h_i} p(x_i, h_i|s)} \frac{\partial}{\partial s} \log p(x_i, h_i|s)$$

Using Bayes rule and setting the expression to zero, we want to find s that satisfy:

$$\sum_i \sum_{h_i} p(h_i|x_i, s) \frac{\partial}{\partial s} \log p(x_i, h_i|s) = 0$$

■ Summary of EM strategy

So following the above reasoning for the Gaussian case, the EM strategy for solving equation for s involves two steps:

1) Given a guess, s=s_t at step t , calculate

$$p(h_i|x_i, s_t) \qquad (E - step)$$

2) Us this to solve

$$\sum_i \sum_{h_i} p(h_i|x_i, s_t) \frac{\partial}{\partial s} \log p(x_i, h_i|s) = 0 \qquad (M - step)$$

to find the next s=s_{t+1} . Then iterate back to the E-step until convergence. Although the EM algorithm does converge, it doesn't necessarily converge to the maximum likelihood estimate.

Expectation Maximization -- Segmentation simulation

We've studied the problem of interpolation given missing data. We motivated the problem by the visual phenomenon of surface completion. Another aspect of surface perception is our ability to take noisy data (e.g. depth cues), and not only interpolate the data, but also decide which of several surfaces the data belong using stereo data (Madarasmi et al., 1993; Kersten & Madarasmi, 1995) or optic flow. A number of ways have been proposed to deal with this problem. EM (described above) is a general statistical technique developed in the 1970's that appears in various forms in many algorithms, including belief propagation. The algorithm has been applied to the surface estimation problem, e.g. from optic flow (Jepson, 1993; Weiss, 1997). We go to Yair Weiss again for a nice simple tutorial and demo.

Consider **Generative Model 1.**

Generative models

Two lines with slopes and intercepts (a1,b1) and (a2,b2).

In[707]:=
```
ndist = NormalDistribution[0, 2];
```

■ Generative model 1

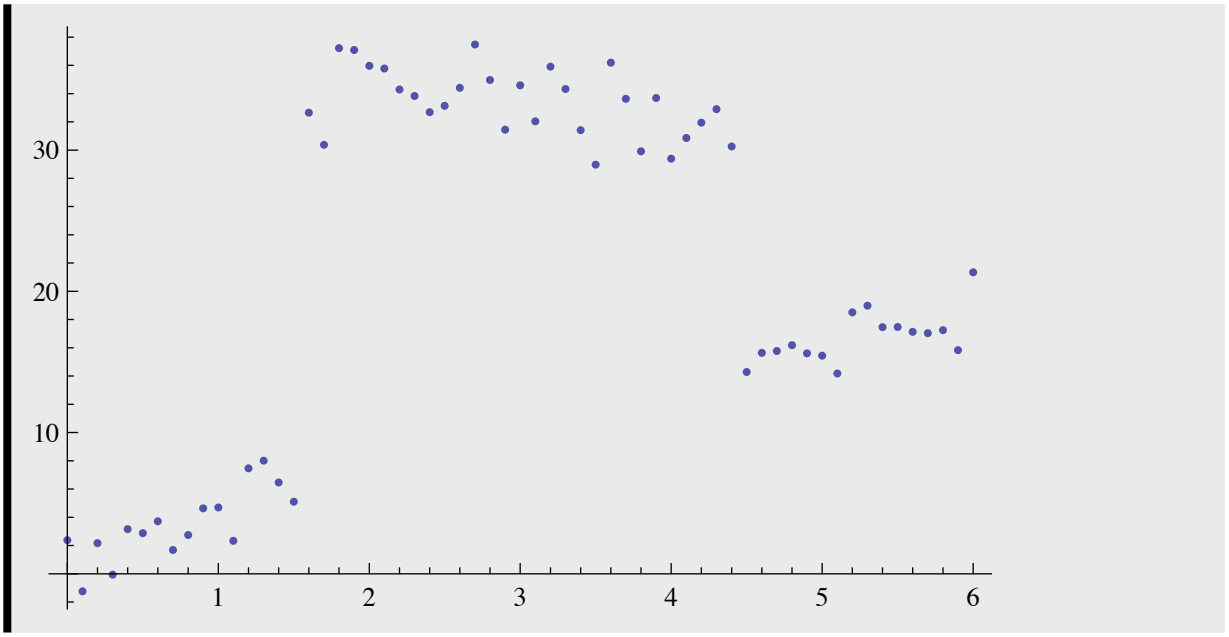
In[708]:=
```
y1[x_] := -2 * x + 40 + Random[ndist];
y2[x_] := 3 * x + 1 + Random[ndist];
data = Table[{x, If[Abs[x - 3] < 1.5, y1[x], y2[x]]}, {x, 0, 6, .1}];
{x, y} = Transpose[data];
```

■ **Generative model 2 (you will need to make the cell property "evaluatable" to work)**

In[626]:=
```
ndist = NormalDistribution[0, .25];
y1[x_] := -2 x + 15 + Random[ndist];
y2[x_] := 3 x + 1 + Random[ndist];
data = Table[{x, If[RandomReal[] < 0.5, y1[x], y2[x]]}, {x, 0, 6, 0.1}];
{x, y} = Transpose[data];
```

In[712]:=
```
gdata = ListPlot[data]
```

Out[712]=



EM algorithm

The basic logic of the EM algorithm is as follows:

Start with random parameter values (slopes and intercepts, or a's and b's) for the two models.

Iterate the E and M steps until convergence:

1. E-step: assign points to the line that are the best fit

2. M-step: update the line parameters using only the points assigned to it

■ Initialize parameters to random values

In[713]:=
```
σ = 0.1;
{a1, b1, a2, b2} = Table[10 RandomReal[], {4}];
```

■ E-step

Compute residuals r1, r2, the error in the predicted and actual y values under each of the two models.

In[723]:=
```
r1 = a1 * x + b1 - y;
r2 = a2 * x + b2 - y;
```

Now we could just assign points based on which line parameters give the smallest residual. But we'd like to take into account our model of the conditional mixing probabilities, which also depend on the noisiness in the data. So we'll compute weights that correspond to the conditional mixing probabilities.

Using the residuals, compute weights. We'll assign these weights to data in the M-step later.

In[725]:=
```
w1 = Exp[-r1^2 / σ] / (Exp[-r1^2 / σ] + Exp[-r2^2 / σ]);
w2 = Exp[-r2^2 / σ] / (Exp[-r1^2 / σ] + Exp[-r2^2 / σ]);
```

■ M-step

Standard linear regression doesn't assume that the data may have come from different sources. The least-squares solution (derivable from i.i.d. gaussian model for the data) is equivalent to solving:

$$\begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & \sum_i 1 \end{pmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix}$$

But if the data come from different sources, with above weights we can compute *weighted* linear regression to estimate the slope and intercept parameters:

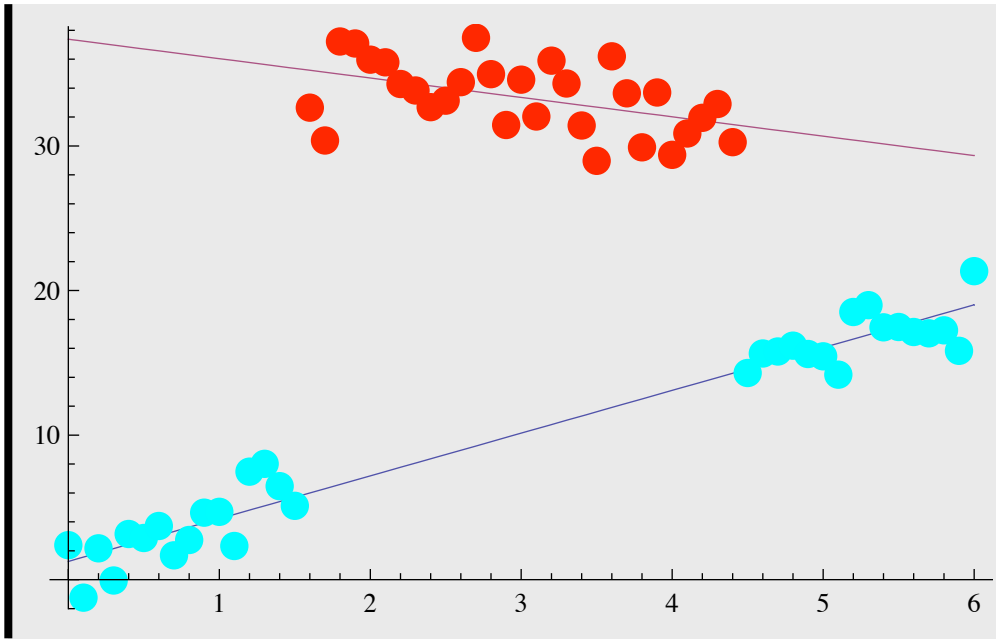
$$\begin{pmatrix} \sum_i w_i x_i^2 & \sum_i w_i x_i \\ \sum_i w_i x_i & \sum_i w_i 1 \end{pmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i w_i x_i y_i \\ \sum_i w_i y_i \end{bmatrix}$$

(For simplicity, we've dropped the subscript on the weights that indicates whether it is line 1 or line 2, and just keep the subscript indicating the weight for point i).

In[727]:=
```
{a1, b1} = Inverse[{{w1.(x x), w1.x}, {w1.x, Apply[Plus, w1]}}].
   {w1.(x y), w1.y};
{a2, b2} = Inverse[{{w2.(x x), w2.x}, {w2.x, Apply[Plus, w2]}}].
   {w2.(x y), w2.y};
```

In[729]:=
```
gfit = Plot[{a1 x + b1, a2 x + b2}, {x, 0, 6}];
Show[
 {gfit,
  Graphics[{PointSize[0.03],
    Transpose[{(Hue[#1] &) /@ w1/2, Point /@ data}]}]}, PlotRange → All]
```

Out[730]=



Now manually run through the E and M steps again...and again, until convergence.

Exercises

Run EM with Generative Model 2. Increase the additive noise. How does attribution accuracy change? ("attribution" means assigning a point to its correct line)

N=5 Multivariate gaussians, only data known: Implement EM to estimate means, variance, and mixing probabilities

---

■ **Implement EM to estimate means, variance, and mixing probabilities**

$$p(a|x_i, \mu_{a,t}) = \frac{p(x_i|a, \mu_{a,t})p(a)}{\sum_a p(x_i|a, \mu_{a,t})p(a)}$$

$$\mu_{a,t+1} = \frac{\sum_i x_i p(a|x_i, \mu_{a,t})}{\sum_i p(a|x_i, \mu_{a,t})}.$$

$$C_a(t+1) = \frac{\sum_i (x_i - \mu_a)(x_i - \mu_a)^T p_t(a|x_i)}{\sum_i p_t(a|x_i)}$$

$$p_{t+1}(a|x_i,) = p_t(a|x_i, \mu_a(t), C_a(t), p_t(a))$$

References

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Roy. Stat. Soc., B39*, 1-38.

Frey, B. J. (1998). *Graphical Models for Machine Learning and Digital Communication*. Cambridge, Massachusetts: MIT Press.

Gershenfeld, N. A. (1999). *The nature of mathematical modeling*. Cambridge ; New York: Cambridge University Press.

Jepson, A., & Black, M. J. (1993). *Mixture models for optical flow computation*. Paper presented at the Proc. IEEE Conf. Comput. Vsion Pattern Recog., New York.

Kersten, D., & Madarasmi, S. (1995). The Visual Perception of Surfaces, their Properties, and Relationships. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 19*, 373-389.

Madarasmi, S., Kersten, D., & Pong, T.-C. (1993). The computation of stereo disparity for transparent and for opaque surfaces. In C. L. Giles & S. J. Hanson & J. D. Cowan (Eds.), *Advances in Neural Information Processing Systems 5*. San Mateo, CA: Morgan Kaufmann Publishers.

Weiss, Y. (1997). *Smoothness in Layers: Motion segmentation using nonparametric mixture estimation*. Paper presented at the Proceedings of IEEE conference on Computer Vision and Pattern Recognition.

Yuille, A., Coughlan J., Kersten D.(1998) (pdf)