

# Introduction to Neural Networks

## Daniel Kersten

### Clustering, K-Means, Expectation-Maximization (EM)

#### Initialize

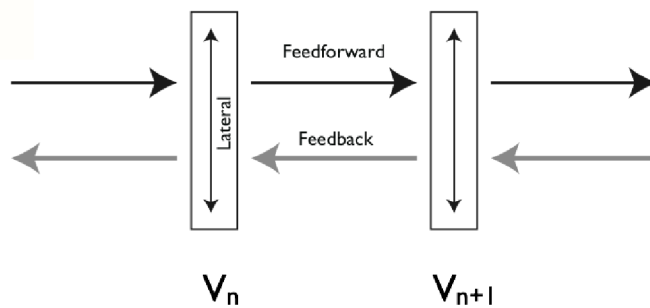
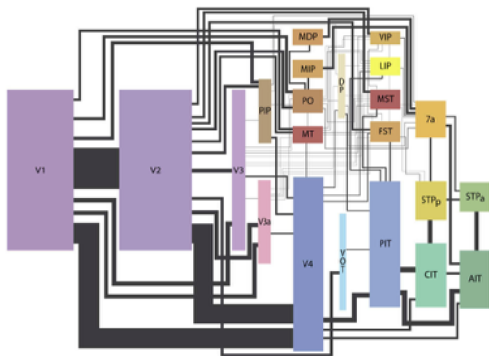
Read in Statistical and Graphics Add-in packages:

```
Off[General::spell1];  
Needs["ErrorBarPlots`"];  
Needs["MultivariateStatistics`"];
```

---

## Review: neural architecture of vision

Hierarchical view of visual neural architecture consisting of lateral, feedforward and feedback computations.



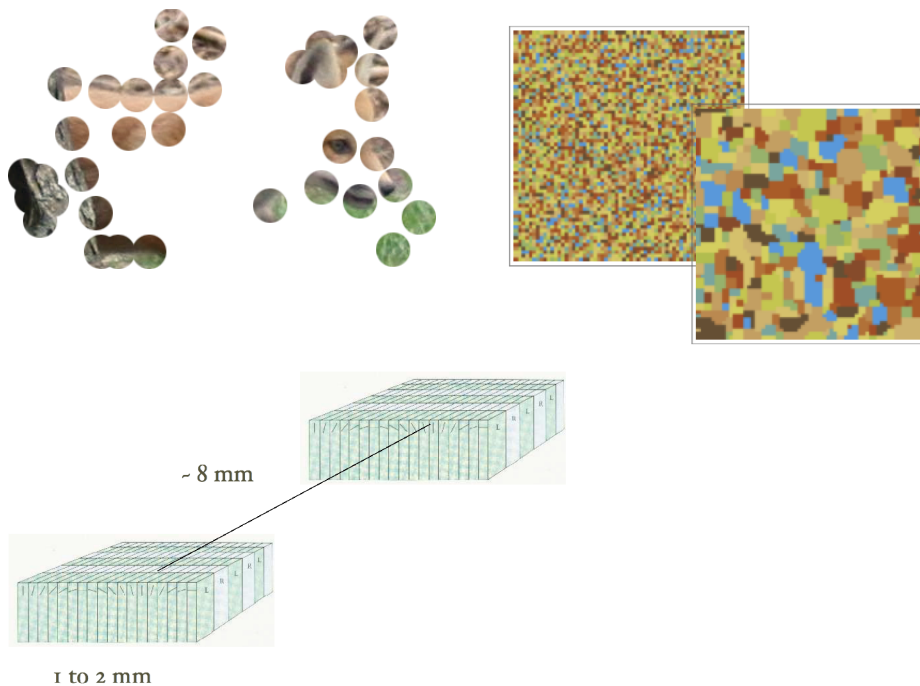
## Lateral organization/computation

Assumption is that lateral organization represents features or “concepts” at a given level of abstraction. E.g. an “edge” is a concept, but so is “elbow”, “face”, “body” and “scene”.

Self-organization of maps based on bringing similar features at the same level of abstraction close together on the cortical gray matter. E.g. edge orientation, spatial frequency, location in V1. Poses of the human head in the columnar structure of temporal areas.

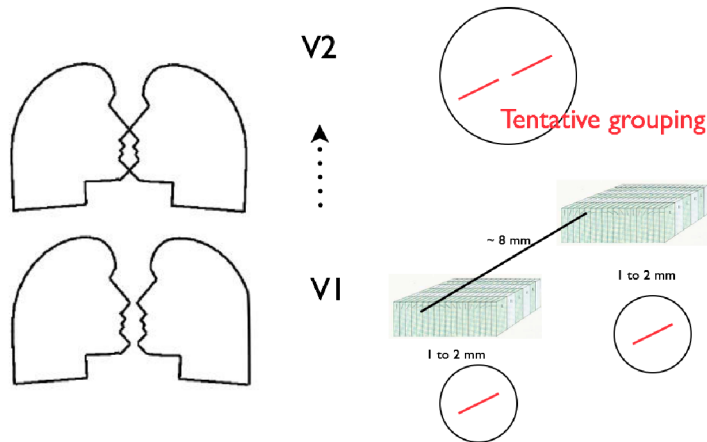
Self-organization of receptive fields based on efficient coding. With connections to classical dimensional-reduction methods such as principal components analysis, sparse coding/dictionary methods (including ICA), and contingent adaptation to maximize representational capacity. The idea is that statistical regularities in visual sensory patterns can be exploited to recode information into forms that are less redundant.

Examples of lateral computations involving active linking and suppression of features that are likely or unlikely to be grouped.



Das A, Gilbert CD (1999) Topography of contextual modulations mediated by short-range interactions in primary visual cortex. *Nature* 399:655-661.

One of the problems of early local linking is that one may suffer from premature commitment--i.e. pass along the wrong assignments to a higher level.

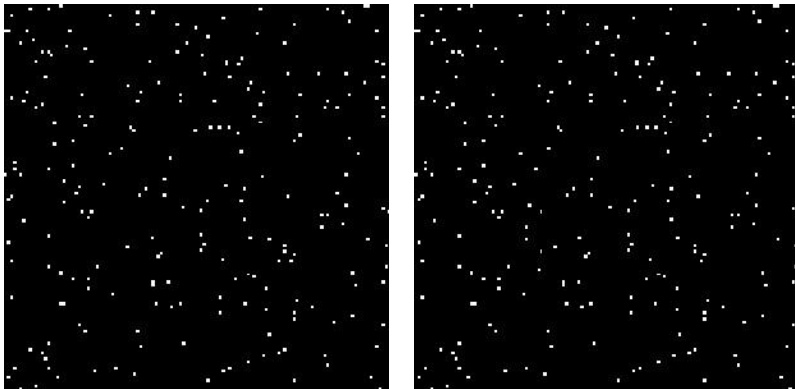
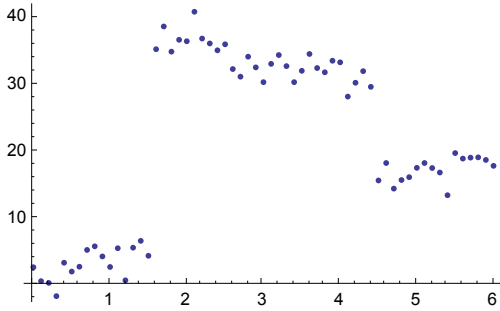


This is an argument for a “probabilistic principle of least commitment”, and for probabilistic representations of information which we discussed last week.

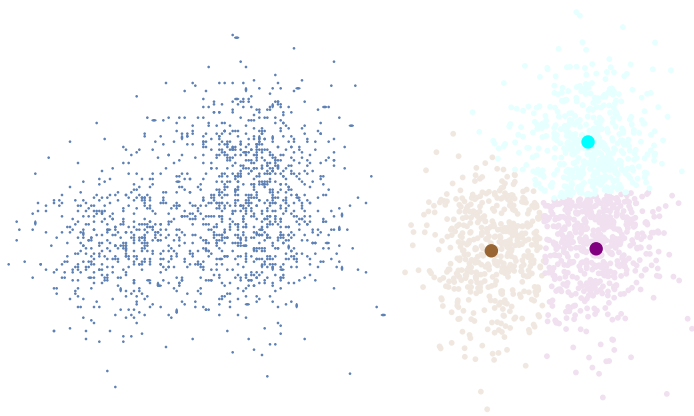
## Motivation for today

Today we are going to study EM, the “expectation maximization” algorithm. We will be estimating the parameters of probability distributions.

We’ll revisit linear regression but where the data come from one of two lines, but we aren’t told which line. This problem is also encountered in vision where the visual system decides which of several surfaces the visual features come from.



Given a distribution of patterns, for example below where each point represents a pattern with 2 features, can one discover an underlying organization that assigns each pattern to one of  $n$  clusters, either exclusively or with a probability of membership:



## Clustering

The larger goal is: given a set of unlabeled data  $x$ , 1) decompose them into  $M$  classes ( $w_1, \dots, w_M$ ), where  $M$  may or may not be known. 2) Learn  $p(x|w)$ ; 3) Discover new classes.

We will look at the case of discovering classes when  $M$  is known, first with a deterministic model (K-means), and then with a probabilistic model where the underlying generative processes has hidden or latent variables (EM).

## K-means

Assume that data  $x$  in  $D$  is clustered around unknown mean values,  $m_a$ . Assume the # of clusters is known -- fixed at  $k$ .

Simultaneously associate data to the means *and* estimate the means by minimizing:

$$\sum_{a=1}^k \sum_{x \in D_a} (x - m_a)^2$$

There are various algorithms. Deterministic K-means works by 1) randomly choosing data points  $x$  and put them in  $k$  sets  $D_a$ ,  $a = 1$  to  $k$ . 2) calculate the mean of each cluster; 3) For each data point calculate how far it is from each of these provisional means and assign it to the nearest one; 4) repeat 2 and 3 until convergence.

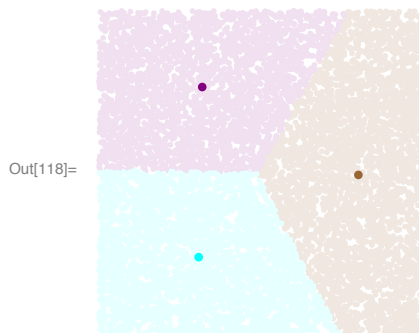
```
In[112]:= data = RandomReal[{-1, 1}, {10^4, 2}];
```

```
In[116]:= cc = ClusteringComponents[data, 3, 1, Method -> "KMeans",
  "DistanceFunction" -> SquaredEuclideanDistance, "RandomSeed" -> 1];
```

```
In[117]:= means = Mean /@ {Pick[data, cc, 1], Pick[data, cc, 2], Pick[data, cc, 3]}
```

```
Out[117]:= {{0.608332, -0.0165405}, {-0.360212, 0.528569}, {-0.382107, -0.526764}}
```

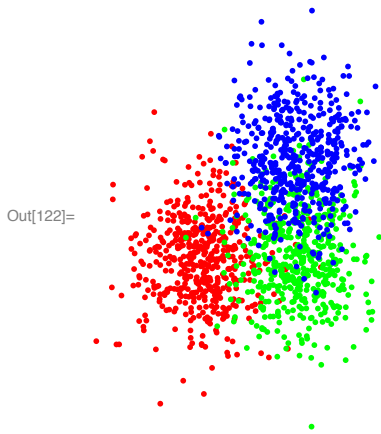
```
In[118]:= Graphics[{LightBrown, Point@Pick[data, cc, 1], LightPurple, Point@Pick[data, cc, 2],
  LightCyan, Point@Pick[data, cc, 3], {Brown, PointSize[Medium], Point@means[[1]]},
  {Purple, PointSize[Medium], Point@means[[2]]},
  {Cyan, PointSize[Medium], Point@means[[3]]}}
```



Here's a second example:

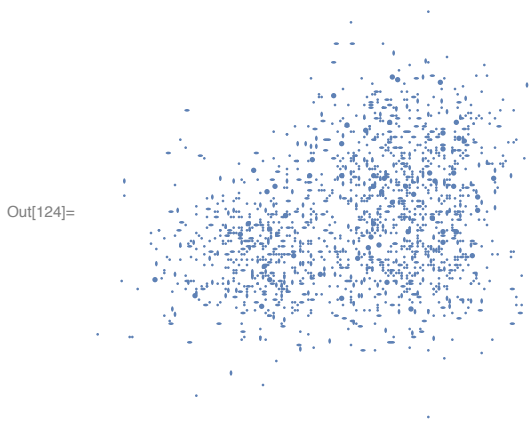
Generative model:

```
In[119]:= x1 = RandomVariate[MultinormalDistribution[{0, 0}, {{1, 0}, {0, 2}}], 500];
x2 = RandomVariate[MultinormalDistribution[{3, 0}, {{1, 0}, {0, 2}}], 500];
x3 = RandomVariate[MultinormalDistribution[{3, 3}, {{1, 0}, {0, 2}}], 500];
Graphics@{Red, Point@x1, Green, Point@x2, Blue, Point@x3}
```



Given unlabeled data:

```
In[123]:= data2 = Join[x1, x2, x3];
ListPlot[data2, AspectRatio -> 1, Axes -> False]
```

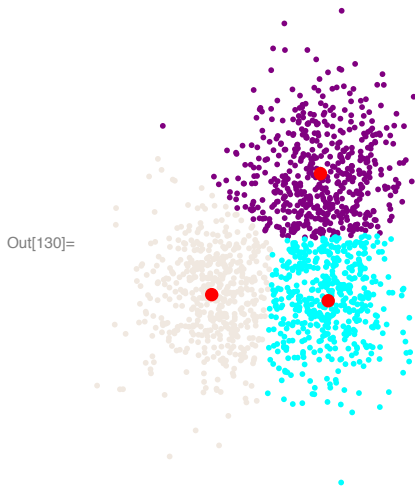


```
In[125]:= cc2 = ClusteringComponents[data2, 3, 1, Method -> "KMeans",
  "DistanceFunction" -> SquaredEuclideanDistance, "RandomSeed" -> 1];
```

```
In[126]:= means = Mean /@ {Pick[data2, cc2, 1], Pick[data2, cc2, 2], Pick[data2, cc2, 3]}
```

```
Out[126]= {{-0.0867872, -0.0846732}, {2.88889, 3.22616}, {3.10053, -0.252354}}
```

```
In[130]:= Graphics[{LightBrown, Point@Pick[data2, cc2, 1], Purple, Point@Pick[data2, cc2, 2],
  Cyan, Point@Pick[data2, cc2, 3], {Red, PointSize[Large], Point@means[[1]]},
  {Red, PointSize[Large], Point@means[[2]]},
  {Red, PointSize[Large], Point@means[[3]]}}]
```



## EM

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

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

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}}]
0.9975
```

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

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

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

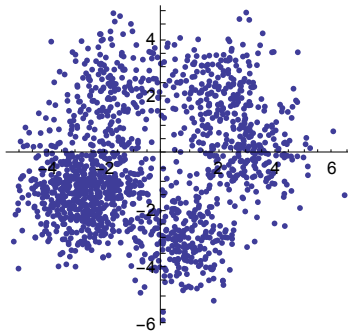
Thus our stochastic generative process can be written:

```
RandomReal[ndist[μ[[t = randomindex]], Σ[[t]]]]
```

```
{-0.412204, -3.19202}
```

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

```
ListPlot[scatterdata, AspectRatio → Automatic, ImageSize → Small]
```

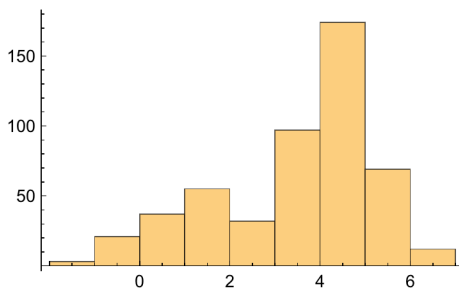


For mixture distributions, we may not be directly interested in the mixing probabilities--but are interested in the parameters of the distributions, e.g. the 5 means and their standard deviations. The means, in an N-dimensional space for example, could represent prototypes in memory, and standard deviations the range of variability in the exemplars.

### Simple case & intuition

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  $x$  of the light intensity.

We want to estimate the two means from a series of measurements (e.g. photon counts corresponding to light intensity),  $\{x_i: i=1, \dots, N\}$ , 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  $a=1,2$ .

(Recall that we've used indicator functions before, as in the example of interpolation with missing data.)



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 hidden variables that 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:

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

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  $p(a | x_i, \mu_a)$  ( $= p(a | x_i)$ ). But the K-means example provides a clue. We could guess the two means, then assign each data point to one of them based on which is closer. Then estimate the means again based on this provisional assignment. And then repeat these two steps.

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

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

We are given a sequence of vector measurements  $\{x_i : i=1, \dots, N\}$  and wish to estimate the means, the covariances and mixing parameters.

Let the probability that  $x_i$  came from mixture component  $a$  be  $p(x_i|a)$ . We assume independence, so the likelihood is:

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

The log-likelihood is given by:

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

For the moment to keep things simple, 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 finding values of the means where the gradient values of  $E$  are zero:

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

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  $\mu_j$ 's ( $= \mu_a$ '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.

*Recall, the problem, of course, is that we don't know  $p(a|x_i)$ . It depends on  $\mu_a$  which we want to estimate..., so to get started we will just guess values for the means,  $\mu_a$ , and proceed....*

## 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,  $x_i$ , belongs to (e.g.  $a$  = 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 proceed to update our estimates of the mean.

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

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

The EM algorithm then proceeds by going back to the E-step to recompute  $p(a|x_i, \mu_a)$  using the updated  $\mu_a$ , followed by another M-step, and so on, until convergence.

(It has been shown that in general, EM will converge to a local minimum of the likelihood function.)

### How about the covariance?

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

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

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  $\mathbf{ab}^T$  is the outer product between vectors  $\mathbf{a}$  and  $\mathbf{b}$ ). The mixing probabilities are estimated by:

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

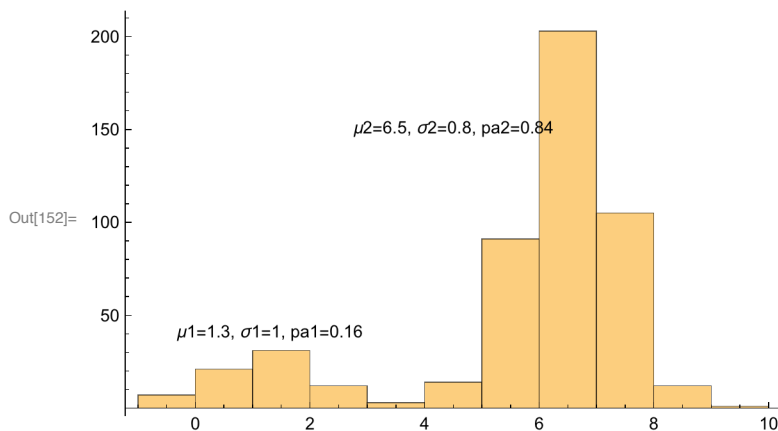
## Demo: Two gaussians, unknown $\mu$ 's $\sigma$ 's , and mixing probabilities

### Generating samples from a Gaussian Mixture Distribution

#### Generative model

```
In[148]:=  $\mu_1 = 1.3; \sigma_1 = 1; \mu_2 = 6.5; \sigma_2 = 0.8; a_1 = 0.16; a_2 = 1 - a_1;$ 
p1dist = NormalDistribution[ $\mu_1$ ,  $\sigma_1$ ];
p2dist = NormalDistribution[ $\mu_2$ ,  $\sigma_2$ ];
p1[x_] := PDF[p1dist, x];
p2[x_] := PDF[p2dist, x];
samplemix := If[RandomReal[] < a1, RandomReal[p1dist], RandomReal[p2dist]];

In[151]:= data = Table[samplemix, {i, 1, 500}];
Histogram[data, Epilog ->
  {Text[" $\mu_1 =$ " <> ToString[ $\mu_1$ ] <> ",  $\sigma_1 =$ " <> ToString[ $\sigma_1$ ] <> ", pa1=" <> ToString[a1],
    { $\mu_1$ , 40}], Text[" $\mu_2 =$ " <> ToString[ $\mu_2$ ] <> ",  $\sigma_2 =$ " <>
    ToString[ $\sigma_2$ ] <> ", pa2=" <> ToString[a2], { $\mu_2 - 2$ , 150}]}]
```



### EM algorithm--Now learn all the parameters--means, standard deviations, and mixing probabilities from the data

```
In[153]:= pxdm[x_, mu_,  $\sigma$ _] := PDF[NormalDistribution[mu,  $\sigma$ ], 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[154]:= 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]]);
SetAttributes[pmcx, Listable]; (*pmcx is made listable so that
  that it will automatically map over the data below*)
```

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

Then iterate 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[157]:= 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]];
  ];
];
```

## Graph of the evolution of the mixing parameters

Blue line shows true values from the generative model.

```
In[161]:= ga = Show[{Plot[{pa[[1]], pa[[2]]}, {x, 0, iter}, PlotStyle -> Hue[0.5]],
  ListPlot[{paparameterList[[1]], paparameterList[[2]], Joined -> True}],
  PlotRange -> {0, 1}, AxesOrigin -> {0, 0}, AxesLabel -> {"iteration", "pa"}];
```

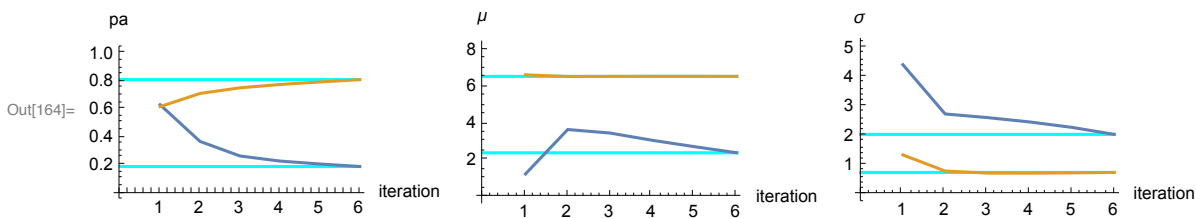
### Graph of the evolution of the means

```
In[162]:= gmeans = Show[{Plot[{μ[[1]], μ[[2]]}, {x, 0, iter}, PlotStyle -> Hue[0.5]],
  ListPlot[{μparameterList[[1]], μparameterList[[2]], Joined -> True}],
  PlotRange -> {0, 8}, AxesOrigin -> {0, 0}, AxesLabel -> {"iteration", "μ"}];
```

### Graph of the evolution of the standard deviations

```
In[163]:= gsds = Show[{Plot[{σ[[1]], σ[[2]]}, {x, 0, iter}, PlotStyle -> Hue[0.5]],
  ListPlot[{σparameterList[[1]], σparameterList[[2]], Joined -> True}],
  PlotRange -> {0, 5}, AxesOrigin -> {0, 0}, AxesLabel -> {"iteration", "σ"}];
```

```
In[164]:= GraphicsRow[{ga, gmeans, gsds}, ImageSize -> Large]
```



## 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  $s$  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  $\{h_i\}$ ) 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  $\{x_i\}$ , given other intervening variables  $h_i$ .

The log-likelihood of the observations is given by:

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

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:

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

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

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

2) Use this to solve

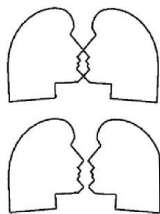
$$\sum_i \sum_{h_i} p(h_i|x_i, s_t) \frac{\partial}{\partial s} \log p(x_i, h_i|s) = 0 \quad (M - step)$$

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.

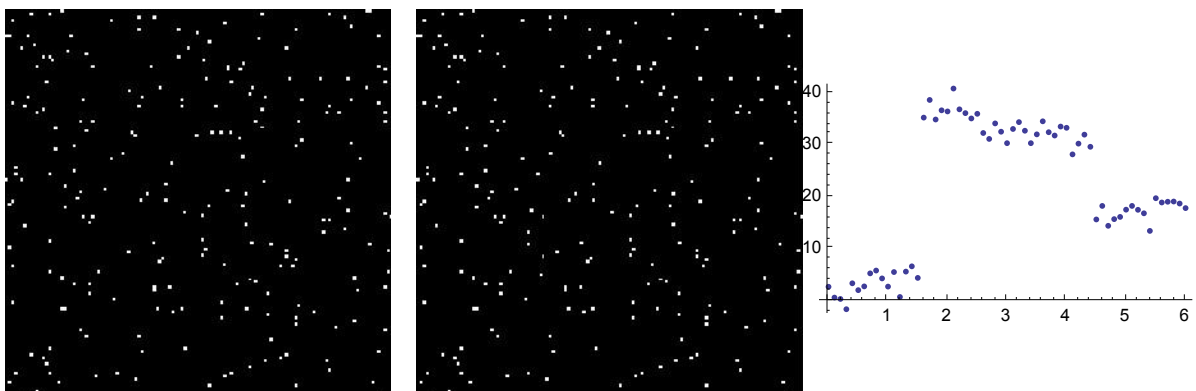
---

## 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. In a previous example, we used a local smoothness constraint. Another way is to do a parametric fit, which imposes global, rather than local, constraints. For example, a global constraint might look for surface fits among planes or quadratic surfaces, or splines. Or even more complex shape models could be invoked. In the earlier ape-human figure, one could use a higher-level, global shape model to disambiguate local edge groupings where the noses meet.



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. This problem appears with stereo and motion data (Madarasmi et al., 1993; Kersten & Madarasmi, 1995) or optic flow (Jepson, 1993; Weiss, 1997).



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 simple tutorial and demo.

Consider first **Generative Model 1**.

## Generative models

Two lines with (slopes, intercepts) =  $(a_1, b_1)$  and  $(a_2, b_2)$ .

```
In[165]:= ndist = NormalDistribution[0, 2];
```

### Generative model I

```
In[166]:= {at1, bt1} = {-2, 40};
{at2, bt2} = {3, 1};
y1[x_] := at1 * x + bt1 + Random[ndist];
y2[x_] := at2 * x + bt2 + 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)

```

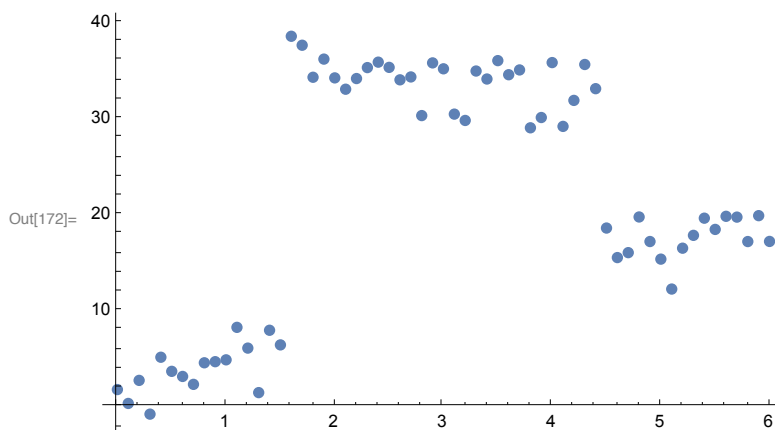
{at1, bt1} = {-2, 15};
{at2, bt2} = {3, 1};

ndist = NormalDistribution[0, .25];
y1[x_] := at1 * x + bt1 + Random[ndist];
y2[x_] := at2 * x + bt2 + Random[ndist];

data = Table[{x, If[RandomReal[] < 0.5, y1[x], y2[x]]}, {x, 0, 6, 0.1}];
{x, y} = Transpose[data];

```

In[172]:= **gdata = ListPlot[data]**



## EM algorithm

We now pretend we don't know the values of the slopes and intercepts, and want to estimate them from the data.

The simplified 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

...but with EM we will do "soft" assignment where probabilities are given to each data point as to whether it might have been caused by line 1 vs. line 2. And in the M-step, we will use these weights to do weighted linear regression.

### Initialize parameters to random values

```

In[173]:=  $\sigma = 0.1;$ 
{a1, b1, a2, b2} = Table[10 RandomReal[], {4}];

```

### E-step

Compute residuals  $r_1, r_2$ , the error in the predicted and actual y values under each of the two models.

```
In[184]:= 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[186]:= 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).

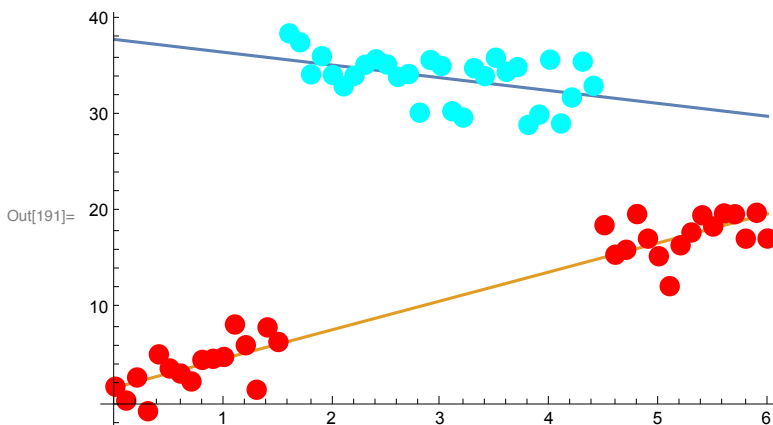
```
In[188]:= {a1, b1} = Inverse[{{w1.(x x), w1.x}, {w1.x, Apply[Plus, w1]}}].{w1.(x y), w1.y};
          {a2, b2} = Inverse[{{w2.(x x), w2.x}, {w2.x, Apply[Plus, w2]}}].{w2.(x y), w2.y};
```

```
In[190]:= 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]
```



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

```
In[192]:= {{at1, bt1}, {a1, b1}, {at2, bt2}, {a2, b2}} // TableForm
Out[192]//TableForm=
  - 2          40
  - 1.32923    37.8966
  3           1
  3.0105      1.72034
```

---

## 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.
- Gershensfeld, 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. Vision Pattern Recog., New York.
- Kersten, D., & Madarasmı, S. (1995). The Visual Perception of Surfaces, their Properties, and Relationships. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, *19*, 373-389.
- Madarasmı, 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)

© 2000, 2001, 2003, 2005, 2007, 2009, 2011, 2012, 2014 Daniel Kersten, Computational Vision Lab, Department of Psychology, University of Minnesota.  
(<http://vision.psych.umn.edu/www/kersten-lab/kersten-lab.html>)