Initialize

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

---

**Last time**

Generative modeling: Multivariate gaussian, mixtures

- Drawing samples
- Mixtures of gaussians
- Will use mixture distributions in a future lecture on EM application to segmentation

---

**Introduction to Optimal inference and task**

**Introduction to Bayesian learning**

---

**Perceptual surface completion**

If you view the left and right images below in stereo (cross your eyes so that the left image goes to the right eye, and the right image to the left eye), you may see a horizontal rectangle floating out in front of the vertical rectangle in the back:
If you can’t perceptually fuse the two images to see the stereo figure, the figure below gives some idea of what it looks like (I’ve changed the color of the horizontal bar for the purposes of illustration).

This is an example of your visual system interpolating a smooth surface from sparse data (cf. Nakayama and Shimojo, 1992). The data is sparse because the information about depth comes from the disparities in the left and right vertical edges of the horizontal rectangle. The random dot stereogram that we saw in an earlier lecture was “dense” rather than sparse. The data is dense because there were lots of potential features to match throughout the background and the square that floated out in depth. Here’s an example of a sparse random dot stereogram:
One sees the white points on dark, central surface floating in front of a background surface, also with a white point texture. How can one model processes of interpolation? We first approach the problem by constructing a cost function to be minimized. And in the second half of the lecture, we re-formulate the problem from a Bayesian perspective in order to introduce Belief Propagation.

**Interpolation using smoothness: Gradient descent**

For simplicity, we'll assume 1-D as in the lecture on sculpting the energy function. In anticipation of formulating the problem in terms of a graph that represents conditional probability dependence, we represent *observable* depth cues by $y^*$, and the true (*"hidden"*) depth estimates by $y$.

(Figure from Weiss (1999).)
First-order smoothness

Earlier we saw that under specific assumptions, biologically plausible neural updating can be seen to decrease the value of an energy (or "cost") function.

One can also start off with an assumed cost function, determined by a set of contraints, and use gradient descent to derive an update rule that minimizes the cost. (See supplementary material). However, such an update rule will not necessarily resemble a biologically plausible neural mechanism.

For an interpolation problem, we can write the energy or cost function by:

\[
J(Y) = \sum_k w_k (y_k - y_k^*)^2 + \lambda \sum_i (y_i - y_{i+1})^2
\]

where \(w_k = xs[[k]]\) is an "indicator function", and \(y_k^* = d\), are the data values. The indicator function is 1 if there is data available, and zero otherwise. The second sum represents the sum of the squared differences between neighboring y-values. Minimizing the first sum encourages the estimates of y to be close to the measured values. Minimizing the second sum encourages nearby y-values to be the same. Thus minimizing \(J(Y)\) encourages fidelity to the data where present, and similarity to nearby values where there is no data.

Gradient descent gives the following local update rule:

\[
y_k \leftarrow y_k + \eta_k \left( \lambda \left( \frac{y_{k-1} + y_{k+1}}{2} - y_k \right) + w_k (y_k^* - y_k) \right)
\]

\(\lambda\) is a free parameter that controls the degree of smoothness, i.e. smoothness at the expense of fidelity to the data.

There are various choices for how to change the smoothness as a function of iterations.

E.g. Gauss-Seidel: \(\eta[k_] := 1/(\lambda xs[[k]])\). Successive over-relaxation (SOR): \(\eta2[k_] := 1.9/(\lambda xs[[k]])\);

A simulation: Straight line with random missing data points

- **Make the data**

Consider the problem of interpolating a set of points with missing data, marked by an indicator function with the following notation:

\(w_k = xs[[k]], y^* = data, y = f\).

We'll assume the true model is that \(f = y = j\), where \(j = 1\) to size. data is a function of the sampling process on \(f = j\)
In[79]:= 
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]}];

The green line shows the a straight line connecting the data points. The red dots on the abscissa mark the points where data are missing.

In[80]:= 
Show[{g2, g3}, ImageSize -> Medium]

Out[80]=

Let's set up two matrices, \( T_m \) and \( S_m \) such that the gradient of the energy is equal to:

\[
T_m \cdot f - S_m \cdot f.
\]

\( S_m \) will be our filter to exclude non-data points. \( T_m \) will express the "smoothness" constraint.

In[81]:= 
Sm = DiagonalMatrix[xs];
Tm = Table[0, {i, 1, size}, {j, 1, size}];
For[i = 1, i <= size, i++,
    Tm[[i, i]] = 2];
Tm[[1, 1]] = 1; Tm[[size, size]] = 1; (*Adjust for the boundaries*)
For[i = 1, i < size, i++,
    Tm[[i + 1, i]] = -1];
For[i = 1, i < size, i++,
    Tm[[i, i + 1]] = -1];
Run gradient descent

\begin{verbatim}
In[87]:= Clear[f, d, λ]
    (λ*Tm.Array[f, size] - Sm.((Array[d, size]) - Array[f, size])) //
        MatrixForm;

In[89]:= Clear[Tf,f1,j];
    dt = 1; λ=2;
    Tf[f1_] := f1 - dt*(1/(λ+xS))*(Tm.f1 - λ*Sm.(data-f1));

We will initialize the state vector to zero, and then run the network for \textit{iter} iterations:

\begin{verbatim}
In[92]:= f0 = Table[0,{i,1,size}];
   result=f0;
   f=f0;
   iter=25;

Now plot the interpolated function.

\begin{verbatim}
In[96]:= Animate[result = Nest[Tf, f, iter];
   f = result;
   g1 = ListPlot[result, Joined -> False, AspectRatio -> Automatic,
       PlotRange -> {{0, size}, {-1, size + 1}}, ImageSize -> Medium];
   Show[{{g1, g2, g3}, PlotRange -> {-1, size + 1]}, {j, 1, 10, 1},
     AnimationRunning -> False]
\end{verbatim}
\end{verbatim}
\end{verbatim}
\end{verbatim}
Try starting with $f = \text{random values}$. Try various numbers of iterations.

Try different sampling functions $xs[[i]]$.

---

**Interpolation using Belief Propagation**

**Same interpolation problem, but now using belief propagation**

Example is taken from Yair Weiss (Weiss, 1999)

![Diagram](image)

**Probabilistic generative model**

\[
\text{data}[[i]] = y'[i] = xs[[i]] y[[i]] + d\text{noise}, \quad d\text{noise} \sim N[0, \sigma_D] \quad (1)
\]

\[
y[[i+1]] = y[[i]] + z\text{noise}, \quad z\text{noise} \sim N[0, \sigma_R] \quad (2)
\]

The first term is the "data formation" model, i.e. how the data is directly influenced by the interaction of the underlying influences or causes:

- $y^*$ is determined by an underlying hidden "y" which can't be directly measured. But we assume we can measure $y^*$, which is determined by sampling some values of $y$ and adding noise.

The second term reflects our prior assumptions about the smoothness of $y$, i.e. nearby $y$'s are correlated, and in fact assumed identical except for some added noise. So with no noise the prior reflects the assumption that lines are horizontal—all $y$'s are the same.

**Some theory**

We'd like to know the distribution of the random variables at each node $i$, conditioned on all the data: I.e. we want the posterior

\[
p(Y_i = u \mid \text{all the data})
\]

If we could find this, we'd be able to: 1) say what the most probable value of the $y$ value is, and 2) give a measure of confidence
Updating the mean and variance given the data at point i, and current beliefs about mean and variance before and after i

Let \( p(Y_i = u | \text{all the data}) \) be normally distributed: \( \text{NormalDistribution}[\mu_i, \sigma_i] \).

Consider the ith unit. The posterior \( p(Y_i = u | \text{all the data}) = \) 
\[
p(Y_i = u | \text{all the data}) \propto p(Y_i = u | \text{data before i}) p(\text{data at i | } Y_i = u) p(Y_i = u | \text{data after i})
\]

(“before” and “after” means to the left and right of i, respectively.)

Suppose that \( p(Y_i = u | \text{data before i}) \) is also gaussian:
\[
p(Y_i = u | \text{data before i}) = a_i[u] \sim \text{NormalDistribution}[\mu_i, \sigma_i]
\]

and so is probability conditioned on the data after i:
\[
p(Y_i = u | \text{data after i}) = b_i[u] \sim \text{NormalDistribution}[\mu_i, \sigma_i]
\]

And the noise model for the data:
\[
p(\text{data at i | } Y_i = u) = L_i[u] \sim \text{NormalDistribution}[\mu_Y, \sigma_Y]
\]

So in terms of these functions, the posterior probability of the ith unit taking on the value u can be expressed as proportional to a product of the three factors:
\[
p(Y_i = u | \text{all the data}) \propto a_i[u] * L_i[u] * b_i[u]
\]

This just another gaussian distribution on \( Y_i = u \). What is its mean and variance? Finding the root enables us to complete the square to see what the numerator looks like. In particular, what the mode (=mean for gaussian) is.
In[30]:= \textbf{Solve}\left[-D\left[-\frac{(u - \mu_\alpha)^2}{2\sigma_\alpha^2} - \frac{(u - \mu_\beta)^2}{2\sigma_\beta^2} - \frac{(u - y_P)^2}{2\sigma_0^2}, \ u\right] = 0, \ u\right]

Out[30]= \left\{ \{ u \rightarrow \frac{\mu_\alpha \sigma_\alpha^2 + \mu_\beta \sigma_\beta^2 + y_P}{\sigma_\alpha^2 + \frac{1}{\sigma_\beta^2} + \frac{1}{\sigma_0^2}} \right\}

This suggests that if we had estimates of $\mu_\alpha, \mu_\beta, \sigma_\alpha^2, \sigma_\beta^2$ and the data, we could update the mean of node $i$ using:

$$u \leftarrow \frac{\mu_\alpha \sigma_\alpha^2 + \mu_\beta \sigma_\beta^2 + y_P}{\sigma_\alpha^2 + \frac{1}{\sigma_\beta^2} + \frac{1}{\sigma_0^2}}$$

Similarly, the update rule for the variance is:

$$\sigma^2 \leftarrow \frac{1}{\sigma_\alpha^2} + \frac{1}{\sigma_\beta^2} + \frac{1}{\sigma_0^2}$$

**How do we get $\mu_\alpha, \mu_\beta, \sigma_\alpha, \sigma_\beta$?**

We express the probability of the $i$th unit taking on the value $u$ in terms of the values of the neighbor before, conditioning on what is known (the observed measurements), and marginalizing over what isn't (the previous "hidden" node value, $v$, at the $i$-th location).

We have three terms to worry about that depend on nodes in the neighborhood preceding $i$:

$$\alpha[u] = \int_{-\infty}^{\infty} \alpha_p[v] \ast S[u] \ast L[v] \ dv = \int_{-\infty}^{\infty} e^{-\frac{(v - y_P)^2}{2\sigma_0^2}} \frac{(u - v)^2}{2\sigma_\alpha^2} \frac{(v - \mu_\alpha)^2}{2\sigma_\beta^2} \ dv$$

$$\alpha_p = \alpha_{i-1}. \ S[u] \text{ is our smoothing term, or transition probability: } S[u] = p(u \mid v). \ L[v] \text{ is the likelihood of } v \text{ given the data previous at the previous node.}$$

In[31]:= Rdist = \text{NormalDistribution}[(v, \sigma_\text{R})];
S[u] = \text{PDF}[Rdist, u];

avdist = \text{NormalDistribution}[(\mu_\alpha, \sigma_\alpha)];
\alpha_p[v] = \text{PDF}[avdist, v];

Lp[v] = \text{PDF}[Ddist, v];
Let's find an expression for the mode of the above calculated expression for $\sigma[u]$

$$D\left[\frac{(u - \mu \sigma_p)^2 \sigma_p^2 + \mu \sigma_p^2 \sigma_R^2 + u^2 \sigma_R^2 + y_p^2 (\sigma_R^2 + \sigma_p^2)}{2 (\sigma_R^2 \sigma_p^2 + \sigma_p^2 (\sigma_R^2 + \sigma_p^2))}, u\right]$$

$$\text{Solve}\left[-\% = 0, u\right]$$

$$\text{Simplify}\left[\frac{\mu \sigma_p \sigma_p^2}{\sigma_R^2 \sigma_p^2 + \sigma_p^2 (\sigma_R^2 + \sigma_p^2)} + \frac{y_p \sigma_p^2}{\sigma_R^2 \sigma_p^2 + \sigma_p^2 (\sigma_R^2 + \sigma_p^2)}\right] / (\sigma_R^2 \sigma_p^2)$$
We now have a rule that tells us how to update the $\alpha(u)=p(y_i=\text{data before } i)$, in terms of the mean and variance parameters of the previous node:

$$
\mu_{a} \leftarrow \frac{\mu_{a}\sigma_{a}^{2} + y_{p}\sigma_{a}^{2}}{\sigma_{a}^{2} + \sigma_{D}^{2}}
= \frac{\mu_{a}^{2} + y_{p}^{2}}{\sigma_{a}^{2} + \sigma_{D}^{2}}
= \frac{\mu_{p}^{2} + y_{p}^{2}}{\sigma_{p}^{2} + \sigma_{D}^{2}}
= \frac{\mu_{p}^{2} + y_{p}^{2}}{\sigma_{p}^{2} + \sigma_{D}^{2}}
$$

The update rule for the variance is:

$$
\sigma_{a}^{2} \leftarrow \sigma_{D}^{2} + \sigma_{R}^{2} + \frac{1}{\sigma_{D}^{2} + \sigma_{R}^{2}}
$$

A similar derivation gives us the rules for $\mu_{b}, \sigma_{b}^{2}$

$$
\mu_{b} \leftarrow \frac{\mu_{b}\sigma_{b}^{2} + y_{a}\sigma_{b}^{2}}{\sigma_{b}^{2} + \sigma_{D}^{2}}
= \frac{\mu_{b}^{2} + y_{a}^{2}}{\sigma_{b}^{2} + \sigma_{D}^{2}}
= \frac{\mu_{a}^{2} + y_{a}^{2}}{\sigma_{a}^{2} + \sigma_{D}^{2}}
= \frac{\mu_{a}^{2} + y_{a}^{2}}{\sigma_{a}^{2} + \sigma_{D}^{2}}
$$

$$
\sigma_{b}^{2} \leftarrow \sigma_{D}^{2} + \sigma_{R}^{2} + \frac{1}{\sigma_{D}^{2} + \sigma_{R}^{2}}
$$

Where the subscript index $p$ (for "previous", i.e. unit $i-1$) is replaced by a (for "after", i.e. unit $i+1$).

Recall that sometimes we have data and sometimes we don't. So replace:

$$
y_{p} \rightarrow x_{S[i-1]} \; \text{data[i-1]} = w_{i-1} \; y'_{i-1}
$$

(6)

And similarly for $y_{a}$.
Summary of update rules

The ratio, \( \left( \frac{\sigma_D^2}{\sigma_R^2} \right) \), plays the role of \( \lambda \) above. If \( \sigma_D^2 > \sigma_R^2 \), there is greater smoothing. If \( \sigma_D^2 < \sigma_R^2 \), there is more fidelity to the data. (Recall \( y^* \rightarrow \text{data}, w_k \rightarrow \text{xs}[k] \)). But now we have a principled way of assigning the relative amount of smoothing.

We'll follow Weiss, and also make a (hopefully not too confusing) notation change to avoid the square superscripts for \( \sigma_D^2 > \sigma_D, \sigma_R^2 > \sigma_R \).

\[
\begin{align*}
\mu_i & \leftarrow \frac{\frac{w_i}{\sigma_D} Y_i^* + \frac{1}{\sigma_i^\alpha} \mu_i^\alpha + \frac{1}{\sigma_i^\beta} \mu_i^\beta}{\frac{w_i}{\sigma_D} + \frac{1}{\sigma_i^\alpha} + \frac{1}{\sigma_i^\beta}} \\
\sigma_i & \leftarrow \frac{1}{\frac{w_i}{\sigma_D} + \frac{1}{\sigma_i^\alpha} + \frac{1}{\sigma_i^\beta}} \\
\mu_i^\alpha & \leftarrow \frac{1}{\sigma_i^\alpha} \mu_{i-1}^\alpha + \frac{w_{i-1}}{\sigma_D} Y_{i-1}^* \\
\sigma_i^\alpha & \leftarrow \sigma_R + \left( \frac{1}{\sigma_i^\alpha} + \frac{w_{i-1}}{\sigma_D} \right)^{-1}
\end{align*}
\]

![Diagram](image.png)
A simulation: Belief propagation for interpolation with missing data

In[42]:= size = 32; xs = Table[0, {i, 1, size}]; xs[[1]] = 1; xs[[size]] = 1;
data = Table[N[j] xs[[j]], {j, 1, size}];
g3bp = ListPlot[Table[N[j], {j, 1, size}], Joined -> True,
   PlotStyle -> {RGBColor[0, 0.5, 0]}];
g2bp = ListPlot[data, Joined -> False,
   PlotStyle -> {Opacity[0.35], RGBColor[0.75, 0., 0., 0], PointSize[Large]}];

The green line shows the a straight line connecting the data points. The red dots on the abscissa mark the points where data are missing.

In[43]:= Show[{g2bp, g3bp}, ImageSize -> Medium]
\section*{Initialization}

\begin{verbatim}
In[44]:= size = 32;
\mu 0 = 1;
\mu a = 1; \sigma a = 100 000; (* large uncertainty *)
\mu b = 1; \sigma b = 100 000; (* large *)
\sigma R = 4.0; \sigma D = 1.0;
\mu = Table[\mu 0, {i, 1, size}];
\sigma = Table[\sigma a, {i, 1, size}];
\mu a = Table[\mu a, {i, 1, size}];
\sigma a = Table[\sigma a, {i, 1, size}];
\mu b = Table[\mu b, {i, 1, size}];
\sigma b = Table[\sigma b, {i, 1, size}];
iter = 0;
i = 1;
j = size;
\end{verbatim}

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 $\pm \sigma_i$. 
Belief Propagation Routine

```math
\text{yfit} = \text{Table}[[0, 0], \{i, 1, \text{size}\}];
\text{glb} = \text{ErrorListPlot}[[\text{yfit}]];
\text{Dynamic}[
\text{Show}[[\text{glb}, \text{g2bp}, \text{g3bp},
\text{Graphics}[[\text{Text}[:,:,\text{Iteration}=\text{ToString}[\text{iter}], \left\{\frac{\text{size}}{2}, \text{size}\}\right\}]],
\text{PlotRange} \to \{-50, 50\}, \text{Axes} \to \{\text{False, True}\}]]]
```

Execute the next cell to run 31 iterations. The display is slowed down so that you can see the progression of the updates in the above graph.
Relation to neural networks
We’ve seen two very different ways of estimating states through iterative updating. It is easy to see how the first, derived from gradient descent, is related to updating in a traditional neural network. The second, belief propagation, gives us a very different view of how information might be represented and updated in a network. For example, in the second case applied to interpolation, the nodes represent probability distributions over depths (summarized by the mean and variance) not just estimates of depth. A critical new feature is the explicit representation of uncertainty, i.e. in the standard deviations of the node values. Later in the course, we will discuss whether the brain might have explicit representations of such uncertainties, how these might be represented in populations of neurons, and how the brain might do computations on these. Does the brain do belief propagation, and if so how?


### Exercises

Run the descent algorithm using successive over-relaxation (SOR): \( \eta_2[k_] := 1.9/(\lambda + \text{xs}[k]) \).

How does convergence compare with Gauss-Seidel?

Run Belief Propagation using: \( \sigma_R = 1.0; \sigma_D = 4.0 \); How does fidelity to the data compare with the original case \( (\sigma_R = 4.0; \sigma_D = 1.0) \).

BP with missing sine wave data

- Generate sine wave with missing data

```math
\text{size} = 64; \text{xs} = \text{Table}[\text{RandomInteger}[1], \{i, 1, \text{size}\}];
\text{data} = \text{Table}[\text{N}[\text{Sin}[\frac{2 \pi j}{20}] \text{xs}[j]], \{j, 1, \text{size}\}];
\text{g3b} = \text{ListPlot}[\text{Table}[\text{N}[\text{Sin}[\frac{2 \pi j}{20}]], \{j, 1, \text{size}\}], \text{Joined} \rightarrow \text{True}, \\
\text{PlotStyle} \rightarrow \{\text{RGBColor}[0, 0.5, 0]\}];
\text{g2b} = \text{ListPlot}[\text{data}, \text{Joined} \rightarrow \text{False}, \text{PlotStyle} \rightarrow \{\text{RGBColor}[0.75, 0. , 0]\}];
```
**Initialize**

\[\begin{align*}
\text{In}[59]:= & \quad \mu_0 = 1; \\
& \text{\quad \mu_\alpha = 1; } \sigma_\alpha = 100\ 000; (\text{large uncertainty} \, *) \\
& \text{\quad \mu_\beta = 1; } \sigma_\beta = 100\ 000; (\text{large} \, *) \\
& \text{\quad \sigma_R = .5; } \sigma_D = .1; \\
& \mu = \text{Table}[\mu_0, \{i, 1, \text{size}\}]; \\
& \sigma = \text{Table}[\sigma_\alpha, \{i, 1, \text{size}\}]; \\
& \mu_\alpha = \text{Table}[\mu_0, \{i, 1, \text{size}\}]; \\
& \sigma_\alpha = \text{Table}[\sigma_\alpha, \{i, 1, \text{size}\}]; \\
& \mu_\beta = \text{Table}[\mu_0, \{i, 1, \text{size}\}]; \\
& \sigma_\beta = \text{Table}[\sigma_\beta, \{i, 1, \text{size}\}]; \\
& \text{iter} = 0; \\
& \text{i = 1; } \\
& \text{j = size; }
\end{align*}\]

\[\begin{align*}
\text{In}[71]:= & \quad \text{yfit = Table}[\{0, 0\}, \{i, 1, \text{size}\}]; \\
& \quad \text{g1bb = ErrorListPlot}[\{\text{yfit}\}]; \\
& \quad \text{Dynamic[Show}\{\text{g1bb, g2b, g3b}, \text{PlotRange} \to \{-2, 2\}, \text{Axes} \to \{\text{False, True}\}\}]
\end{align*}\]
SINE WAVE DEMO: Belief Propagation Routine

In[74]:= Do[
  Pause[0.2];
  \[mu\][i] = \frac{xs[i] data[i] + \mu_0[i] + 1. \mu_{\theta}[i]}{\sigma D} + \frac{1}{\sigma [i]} + \frac{1}{\sigma_{\theta}[i]};

  \[sigma\][i] = \frac{1.}{\frac{xs[i]}{\sigma D} + \frac{1}{\sigma [i]} + \frac{1}{\sigma_{\theta}[i]}};

  \[mu\][j] = \frac{xs[j] data[j] + \mu_0[j] + 1. \mu_{\theta}[j]}{\sigma D} + \frac{1}{\sigma [j]} + \frac{1}{\sigma_{\theta}[j]};

  \[sigma\][j] = \frac{1.}{\frac{xs[j]}{\sigma D} + \frac{1}{\sigma [j]} + \frac{1}{\sigma_{\theta}[j]}};

  nextj = j - 1;

  \[mu_\alpha][nextj] = \frac{xs[j] data[j] + 1. \mu_\alpha[j]}{\sigma D} + \frac{1}{\sigma [j]};

  \[sigma_\alpha][nextj] = \sigma R + \frac{1.}{\frac{xs[j]}{\sigma D} + \frac{1}{\sigma [j]}};

  nexti = i + 1;

  \[mu_\beta][nexti] = \frac{xs[i] data[i] + 1. \mu_{\theta}[i]}{\sigma D} + \frac{1}{\sigma_{\theta}[i]};

  \[sigma_\beta][nexti] = \sigma R + \frac{1.}{\frac{xs[i]}{\sigma D} + \frac{1}{\sigma_{\theta}[i]}};

  j--; i++; iter++; yfit = Table[{\mu[i1], \sigma[i1]}, {i1, 1, size}]; g1bb = ErrorListPlot[{yfit}]; , {size - 1}]
References


For notes on Graphical Models, see:http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html