Introduction to Neural Networks
U. Minn. Psy 5038
Daniel Kersten
Belief Propagation

Initialize

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

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

Last time

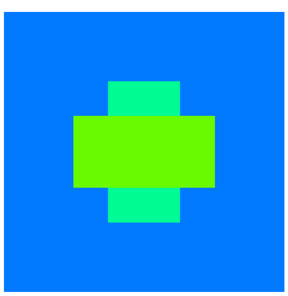
Generative modeling: Multivariate gaussian, mixtures

- **Drawing samples**
- **Mixtures of gaussians**
- **Will use mixture distributions in the next lecture on EM application to segmentation**

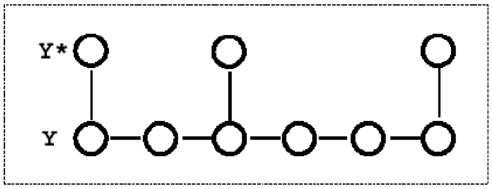
Introduction to Optimal inference and task

Introduction to Bayesian learning

Interpolation using smoothness revisited: 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

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 the indicator function, and y_k^*= d, are the data values. The indicator function is 1 if there is data available, and zero otherwise. (See supplementary material in Lecture 20).

Gradient descent gives the following local update rule:

$$y_k \leftarrow y_k + \eta_k (\lambda(\frac{y_{k-1} + y_{k+1}}{2} - y_k) + w_k (y_k^* - y_k))$$

λ controls the degree of smoothness, i.e. smoothness at the expense of fidelity to the data.

Gauss-Seidel: η[k_]:=1/(λ+xs[[k]])

Successive over-relaxation (SOR): η2[k_]:=1.9/(λ+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.**

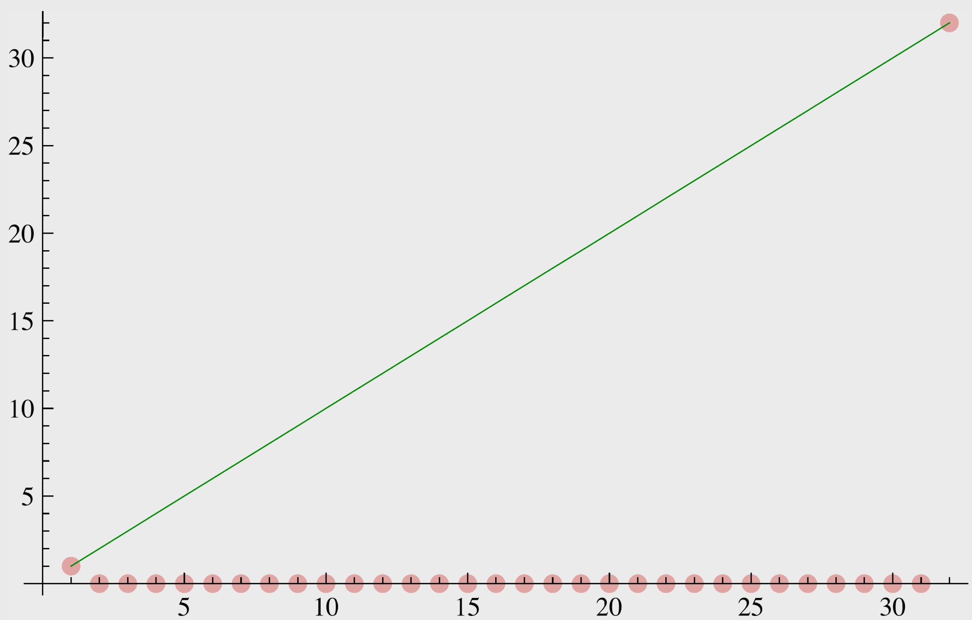
We'll assume the true model is that f=y=j, where j=1 to size. data is a function of the sampling process on f=j

In[3]:=
```
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 is missing.

In[4]:=
```
Show[g2, g3]
```

Out[4]=



Let's set up two matrices, **Tm** and **Sm** such that the gradient of the energy is equal to:

Tm . f - Sm . f.

Sm will be our filter to exclude non-data points. **Tm** will express the "smoothness" constraint.

In[5]:=
```
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];
```

Check the update rule code for small size=10:

In[11]:=
```
Clear[f, d, λ]
(λ * Tm.Array[f, size] - Sm.((Array[d, size]) - Array[f, size])) //
  MatrixForm ;
```

■ Run gradient descent

In[13]:=
```
Clear[Tf,f1];
dt = 1; λ=2;
Tf[f1_] := f1 - dt*(1/(λ+xs))*(Tm.f1 - λ*Sm.(data-f1));
```

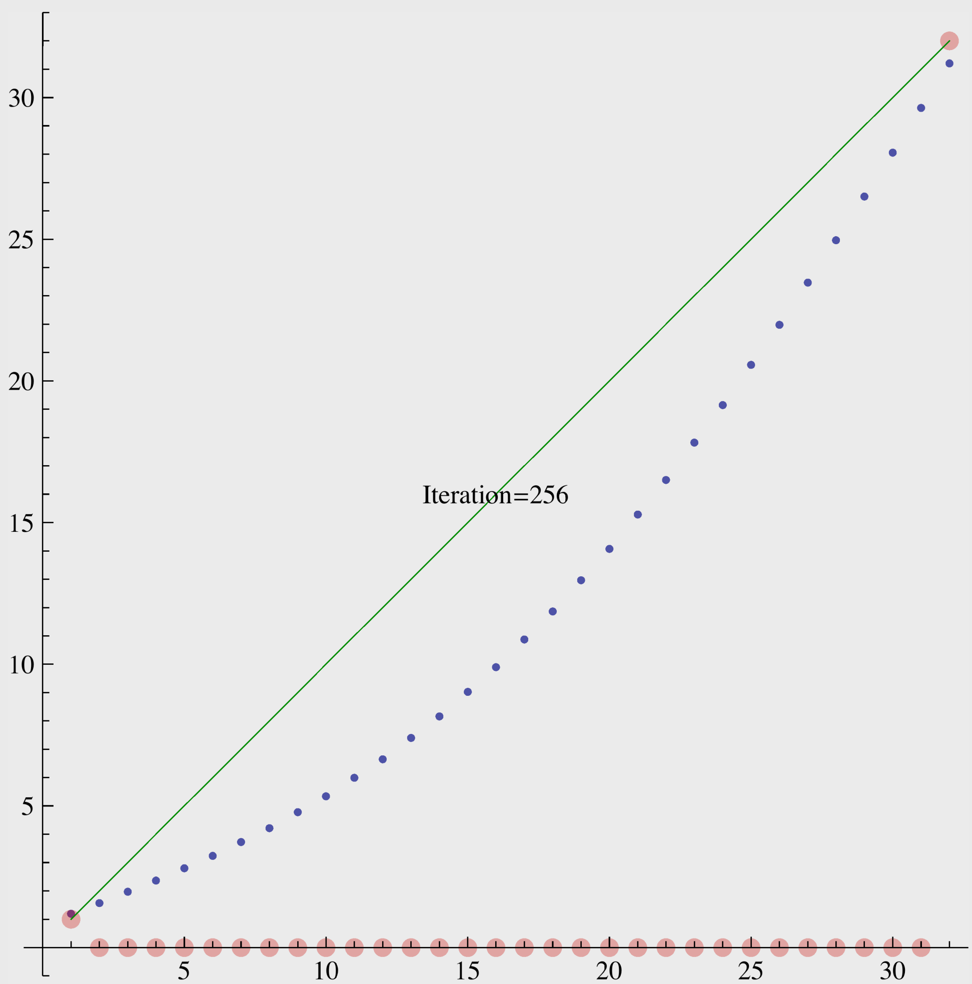
We will initialize the state vector to zero, and then run the network for **iter** iterations:

In[16]:=
```
iter=256;
f = Table[0,{i,1,size}];
result = Nest[Tf,f,iter];
```

Now plot the interpolated function.

In[19]:=
```
g1 = ListPlot[result, Joined → False, AspectRatio → Automatic,
   PlotRange → {{0, size}, {-1, size + 1}}];
Show[
 {g1, g2, g3,
   Graphics[{Text["Iteration=" <> ToString[iter], {size/2, size/2}]}]},
  PlotRange → {-1, size + 1}]
```

Out[19]=



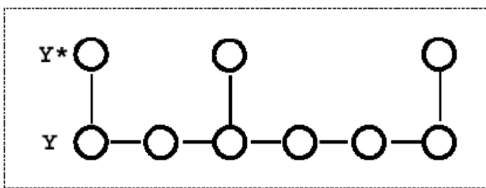
Try starting with f = random values between 0 and 40. Try various numbers of iterations.

Try different sampling functions xs[[i]].

Belief Propagation

Same interpolation problem, but now using belief propagation

Example is taken from Yair Weiss.(Weiss, 1999)



Probabilistic generative model

$$\text{data}[[i]] = y^*[i] = \text{xs}[[i]]\, y[[i]] + \text{dnoise},\ \text{dnoise} \sim N[0, \sigma_D] \quad (1)$$

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

The first term is the "data formation" model, i.e. how the data is directly influenced by the interaction of the underlying causes, y with the sampling and noise. The second term reflects our prior assumptions about the smoothness of y, i.e. nearby y's are correlated, and in fact 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 |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

Let p(Y_i=u |all the data) be normally distributed: NormalDistribution[μi,σi].

Consider the ith unit. The posterior p(Y_i=u|all the data) =

p(Y_i=u|all the data) ∝ p(Y_i=u|data before i) p(data at i| Y_i=u) p(Y_i=u|data after i) (3)

Suppose that p(Y_i=u | data before i) is also gaussian:

p(Y_i=u|data before i) = α[u] ~ NormalDistribution[$\mu\alpha$,$\sigma\alpha$]

and so is probability conditioned on the data after i:

p(Y_i=u|data after i)= β[u] ~ NormalDistribution[$\mu\beta$,$\sigma\beta$]

And the noise model for the data:

p(data at i| Y_i=u) = L[u]~ NormalDistribution[yp, σ_D]

yp=data[[i]]

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|all the data) ∝ α[u]*L[u]*β[u] (4)

```
αudist = NormalDistribution[μα, σα];
α[u] = PDF[αudist, u];

Ddist = NormalDistribution[yp, σD];
L[u] = PDF[Ddist, u];

βudist = NormalDistribution[μβ, σβ];
β[u] = PDF[βudist, u];

α[u] * L[u] * β[u]
```

$$\frac{e^{-\frac{(u-\mu\alpha)^2}{2\,\sigma\alpha^2}-\frac{(u-\mu\beta)^2}{2\,\sigma\beta^2}-\frac{(u-y_p)^2}{2\,\sigma_D^2}}}{2\sqrt{2}\,\pi^{3/2}\,\sigma\alpha\,\sigma\beta\,\sigma_D}$$

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.

```
Solve[-D[-(u - μα)^2/(2 σα^2) - (u - μβ)^2/(2 σβ^2) - (u - y_p)^2/(2 σ_D^2), u] == 0, u]
```

$$\left\{\left\{u \to \frac{\frac{\mu\alpha}{\sigma\alpha^2} + \frac{\mu\beta}{\sigma\beta^2} + \frac{y_p}{\sigma_D^2}}{\frac{1}{\sigma\beta^2} + \frac{1}{\sigma_D^2} + \frac{1}{\sigma\alpha^2}}\right\}\right\}$$

The update rule for the variance is:

$$\sigma^2 \to \frac{1}{\sigma\alpha^2} + \frac{1}{\sigma\beta^2} + \frac{1}{\sigma_D^2}$$

How do we get $\mu\alpha, \mu\beta$, $\sigma\alpha, \sigma\beta$?

We express the probability of the ith 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-1th location).

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

$$\alpha[u] = \int_{-\infty}^{\infty} \alpha_p[v] * S[u] * L[v] \, dv \propto \int_{-\infty}^{\infty} e^{-\frac{(v-y_p)^2}{2\sigma_D^2} - \frac{(u-v)^2}{2\sigma_R^2} - \frac{(v-\mu\alpha_p)^2}{2\sigma\alpha_p^2}} \, dv \qquad (5)$$

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

```
Rdist = NormalDistribution[v, σR];
S[u] = PDF[Rdist, u];

αvdist = NormalDistribution[μαp, σαp];
αp[v] = PDF[αvdist, v];

Lp[v] = PDF[Ddist, v];
```

```
Integrate[αp[v] * S[u] * Lp[v], {v, -Infinity, Infinity}]
```

$$\frac{1}{2\sqrt{2}\,\pi^{3/2}\,\sigma_D\,\sigma_R\,\sigma\alpha_p}$$

$$\text{If}\Big[\text{Re}\Big(\frac{1}{\sigma_R^2}+\frac{1}{\sigma\alpha_p^2}+\frac{1}{\sigma_D^2}\Big)>0,\ \frac{e^{-\frac{(\sigma_R^2+\sigma\alpha_p^2)\,y_p^2-2\,(\mu\alpha_p\,\sigma_R^2+u\,\sigma\alpha_p^2)\,y_p+(u-\mu\alpha_p)^2\,\sigma_D^2+\mu\alpha_p^2\,\sigma_R^2+u^2\,\sigma\alpha_p^2}{2\,((\sigma_R^2+\sigma\alpha_p^2)\,\sigma_D^2+\sigma_R^2\,\sigma\alpha_p^2)}}\sqrt{2\pi}}{\sqrt{\frac{1}{\sigma_R^2}+\frac{1}{\sigma\alpha_p^2}+\frac{1}{\sigma_D^2}}},$$

$$\text{Integrate}\Big[e^{\frac{1}{2}\left(-\frac{(u-v)^2}{\sigma_R^2}-\frac{(v-y_p)^2}{\sigma_D^2}-\frac{(v-\mu\alpha_p)^2}{\sigma\alpha_p^2}\right)},\{v,-\infty,\infty\},\ \text{Assumptions}\to\text{Re}\Big(\frac{1}{\sigma_R^2}+\frac{1}{\sigma\alpha_p^2}+\frac{1}{\sigma_D^2}\Big)\le 0\Big]\Big]$$

■ Some uninspired *Mathematica* manipulations

To find an expression for the mode of the above calculated expression for α[u]

$$\text{D}\Big[-\frac{(u-\mu\alpha_p)^2\,\sigma_D^2+\mu\alpha_p^2\,\sigma_R^2+u^2\,\sigma\alpha_p^2+y_p^2\,(\sigma_R^2+\sigma\alpha_p^2)-2\,y_p\,(\mu\alpha_p\,\sigma_R^2+u\,\sigma\alpha_p^2)}{2\,(\sigma_R^2\,\sigma\alpha_p^2+\sigma_D^2\,(\sigma_R^2+\sigma\alpha_p^2))},\ u\Big]$$

$$-\frac{2\,(u-\mu\alpha_p)\,\sigma_D^2+2\,u\,\sigma\alpha_p^2-2\,y_p\,\sigma\alpha_p^2}{2\,((\sigma_R^2+\sigma\alpha_p^2)\,\sigma_D^2+\sigma_R^2\,\sigma\alpha_p^2)}$$

```
Solve[-% == 0, u]
```

$$\Big\{\Big\{u\to\frac{\mu\alpha_p\,\sigma_D^2+y_p\,\sigma\alpha_p^2}{\sigma_D^2+\sigma\alpha_p^2}\Big\}\Big\}$$

$$\text{Simplify}\Big[\Big(\frac{\mu\alpha_p\,\sigma_D^2}{\sigma_R^2\,\sigma\alpha_p^2+\sigma_D^2\,(\sigma_R^2+\sigma\alpha_p^2)}+\frac{y_p\,\sigma\alpha_p^2}{\sigma_R^2\,\sigma\alpha_p^2+\sigma_D^2\,(\sigma_R^2+\sigma\alpha_p^2)}\Big)\Big/(\sigma_D^2*\sigma\alpha_p^2)\Big]$$

$$\frac{\mu\alpha_p\,\sigma_D^2+y_p\,\sigma\alpha_p^2}{\sigma\alpha_p^2\,(\sigma_R^2+\sigma\alpha_p^2)\,\sigma_D^4+\sigma_R^2\,\sigma\alpha_p^4\,\sigma_D^2}$$

```
Simplify[(σD^2/(σR^2 σαp^2 + σD^2 (σR^2 + σαp^2)) + σαp^2/(σR^2 σαp^2 + σD^2 (σR^2 + σαp^2))) / (σD^2 * σαp^2)]
```

$$\frac{\sigma_D^2+\sigma\alpha_p^2}{\sigma\alpha_p^2\left(\sigma_R^2+\sigma\alpha_p^2\right)\sigma_D^4+\sigma_R^2\,\sigma\alpha_p^4\,\sigma_D^2}$$

```
((μαp σD^2 + yp σαp^2)/(σD^2 σR^2 σαp^4 + σD^4 σαp^2 (σR^2 + σαp^2))) / ((σD^2 + σαp^2)/(σD^2 σR^2 σαp^4 + σD^4 σαp^2 (σR^2 + σαp^2)))
```

$$\frac{\mu\alpha_p\,\sigma_D^2+y_p\,\sigma\alpha_p^2}{\sigma_D^2+\sigma\alpha_p^2}$$

So we now have rule that tells us how to update the α(u)=p(yi=u|data before i), in terms of the mean and variance parameters of the previous node:

$$\mu\alpha \leftarrow \frac{\mu\alpha_p\,\sigma_D^2+y_p\,\sigma\alpha_p^2}{\sigma_D^2+\sigma\alpha_p^2} = \frac{\frac{\mu\alpha_p\,\sigma_D^2}{\sigma\alpha_p^2\,\sigma_D^2}+\frac{y_p\,\sigma\alpha_p^2}{\sigma\alpha_p^2\,\sigma_D^2}}{\frac{\sigma_D^2}{\sigma\alpha_p^2\,\sigma_D^2}+\frac{\sigma\alpha_p^2}{\sigma\alpha_p^2\,\sigma_D^2}} = \frac{\frac{\mu\alpha_p}{\sigma\alpha_p^2}+\frac{y_p}{\sigma_D^2}}{\frac{1}{\sigma\alpha_p^2}+\frac{1}{\sigma\alpha_p^2}}$$

The update rule for the variance is:

$$\sigma\alpha^2 \leftarrow \sigma_R^2+\frac{1}{\frac{1}{\sigma_D^2}+\frac{1}{\sigma\alpha_p^2}}$$

A similar derivation gives us the rules for $\mu\beta$, $\sigma\beta^2$

$$\mu\beta\leftarrow\frac{\frac{\mu\beta_a}{\sigma\beta_a^2}+\frac{y_a}{\sigma_D^2}}{\frac{1}{\sigma\beta_a^2}+\frac{1}{\sigma\alpha_a^2}}$$

$$\sigma\beta^2 \leftarrow \sigma_R^2+\frac{1}{\frac{1}{\sigma_D^2}+\frac{1}{\sigma\beta_a^2}}$$

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:

$$y_p \to xs[i-1]\,data[i-1] = w_{i-1}\,y_{i-1}^{*} \quad (6)$$

And similarly for y_a.

■ Summary of update rules

The ratio, $\left(\frac{\sigma_D}{\sigma_R}\right)^2$ plays the role of λ above. If σ_D^2>>σ_R^2, there is greater smoothing. If σ_D^2<<σ_R^2, there is more fidelity to the data. (Recall $y^* \to$ data.$w_k \to$ xs[[k]])

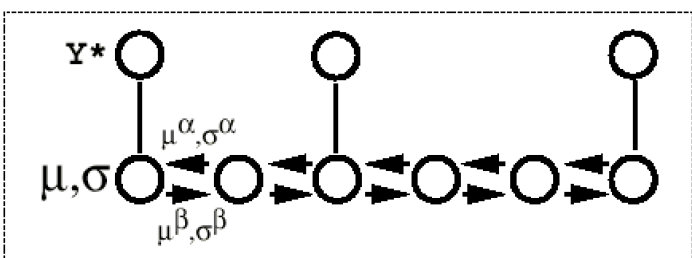
We'll follow Weiss, and also make a (hopefully not too confusing) notation change to avoid the square superscripts for σ_D^2->σ_D, σ_R^2->σ_R.

$$\mu_i \leftarrow \frac{\frac{w_i}{\sigma_D} Y_i^* + \frac{1}{\sigma_i^\alpha}\mu_i^\alpha + \frac{1}{\sigma_i^\beta}\mu_i^\beta}{\frac{w_i}{\sigma_D} + \frac{1}{\sigma_i^\alpha} + \frac{1}{\sigma_i^\beta}}$$

$$\sigma_i \leftarrow \frac{1}{\frac{w_i}{\sigma_D} + \frac{1}{\sigma_i^\alpha} + \frac{1}{\sigma_i^\beta}}$$

$$\mu_i^\alpha \leftarrow \frac{\frac{1}{\sigma_{i-1}^\alpha}\mu_{i-1}^\alpha + \frac{w_{i-1}}{\sigma_D} Y_{i-1}^*}{\frac{1}{\sigma_{i-1}^\alpha} + \frac{w_{i-1}}{\sigma_D}}$$

$$\sigma_i^\alpha \leftarrow \sigma_R + \left(\frac{1}{\sigma_{i-1}^\alpha} + \frac{w_{i-1}}{\sigma_D}\right)^{-1}$$



A simulation: Belief propagation for interpolation with missing data

■ Initialization

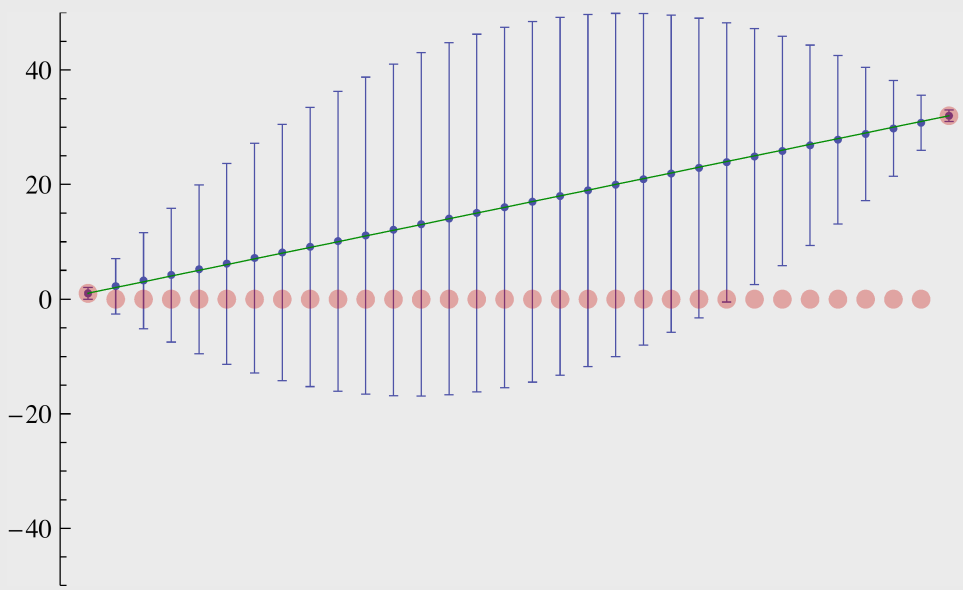
In[37]:=
```
size = 32;
μ0 = 1;
μα = 1; σα = 100 000; (*large uncertainty *)
μβ = 1; σβ = 100 000; (*large*)
σR = 4.0; σD = 1.0;
μ = Table[μ0, {i, 1, size}];
σ = Table[σα, {i, 1, size}];
μα = Table[μ0, {i, 1, size}];
σα = Table[σα, {i, 1, size}];
μβ = Table[μ0, {i, 1, size}];
σβ = Table[σβ, {i, 1, size}];
iter = 0;
i = 1;
j = size;
```

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: Execute this cell "manually" for each iteration

In[50]:=
```
yfit = Table[{0, 0}, {i1, 1, size}];
g1b = ErrorListPlot[{yfit}];
Dynamic[
 Show[
  {g1b, g2, g3,
   Graphics[{Text["Iteration=" <> ToString[iter], {size/2, size}]}]},
  PlotRange → {-50, 50}, Axes → {False, True}]]
```

Out[52]=



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.

In[53]:=

```
Do[
  Pause[.333];
  μ[[i]] = (xs[[i]] data[[i]]/σD + μα[[i]]/σα[[i]] + 1. μβ[[i]]/σβ[[i]]) / (xs[[i]]/σD + 1/σα[[i]] + 1/σβ[[i]]);
  σ[[i]] = 1. / (xs[[i]]/σD + 1/σα[[i]] + 1/σβ[[i]]);
  μ[[j]] = (xs[[j]] data[[j]]/σD + μα[[j]]/σα[[j]] + 1. μβ[[j]]/σβ[[j]]) / (xs[[j]]/σD + 1/σα[[j]] + 1/σβ[[j]]);
  σ[[j]] = 1. / (xs[[j]]/σD + 1/σα[[j]] + 1/σβ[[j]]);
  nextj = j - 1;
  μα[[nextj]] = (xs[[j]] data[[j]]/σD + 1. μα[[j]]/σα[[j]]) / (xs[[j]]/σD + 1/σα[[j]]);
  σα[[nextj]] = σR + 1. / (xs[[j]]/σD + 1/σα[[j]]);
  nexti = i + 1;
  μβ[[nexti]] = (xs[[i]] data[[i]]/σD + 1. μβ[[i]]/σβ[[i]]) / (xs[[i]]/σD + 1/σβ[[i]]);
  σβ[[nexti]] = σR + 1. / (xs[[i]]/σD + 1/σβ[[i]]);
  j--;
  i++;
  iter++;
  yfit = Table[{μ[[i1]], σ[[i1]]}, {i1, 1, size}];
  g1b = ErrorListPlot[{yfit}];
  , {size - 1}];
```

Exercises

Run the descent algorithm using successive over-relaxation (SOR): η2[k_]:=1.9/(λ+xs[[k]]).
How does convergence compare with Gauss-Seidel?

Run Belief Propation using: σR=1.0; σD=4.0; How does fidelity to the data compare with the original case
(σR=4.0; σD=1.0).

BP with missing sine wave data

■ Generate sine wave with missing data

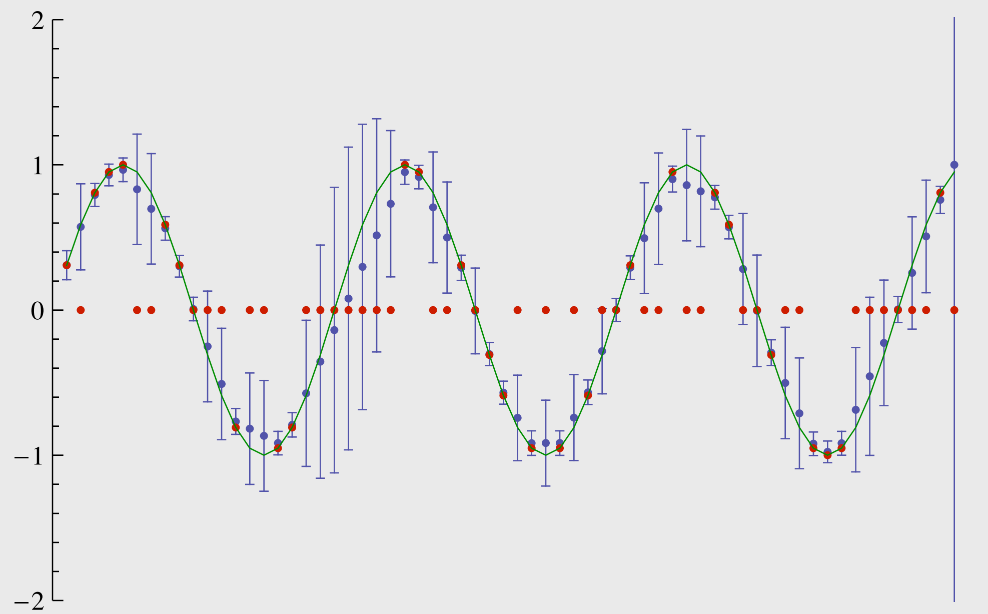
In[70]:=
```
size = 64; xs = Table[RandomInteger[1], {i, 1, size}];
data = Table[N[Sin[2 π j / 20] xs[[j]]], {j, 1, size}];
g3b = ListPlot[Table[N[Sin[2 π j / 20]], {j, 1, size}], Joined → True,
  PlotStyle → {RGBColor[0, 0.5, 0]}];
g2b = ListPlot[data, Joined → False, PlotStyle → {RGBColor[0.75, 0., 0]}];
```

■ Initialize

In[71]:=
```
μ0 = 1;
μα = 1; σα = 100 000; (*large uncertainty *)
μβ = 1; σβ = 100 000; (*large*)
σR = .5; σD = .1;
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[83]:=
```
yfit = Table[{0, 0}, {i1, 1, size}];
g1bb = ErrorListPlot[{yfit}];
Dynamic[Show[{g1bb, g2b, g3b}, PlotRange → {-2, 2}, Axes → {False, True}]]
```

Out[85]=



■ SINE WAVE DEMO: Belief Propagation Routine

In[86]:=

```
Do[
 Pause[0.2];
 μ[[i]] = (xs[[i]] data[[i]]/σD + μα[[i]]/σα[[i]] + 1. μβ[[i]]/σβ[[i]]) / (xs[[i]]/σD + 1/σα[[i]] + 1/σβ[[i]]);
σ[[i]] = 1. / (xs[[i]]/σD + 1/σα[[i]] + 1/σβ[[i]]);
μ[[j]] = (xs[[j]] data[[j]]/σD + μα[[j]]/σα[[j]] + 1. μβ[[j]]/σβ[[j]]) / (xs[[j]]/σD + 1/σα[[j]] + 1/σβ[[j]]);
σ[[j]] = 1. / (xs[[j]]/σD + 1/σα[[j]] + 1/σβ[[j]]);
nextj = j - 1;
μα[[nextj]] = (xs[[j]] data[[j]]/σD + 1. μα[[j]]/σα[[j]]) / (xs[[j]]/σD + 1/σα[[j]]);
σα[[nextj]] = σR + 1. / (xs[[j]]/σD + 1/σα[[j]]);
nexti = i + 1;
μβ[[nexti]] = (xs[[i]] data[[i]]/σD + 1. μβ[[i]]/σβ[[i]]) / (xs[[i]]/σD + 1/σβ[[i]]);
σβ[[nexti]] = σR + 1. / (xs[[i]]/σD + 1/σβ[[i]]);
j --;
i ++;
iter ++;
yfit = Table[{μ[[i1]], σ[[i1]]}, {i1, 1, size}];
g1bb = ErrorListPlot[{yfit}];
 , {size - 1}]
```

References

Applebaum, D. (1996). Probability and Information . Cambridge, UK: Cambridge University Press.

Frey, B. J. (1998). *Graphical Models for Machine Learning and Digital Communication*. Cambridge, Massachusetts: MIT 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. and P.W. Schrater (2000), *Pattern Inference Theory: A Probabilistic Approach to Vision*, in *Perception and the Physical World*, R. Mausfeld and D. Heyer, Editors. , John Wiley & Sons, Ltd.: Chichester. (pdf)

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.

Pearl, Judea. (1997) Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference. (amazon.com link)

Ripley, B. D. (1996). *Pattern Reco gnition and Neural Networks*. Cambridge, UK: Cambridge University Press.

Weiss Y. (1999) Bayesian Belief Propagation for Image Understanding submitted to *SCTV* 1999. (gzipped postscript 297K)

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)

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