Initialize

- Read in Statistical Add-in packages:

```mathematica
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 \( y \).
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 \( w_k = x[s[k]] \) 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:

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

\( \lambda \) controls the degree of smoothness, i.e. smoothness at the expense of fidelity to the data.

Gauss-Seidel: \( \eta[k_] := 1/(\lambda + x[s[k]]) \)

Successive over-relaxation (SOR): \( \eta[2[k_]] := 1.9/(\lambda + x[s[k]]) \);

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:

\( w_k = x[s[k]], y^* = \text{data}, y = f \).

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 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, \(T_m\) 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[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:
Run gradient descent

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

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

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}]}],
   PlotRange -> {{0, size}, {-1, size + 1}}]
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)

![Graph of Belief Propagation](image)

**Probabilistic generative model**

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

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

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

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.

Let \( p(Y_i = u | \text{all the data}) \) be normally distributed: \( \text{NormalDistribution}[\mu_i, \sigma_i] \).
Consider the ith unit. The posterior $p(Y_i=u|\text{all the data}) = p(Y_i=u|\text{data before i}) \cdot p(\text{data at i}|Y_i=u) \cdot p(Y_i=u|\text{data after i})$

\begin{equation}
  p(Y_i=u|\text{all the data}) \propto p(Y_i=u|\text{data before i}) \cdot p(\text{data at i}|Y_i=u) \cdot p(Y_i=u|\text{data after i})
\end{equation}

Suppose that $p(Y_i=u|\text{data before i})$ is also gaussian:

\[ p(Y_i=u|\text{data before i}) = a[u] \sim \text{NormalDistribution}[\mu_\alpha, \sigma_\alpha] \]

and so is probability conditioned on the data after i:

\[ p(Y_i=u|\text{data after i}) = b[u] \sim \text{NormalDistribution}[\mu_\beta, \sigma_\beta] \]

And the noise model for the data:

\[ p(\text{data at i}|Y_i=u) = L[u] \sim \text{NormalDistribution}[\mu_p, \sigma_D] \]

\[ yp = \text{data}[i] \]

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:

\begin{equation}
  p(Y_i=u|\text{all the data}) \propto a[u] \cdot L[u] \cdot b[u]
\end{equation}

\begin{verbatim}
audist = NormalDistribution[\mu_\alpha, \sigma_\alpha];
a[u] = PDF[audist, u];

Ddist = NormalDistribution[\mu_\beta, \sigma_\beta];
L[u] = PDF[Ddist, u];

\betaudist = NormalDistribution[\mu_\beta, \sigma_\beta];
\beta[u] = PDF[\betaudist, u];

\[ a[u] \cdot L[u] \cdot \beta[u] \]
\end{verbatim}

\[ a(u-\mu_\alpha)^2 \frac{1}{2\sigma_\alpha^2} - \frac{(u-\mu_\beta)^2}{2\sigma_\beta^2} - \frac{(u-\mu_p)^2}{2\sigma_D^2} \]

\[ 2\sqrt{2 \pi}^{3/2} \sigma_\alpha \sigma_\beta \sigma_D \]

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.
\[ \text{Solve} \left[ -D \left[ -\frac{(u - \mu \alpha)^2}{2 \sigma \alpha^2} - \frac{(u - \mu \beta)^2}{2 \sigma \beta^2} - \frac{(u - y_p)^2}{2 \sigma_d^2} \right], u \right] = 0, u] \]

\[ \left\{ \left\{ u \rightarrow \frac{\mu \alpha}{\sigma \alpha^2} + \frac{\mu \beta}{\sigma \beta^2} + \frac{y_p}{\sigma_d^2} \right\} \right\} \]

The update rule for the variance is:

\[ \sigma^2 \rightarrow \frac{1}{\sigma \alpha^2} + \frac{1}{\sigma \beta^2} + \frac{1}{\sigma_d^2} \]

How do we get \( \mu \alpha, \mu \beta, \sigma \alpha, \sigma \beta \)?

We express the probability of the \( i \)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 \( i \):

\[ \alpha[u] = \int_{-\infty}^{\infty} \alpha_p[v] * S[u] * L[v] dv = \int_{-\infty}^{\infty} e^{-\frac{(v - y_p)^2}{2 \sigma_d^2} - \frac{(u - v)^2}{2 \sigma_R^2} - \frac{(v - \mu \alpha_p)^2}{2 \sigma_p^2}} dv \]

(5)

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

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

avdist = NormalDistribution[μα_p, σα_p];
α_p[v] = PDF[avdist, v];

Lp[v] = PDF[Ddist, v];
```
Integrate[$\alpha_p[v] \ast S[u] \ast Lp[v], \{v, -\infty, \infty\}$]

\[
\frac{1}{2 \sqrt{2 \pi^{3/2}} \sigma_D \sigma_R \sigma_{\alpha_p}} \left\{ \right. \\
\left. e^{-\frac{(\sigma_R^2 + \sigma_{\alpha_p}^2) (u - \mu_{\alpha_p})^2 + (\sigma_D^2 + \sigma_R^2 + \sigma_{\alpha_p}^2) u^2 + \sigma_{\alpha_p}^2 \alpha^2}{2((\sigma_R^2 + \sigma_{\alpha_p}^2)^2 \sigma_D^2 + \sigma_D^2 \sigma_R^2)}}, \{v, -\infty, \infty\}, \text{Assumptions} \to \text{Re} \left( \frac{1}{\sigma_R^2 + \frac{1}{\sigma_{\alpha_p}^2} + \frac{1}{\sigma_D^2}} \right) > 0 \right\}
\]

\[\text{Some uninspired Mathematica manipulations}\]

To find an expression for the mode of the above calculated expression for $\alpha[u]$

\[
D\left[ -\frac{(u - \mu_{\alpha_p})^2 \sigma_D^2 + \mu_{\alpha_p}^2 \sigma_R^2 + u^2 \sigma_{\alpha_p}^2 + y_p^2 \left( \sigma_R^2 + \sigma_{\alpha_p}^2 \right)}{2 \left( \sigma_D^2 \sigma_R^2 \sigma_{\alpha_p}^2 \right)} + \frac{1}{\sigma_D^2 + \sigma_R^2 + \sigma_{\alpha_p}^2}, u \right]
\]

\[2(u - \mu_{\alpha_p}) \sigma_D^2 + 2u \sigma_{\alpha_p}^2 - 2y_p \sigma_{\alpha_p}^2 \]

\[\frac{2((\sigma_R^2 + \sigma_{\alpha_p}^2)^2 \sigma_D^2 + \sigma_R^2 \sigma_{\alpha_p}^2)}{2 \left( \sigma_D^2 \sigma_R^2 \sigma_{\alpha_p}^2 \right)}\]

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

\[\left\{ \left\{ u \to \frac{\mu_{\alpha_p} \sigma_D^2 + y_p \sigma_{\alpha_p}^2}{\sigma_D^2 + \sigma_R^2 + \sigma_{\alpha_p}^2} \right\} \right\}\]

\[\text{Simplify}\left[ \frac{\mu_{\alpha_p} \sigma_D^2}{\sigma_R^2 \sigma_{\alpha_p}^2 + \sigma_D^2 \left( \sigma_R^2 + \sigma_{\alpha_p}^2 \right)} + \frac{y_p \sigma_{\alpha_p}^2}{\sigma_R^2 \sigma_{\alpha_p}^2 + \sigma_D^2 \left( \sigma_R^2 + \sigma_{\alpha_p}^2 \right)} \right] \]

\[\frac{\mu_{\alpha_p} \sigma_D^2 + y_p \sigma_{\alpha_p}^2}{\sigma_D^2 \left( \sigma_R^2 + \sigma_{\alpha_p}^2 \right) + \sigma_R^2 \sigma_{\alpha_p}^2 \sigma_D^2} \]
So we now have 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:

\[
\mu_a \leftarrow \frac{\mu_a \sigma_a^2 + y_p \sigma_a^2}{\sigma_a^2 + \sigma_a^2} = \frac{\mu_a \sigma_a^2 + y_p \sigma_a^2}{\sigma_a^2 + \sigma_a^2} = \frac{1}{1 + \frac{1}{\sigma_a^2}}
\]

The update rule for the variance is:

\[
\sigma_a^2 \leftarrow \sigma_a^2 + \frac{1}{1 + \frac{1}{\sigma_a^2}}
\]

A similar derivation gives us the rules for \( \mu_b, \sigma_b^2 \):

\[
\mu_b \leftarrow \frac{\mu_b \sigma_b^2 + y_b \sigma_b^2}{\sigma_b^2 + \sigma_b^2} = \frac{\mu_b \sigma_b^2 + y_b \sigma_b^2}{\sigma_b^2 + \sigma_b^2} = \frac{1}{1 + \frac{1}{\sigma_b^2}}
\]

\[
\sigma_b^2 \leftarrow \sigma_b^2 + \frac{1}{1 + \frac{1}{\sigma_b^2}}
\]

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:

\[
y_p \rightarrow \text{xs}[i-1] \text{ data}[i-1] = w_{1.i-1} y_{i-1}
\]

And similarly for \( y_a \).
Summary of update rules

The ratio, \((\frac{\sigma_D}{\sigma_R})^2\), 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 \(y^* \rightarrow w_k \rightarrow \xi_s[[k]]\))

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

\[
\mu_i \leftarrow \frac{\frac{w_i}{\sigma_D} Y^*_i + \frac{1}{\sigma_i^\alpha} \mu_i^\alpha + \frac{1}{\sigma_i^\beta} \mu_i^\beta}{\frac{w_i}{\sigma_D} + \frac{1}{\sigma_i^\alpha} + \frac{1}{\sigma_i^\beta}}
\]

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

\[
\mu_i^\alpha \leftarrow \frac{\frac{1}{\sigma_i^\alpha} \mu_{i-1}^\alpha + \frac{w_{i-1}}{\sigma_D} Y^*_{i-1}}{\frac{1}{\sigma_i^\alpha} + \frac{w_{i-1}}{\sigma_D}}
\]

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

![Diagram](Lect_24_BeliefProp.nb)
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 $y_i = μ$, and the error bars show $±σ_i$. 
Belief Propagation Routine: Execute this cell "manually" for each iteration

```
In[50]:=

yfit = Table[{0, 0}, {il, 1, size}];
g1b = ErrorListPlot[{|yfit|};
Dynamic[
   Show[
      {g1b, g2, g3, 
       Graphics[{Text["Iteration" <> ToString[iter], \(\frac{\text{size}}{2}, \text{size}\)]},
        PlotRange -> {-50, 50}, Axes -> {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.
In[53]:= Do[
    Pause[.333];
    μ[i] = 
        xs[i] data[i] + μa[i] + μβ[i]
        σD + 1/σa[i] + 1/σβ[i];

    σ[i] = 
        1./
        σD + 1/σa[i] + 1/σβ[i];

    μ[j] = 
        σD + 1/σa[j] + 1/σβ[j];

    σ[j] = 
        1./
        σD + 1/σa[j] + 1/σβ[j];

    nextj = j - 1;

    μα[nextj] = 
        xs[j] data[j] + 1. μa[j]
        σD + 1/σa[j];

    σα[nextj] = σR + 
        1./
        σD + 1/σa[j];

    nexti = i + 1;

    μβ[nexti] = 
        xs[i] data[i] + 1. μβ[i]
        σD + 1/σβ[i];

    σβ[nexti] = σR + 
        1./
        σD + 1/σβ[i];

    j --;
    i ++;
    iter ++;
    yfit = Table[{μ[i], σ[i]}, {i, 1, size}];
    g1b = ErrorListPlot[{yfit}];
    , {size - 1}];
**Exercises**

Run the descent algorithm using successive over-relaxation (SOR): \( \eta_{2[k]}:=1.9/(\lambda\times s[k]) \).

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
\begin{align*}
\text{In[70]:=} & \quad \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}] \times \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} \to \text{True}, \\
& \quad \text{PlotStyle} \to \{\text{RGBColor}[0, 0.5, 0]\}]; \\
& \text{g2b} = \text{ListPlot}[\text{data}, \text{Joined} \to \text{False}, \text{PlotStyle} \to \{\text{RGBColor}[0.75, 0., 0]\}];
\end{align*}
```

- Initialize

```math
\begin{align*}
\text{In[71]:=} & \quad \mu_0 = 1; \\
& \quad \mu_\alpha = 1; \sigma_\alpha = 100 000; (* \text{large uncertainty *}) \\
& \quad \mu_\beta = 1; \sigma_\beta = 100 000; (* \text{large*}) \\
& \quad \sigma_R = .5; \sigma_D = .1; \\
& \quad \mu = \text{Table}[\mu_0, \{i, 1, \text{size}\}]; \\
& \quad \sigma = \text{Table}[\sigma_\alpha, \{i, 1, \text{size}\}]; \\
& \quad \mu_\alpha = \text{Table}[\mu_0, \{i, 1, \text{size}\}]; \\
& \quad \sigma_\alpha = \text{Table}[\sigma_\alpha, \{i, 1, \text{size}\}]; \\
& \quad \mu_\beta = \text{Table}[\mu_0, \{i, 1, \text{size}\}]; \\
& \quad \sigma_\beta = \text{Table}[\sigma_\beta, \{i, 1, \text{size}\}]; \\
& \quad \text{iter} = 0; \\
& \quad i = 1; \\
& \quad j = \text{size};
\end{align*}
```
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]=

Lect_24_BeliefProp.nb 17
SINE WAVE DEMO: Belief Propagation Routine

In[86]:=
Do[
  Pause[0.2];
  \(\mu[i] = \frac{xs[i] data[i] + \mu a[i] + 1. \mu b[i]}{\sigma a[i]} + \frac{1}{\sigma b[i]}\); 
  \(\sigma[i] = \frac{1}{\sigma a[i]} + \frac{1}{\sigma b[i]}\);
  \(\mu[j] = \frac{xs[j] data[j] + \mu a[j] + 1. \mu b[j]}{\sigma a[j]} + \frac{1}{\sigma b[j]}\);
  \(\sigma[j] = \frac{1}{\sigma a[j]} + \frac{1}{\sigma b[j]}\);

  nextj = j - 1;
  \(\mu a[nextj] = \frac{xs[j] data[j] + 1. \mu a[j]}{\sigma a[j]}\);
  \(\sigma a[nextj] = \sigma R + \frac{1}{\sigma a[j]}\);

  nexti = i + 1;
  \(\mu b[nexti] = \frac{xs[i] data[i] + 1. \mu b[i]}{\sigma b[i]}\);
  \(\sigma b[nexti] = \sigma R + \frac{1}{\sigma b[i]}\);

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


References


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