Introduction to Neural Networks
U. Minn. Psy 5036
Daniel Kersten

Initialize

■ Read in Statistical Add-in packages:

```
Off[General::spell1];
<< Statistics`NormalDistribution`
```

```
<< Statistics`ContinuousDistributions`
<< Statistics`MultinormalDistribution`
```

```
<< Graphics`MultipleListPlot`
```

Graphical Models of dependence

■ Conditional independence

Two random variables may only become independent, once the value of some third variable is known. This is called conditional independence.

Influences between variables are represented by conditioning, and a graphical model can be designed to express the conditional independencies between variables.

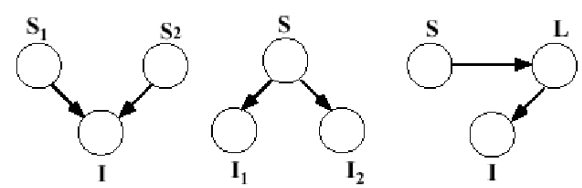
Recall from that two random variables are independent if and only if their joint probability is equal to the product of their individual probabilities. Thus, if p(A,B) = p(A)p(B), then A and B are independent. If p(A,B|C) = p(A|C)p(B|C), then A and B are conditionally independent.

■ Graphs & causal structure

The idea is to represent the probabilistic structure of the joint distribution P(S,L,I) by a Bayes net (e.g. Ripley, 1996}, which is a graphical model that expresses how random variables influence each other. Each node in the graph represents a random variable with an associated probability distribution. Markov Random Fields are one type of graphical model in which the random variables are represented in undirected graphs where the links between nodes do not have directions.

We are going to focus on directed acyclic graphs. Directed graphs connect the nodes with arrows, where the arrow is interpreted as a direction of influence or cause. One advantage of directed graphs is that one can use intuitions about the generative processes in constructing the linkages. Acyclic graphs avoid loops.

Beyond the two-node case, there are just three basic building blocks: converging, diverging, and intermediate nodes. For example, multiple causal variables causing a given measurement, a single variable producing multiple measurements, or a cause indirectly influencing a measurement through an intermediate variable. These types of influence provide a first step towards modeling the joint distribution and the means to compute probabilities of the unknown variables given known values.



In general, the product rule says that we can write:

p(S,L,I)=P(S,L|I)P(I)=P(S|L,I)P(L|I)P(I)

or

p(S,L,I)=P(S) P(L|S) P(I|S,L)

or...

But suppose we have a model that specifies the influence relationships as shown in one of the three above graphs.

The arrows tell us how to factor the joint probability into conditionals. So for the three examples above, we have:

p(S1,S2,I)=p(I|S1,S2)p(S1)p(S2)

p(S,I1,I2)=p(I1|S)p(I2|S)p(S)

p(S,L,I)=p(I|L)p(L|S)p(S)

The children of a node are at the arrow tip, and parents at the beginning of the arrow. In more complicated graphs, a node may have ancestors and descendents that are not parents or children (grand-parents, etc..).There can also be non-descendents.

Examples

■ "Explaining away"

Make the directed graph to help explain the corn/hay fever phenomenon

---

When corn prices drop in the summer, hay fever incidence goes up. However, if the joint on corn price and hay fever is conditioned on ``ideal weather for corn and ragweed", the correlation between corn prices and hay fever drops. This is because corn price and hay fever symptoms are conditionally independent.

There is a correlation between eating ice cream and drowning. Why? What event should you condition on to make the dependence go away?

■ What is noise? Primary and secondary variables

A common problem in perception is estimating a signal in noise. The graph for this is on the left above.

The data measurements (I) are determined by a typically non-linear function (ϕ) of primary signal variables (S1) and confounding secondary variables (S2). Knowledge is represented by the joint probability p(I,S1,S2). In general, the causal structure of natural data (e.g. image or speech) patterns is complex and consequently requires elaboration of its dependency structure. A graphical representation is one way to do this. For inference, the task is to make a decision about the signal hypotheses or primary signal variables (say S1), while discounting the noise or secondary variables (say S2). Thus optimal perceptual decisions are determined by p(I,S1), which is derived by summing over the secondary variables (i.e. marginalizing with respect to the secondary variables): $\int_{S2} p(I, S1, S2)\, dS2$, and if I is a known measurement, by the posterior p(S1|I): $\int_{S2} \frac{p(I,S1,S2)}{p(I)}\, dS2$

Noise is whatever you don't care to estimate, but contributes to the data. The secondary variables are noise.

Optimal Inference and task dependence: Fruit example

(due to James Coughlan; in Yuille, Coughlan, Kersten & Schrater).

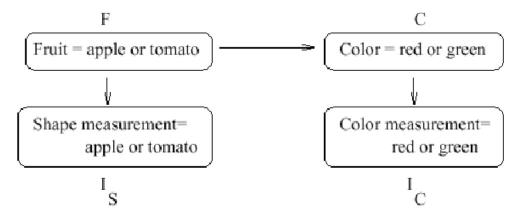


Figure from Yuille, Coughlan, Kersten & Schrater.

The the graph specifies how to decompose the joint probability:

p[F, C, Is, Ic] = p[Ic | C] p[C | F] p[Is | F] p[F]

The prior model on hypotheses, F & C

More apples (F=1) than tomatoes (F=2), and:

```
ppF[F_] := If[F == 1, 9 / 16, 7 / 16];
TableForm[Table[ppF[F], {F, 1, 2}], TableHeadings -> {{"F=a", "F=t"}}]
```

F=a	$\frac{9}{16}$
F=t	$\frac{7}{16}$

The conditional probability **cpCF[C|F]**:

```
cpCF[F_, C_] := Which[F == 1 && C == 1, 5 / 9,
   F == 1 && C == 2, 4 / 9, F == 2 && C == 1, 6 / 7, F == 2 && C == 2, 1 / 7];
TableForm[Table[cpCF[F, C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
```

	C=r	C=g
F=a	$\frac{5}{9}$	$\frac{4}{9}$
F=t	$\frac{6}{7}$	$\frac{1}{7}$

So the joint is:

```
jpFC[F_, C_] := cpCF[F, C] ppF[F];
TableForm[Table[jpFC[F, C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
```

	C=r	C=g
F=a	$\frac{5}{16}$	$\frac{1}{4}$
F=t	$\frac{3}{8}$	$\frac{1}{16}$

We can marginalize to get the prior probability on color alone is:

$$ppC[C\_] := \sum_{F=1}^{2} jpFC[F, C]$$

Question: Is fruit identity independent of material color--i.e. is F independent of C?

■ Answer

No.

```
TableForm[Table[jpFC[F, C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
TableForm[Table[ppF[F] ppC[C], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
```

	C=r	C=g
F=a	$\frac{5}{16}$	$\frac{1}{4}$
F=t	$\frac{3}{8}$	$\frac{1}{16}$

	C=r	C=g
F=a	$\frac{99}{256}$	$\frac{45}{256}$
F=t	$\frac{77}{256}$	$\frac{35}{256}$

The generative model: Imaging probabilities

Analogous to collecting histograms for the two switch positions in the SDT experiment, suppose that we have gathered some "image statistics" which provides us knowledge of how the image measurements for shape Is, and for color Ic depend on the type of fruit F, and material color, C. For simplicity, our measurements are discrete and binary (a more realistic case, they would have continuous values), say Is = {am, tm}, and Ic = {rm, gm}.

P(I_S=am,tm | F=a) = {11/16, 5/16}

P(I_S=am,tm | F=t) = {5/8, 3/8}

P(I_C=rm,gm | C=r) = {9/16, 7/16}

P(I_C=rm,gm | C=g) = {1/2, 1/2}

We use the notation am, tm, rm, gm because the measurements are already suggestive of the likely cause. So there is a correlation between apple and apple-like shapes, am; and between red material, and "red" measurements. In general, there may not be an obvious correlation like this.

We define a function for the probability of Ic given C, **cpIcC[Ic | C]**:

```
cpIcC[Ic_, C_] := Which[Ic == 1 && C == 1, 9 / 16,
  Ic == 1 && C == 2, 7 / 16, Ic == 2 && C == 1, 1 / 2, Ic == 2 && C == 2, 1 / 2];
TableForm[Table[cpIcC[Ic, C], {Ic, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"Ic=rm", "Ic=gm"}, {"C=r", "C=g"}}]
```

	C=r	C=g
Ic=rm	$\frac{9}{16}$	$\frac{7}{16}$
Ic=gm	$\frac{1}{2}$	$\frac{1}{2}$

The probability of Is conditional on F is **cpIsF[Is | F]**:

```
cpIsF[Is_, F_] := Which[Is == 1 && F == 1, 11 / 16,
   Is == 1 && F == 2, 5 / 8, Is == 2 && F == 1, 5 / 16, Is == 2 && F == 2, 3 / 8];
TableForm[Table[cpIsF[Is, F], {Is, 1, 2}, {F, 1, 2}],
 TableHeadings -> {{"Is=am", "Is=tm"}, {"F=a", "F=t"}}]
```

	F=a	F=t
Is=am	$\frac{11}{16}$	$\frac{5}{8}$
Is=tm	$\frac{5}{16}$	$\frac{3}{8}$

The total joint probability

We now have enough information to put probabilities on the 2x2x2 "universe" of possibilities, i.e. all possible combinations of fruit, color, and image measurements. Looking at the graphical model makes it easy to use the product rule to construct the total joint, which is:

p[F, C, Is, Ic] = p[Ic | C] p[C | F] p[Is | F] p[F]:

```
jpFCIsIc[F_, C_, Is_, Ic_] := cpIcC[ Ic, C ] cpCF[F, C ] cpIsF[Is, F ] ppF[F ]
```

Usually, we don't need the probabilities of the image measurements (because once the measurements are made, they are fixed and we want to compare the probabilities of the hypotheses. But in our simple case here, once we have the joint, we can calculate the probabilities of the image measurements through marginalization p(Is,Ic)=$\sum_C \sum_F p(F, C, \text{Is}, \text{Ic})$, too:

```
jpIsIc[Is_, Ic_] := Sum[jpFCIsIc[F, C, Is, Ic], {C, 1, 2}, {F, 1, 2}]
```

$$\texttt{jpIsIc[Is\_, Ic\_]} := \sum_{C=1}^{2} \sum_{F=1}^{2} \texttt{jpFCIsIc[F, C, Is, Ic]}$$

Three MAP tasks

Suppose that we measure Is=am, and Is = rm. The measurements suggest "red apple", but to find the most probable, we need to take into account the priors too.

■ Define argmax[] function:

```
argmax[x_] := Position[x, Max[x]];
```

■ Pick most probable fruit AND color--Answer "red tomato"

Using the total joint, p(F,C | Is, Ic) = $\frac{p(F,C,\text{Is},\text{Ic})}{p(\text{Is},\text{Ic})} \propto$ p(F,C,Is,Ic)

```
TableForm[jpFCIsIcTable = Table[jpFCIsIc[F, C, 1, 1], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
Max[jpFCIsIcTable]
argmax[jpFCIsIcTable]
```

	C=r	C=g
F=a	$\frac{495}{4096}$	$\frac{77}{1024}$
F=t	$\frac{135}{1024}$	$\frac{35}{2048}$

$\frac{135}{1024}$

```
{{2, 1}}
```

"Red tomato" is the most probable once we take into account the difference in priors.

Calculating p(F,C | Is, Ic). We didn't actually need p(F,C | Is, Ic), but we can calculate it by conditioning the total joint on the probability of the measurments:


```
jpFCcIsIc[F_, C_, Is_, Ic_] := jpFCIsIc[F, C, Is, Ic] / jpIsIc[Is, Ic]
```

```
TableForm[jpFCcIsIcTable = Table[jpFCcIsIc[F, C, 1, 1], {F, 1, 2}, {C, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}, {"C=r", "C=g"}}]
Max[jpFCcIsIcTable]
argmax[jpFCcIsIcTable]
```

	C=r	C=g
F=a	$\frac{55}{157}$	$\frac{308}{1413}$
F=t	$\frac{60}{157}$	$\frac{70}{1413}$

$\frac{60}{157}$

```
{{2, 1}}
```

■ Pick most probable color--Answer "red"

In this case, we want maximize the posterior:

p(C | Is, Ic)=$\sum_{F=1}^{2} p(F, C \mid \text{Is}, \text{Ic})$

$$\texttt{pC[C\_, Is\_, Ic\_]} := \sum_{F=1}^{2} \texttt{jpFCcIsIc[F, C, Is, Ic]}$$

```
TableForm[pCTable = Table[pC[C, 1, 1], {C, 1, 2}],
 TableHeadings -> {{"C=r", "C=g"}}]
Max[pCTable]
argmax[pCTable]
```

C=r	$\frac{115}{157}$
C=g	$\frac{42}{157}$

$\frac{115}{157}$

```
{{1}}
```

Answer is that the most probable material color is C = r, "red".

■ **Pick most probable fruit--Answer "apple"**

p(F | Is, Ic)

```
pF[F_, Is_, Ic_] := Sum[jpFCcIsIc[F, C, Is, Ic], {C, 1, 2}]
```

```
TableForm[pFTable = Table[pF[F, 1, 1], {F, 1, 2}],
 TableHeadings -> {{"F=a", "F=t"}}]
Max[pFTable]
argmax[pFTable]
```

F=a	$\frac{803}{1413}$
F=t	$\frac{610}{1413}$

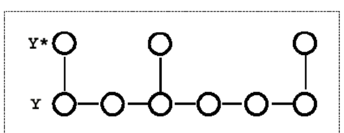
$\frac{803}{1413}$

{{1}}

The answer is "apple"

■ **Moral of the story: Optimal inference depends on the precise definition of the task**

Interpolation using smoothness revisited: Gradient descent



We use an example (and notation) due to Weiss.

First-order smoothness

Recall that the energy or cost function is given 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.

Gradient descent gives the following local update rule:

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

As before, 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**

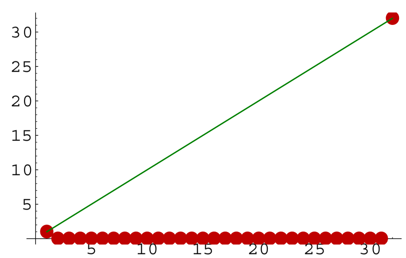
We return to the problem of interpolating a set of points with missing data, marked by an indicator function **xs**.

In[4838]:=

```
size = 32;
xs = Table[0, {i,1,size}]; xs[[1]]=1;xs[[size]]=1;(*xs[[size/2]]=1;*)
data = Table[N[j] xs[[j]],
{j, 1, size}];
g3 = ListPlot[Table[N[j],{j,1,size}],PlotJoined→True,
    DisplayFunction→Identity,PlotStyle→{RGBColor[0,.5,0]}];
g2 = ListPlot[data,PlotJoined->False,
PlotStyle->{RGBColor[.75,.0,0]}, Prolog-> AbsolutePointSize[5],
DisplayFunction->Identity];
```

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

In[4843]:=
```
Show[g2,g3,DisplayFunction->$DisplayFunction];
```



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[4844]:=
```
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:

```
Clear[f, d, λ]
(λ*Tm.Array[f, size] - Sm.((Array[d, size]) - Array[f, size])) // MatrixForm
```

■ Run gradient descent

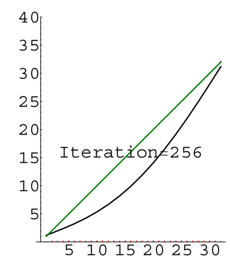
In[4863]:=
```
Clear[Tf,f1];
dt = 1; λ=2;
Tf[f1_] := f1 - dt*(1/(λ+xs))*(Tm.f1 - λ*Sm.(data-f1));
```

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

In[4871]:=
```
iter=256;
f = Table[0,{i,1,size}];
result = Nest[Tf,f,iter];
```

Now plot the interpolated function.

In[4878]:=
```
g1 = ListPlot[result,PlotJoined->True, AspectRatio->Automatic,PlotRange-
>{{1,size},{1,size}}, DisplayFunction->Identity];
Show[{g1,g2,g3,Graphics[{Text["Iteration="<>ToString[iter],{size/2,size/2}]
}]}, DisplayFunction->$DisplayFunction,PlotRange->{0,40}];
```



Same interpolation problem, but now using belief propagation

Belief propagation

Example is taken from Weiss.

■ Probabilistic generative model

$$y[[i]] = xs[[i]]\, d + dnoise, \quad dnoise \sim N[0, \sigma_D] \tag{1}$$

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

The ratio, $\left(\frac{\sigma_D}{\sigma_R}\right)^2$ plays the role of λ 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. We've changed notation $y^* \to d.w_k \to$ xs[[k]].

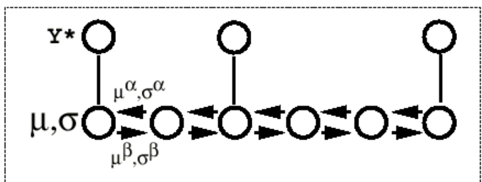
We'll make a notation change to avoid the square superscripts for $\sigma_D^2 \text{->} \sigma_D$, $\sigma_R^2 \text{->} \sigma_R$.

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

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

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

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



A simulation: Belief propagation for interpolation with missing data

In[4762]:=
```
μ0 = 1;
μα = 1; σα = 100000;
μβ = 1; σβ = 100000;
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 $\pm\sigma$.

$$\mu[[i]] = \frac{\frac{xs[[i]]}{\sigma D} * data[[i]] + \frac{1}{\sigma\alpha[[i]]} * \mu\alpha[[i]] + \frac{1.0}{\sigma\beta[[i]]} * \mu\beta[[i]]}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\alpha[[i]]} + \frac{1}{\sigma\beta[[i]]}};$$

$$\sigma[[i]] = \frac{1.0}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\alpha[[i]]} + \frac{1}{\sigma\beta[[i]]}};$$

$$\mu[[j]] = \frac{\frac{xs[[j]]}{\sigma D} * data[[j]] + \frac{1}{\sigma\alpha[[j]]} * \mu\alpha[[j]] + \frac{1.0}{\sigma\beta[[j]]} * \mu\beta[[j]]}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]} + \frac{1}{\sigma\beta[[j]]}};$$

$$\sigma[[j]] = \frac{1.0}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]} + \frac{1}{\sigma\beta[[j]]}};$$

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

$$\mu\alpha[[nextj]] = \frac{\frac{xs[[j]]}{\sigma D} * data[[j]] + \frac{1.0}{\sigma\alpha[[j]]} * \mu\alpha[[j]]}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]}};$$

$$\sigma\alpha[[nextj]] = \sigma R + \frac{1.0}{\frac{xs[[j]]}{\sigma D} + \frac{1}{\sigma\alpha[[j]]}};$$

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

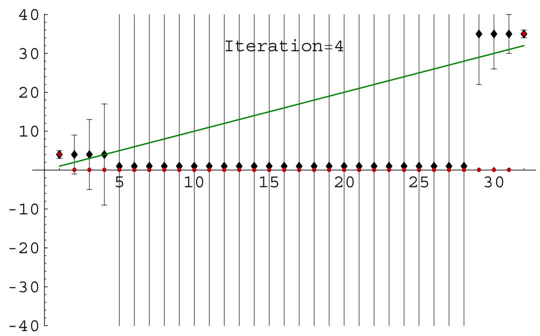
$$\mu\beta[[nexti]] = \frac{\frac{xs[[i]]}{\sigma D} * data[[i]] + \frac{1.0}{\sigma\beta[[i]]} * \mu\beta[[i]]}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\beta[[i]]}};$$

$$\sigma\beta[[nexti]] = \sigma R + \frac{1.0}{\frac{xs[[i]]}{\sigma D} + \frac{1}{\sigma\beta[[i]]}};$$

```
j--;
i++;

iter++;

yfit = Table[{μ[[i1]], ErrorBar[σ[[i1]]]}, {i1, 1, size}];
g1b = MultipleListPlot[yfit, DisplayFunction → Identity];
Show[{g1b, g2, g3,
   Graphics[{Text["Iteration=" <> ToString[iter], {size / 2, size}]}]},
  DisplayFunction → $DisplayFunction, PlotRange → {-40, 40}];
```



Appendices

Using *Mathematica* lists to manipulate discrete priors, likelihoods, and posteriors

■ A note on list arithmetic

We haven't done standard matrix/vector operations above to do conditioning. We've take advantage of how *Mathematica* divides a 2x3 array by a 2-element vector:

```
M=Array[m,{2,3}]
X = Array[x,{2}]
```

$$\begin{pmatrix} m(1,1) & m(1,2) & m(1,3) \\ m(2,1) & m(2,2) & m(2,3) \end{pmatrix}$$

$$\{x(1), x(2)\}$$

```
M/X
```

$$\begin{pmatrix} \frac{m(1,1)}{x(1)} & \frac{m(1,2)}{x(1)} & \frac{m(1,3)}{x(1)} \\ \frac{m(2,1)}{x(2)} & \frac{m(2,2)}{x(2)} & \frac{m(2,3)}{x(2)} \end{pmatrix}$$

■ Putting the probabilities back together again to get the joint

```
Transpose[Transpose[pHx] px]
```

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$$

```
pxH pH
```

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$$

■ Getting the posterior from the priors and likelihoods:

One reason Bayes' theorem is so useful is that it is often easier to formulate the likelihoods (e.g. from a causal or generative-model of how the data could have occurred), and the priors (often from heuristics, or in computational vision empirically testable models of the external visual world). So let's use *Mathematica* to derive **p(H|x)** from **p(x|H)** and **p(H)** , (i.e. pHx from pxH and pH).

```
px2 = Plus @@ (pxH pH)
```

$$\left\{\frac{5}{12}, \frac{1}{4}, \frac{1}{3}\right\}$$

```
Transpose[Transpose[(pxH pH)] / Plus @@ (pxH pH)]
```

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{2} \\ \frac{4}{5} & \frac{2}{3} & \frac{1}{2} \end{pmatrix}$$

■ Show that this joint probability has a uniform prior (i.e. both priors equal).

```
p = {{1/8, 1/8, 1/4}, {1/4, 1/8, 1/8}}
```

$\{\{\frac{1}{8}, \frac{1}{8}, \frac{1}{4}\}, \{\frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}\}$

Marginalization and conditioning: A small dimensional example using list manipulation in *Mathematica*

■ A discrete joint probability

All of our knowledge regarding the signal discrimination problem can be described in terms of the joint probability of the hypotheses, **H** and the possible data measurements, **x.** The probability function assigns a number to all possible combinations:

p[H, x]

That is, we are assuming that both the hypotheses and the data are discrete random variables.

$$H = \begin{cases} S1 \\ S2 \end{cases}$$

$$x \in \{1, 2, \ldots\}$$

Let's assume that x can only take on one of three values, 1, 2, or 3. And suppose the joint probability is:

$$p = \{\{\frac{1}{12}, \frac{1}{12}, \frac{1}{6}\}, \{\frac{1}{3}, \frac{1}{6}, \frac{1}{6}\}\}$$

$$\begin{pmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$$


```
TableForm[p, TableHeadings -> {{"H=S1", "H=S2"}, {"x=1", "x=2", "x=3"}}]
```

	x=1	x=2	x=3
H=S1	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{6}$
H=S2	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$

The total probability should sum up to one. Let's test to make sure. We first turn the list of lists into a singel list of scalars using **Flatten[]**. And then we can sum either with **Apply[Plus,Flatten[p]].**

```
Plus @@ Flatten[p]
```

1

We can pull out the first row of p like this:

```
p[[1]]
```

$\{\frac{1}{12}, \frac{1}{12}, \frac{1}{6}\}$

Is this the probability of x? No. For a start, the numbers don't sum to one. But we can get it through the two processes of marginalization and conditioning.

■ Marginalizing

What are the probabilities of the data, p(x)? To find out, we use the *sum rule* to sum over the columns:

```
px = Apply[Plus, p]
```

$\{\frac{5}{12}, \frac{1}{4}, \frac{1}{3}\}$

"Summing over "is also called **marginalization** or **"integrating out".** Note that marginalization turns a probability function with higher degrees of freedom into one of lower degrees of freedom.

What are the prior probabilities? p(H)? To find out, we sum over the rows:

```
pH = Apply[Plus, Transpose[p]]
```

$\left\{\frac{1}{3}, \frac{2}{3}\right\}$

■ Conditioning

Now that we have the marginals, we can get use the *product rule* to obtain the conditional probability through conditioning of the joint:

$$p[x \mid H] = \frac{p[H,\ x]}{p[H]}$$

In the Exercises, you can see how to use *Mathematica* to do the division for conditioning. The syntax is simple:

```
pxH = p / pH
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Note that the probability of x conditional on H sums up to 1 over x, i.e. each row adds up to 1. But, the columns do not. **p[x|H]** is a **probability** function of x, but a **likelihood** function of H. The posterior probability is obtained by conditioning on x:

$$p[H \mid x] = \frac{p[H,\ x]}{p[x]}$$

Syntax here is a bit more complicated, because the number of columns of px don't match the number of rows of p. We use Transpose[] to exchange the columns and rows of p before dividing, and then use Transpose again to get back the 2x3 form:

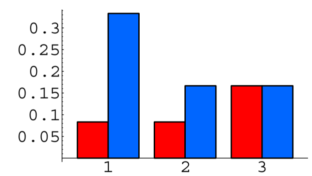
```
pHx = Transpose[Transpose[p] / px]
```

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{2} \\ \frac{4}{5} & \frac{2}{3} & \frac{1}{2} \end{pmatrix}$$

Plotting the joint

The following BarChart[] graphics function requires in add-in package (**<< Graphics`Graphics`**), which is specified at the top of the notebook. You could also use **ListDensityPlot[]**.


```
BarChart[p[[1]], p[[2]]];
```



Marginalization and conditioning: An example using *Mathematica* functions

■ A discrete joint probability

All of our knowledge regarding the signal discrimination problem can be described in terms of the joint probability of the hypotheses, **H** and the possible data measurements, **x.** The probability function assigns a number to all possible combinations:

p[H, x]

That is, we are assuming that both the hypotheses and the data are discrete random variables.

$$H = \begin{cases} S1 \\ S2 \end{cases}$$

```
x ∈ {1, 2, ...}
```

Let's assume that x can only take on one of three values, 1, 2, or 3. And suppose the joint probability is:

```
p[H_, x_] := Which[H == 1 && x == 1, 1 / 12, H == 1 && x == 2, 1 / 12, H == 1 && x == 3,
   1 / 6, H == 2 && x == 1, 1 / 3, H == 2 && x == 2, 1 / 6, H == 2 && x == 3, 1 / 6];
```

```
TableForm[Table[p[H, x], {H, 1, 2}, {x, 1, 3}],
 TableHeadings -> {{"H=s1", "H=s2"}, {"X=1", "X=2", "X=3"}}]
```

	X=1	X=2	X=3
H=s1	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{6}$
H=s2	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$

The total probability should sum up to one. Let's test to make sure. We first turn the list of lists into a singel list of scalars using **Flatten[]**. And then we can sum either with **Apply[Plus,Flatten[p]].**

```
Sum[p[H, x], {H, 1, 2}, {x, 1, 3}]
```

```
1
```

We can pull out the first row of p like this:

```
Table[p[1, x], {x, 1, 3}]
```

$$\left\{\frac{1}{12}, \frac{1}{12}, \frac{1}{6}\right\}$$

Is this the probability of x? No. For a start, the numbers don't sum to one. But we can get it through the two processes of marginalization and conditioning.

■ Marginalizing

What are the probabilities of the data, p(x)? To find out, we use the *sum rule* to sum over the columns:

```
px[x_] := Sum[p[H, x], {H, 1, 2}];
```

```
Table[px[x], {x, 1, 3}]
```

$$\left\{\frac{5}{12}, \frac{1}{4}, \frac{1}{3}\right\}$$

"Summing over "is also called **marginalization** or **"integrating out".** Note that marginalization turns a probability function with higher degrees of freedom into one of lower degrees of freedom.

What are the prior probabilities? p(H)? To find out, we sum over the rows:

```
pH[H_] := Sum[p[H, x], {x, 1, 3}];
```

```
Table[pH[H], {H, 1, 2}]
```

$$\left\{\frac{1}{3}, \frac{2}{3}\right\}$$

■ Conditioning

Now that we have the marginals, we can get use the *product rule* to obtain the conditional probability through conditioning of the joint:

$$p[x \mid H] = \frac{p[H, x]}{p[H]}$$

We use function definition in *Mathematica* to do the division for conditioning. The syntax is simple:

```
pxH[H_, x_] := p[H, x] / pH[H];
```

```
Table[pxH[H, x], {H, 1, 2}, {x, 1, 3}]
```

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

Note that the probability of x conditional on H sums up to 1 over x, i.e. each row adds up to 1. But, the columns do not. **p[x|H]** is a **probability** function of x, but a **likelihood** function of H. The posterior probability is obtained by conditioning on x:

$$p[H \mid x] = \frac{p[H, x]}{p[x]}$$

```
pHx[H_, x_] := p[H, x] / px[x];
```

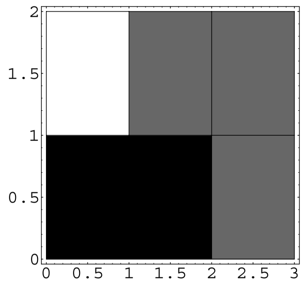
```
Table[pHx [H, x], {H, 1, 2}, {x, 1, 3}]
```

$$\begin{pmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{2} \\ \frac{4}{5} & \frac{2}{3} & \frac{1}{2} \end{pmatrix}$$

Plotting the joint

We use **ListDensityPlot[]**.

```
ListDensityPlot[Table[p[H, x], {H, 1, 2}, {x, 1, 3}]];
```



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.

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)

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

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

http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html

© 2000, 2001 Daniel Kersten, Computational Vision Lab, Department of Psychology, University of Minnesota.
(http://vision.psych.umn.edu/www/kersten-lab/kersten-lab.html)