INFORMATION THEORY

INFORMATION, ENTROPY, KL DISTANCE

Motivation

We have been using data points without acknowledging that such points take up space and take time to transmit

Information theory tackles methods for coding data so that it takes up less space

Since data is commonly probabilistic, information theory works with probability distributions

Information

So far we have talked about probability as the important indicator of where in a state space data is hiding out. Now, we introduce information to address the idea that the data have to be accessed somehow. It could be transmitted over time e.g. a radio, or transmitted over space as in the brain building memory representations.

Let's start with:

The information content of a set of N_m messages is defined to be

$$I_m = \log N_m$$

In other words, information is "the log of the number of messages."

Information Content

Information theory needs a place to put the symbols. This is the **channel**. A channel might be spatial, as in the page of a book, or temporal, as in a slice of time used to broadcast messages.

Let n be the number of discrete symbols $S_1, S_2, \ldots, S_n$ that can be used, and let m be the length of the message. Then the number of messages is

$$M = n^m$$

The **information capacity** of a channel with n symbols and m locations for symbols is defined as the logarithm of the number of messages. The reason for this definition is the intuition that information should be additive. Doubling the size of the channel should double the amount of information. This leads to

$$C_m = km \ln n$$

where k is a constant of proportionality. The next step is to choose a value for k. To do this the unit of channel capacity is defined as the capacity of a channel, which has just two symbols, thus

$$C_0 = 1 = k \ln 2$$

so that

$$k = \frac{1}{\ln 2}$$

Thus in general,

$$C = \frac{\ln n}{\ln 2} = \log_2 n$$

The dimensionless quantity is nonetheless referred to in terms of “bits.”

Channel Capacity

Now let us turn our attention to the messages that are to be placed in the channel. The information content of a set of N_m messages is defined to be $I_m = \log N_m$. Think of this as the minimum information capacity into which the information can be reversibly encoded assuming all the messages are equally frequent. Naturally in designing a channel you would want the capacity to be greater than the information content of the messages, that is,

$$I_m = \log N_m < C_m$$

For illustration, consider the database of nine names shown in the left-hand column of Table 1. Using a code of 27 symbols (26 letters plus a blank) and names of length 5 letters allows 27^5 messages. Thus the channel capacity is

$$C_5 = \log 27^5 = 15 \log 3 = 23.78 \text{ bits}$$

It is easy to verify that the channel capacity is adequate for the nine messages:

$$I_m = \log N_m = \log 9 < 23.78$$

Since the information is so much less than the channel capacity, one suspects that the signal could be better encoded. Let's try the 4-bit binary code in Table 1. Now the channel capacity is just

$$C = 4 \text{ bits}$$

This is still less than the best code, which could use the beginning letter of each name as a 9-element character set. Then the capacity would be

$$C = \log 9 \text{ bits}$$

e 1. Binary encoding for nine names.

Name	Code
Alfie	0000
Brad	0001
Carol	0010
Derek	0011
Edwin	0100
Frank	0101
Greg	0110
Helga	1000
Irene	1001

Entropy

The foregoing discussion assumes that the channel is being used to send one symbol at a time. The more useful case occurs when the channel is used to send many symbols, where each symbol in the message occurs with a given **frequency**. For this case we need the concept of **entropy**, or information rate.

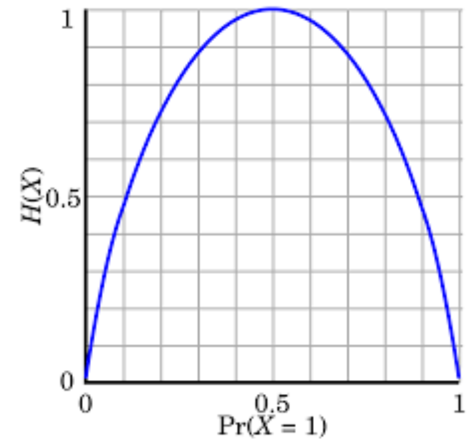
Entropy can also be thought of as a measure of the uncertainty of the contents of the message before it has been received. If there are n messages that have frequencies $p_i, i = 1, \ldots, n$, then entropy H is defined as

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{1}$$

Computer Bits vs Information bits

It is important to understand the relationship between "bits" as used in computers and bits as used in information theory. Suppose that a computer bit can take on the values 0 and 1 with equal probability. In this special case the entropy is

$$H = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$



In other words, one bit in the computer, given equal probabilities, produces one bit of information.

It can be shown that

$$0 \leq H \leq \log n$$

The lower bound is simply understood. Imagine that the message is known; then for that message, its probability is 1 and the rest are 0. Thus all the terms in the expression are 0, either by virtue of p_i being 0 or the one instance when $\log(1)$ is 0. Thus $0 \leq H$. For the upper bound, try to pick the probabilities to maximize H subject to the constraint that the sum of the probabilities is 1, that is, $\sum_{i=1}^{n} p_i = 1$. and the answer is that all the probabilities have to be equal, which makes them each $\frac{1}{n}$, so that $H_{max} = \log n$.

In summary, the basic result is that making the codes equally probable maximizes entropy.

Entropy = Average Information:

An illustrative example

Consider the case of a binary channel with only ones and zeros. Further constrain all messages to be of length m and to have exactly m_1 ones and m_2 zeros. Naturally $m_1 + m_2 = m$. The number of different possible messages of this distribution of ones and zeros is just

$$N_m = \binom{m}{m_1} = \frac{m!}{m_1!m_2!}$$

Thus the information in the ensemble of these messages is just

$$I_m = \log N_m = \log m! - \log m_1! - \log m_2!$$

If the $m_i, i = 1, 2$ are so large that $\log m_i \gg 1$, then you can approximate the preceding equation as

Stirling's approximation

$$\log N_m = m \log m - m_1 \log m_1 - m_2 \log m_2$$

So the average information, or entropy, $H = I_m/m$, can be obtained as:

$$H = \log m - \frac{m_1}{m} \log m_1 - \frac{m_2}{m} \log m_2$$

This can be rearranged as

$$-\frac{m_1}{m} \log \frac{m_1}{m} - \frac{m_2}{m} \log \frac{m_2}{m}$$

Finally, interpreting $\frac{m_i}{m}$ as the probability p_i leads to

$$H = -\sum_{i=1}^{2} p_i \log p_i$$

Entropy calculation

Coding theory: x discrete with 8 possible states; how many bits to transmit the state of x?

All states equally likely

$$\mathrm{H}[x] = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = 3 \text{ bits.}$$

Minimum code length

Think of sending a stream of messages of words of length of length l_i.
The average length of a message is given by

$$\sum p_i l_i$$

An information rate of $-\sum p_i \log p_i$ is the best we can do. Thus we expect that the average rate (or length) is going to be greater than this—that is, that

$$-\sum p_i \log p_i \leq \sum p_i l_i$$

where l_i is the length of the i^{th} code word. From this it is seen that equality occurs when

$$l_i = -\log p_i$$

and this is in fact the best strategy for picking the lengths of the code words.

Huffman coding finds minimal, reversible codes

$$p(1) = \frac{7}{8} \qquad p(0) = \frac{1}{8}$$

Suppose you choose blocks of three characters to be encoded. Then the possibilities, along with their probabilities of occurrence, are shown in Table 3.

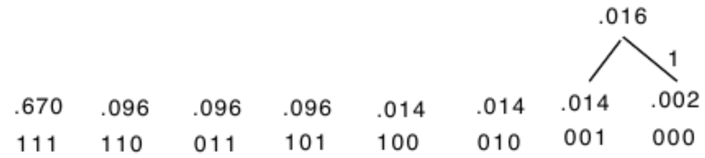
Table 2. Huffman encoding for the example.

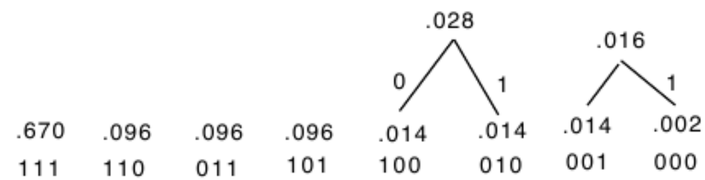
Block	Probability
111	.670
110	.096
011	.096
101	.096
100	.014
010	.014
001	.014
000	.002

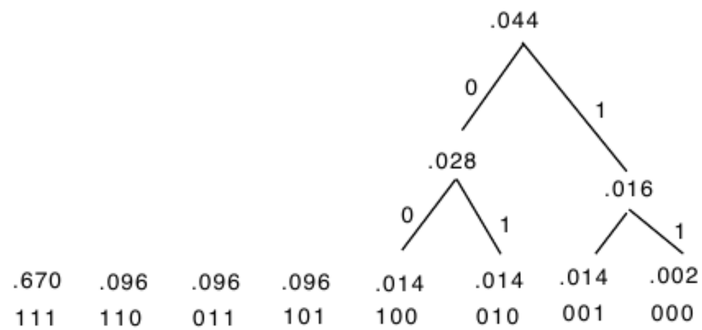
Algorithm Huffman Coding

1. Divide the data into blocks and generate the probabilities of the blocks according to the product of the probabilities of each of the symbols composing the block.
2. Pick the two smallest probabilities, add them, and record the result as the root of a tree).
3. Repeat this process, using the roots of trees as candidate probabilities along with any remaining code word probabilities.
4. When there is just one root, use the tree to generate the code words.

To generate the code words, the two smallest probabilities in the table are summed and form the root of a tree. This process is repeated, as described in the algorithm, until a single tree is formed.





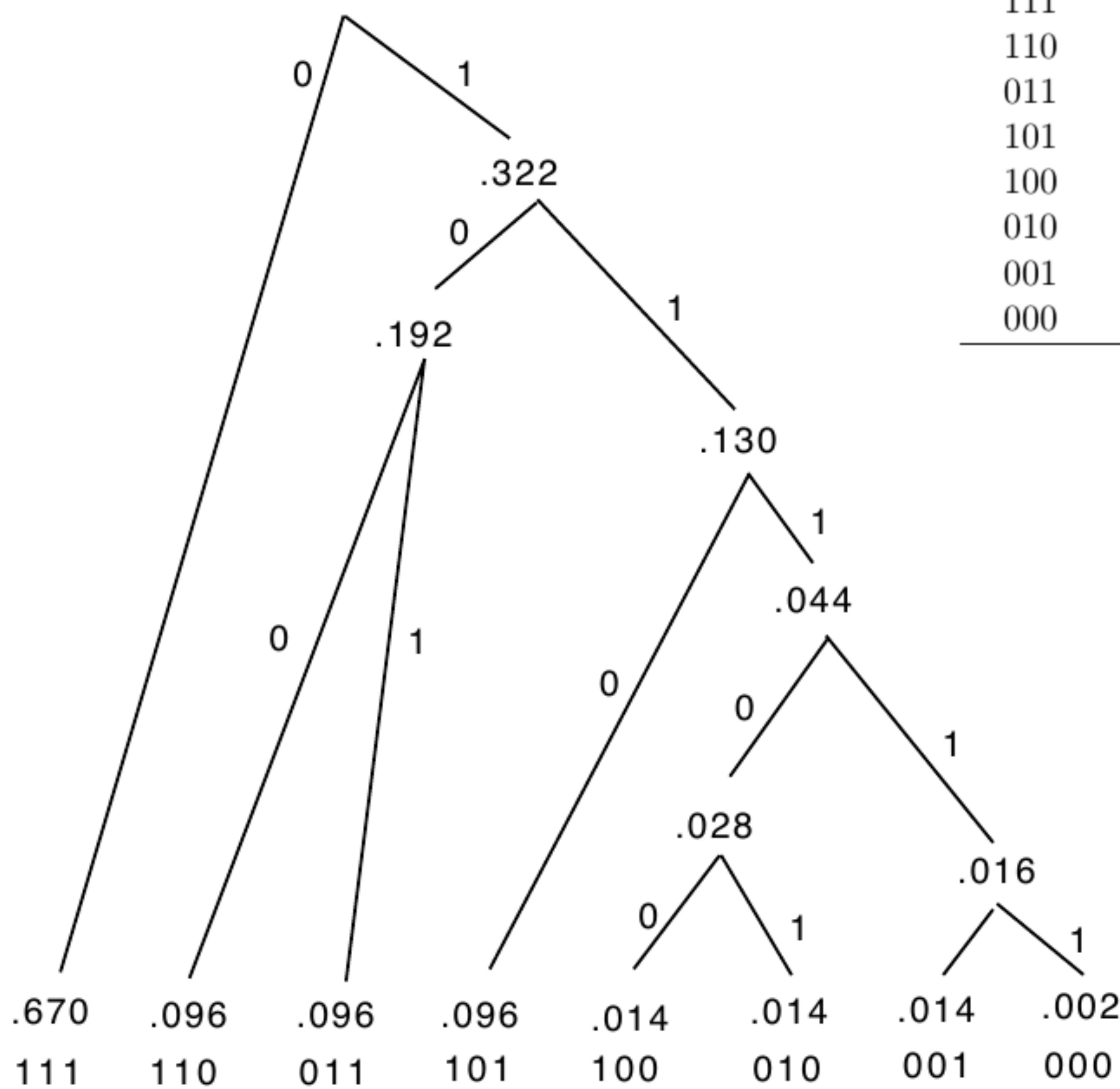


The completed tree shows the result of applying the algorithm to the data in the previous table. Given that tree, the code words are generated by labeling the left branches of the tree with zeros and the right branches with ones. Next, for each block to be encoded, the concatenation of the symbols on a path from the root to the block is used as the code word. For example, 100 is encoded as 11100.

Using this code, the first 21 characters of the original string can be encoded as

1000001100100

This result is 13 bits long, which is greater than the expected 11.30, but this difference is just due to the short length of the string.



Block	Probability	Code Word
111	.670	0
110	.096	100
011	.096	101
101	.096	110
100	.014	11100
010	.014	11101
001	.014	11110
000	.002	11111

Checking the example

x	a	b	c	d	e	f	g	h
$p(x)$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{1}{64}$
code	0	10	110	1110	111100	111101	111110	111111

$$\begin{aligned}\mathrm{H}[x] &= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{16}\log_2\frac{1}{16} - \frac{4}{64}\log_2\frac{1}{64} \\ &= 2 \text{ bits}\end{aligned}$$

$$\begin{aligned}\text{average code length} &= \frac{1}{2}\times 1 + \frac{1}{4}\times 2 + \frac{1}{8}\times 3 + \frac{1}{16}\times 4 + 4\times\frac{1}{64}\times 6 \\ &= 2 \text{ bits}\end{aligned}$$

Differential Entropy

Put bins of width Δ along the real line

$$\lim_{\Delta \to 0} \left\{ -\sum_i p(x_i)\Delta \ln p(x_i) \right\} = -\int p(x) \ln p(x) \, \mathrm{d}x$$

The tacit understanding is that differential entropy is going to be a *relative* measure.

$$\begin{aligned} H(X) &= -\lim_{\delta x \to 0} \sum_{k=-\infty}^{\infty} p_X(x_k)\, \delta x \log(p_X(x_k)\, \delta x) \\ &= -\lim_{\delta x \to 0} \left[\sum_{k=-\infty}^{\infty} p_X(x_k)(\log p_X(x_k))\, \delta x + \log \delta x \sum_{k=-\infty}^{\infty} p_X(x_k)\, \delta x \right] \\ &= -\int_{-\infty}^{\infty} p_X(x) \log p_X(x)\, dx - \lim_{\delta x \to 0} \log \delta x \int_{-\infty}^{\infty} p_X(x)\, dx \\ &= h(X) - \lim_{\delta x \to 0} \log \delta x \end{aligned}$$

$$p_X(x) = \begin{cases} \frac{1}{a}, & 0 \leq x \leq a \\ 0, & \text{otherwise} \end{cases}$$

The differential entropy of X is

$$\begin{aligned} h(X) &= -\int_0^a \frac{1}{a} \log\left(\frac{1}{a}\right) dx \\ &= \log a \end{aligned}$$

$$h(X + c) = h(X) \tag{10.14}$$

where c is constant.

Another useful property of $h(X)$ is described by

$$h(aX) = h(X) + \log |a| \tag{10.15}$$

where a is a scaling factor. To prove this property, we first recognize that since the area under the curve of a probability density function is unity, then

$$p_Y(y) = \frac{1}{|a|} p_Y\left(\frac{y}{a}\right) \tag{10.16}$$

Next, using the formula of Eq. (10.10), we may write

$$\begin{aligned} h(Y) &= -\mathbb{E}[\log p_Y(y)] \\ &= -\mathbb{E}\left[\log\left(\frac{1}{|a|} p_Y\left(\frac{y}{a}\right)\right)\right] \\ &= -\mathbb{E}\left[\log p_Y\left(\frac{y}{a}\right)\right] + \log |a| \end{aligned} \tag{10.17}$$

Differential Entropy of a Gaussian

Differential entropy maximized (for fixed σ^2) when

$$p(x) = \mathcal{N}(x|\mu, \sigma^2)$$

in which case

$$\mathrm{H}[x] = \frac{1}{2}\left\{1 + \ln(2\pi\sigma^2)\right\}.$$

$$\begin{aligned}\mathrm{H}[x] &= -\int p(x)\ln p(x)\,\mathrm{d}x \\ &= -\int p(x)\left(-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2}\right)\mathrm{d}x\end{aligned}$$

Conditional Entropy

$$\mathrm{H}[\mathbf{y}|\mathbf{x}] = -\iint p(\mathbf{y},\mathbf{x}) \ln p(\mathbf{y}|\mathbf{x}) \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}$$

$$\mathrm{H}[\mathbf{x},\mathbf{y}] = \mathrm{H}[\mathbf{y}|\mathbf{x}] + \mathrm{H}[\mathbf{x}]$$

Kullback-Leibler Divergence

Measures how close a distribution q(x) is to the ‘ideal’ (p(x)

$$
\begin{aligned}
\mathrm{KL}(p\|q) &= -\int p(\mathbf{x}) \ln q(\mathbf{x})\,\mathrm{d}\mathbf{x} - \left(-\int p(\mathbf{x}) \ln p(\mathbf{x})\,\mathrm{d}\mathbf{x}\right) \\
&= -\int p(\mathbf{x}) \ln \left\{\frac{q(\mathbf{x})}{p(\mathbf{x})}\right\} \mathrm{d}\mathbf{x}.
\end{aligned}
$$

The Kullback-Leibler Divergence $\mathrm{KL}(p\|q) \geqslant 0$?

$$f\left(\sum_{i=1}^{n} \lambda_i \mathbf{x}_i\right) \leq \sum_{i=1}^{n} \lambda_i f(\mathbf{x}_i)$$

f convex

$$-\mathbb{KL}\left(p||q\right) \quad = \quad -\sum_{x \in A} p(x) \log \frac{p(x)}{q(x)} = \sum_{x \in A} p(x) \log \frac{q(x)}{p(x)}$$

log concave

The Kullback-Leibler Divergence

$$f\left(\sum_{i=1}^{n}\lambda_i\mathbf{x}_i\right) \leq \sum_{i=1}^{n}\lambda_i f(\mathbf{x}_i)$$

f convex

$$\begin{aligned} -\mathbb{KL}\left(p||q\right) &= -\sum_{x\in A} p(x)\log\frac{p(x)}{q(x)} = \sum_{x\in A} p(x)\log\frac{q(x)}{p(x)} \\ &\leq \log\sum_{x\in A} p(x)\frac{q(x)}{p(x)} = \log\sum_{x\in A} q(x) \end{aligned}$$

log concave

The Kullback-Leibler Divergence

$$f\left(\sum_{i=1}^{n} \lambda_i \mathbf{x}_i\right) \leq \sum_{i=1}^{n} \lambda_i f(\mathbf{x}_i)$$

f convex

log concave

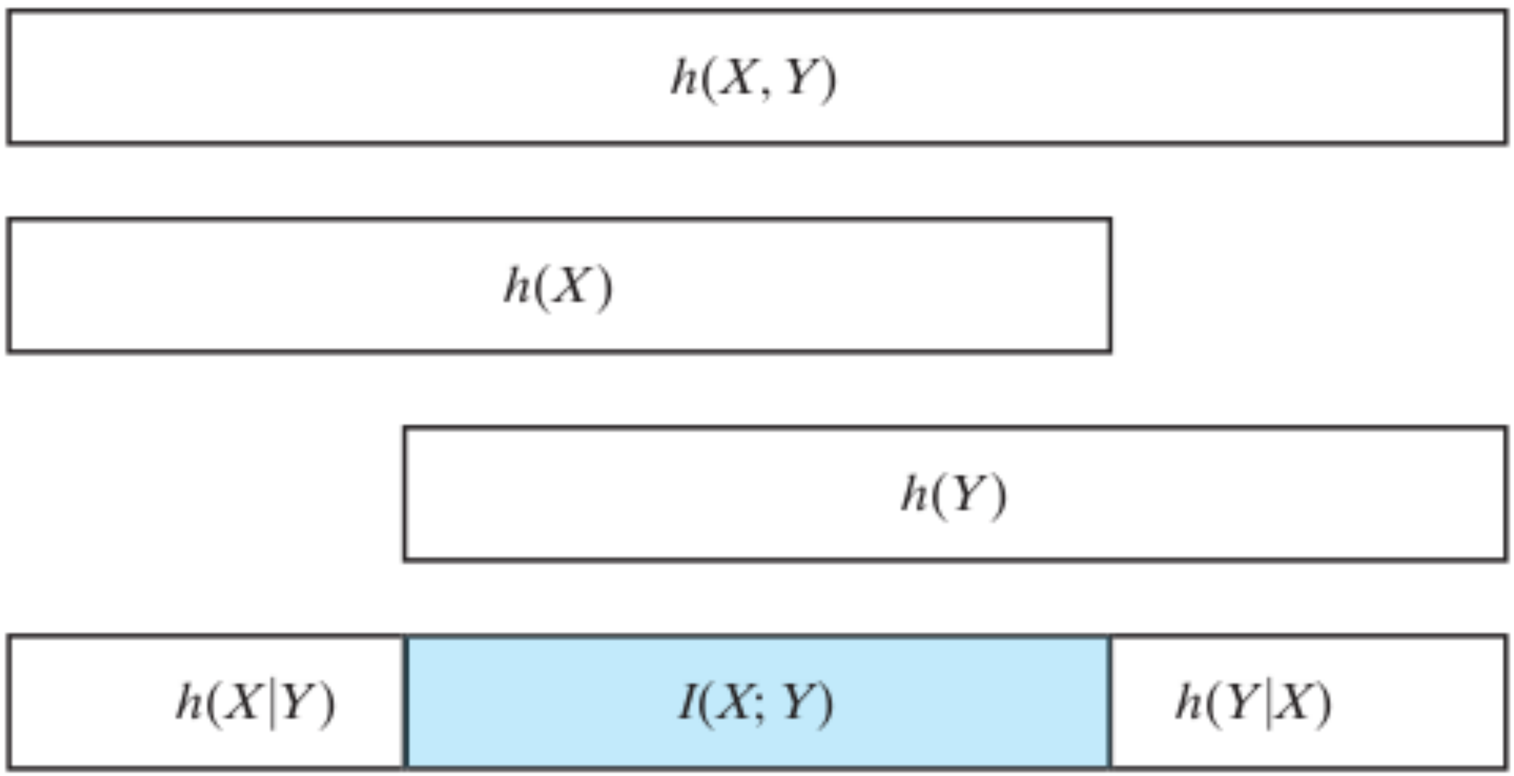
$$\begin{aligned}
-\mathbb{KL}(p||q) &= -\sum_{x \in A} p(x) \log \frac{p(x)}{q(x)} = \sum_{x \in A} p(x) \log \frac{q(x)}{p(x)} \\
&\leq \log \sum_{x \in A} p(x) \frac{q(x)}{p(x)} = \log \sum_{x \in A} q(x) \\
&\leq \log \sum_{x \in \mathcal{X}} q(x) = \log 1 = 0
\end{aligned}$$

Mutual Information

$$\mathrm{I}[\mathbf{x},\mathbf{y}] = \mathrm{H}[\mathbf{x}] - \mathrm{H}[\mathbf{x}|\mathbf{y}] = \mathrm{H}[\mathbf{y}] - \mathrm{H}[\mathbf{y}|\mathbf{x}]$$

$$\begin{aligned}
\mathrm{I}[\mathbf{x},\mathbf{y}] &\equiv \mathrm{KL}(p(\mathbf{x},\mathbf{y})\|p(\mathbf{x})p(\mathbf{y})) \\
&= -\iint p(\mathbf{x},\mathbf{y}) \ln\left(\frac{p(\mathbf{x})p(\mathbf{y})}{p(\mathbf{x},\mathbf{y})}\right) \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y}
\end{aligned}$$

Summary



Minimum Description Length

The idea is that of sending a message that describes a theory. One way to do so would be just to send all the data, as in a sense this is a literal description of the theory. But the intuition behind Occam's razor is to favor compact theories. MDL captures this by allowing the message to have the form of a description of the theory plus a description of the data when encoded by the theory. The assumption is that the sender and receiver agree on the semantics of a language for the message and the cost is then the length of the code for the message. Thus the combined length of the message, $L(M, D)$, is a sum of two parts,

$$|L(M, D)| = |L(M)| + |L(D \text{ encoded using } M)|$$

Bayesian Approach

In terms of Bayes' rule, the probability of a model given data can be expressed as

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

Picking the best model can be expressed as maximizing $P(M|D)$, or

$$\max_M P(D|M)P(M)$$

You do not have to consider $P(D)$ here because it is constant across all models. Now maximizing this expression is equivalent to maximizing its logarithm, as the logarithm is monotonic and will not affect the outcome. So

$$\max_M [P(D|M)P(M)] = \max_M [\log P(D|M) + \log P(M)]$$

and this is the same as minimizing its negative:

$$\min_M [-\log P(D|M) - \log P(M)] \tag{2}$$

But now remember the earlier result that for a minimal code that has probability of being sent P, the length of the code is

$$-\log P \tag{3}$$

MDL cost function for Gaussians

Assume the residuals are distributed in the form of a Gaussian with variance α. Then

$$p(D|M) = \left(\frac{1}{2\pi\alpha}\right)^{\frac{N}{2}} e^{-\frac{1}{2\alpha}\sum_{i=1}^{n}(x_i - m_i)^2}$$

If in turn the model is a neural network with a set of parameters $w_i, i = 1, \ldots, W$, then we can assume that they also are distributed according to a Gaussian, with variance β. Therefore,

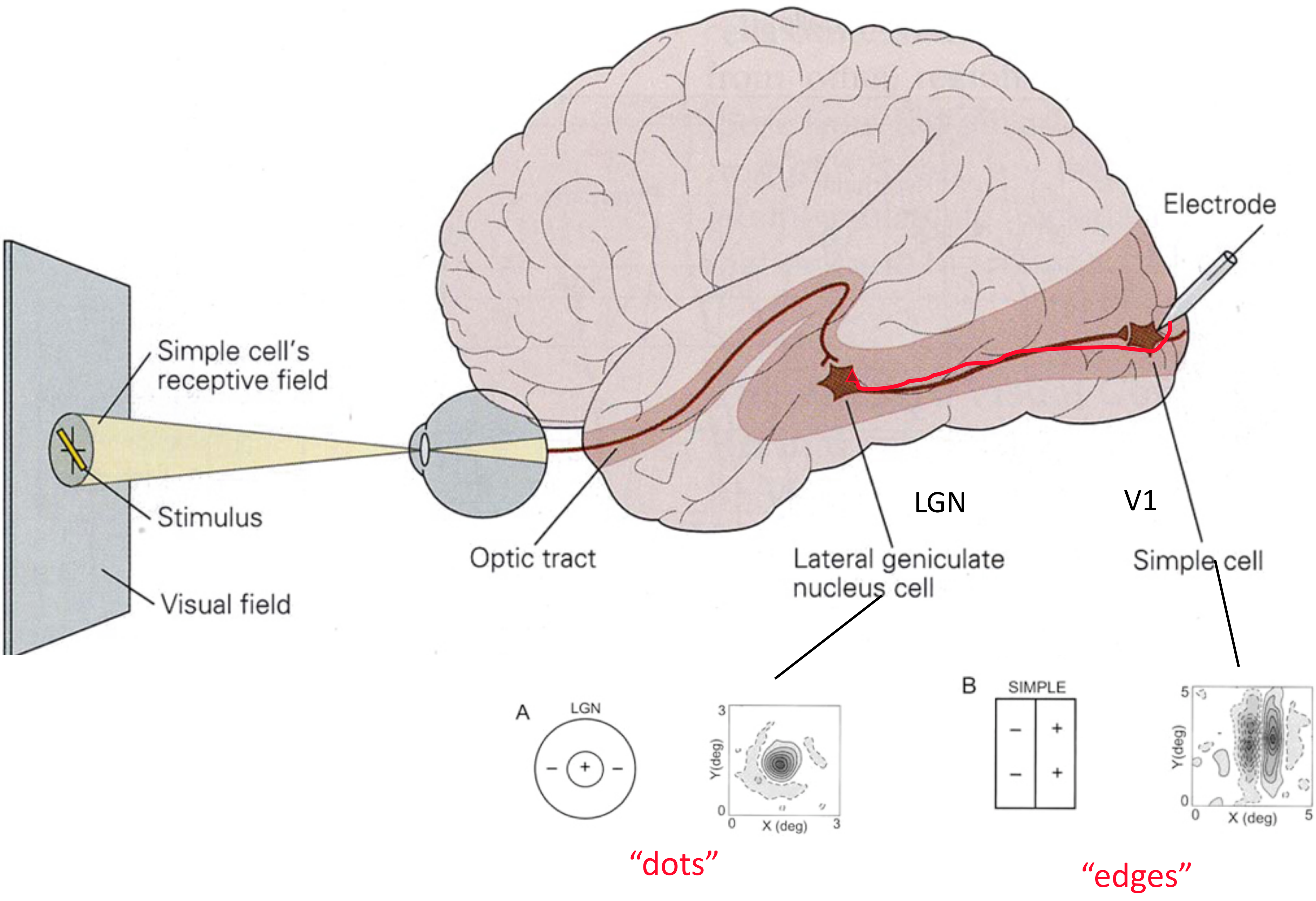
$$p(M) = \left(\frac{1}{2\pi\beta}\right)^{\frac{W}{2}} e^{-\frac{1}{2\beta}\sum_i w_i^2}$$

Substituting these two equations into Equation 2,

$$\min_M[-\log P(D|M) - \log P(M)] = \frac{1}{2\alpha}\sum_{i=1}^{n}(x_i - m_i)^2 + \frac{1}{2\beta}\sum_i w_i^2 + \text{ const.}$$

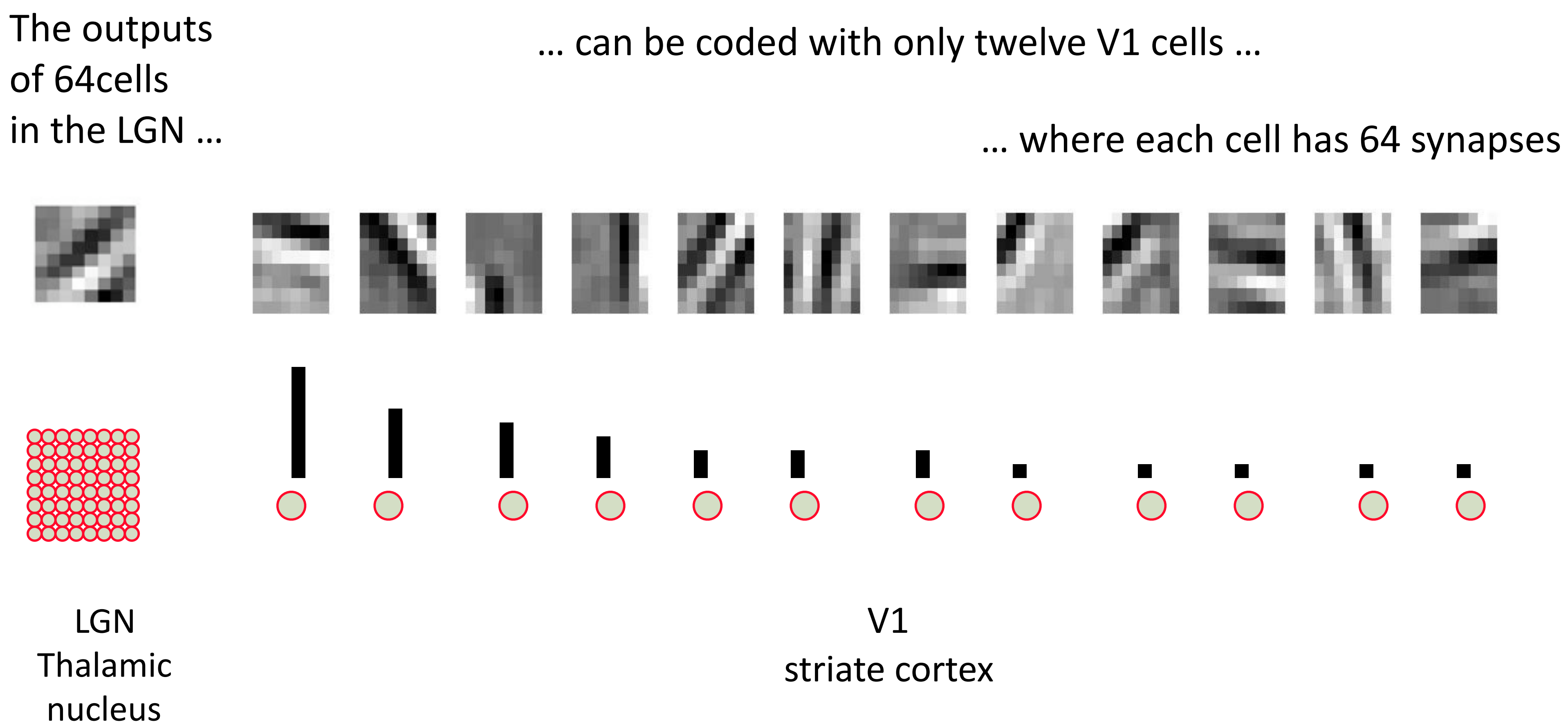
Reduces to a regression problem!

The basic circuit for study: The brain’s visual memory

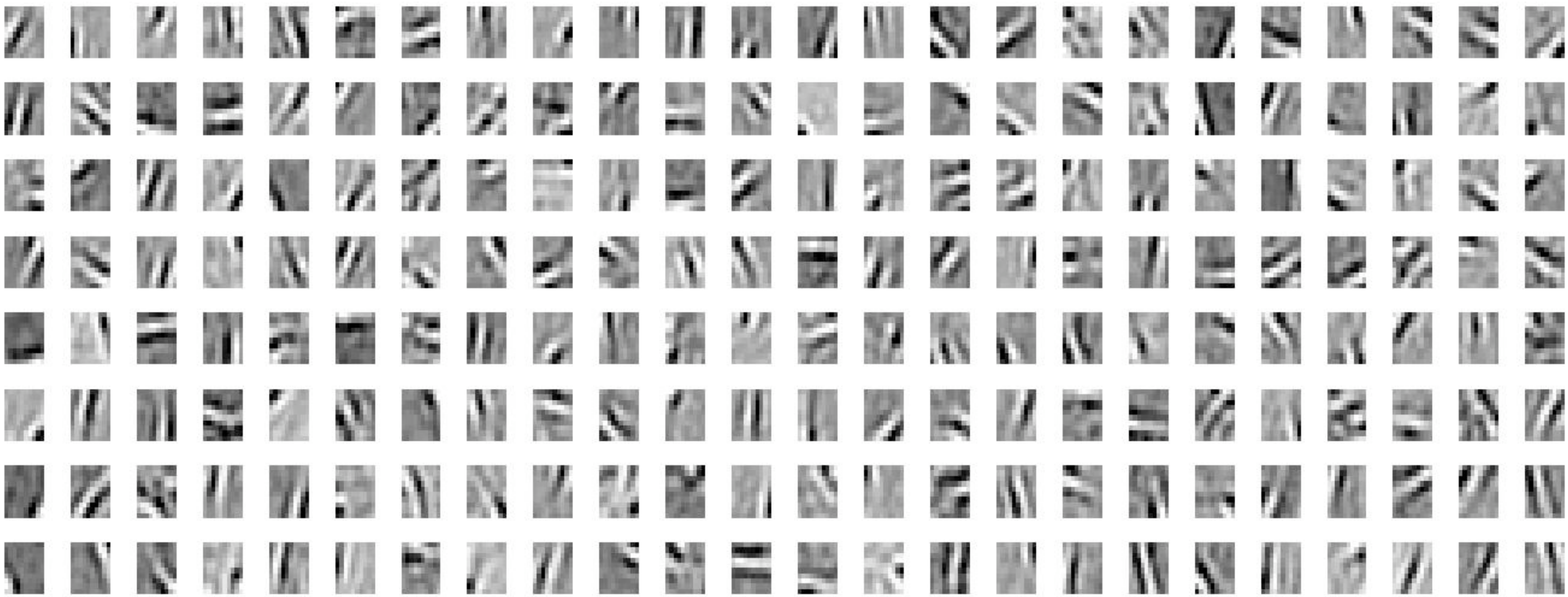


DeAngelis et al. (1995)

Approximating an image patch w basis functions



The neural coding library of learned RFs



Because there are more than we need - *Overcomplete* (192 vs 64) - the number of cells that need to send spikes at any moment is *Sparse* (12 vs 64).