Sampling Lecture 1 Summary

Factored distributions can be sampled : Ancestral sampling

One dimensional pdfs with inverses can be sampled

The trick:One can use the *uniform distribution* to create closed form algebraic formulas

Gaussians can be sampled

Use the trick to sample 2d then transform

But what about arbitrary distributions?

There are methods for the case where the distribution is known up to a scaling factor

Rejection sampling is simple but expensive for his dimensional pdfs

Because almost all samples are rejected

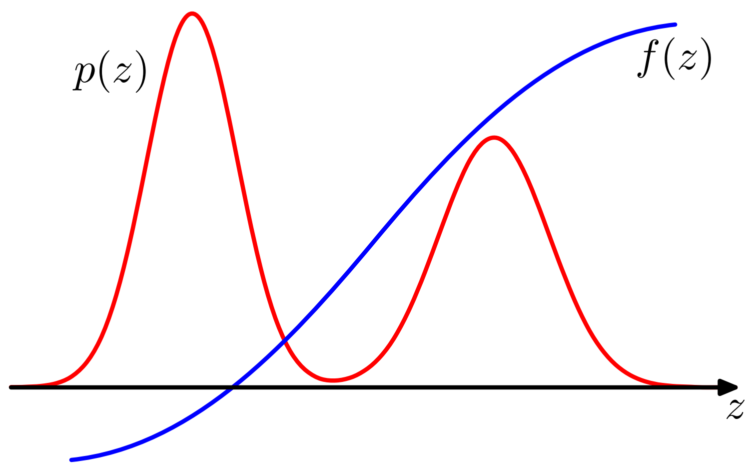
Importance sampling takes advantage of computing functions but expensive for his dimensional pdfs

Because most samples are either unimportant or low probability

Sampling Lecture 2
Markov Chain Monte Carlo (MCMC)

A very general framework that allows
sampling from a large class of distributions that scales well
with the dimensionality of the sample space.

In computing the expectation of a function we would like to sample places where **f(z)** and p**(z)** are both large so their product has the most weight in the sum. But in the previous methods there was no guarantee that the proposal distribution **q(z)** that we could sample from would be related to **f(z)**



$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}$$

Markov Chain Monte Carlo fixes this but taking account of the relative values to guide the sampling process.

Previously the samples were independent, but now …

1. Keep track of samples $\mathbf{z}^{(\tau)}$
2. Proposal distribution $q(\mathbf{z}|\mathbf{z}^{(\tau)})$ depends on current state
3. The samples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)} \ldots$ form a *Markov Chain*

Metropolis algorithm

$\text{Assume symmetry i.e. } q(\mathbf{z}_a|\mathbf{z}_b) = q(\mathbf{z}_a|\mathbf{z}_b)$

Candidate sample $\mathbf{z}^*$ is accepted with probability

$$A(\mathbf{z}^*, \mathbf{z}^\tau) = \min\left(1, \frac{\tilde{p}(\mathbf{z}^*}{\tilde{p}(\mathbf{z}^\tau}\right)$$

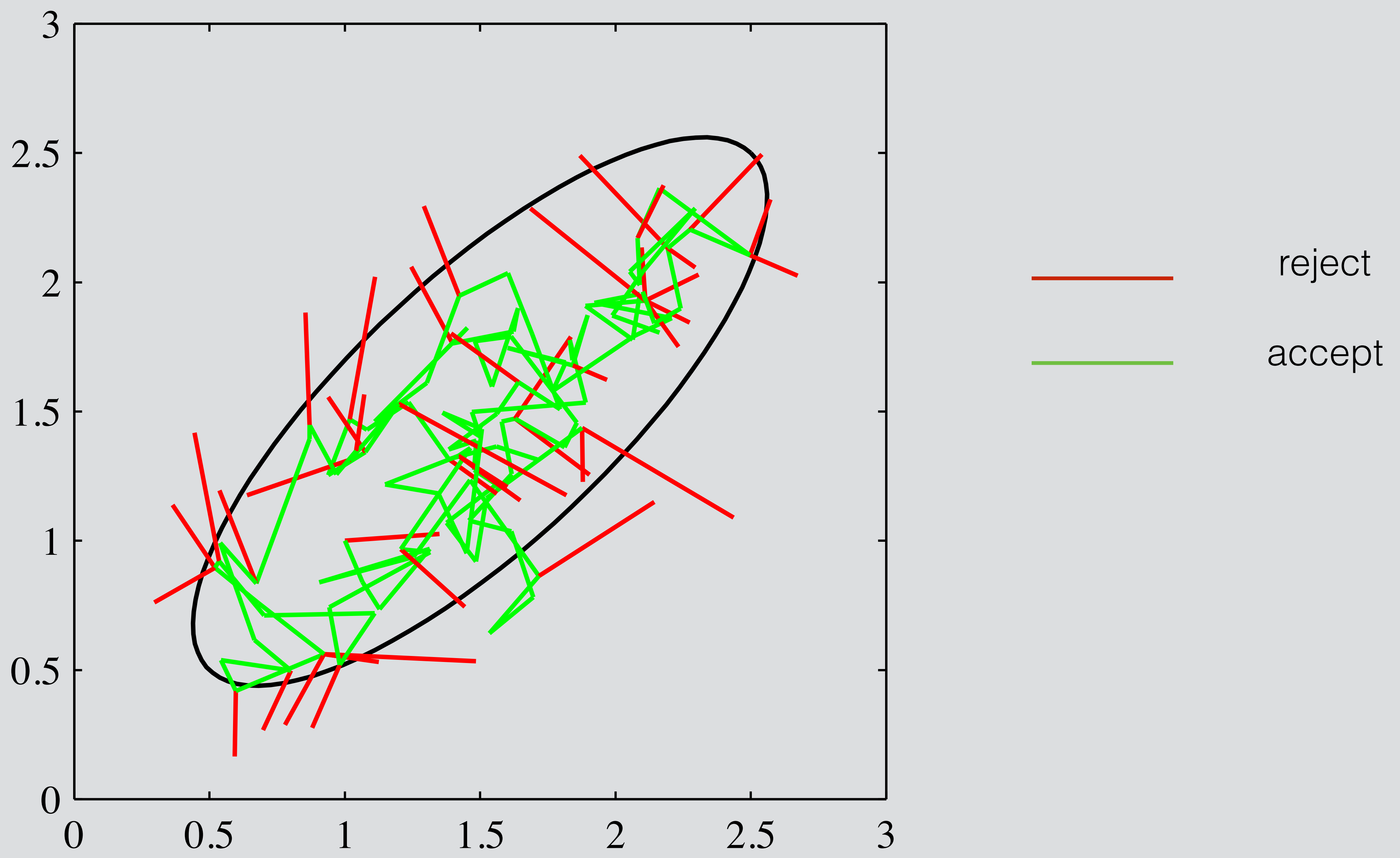
How to do this?

If accept $\quad \mathbf{z}^{(\tau+1)} = \mathbf{z}^*$

Else $\quad \mathbf{z}^{(\tau+1)} = \mathbf{z}^\tau$

Consequences ?

Metropolis Algorithm sampling a Gaussian



Proposal distribution is symmetric Gaussian $N(0,0.2)$

Markov Chains

Markov property $$p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(m)})$$

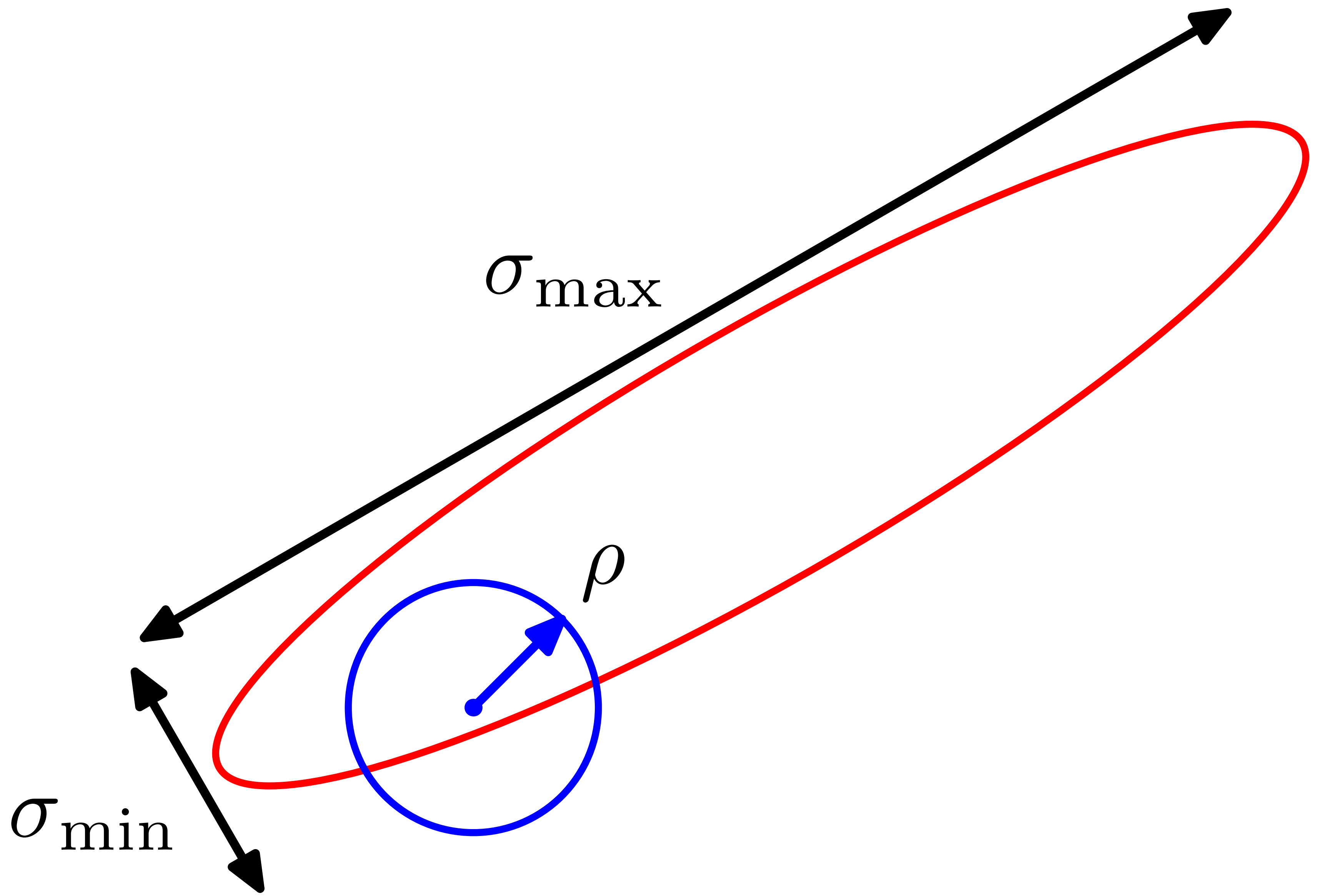
Marginal Probability $$p(\mathbf{z}^{(m+1)}) = \sum_{\mathbf{z}^{(m)}} p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(m)})p(\mathbf{z}^{(m)})$$

Invariant $$p^\star(\mathbf{z}) = \sum_{\mathbf{z}'} T(\mathbf{z}',\mathbf{z})p^\star(\mathbf{z}').$$

Detailed balance:
sufficient condition for invariance $$p^\star(\mathbf{z})T(\mathbf{z},\mathbf{z}') = p^\star(\mathbf{z}')T(\mathbf{z}',\mathbf{z})$$

It works because… $$\sum_{\mathbf{z}'} p^\star(\mathbf{z}')T(\mathbf{z}',\mathbf{z}) = \sum_{\mathbf{z}'} p^\star(\mathbf{z})T(\mathbf{z},\mathbf{z}') = p^\star(\mathbf{z})\sum_{\mathbf{z}'} p(\mathbf{z}'|\mathbf{z}) = p^\star(\mathbf{z}).$$

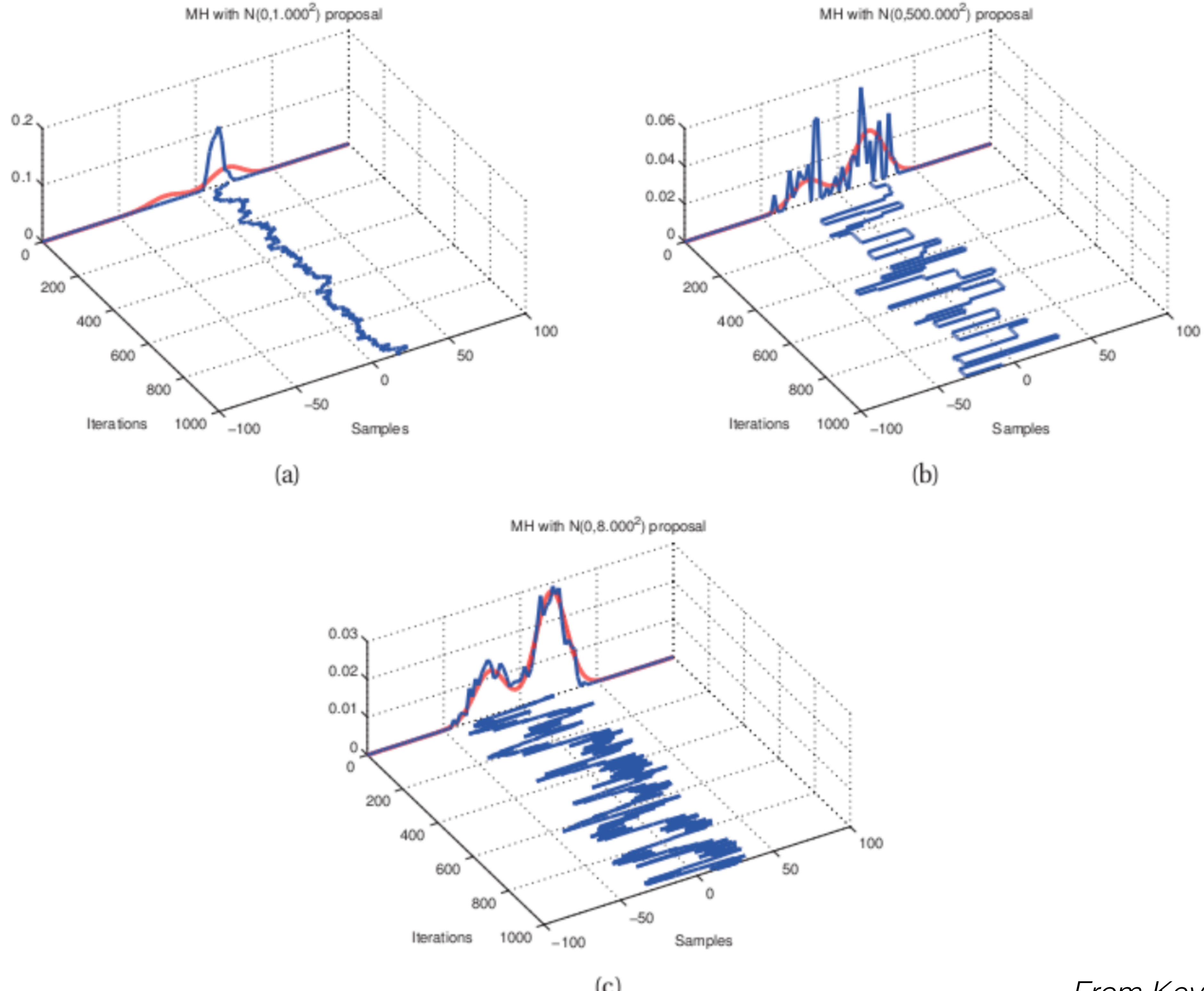
Issues when sampling with Metropolis-Hastings



If proposal Gaussian is too small, it takes a long time to walk elongated distribution

If proposal Gaussian is too large, many proposed samples are rejected

Experiments sampling from a sum of two Gaussians



From Kevin Murphy ML

Metropolis Hastings Algorithm

When proposal distribution is *not symmetric* have to make adjustments

$$A_k(\mathbf{z}^\star, \mathbf{z}^{(\tau)}) = \min\left(1, \frac{\widetilde{p}(\mathbf{z}^\star) q_k(\mathbf{z}^{(\tau)}|\mathbf{z}^\star)}{\widetilde{p}(\mathbf{z}^{(\tau)}) q_k(\mathbf{z}^\star|\mathbf{z}^{(\tau)})}\right)$$

Make identifications

$$\mathbf{z} = \mathbf{z}^{(T)}$$

$$\mathbf{z}' = \mathbf{z}^\star$$

$$\begin{aligned} p(\mathbf{z}) q_k(\mathbf{z}'|\mathbf{z}) A_k(\mathbf{z}', \mathbf{z}) &= \min\left(p(\mathbf{z}) q_k(\mathbf{z}'|\mathbf{z}), p(\mathbf{z}') q_k(\mathbf{z}|\mathbf{z}')\right) \\ &= \min\left(p(\mathbf{z}') q_k(\mathbf{z}|\mathbf{z}'), p(\mathbf{z}) q_k(\mathbf{z}'|\mathbf{z})\right) \\ &= p(\mathbf{z}') q_k(\mathbf{z}|\mathbf{z}') A_k(\mathbf{z}, \mathbf{z}') \end{aligned}$$

With the appropriate identifications $p^\star(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^\star(\mathbf{z}') T(\mathbf{z}', \mathbf{z})$

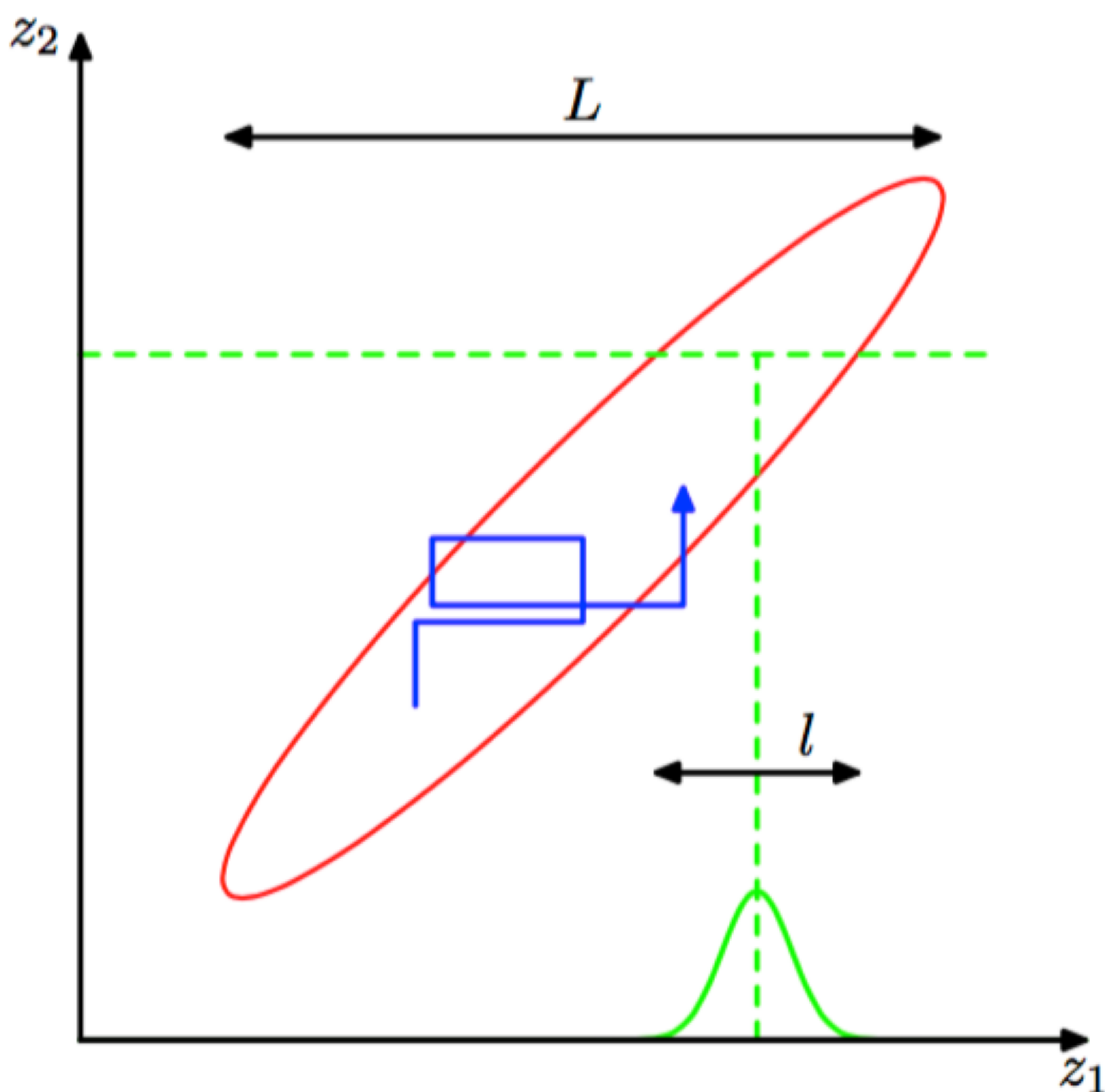
Why would we want an asymmetric proposal distribution?

Gibbs Sampling

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.

     $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.



z_2
L
l
z_1

Rosenbrock function

From Wikipedia, the free encyclopedia

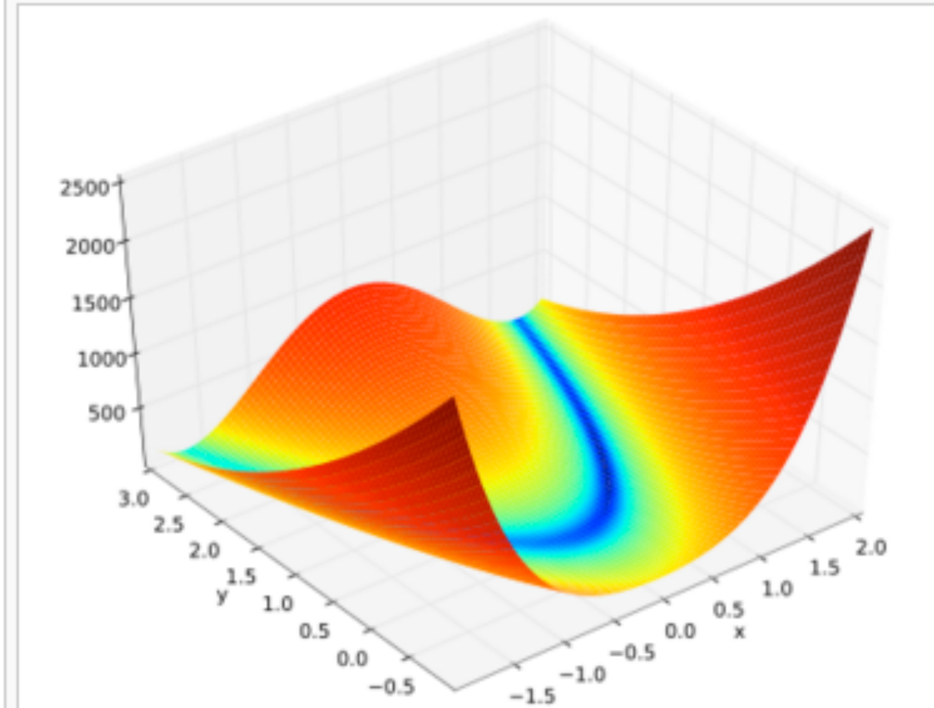
In mathematical optimization, the **Rosenbrock function** is a non-convex function used as a performance test problem for optimization algorithms introduced by Howard H. Rosenbrock in 1960.[1] It is also known as **Rosenbrock's valley** or **Rosenbrock's banana function**.

The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the global minimum, however, is difficult.

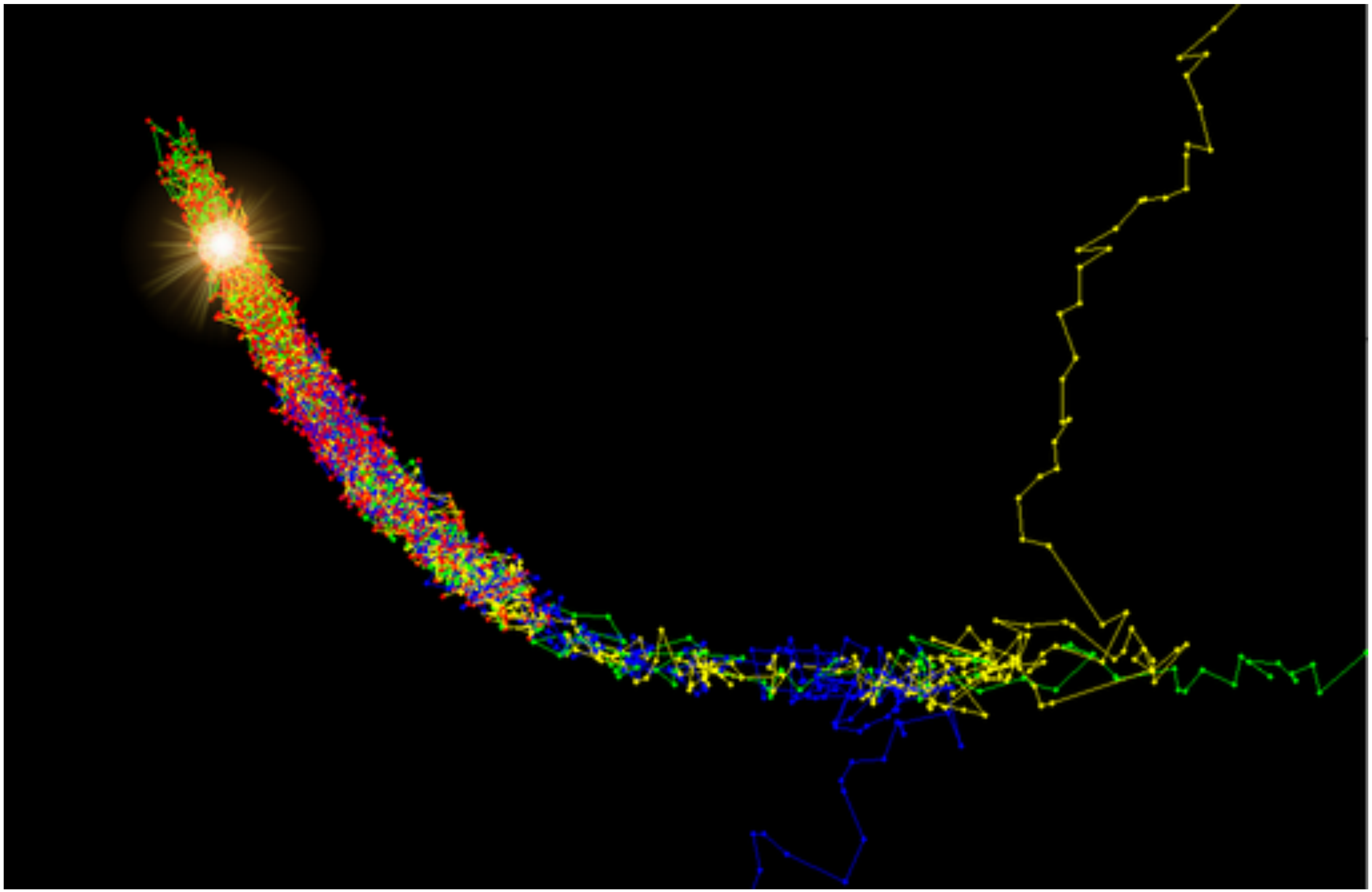
The function is defined by

$$f(x,y)=(a-x)^2+b(y-x^2)^2$$

It has a global minimum at $(x,y)=(a,a^2)$, where $f(x,y)=0$. Usually $a=1$ and $b=100$.



Plot of the Rosenbrock function of two variables.



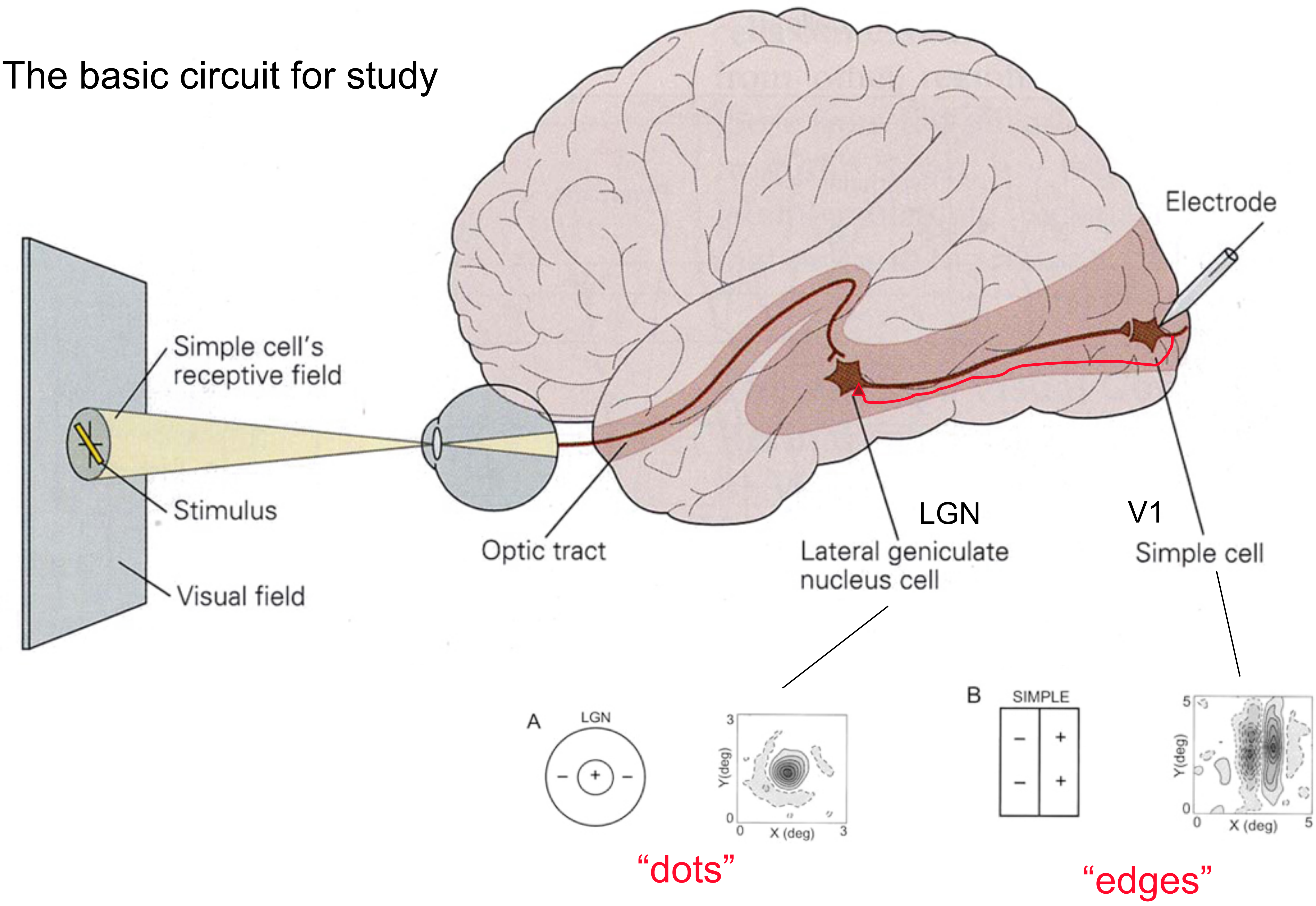
For sampling a multidimensional Gaussian, the successive samples can be correlated. A heuristic to exploit this correlation is to modify the samples with the following formula:

$$z_i' = \mu_i + \alpha(z_i - \mu_i) + \sigma_i(1 - \alpha^2)^{\frac{1}{2}}\nu$$

where μ_i is the mean of z_i and σ_i^2 is its variance and where ν is a Gaussian random variable of zero mean and unit variance and $-1 < \alpha < 1$.

Show that the mean of z_i' is also μ_i and its variance is also σ_i^2.

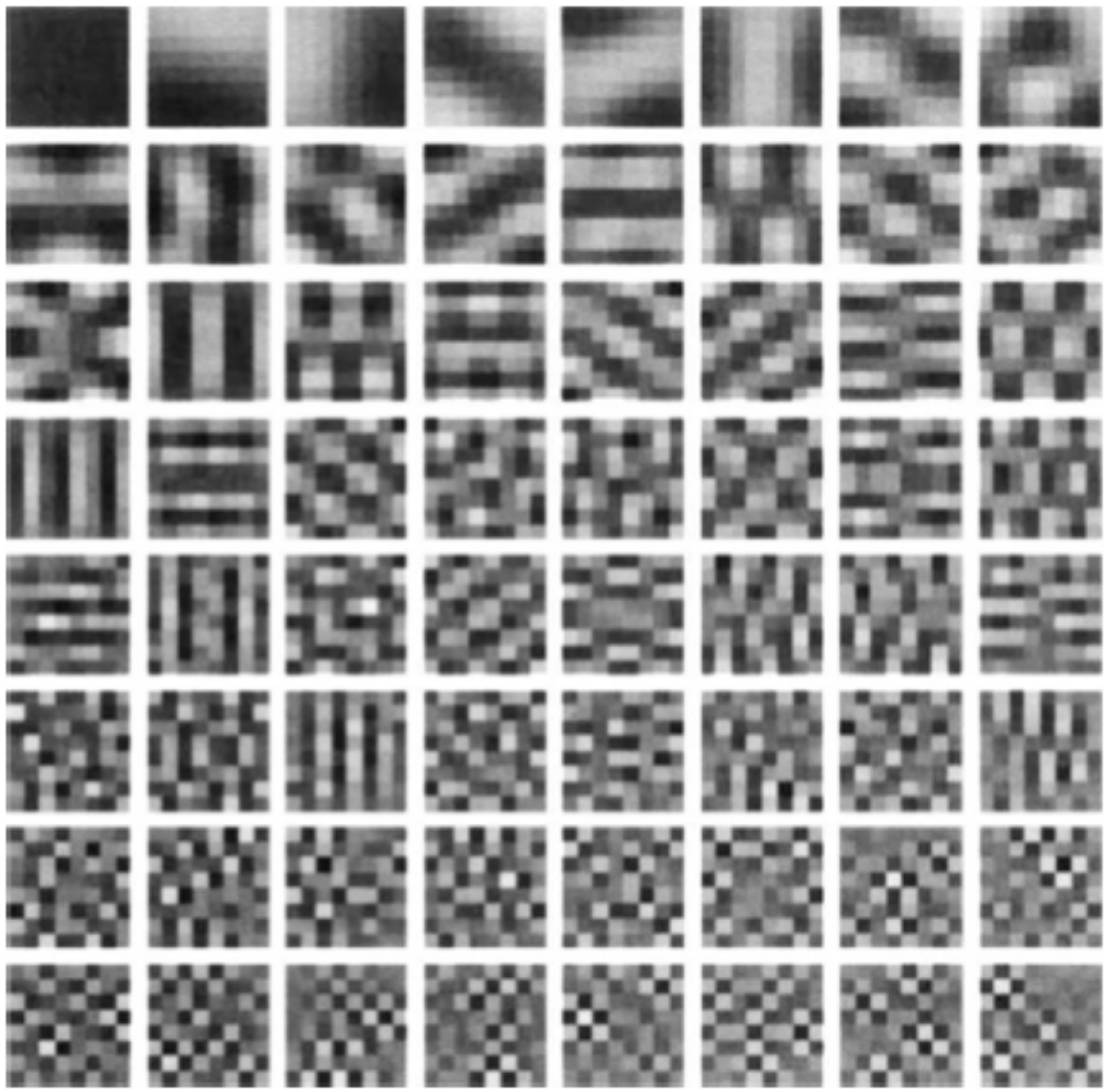
The basic circuit for study

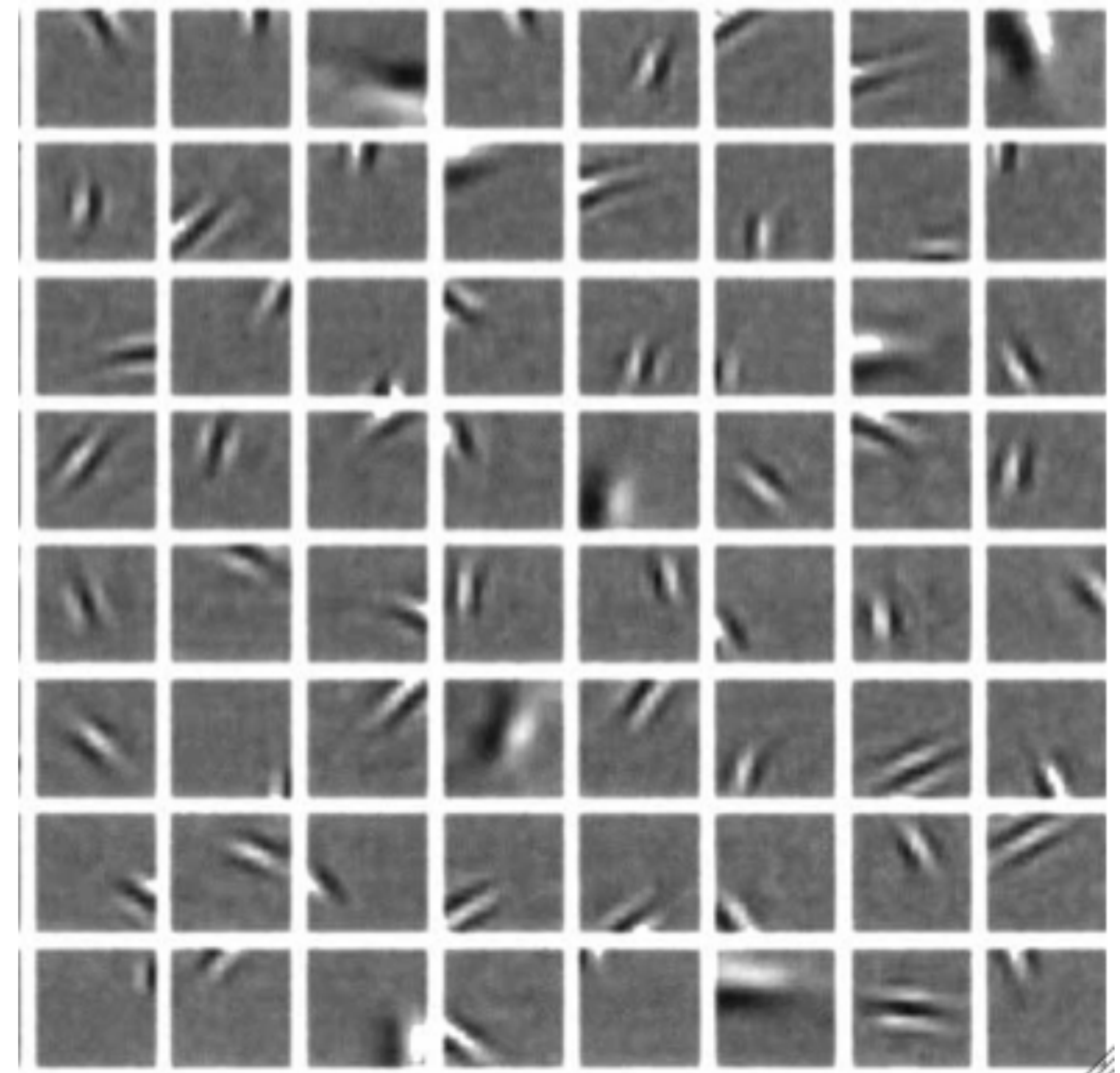


DeAngelis et al. (1995)

Learning basis functions with different cost metrics

the same set has to work for any input image ***I***





$$\min_{\boldsymbol{r}} \quad ||\boldsymbol{I} - U\boldsymbol{r}||_2$$

Learn receptive fields
by minimizing the fit only.

$$\min_{\boldsymbol{r}} \quad ||\boldsymbol{I} - U\boldsymbol{r}||_2 + \lambda||\boldsymbol{r}||_1$$

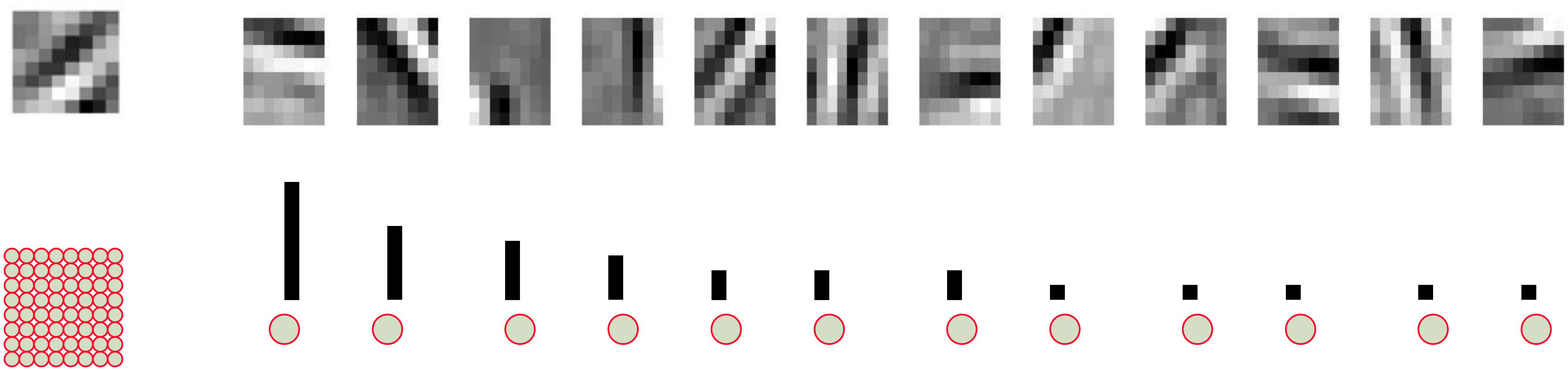
Learn receptive fields
by adding a term that penalizes large synapses
$||_1$ means absolute value

Approximating an image patch w basis functions

The outputs
of 64cells
in the LGN …

… can be coded with only twelve V1 cells …

… where each cell has 64 synapses



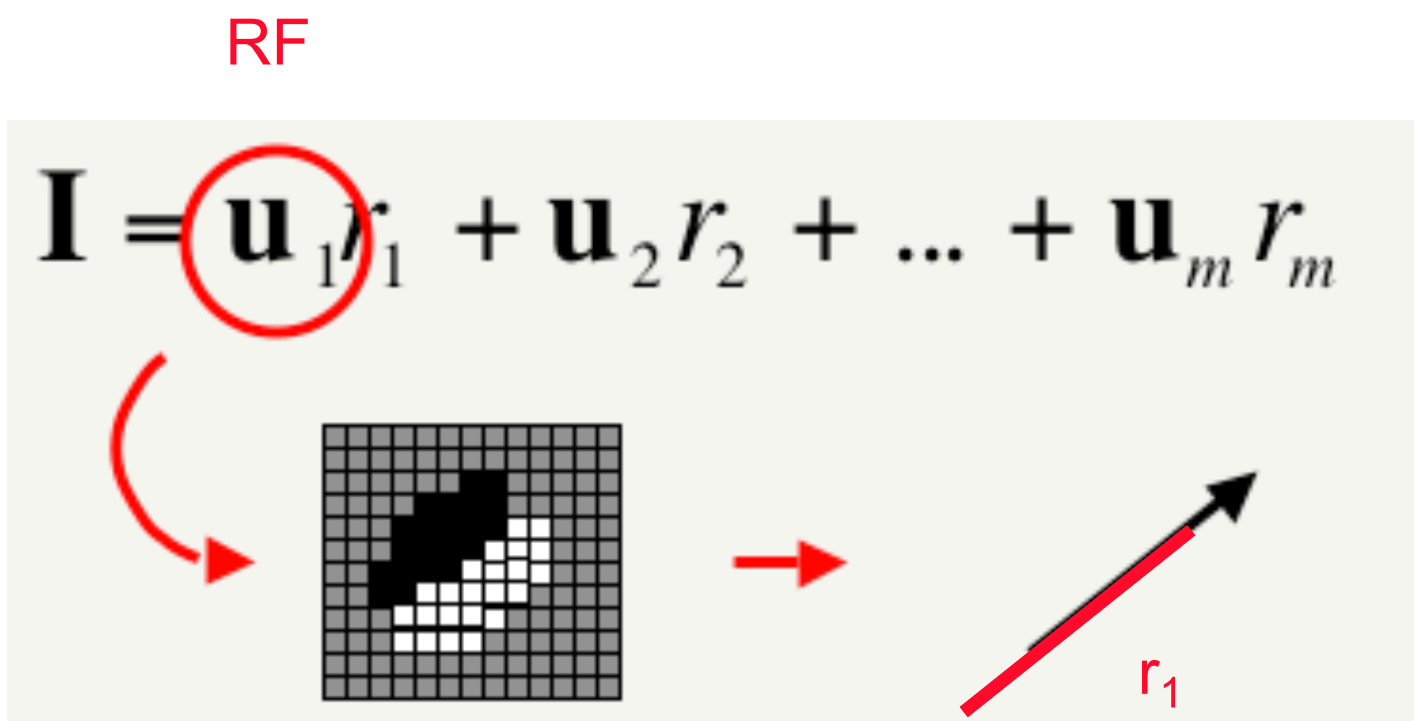
LGN
Thalamic
nucleus

V1
striate cortex

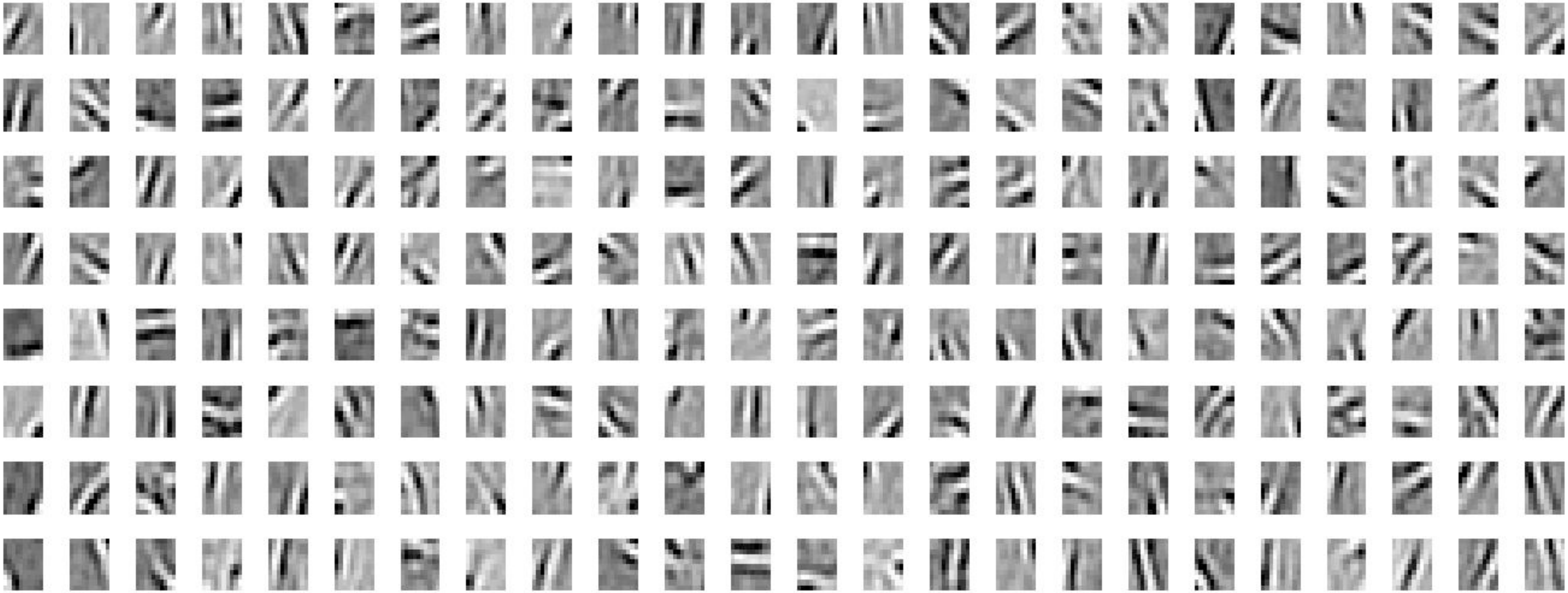
Approximating an image patch w basis functions

LGN

V1

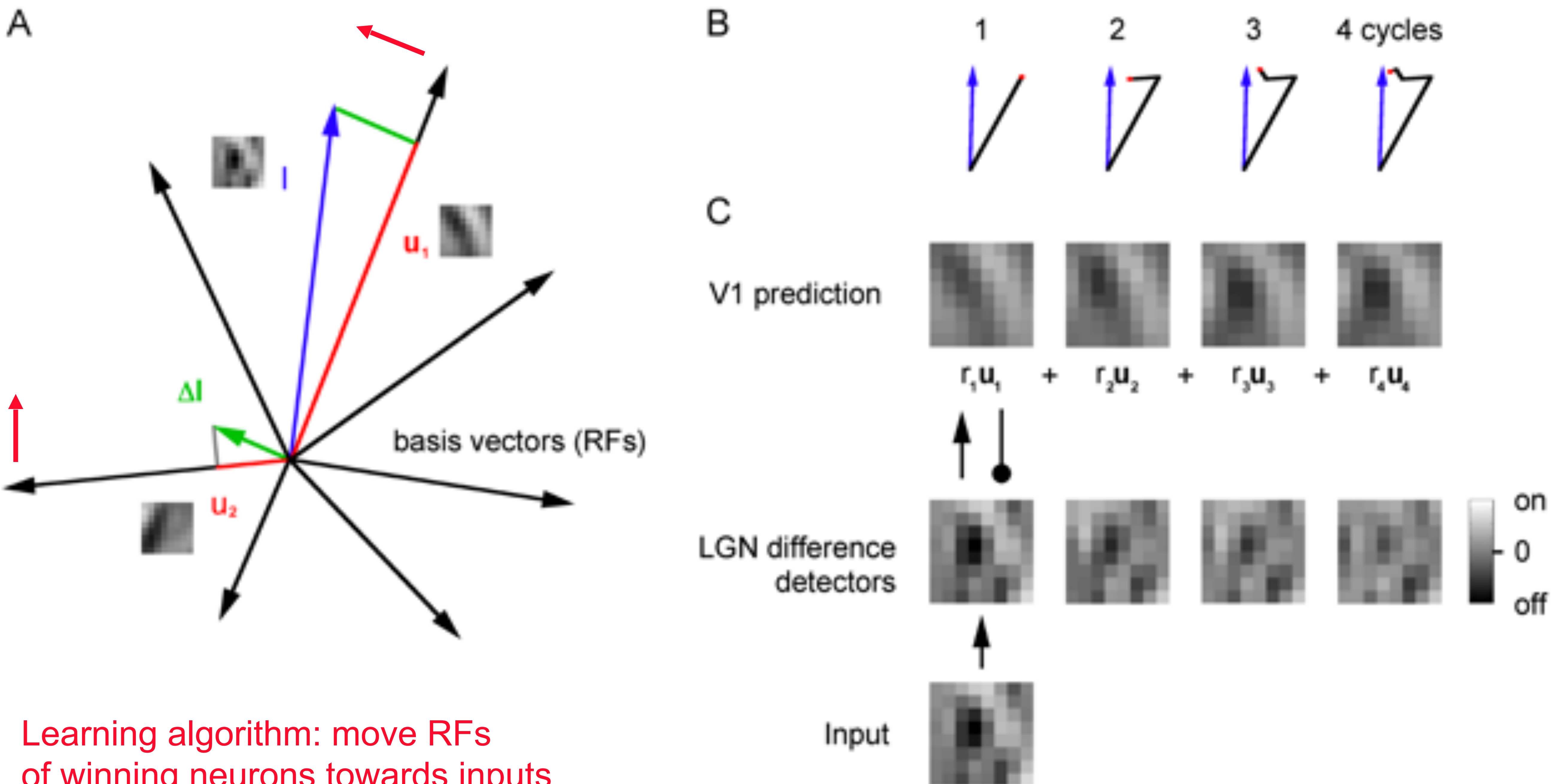


The neural coding library of learned RFs



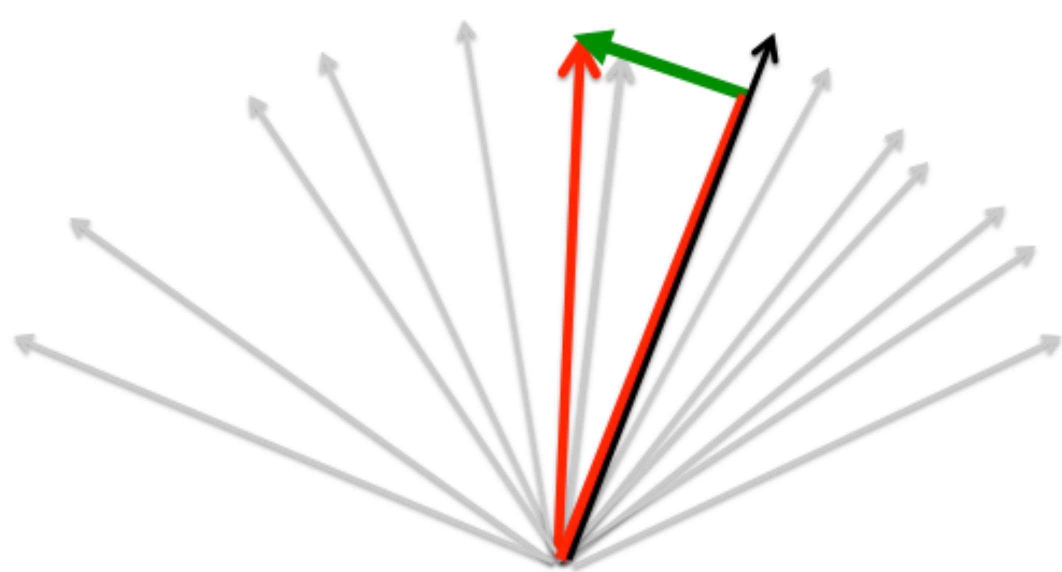
Because there are more than we need - *Overcomplete* (192 vs 64) - the number of cells that need to send spikes at any moment is *Sparse* (12 vs 64).

The sequential algorithm: Matching Pursuit



Learning algorithm: move RFs
of winning neurons towards inputs

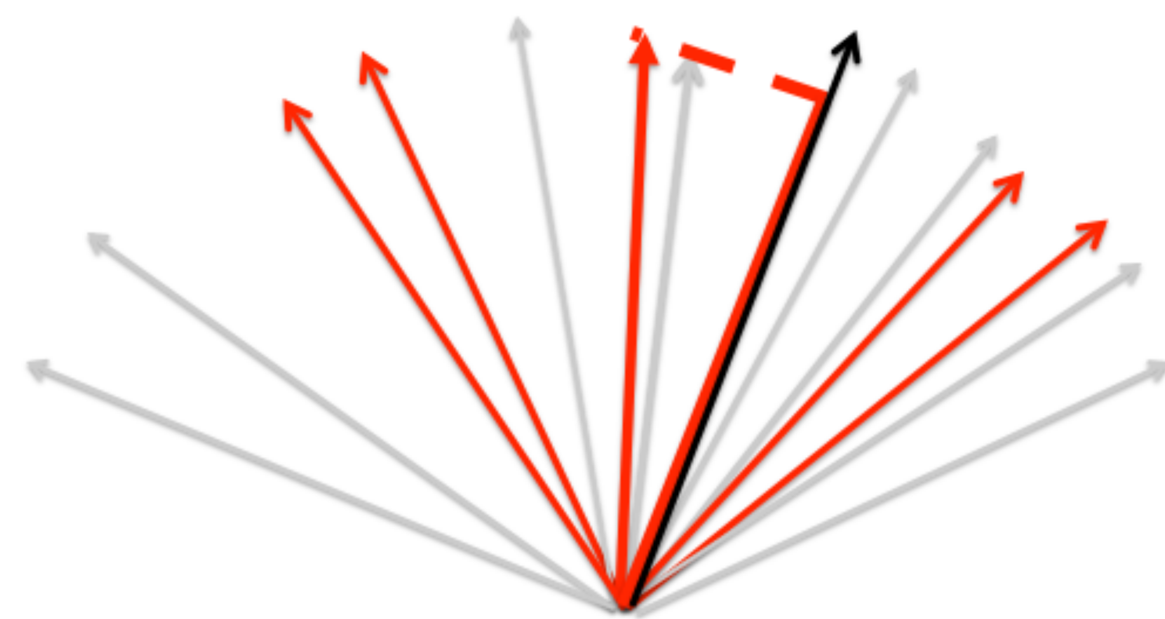
Two methods of coding input



Serial

Choose a basis function
(projection is its probability of being chosen)
Calculate the residual
Recurse

Too expensive: a residual takes a gamma cycle



Parallel

Choose 50 basis functions
Sum them
Normalize the result

Trial averaging method applied to motion cells