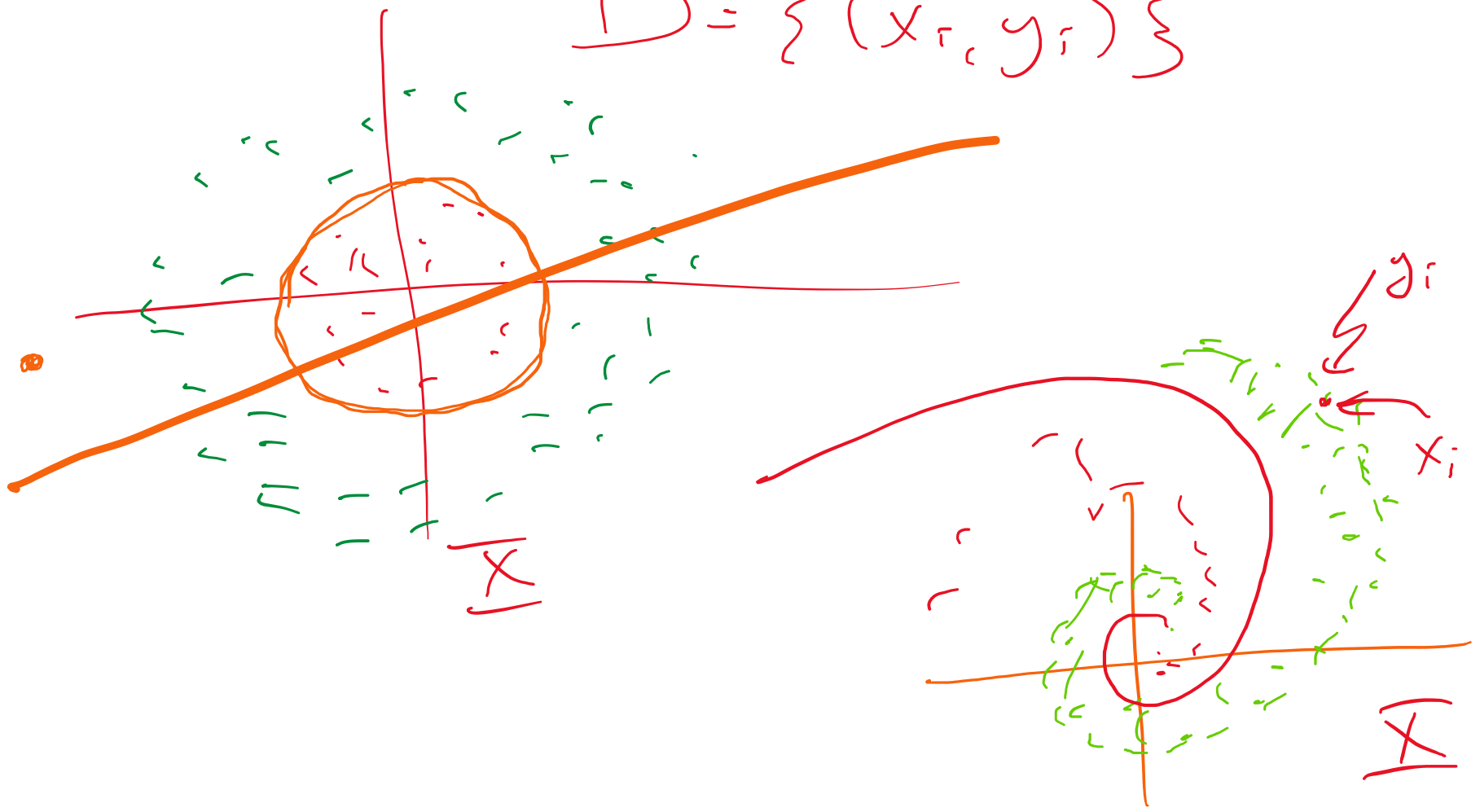
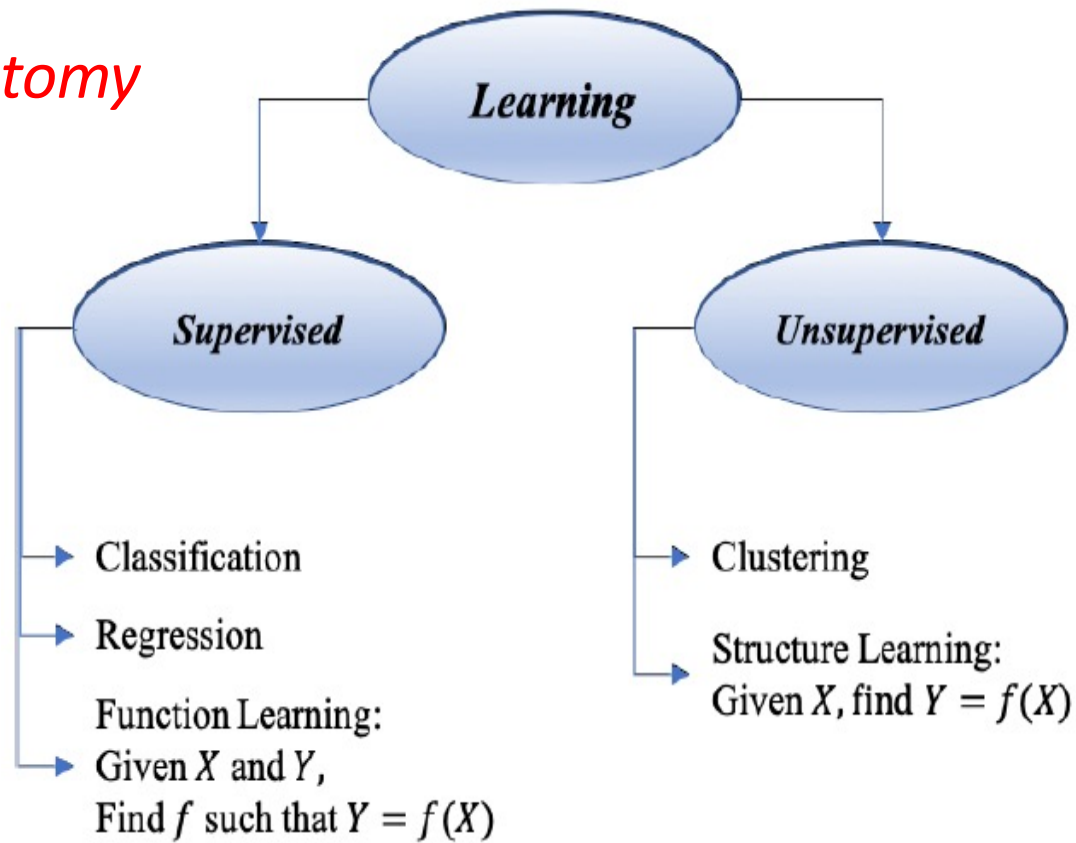


$$D = \{ (x_i, y_i) \}$$



# Learning Dichotomy



Labelled data  
Learn a function

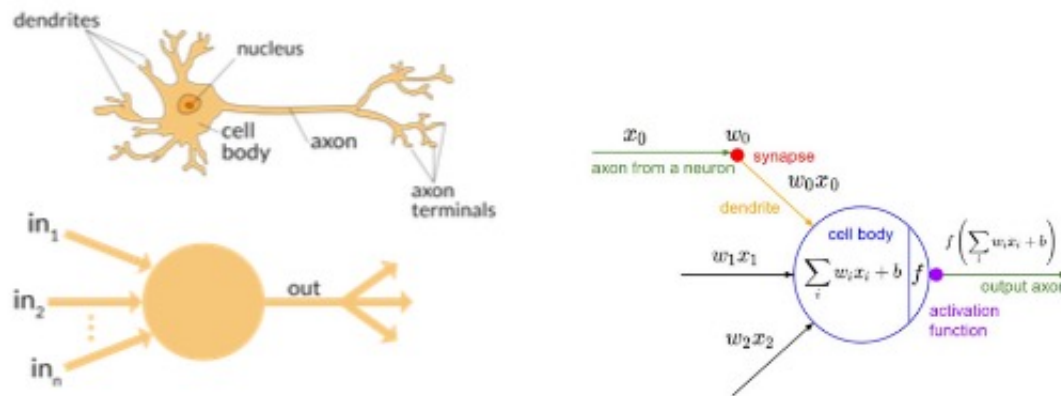
Pattern  
Structure

Well...at least one more  
- re-enforcement learning

And now for something completely different

## Chapter 6 – Artificial Neural Networks

### Neural Networks (NN)



Suddenly ANN is best – why here why now? - I thought it was a crazy idea when I first heard of it.

Advancements of technology

- Stochastic Gradient Descent.
- The computational GPU (Graphic Processing Unites), and the huge evolution of computations speed.



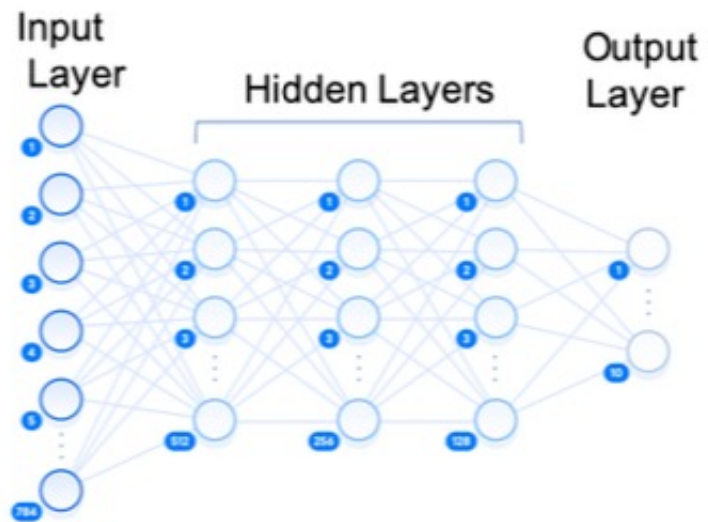
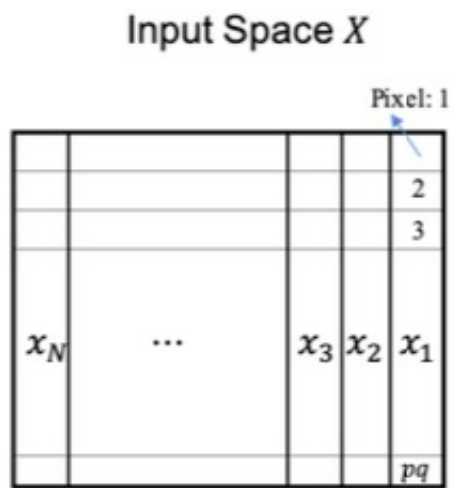
The problem – an example data set USPS Figs –

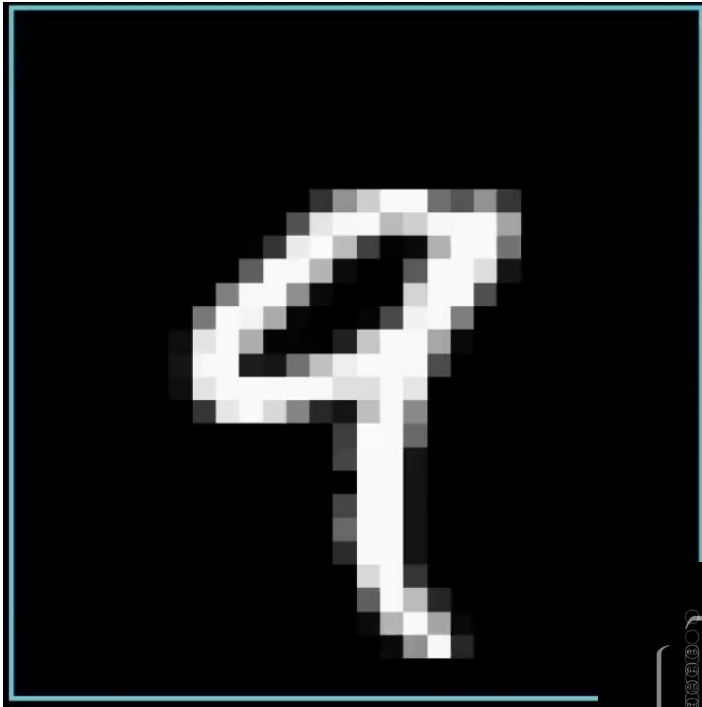
A supervised learning problem



The basis picture of data input into a ANN

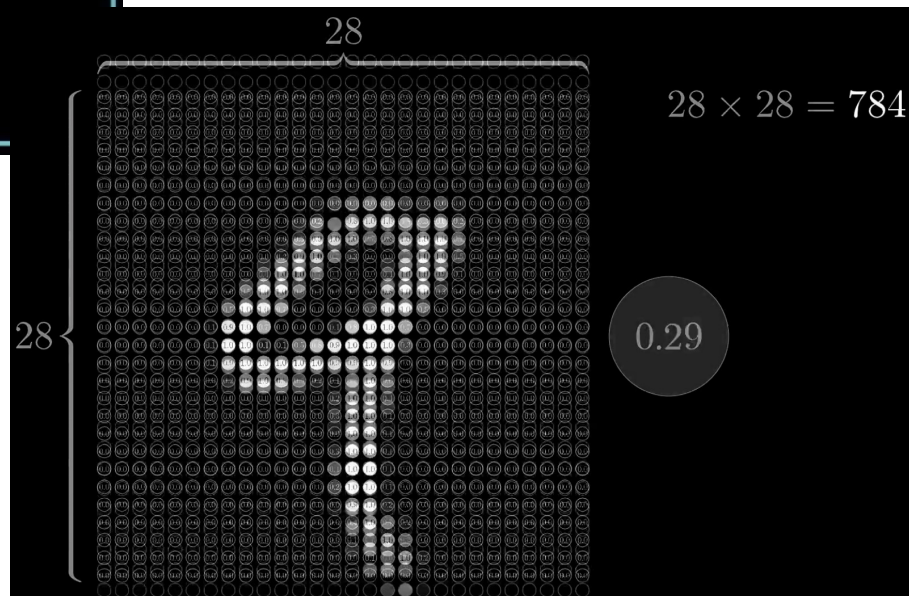
- input layer
- hidden inner layers
- output layer





This datum is a 784 numbers between 0 and 1.

Data  $\{(x_i, y_i)\}_{i=1..784}$

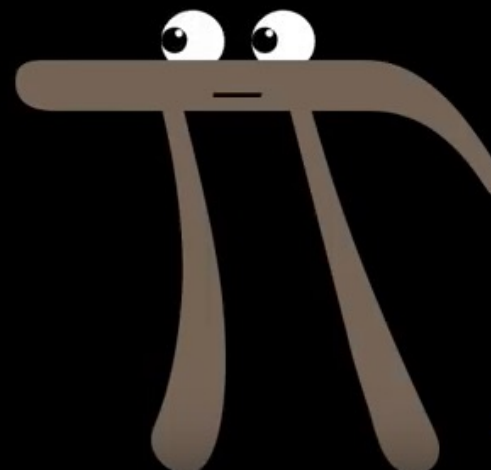
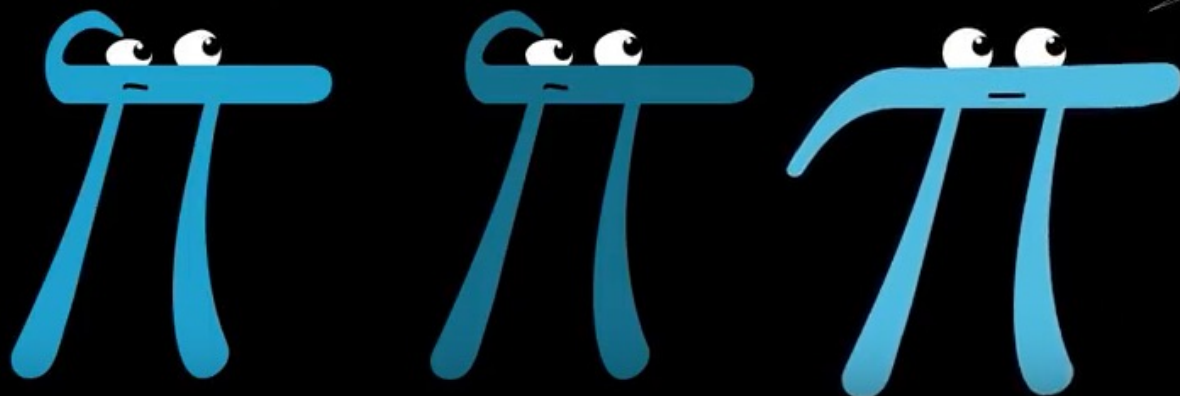
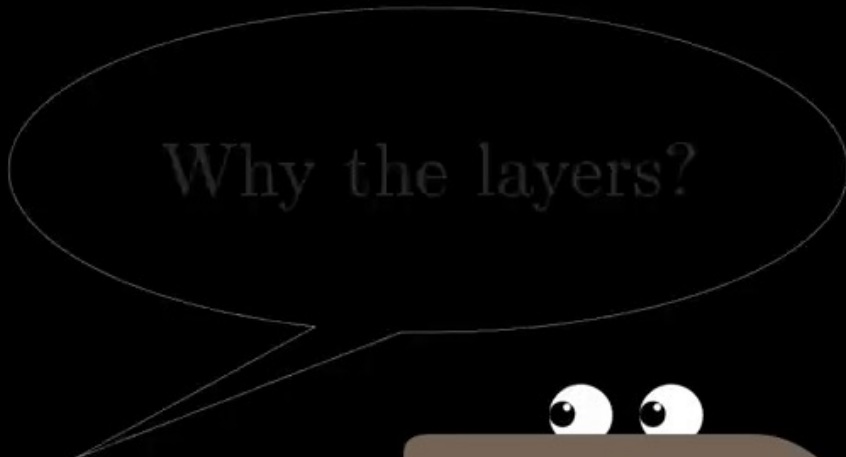
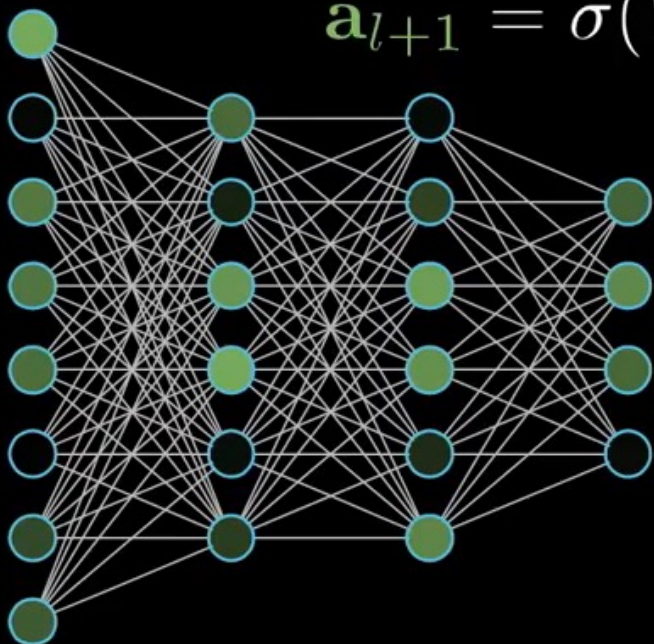


The data is a collection on ordered pairs, vector valued figures together with a symbol

(0, 0)	(6, 6)	(3, 3)	(6, 6)	(7, 7)	(8, 8)	(0, 0)	(9, 9)
(5, 5)	(4, 4)	(3, 3)	(6, 6)	(5, 5)	(8, 8)	(9, 9)	(5, 5)
(4, 4)	(4, 4)	(7, 7)	(2, 2)	(0, 0)	(3, 3)	(2, 2)	(8, 8)
(9, 9)	(1, 1)	(9, 9)	(2, 2)	(2, 2)	(7, 7)	(9, 9)	(4, 4)
(8, 8)	(7, 7)	(4, 4)	(1, 1)	(3, 3)	(1, 1)	(5, 5)	(3, 3)
(2, 2)	(3, 3)	(9, 9)	(0, 0)	(9, 9)	(9, 9)	(1, 1)	(5, 5)
(8, 8)	(4, 4)	(7, 7)	(7, 7)	(4, 4)	(4, 4)	(4, 4)	(2, 2)
(0, 0)	(7, 7)	(2, 2)	(4, 4)	(8, 8)	(2, 2)	(6, 6)	(9, 9)
(9, 9)	(2, 2)	(8, 8)	(7, 7)	(6, 6)	(1, 1)	(1, 1)	(2, 2)
(3, 3)	(9, 9)	(1, 1)	(6, 6)	(5, 5)	(1, 1)	(1, 1)	(0, 0)

# Machine learning Neural network

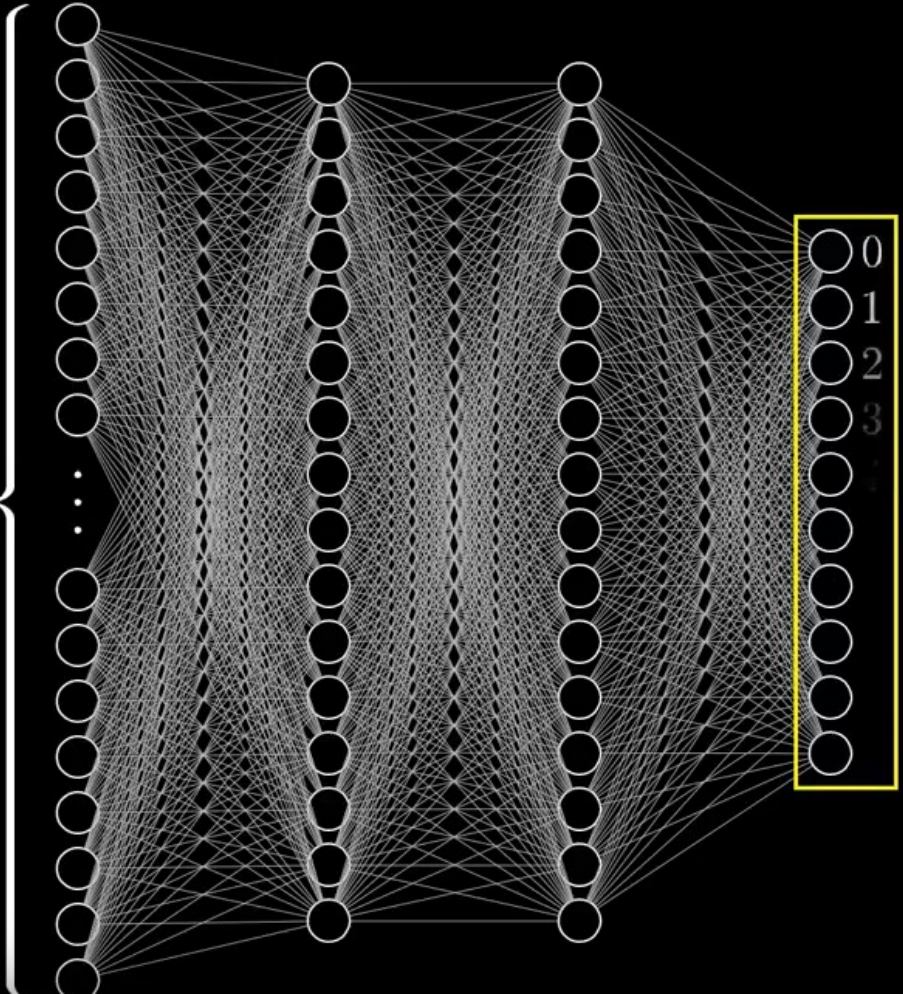
$$\mathbf{a}_{l+1} = \sigma(W_l \mathbf{a}_l + b_l)$$

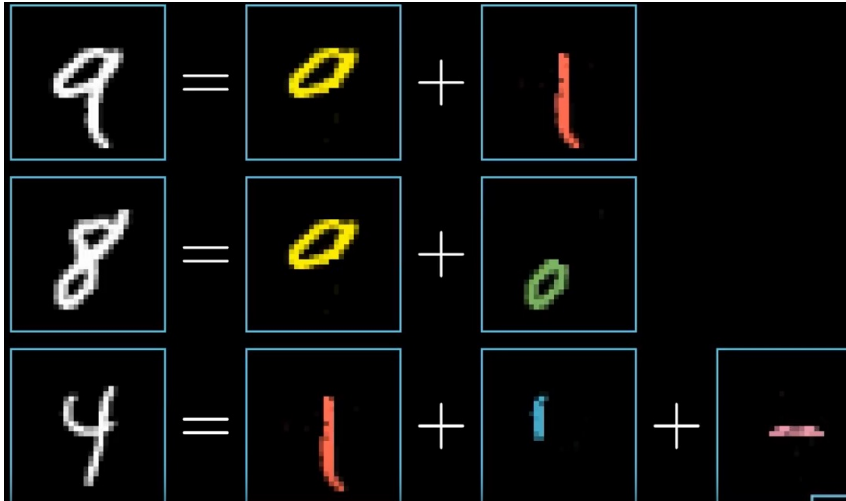




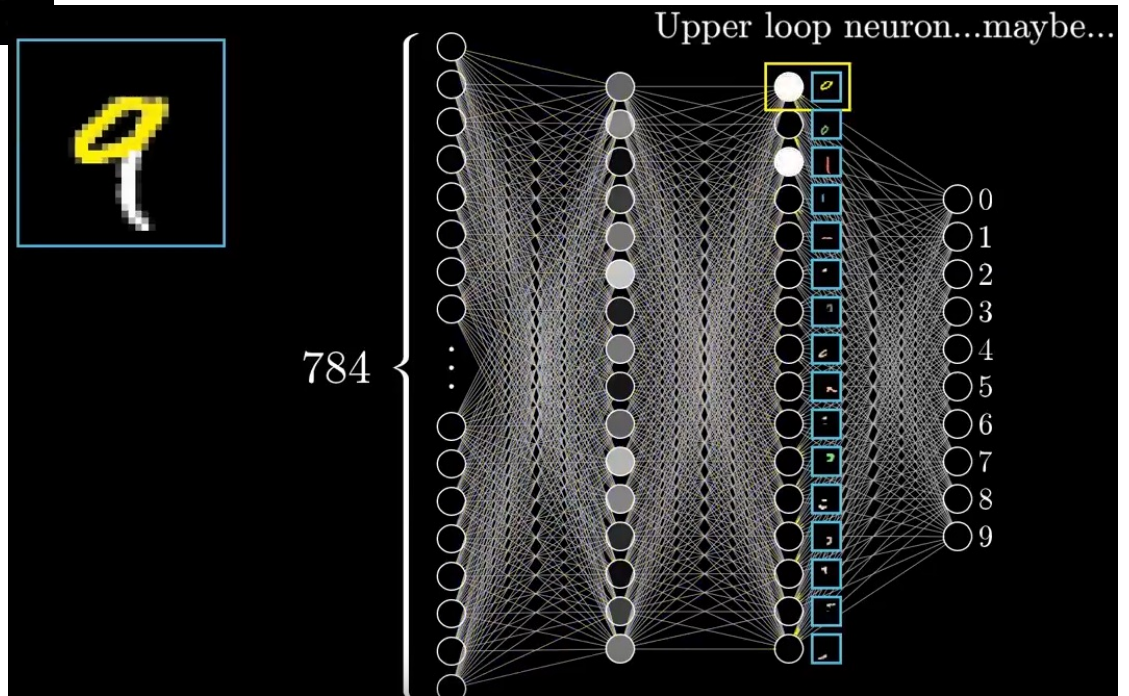


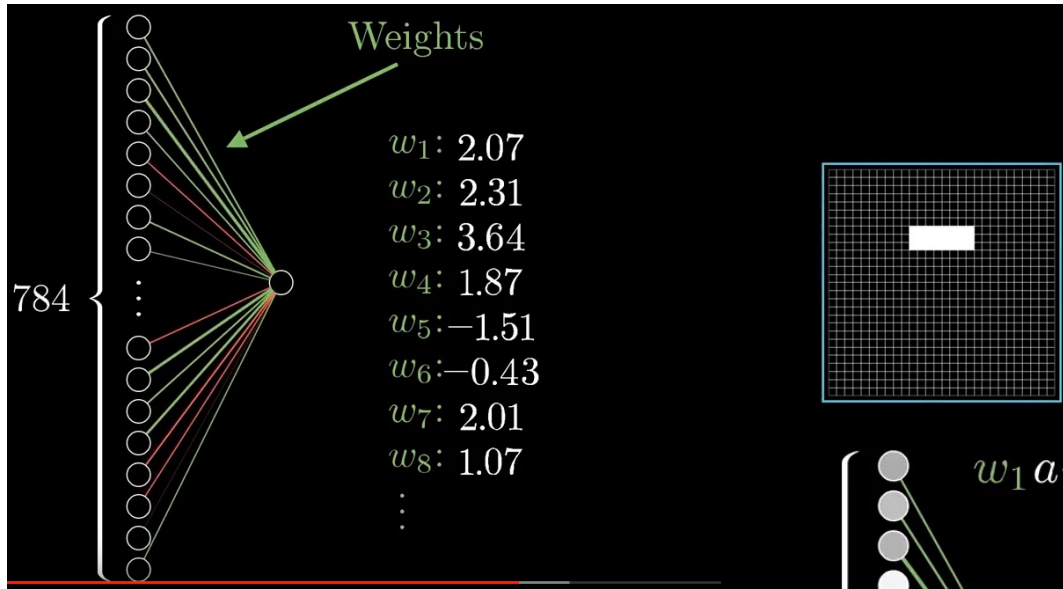
784





What do you wish in the “neurons”?

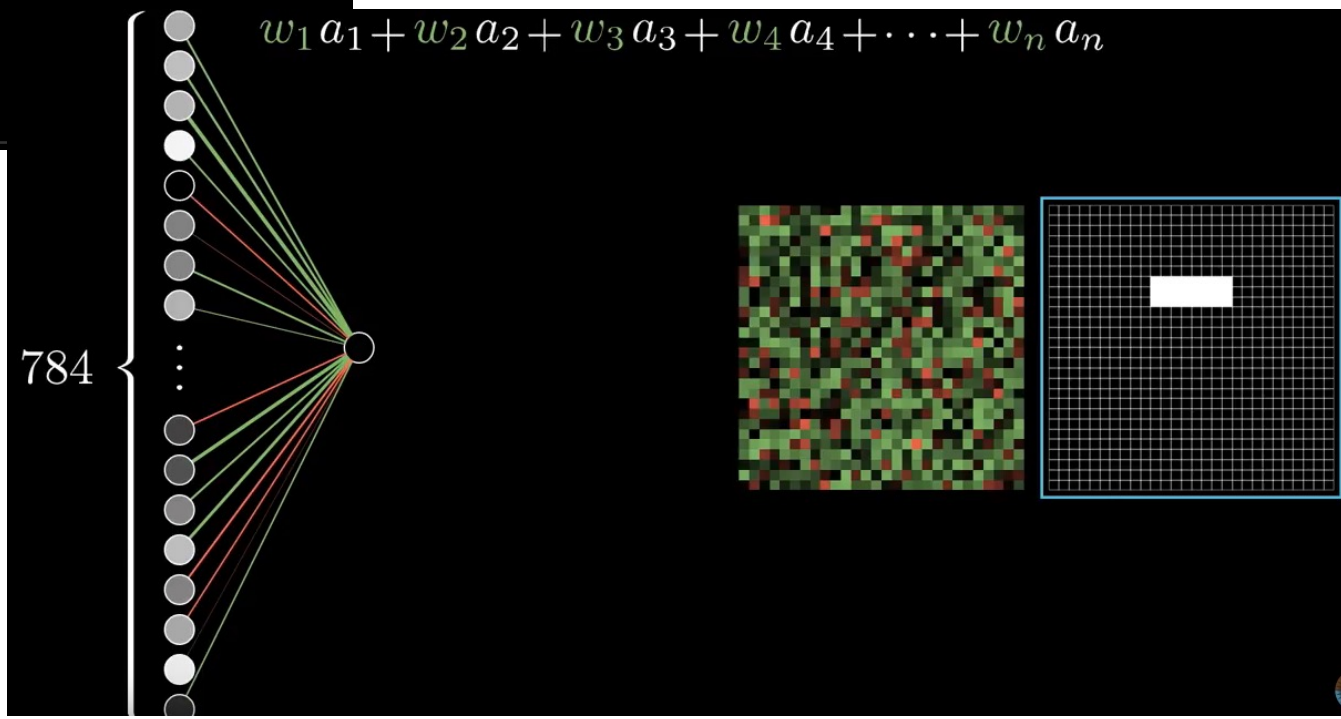




What do you wish in the “neurons”?

Vs

What do we get in the neurons.





### Sigmoid

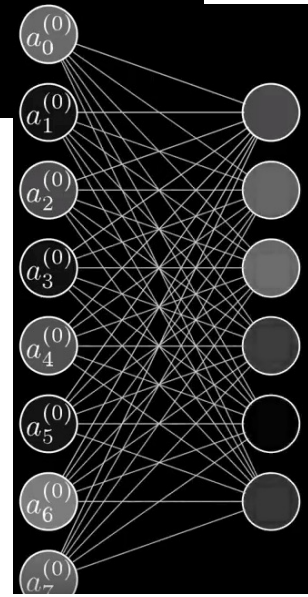
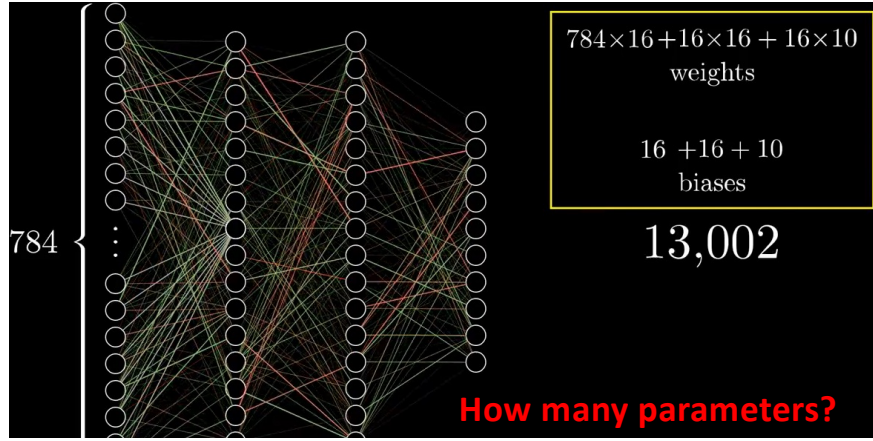
$$a_0^{(1)} = \sigma \left( w_{0,0} a_0^{(0)} + w_{0,1} a_1^{(0)} + \dots + w_{0,n} a_n^{(0)} + b_0 \right)$$

↓
↑  
Bias

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}$$

## The ANN – Mathematical Process

The matrix form of a layer, and composition



$$\sigma \left( \mathbf{W} \mathbf{a}^{(0)} + \mathbf{b} \right)$$

$$\sigma \left( \begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

The basic nodal statement.

$$y = \sigma \left( \sum_{i=1}^n w_i x_i + b \right)$$

Favorite activation functions

- Linear function:  $\sigma(x) = ax + b$ , with  $a$  and  $b$  are constants.
- Sigmoid (Logistic) function:  $\sigma(x) = \frac{1}{1+e^{-x}}$ .
- Hyperbolic Tangent:  $\sigma(x) = \tanh(x)$ .
- Rectified Linear Unit (ReLU):  $\sigma(x) = \max\{0, x\}$ .

What's the cost?

Vs what?

Vs what we want?

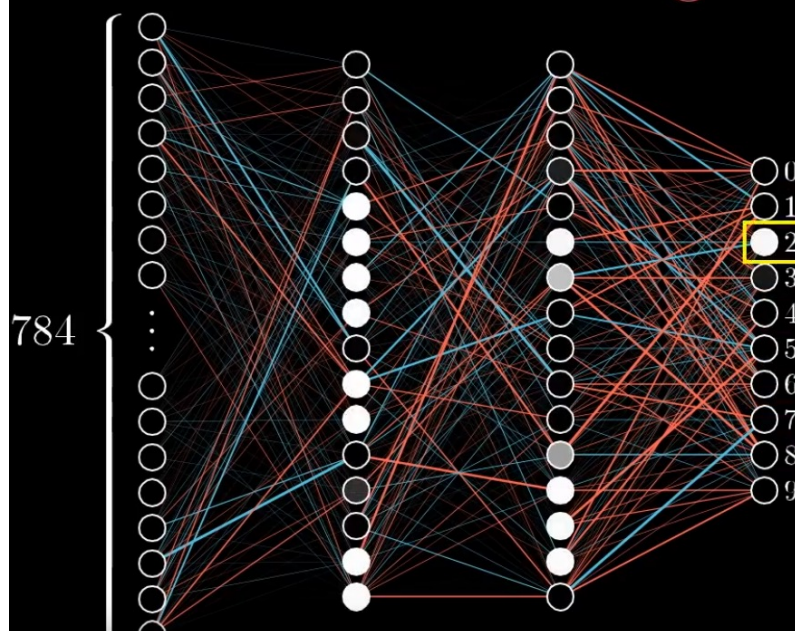
Testing data



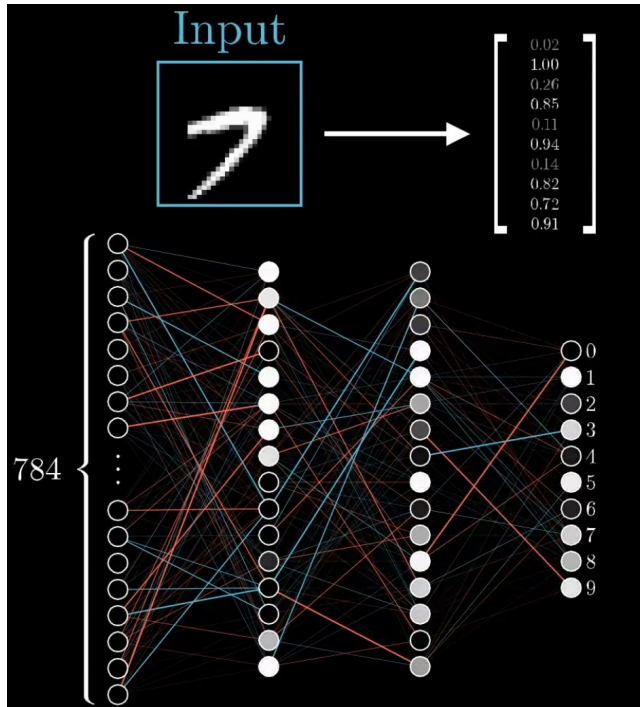
Guess → 2

Wrong!

$$\frac{\text{Number correct}}{\text{total}} = \frac{62}{64} = 0.969$$



Wrong!



## Neural network function

Input: 784 numbers (pixels)

Output: 10 numbers

Parameters:

Average cost of all training data...

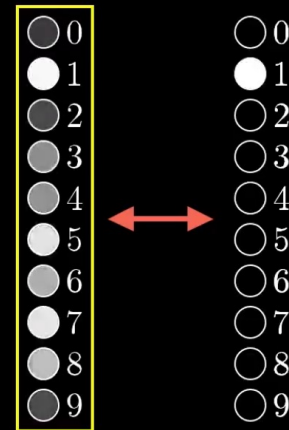
Let's write out the cost function

Cost of



$$\left\{ \begin{array}{l} (0.23 - 0.00)^2 + \\ (0.98 - 1.00)^2 + \\ (0.31 - 0.00)^2 + \\ (0.56 - 0.00)^2 + \\ (0.58 - 0.00)^2 + \\ (0.90 - 0.00)^2 + \\ (0.69 - 0.00)^2 + \\ (0.91 - 0.00)^2 + \\ (0.76 - 0.00)^2 + \\ (0.31 - 0.00)^2 \end{array} \right.$$

What's the "cost" of this difference?



Utter trash

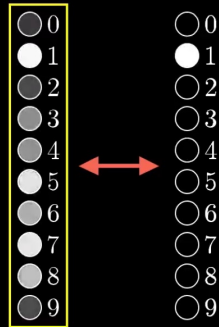
Average cost of all training data...

Cost of



$$\left\{ \begin{array}{l} (0.23 - 0.00)^2 + \\ (0.98 - 1.00)^2 + \\ (0.31 - 0.00)^2 + \\ (0.56 - 0.00)^2 + \\ (0.58 - 0.00)^2 + \\ (0.90 - 0.00)^2 + \\ (0.69 - 0.00)^2 + \\ (0.91 - 0.00)^2 + \\ (0.76 - 0.00)^2 + \\ (0.31 - 0.00)^2 \end{array} \right.$$

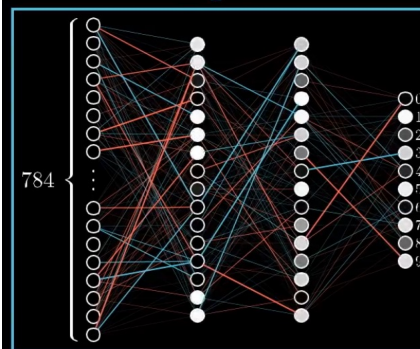
What's the "cost" of this difference?



Utter trash

Let's write out the cost function

Input



Cost: 5.4

Cost function

Input: 13,002 weights/biases

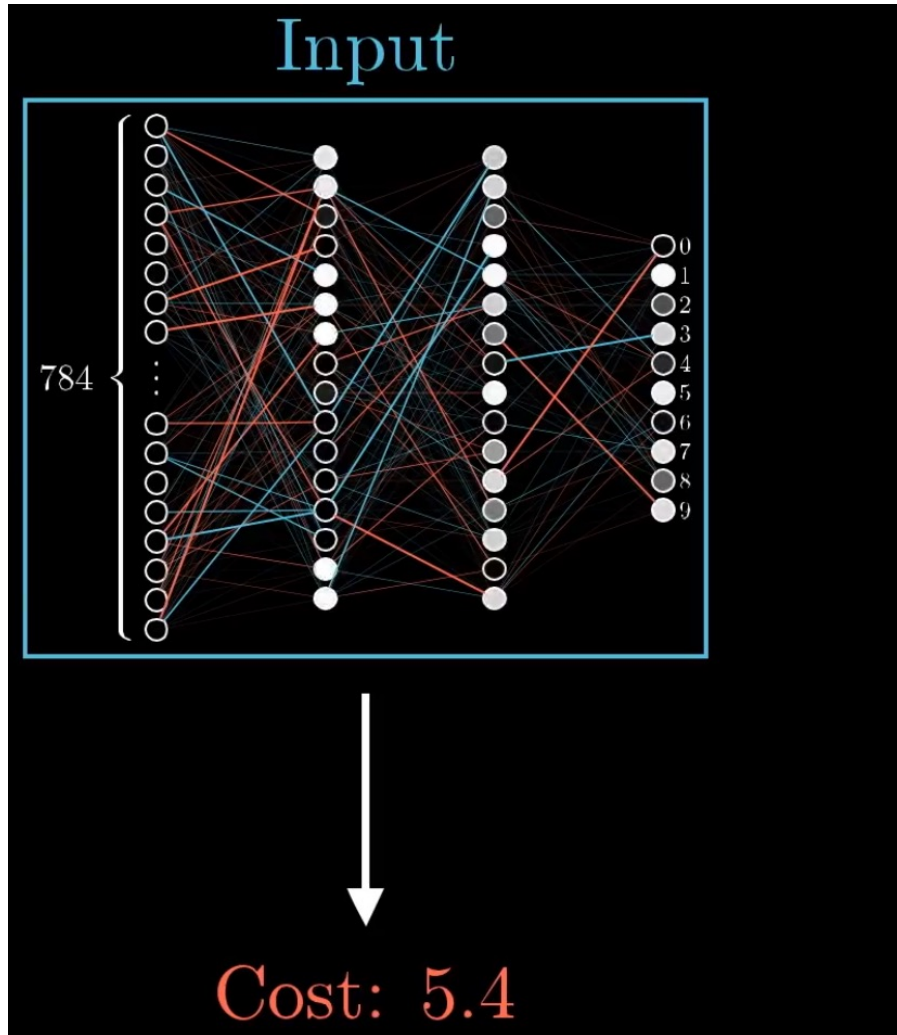
Output: 1 number (the cost)

Parameters: Many, many, many training examples

(2, 2)



Let's write out the cost function




## Cost function

Input: 13,002 weights/biases

Output: 1 number (the cost)

Parameters: Many, many, many training examples

(  , 2 )

# Cost function

$$C(w_1, w_2, \dots, w_{13,002})$$

## Weights and biases

Average cost of all training data...

Cost of **7**

$$\left\{ \begin{array}{l} (0.10 - 0.00)^2 + \\ (0.54 - 0.00)^2 + \\ (0.94 - 0.00)^2 + \\ (0.12 - 0.00)^2 + \\ (0.40 - 0.00)^2 + \\ (0.69 - 0.00)^2 + \\ (0.72 - 0.00)^2 + \\ (0.87 - 1.00)^2 + \\ (0.24 - 0.00)^2 + \\ (0.25 - 0.00)^2 \end{array} \right.$$

What's the "cost" of this difference?

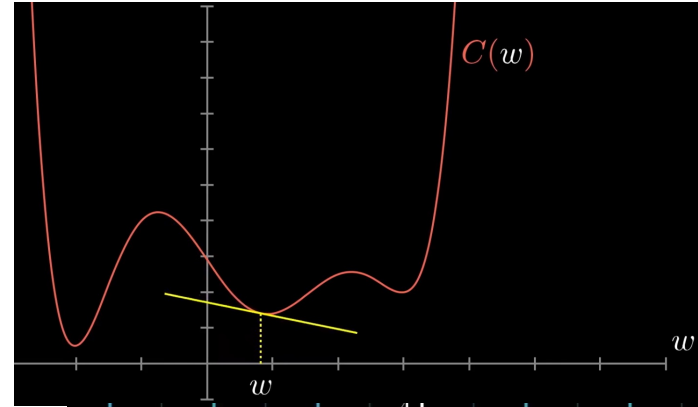
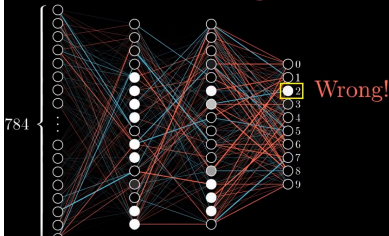
- |                       |   |                                  |   |
|-----------------------|---|----------------------------------|---|
| <input type="radio"/> | 0 | <input type="radio"/>            | 0 |
| <input type="radio"/> | 1 | <input type="radio"/>            | 1 |
| <input type="radio"/> | 2 | <input type="radio"/>            | 2 |
| <input type="radio"/> | 3 | <input type="radio"/>            | 3 |
| <input type="radio"/> | 4 | <input type="radio"/>            | 4 |
| <input type="radio"/> | 5 | <input type="radio"/>            | 5 |
| <input type="radio"/> | 6 | <input type="radio"/>            | 6 |
| <input type="radio"/> | 7 | <input checked="" type="radio"/> | 7 |
| <input type="radio"/> | 8 | <input type="radio"/>            | 8 |
| <input type="radio"/> | 9 | <input type="radio"/>            | 9 |

Utter trash

Testing data

**3** Guess **2**  
Wrong!

$$\frac{\text{Number correct}}{\text{total}} = \frac{62}{64} = 0.969$$

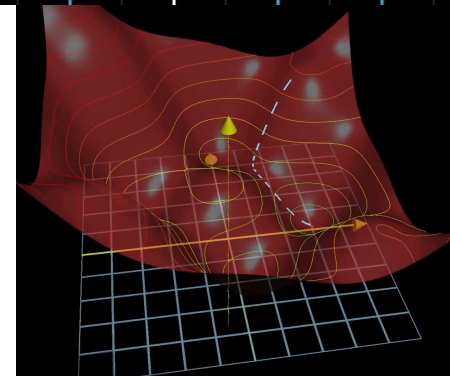


Input space  $y$

"Gradient", the direction of steepest increase

$$\nabla C(x, y)$$

Which direction decreases  $C(x, y)$  most quickly?



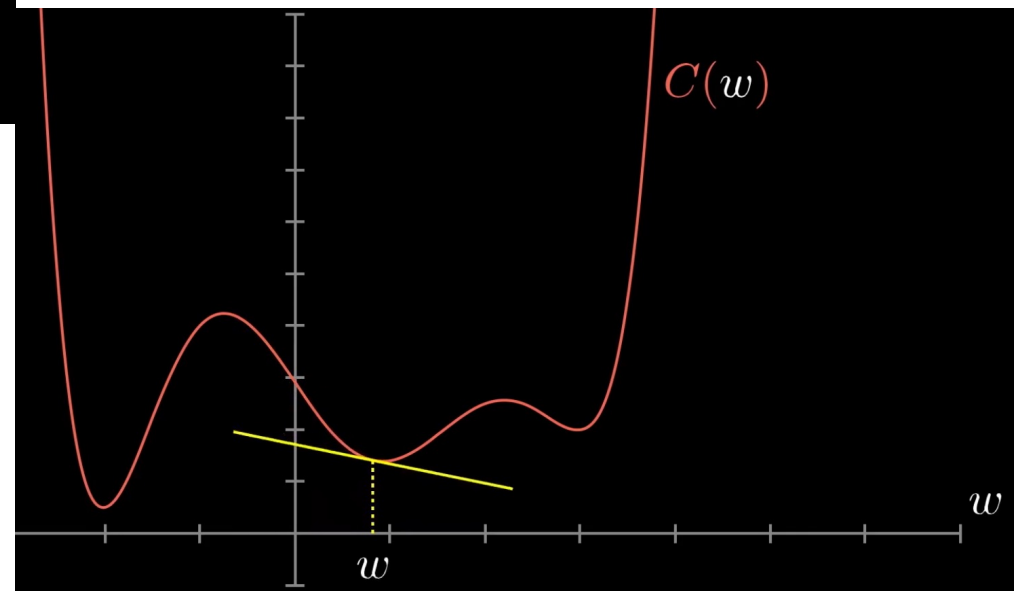
Optimization methods

# Cost function

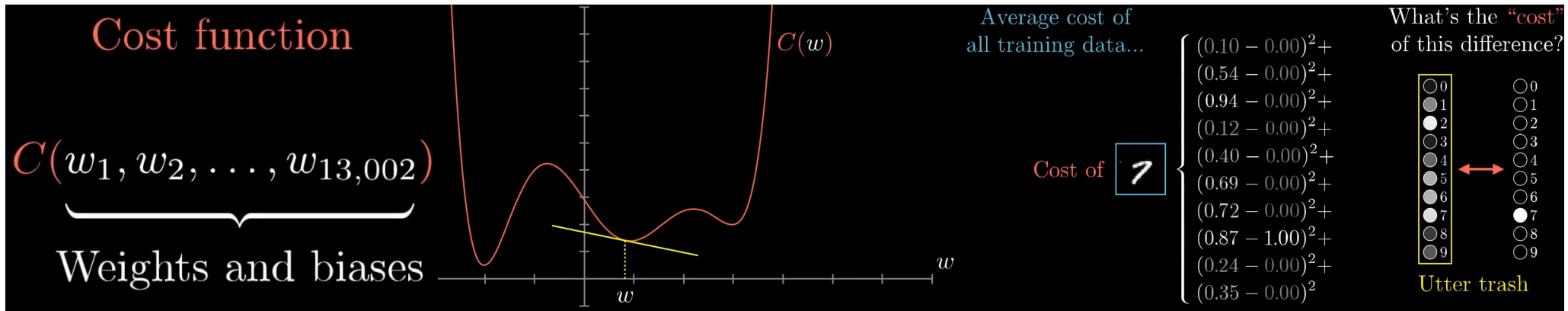
$$C(w_1, w_2, \dots, w_{13,002})$$

Weights and biases

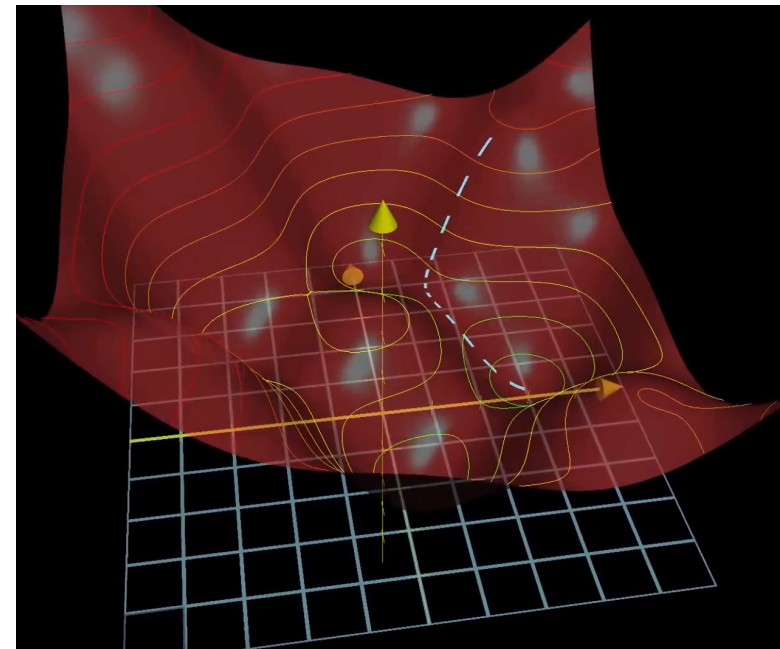
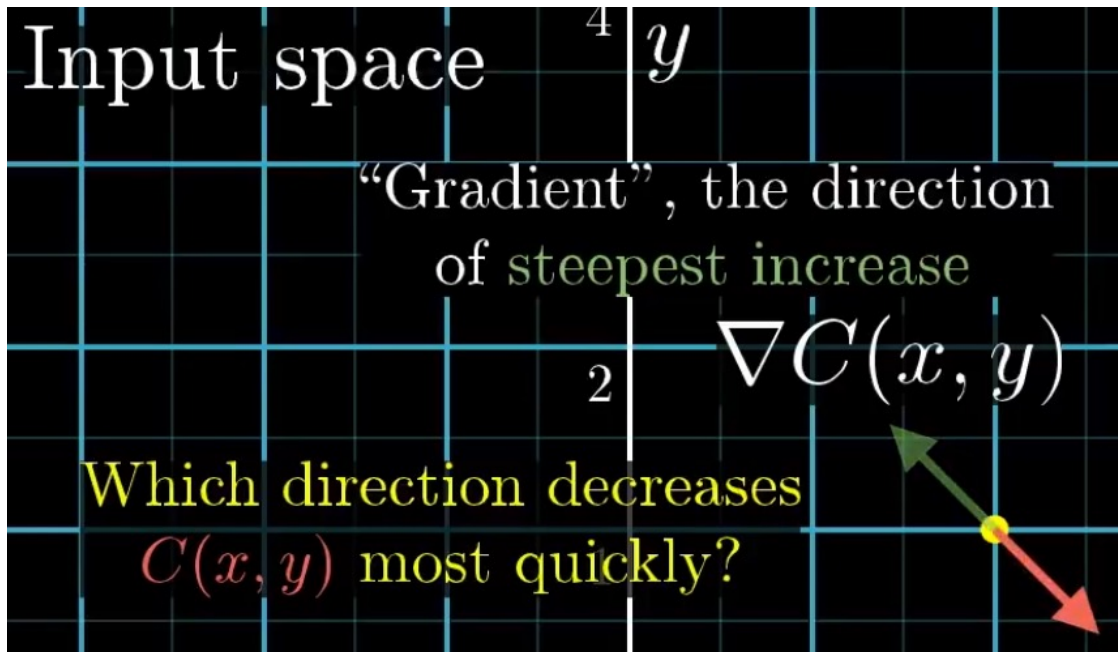
An optimization problem



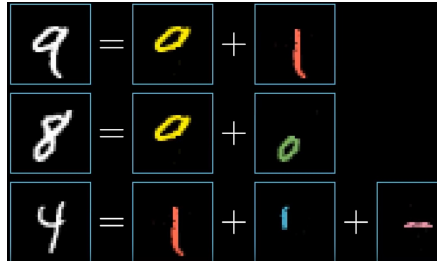




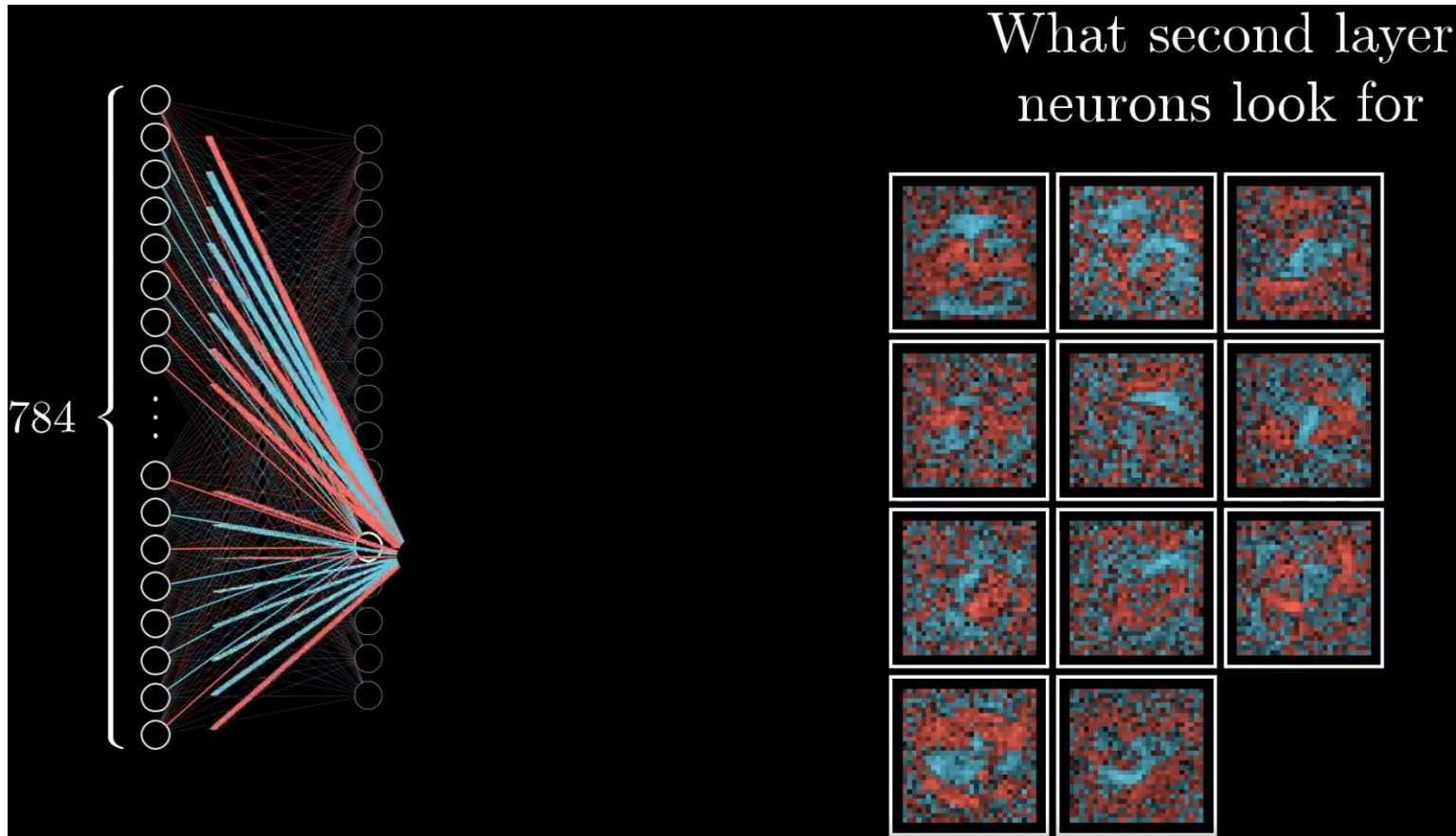
### Optimization methods



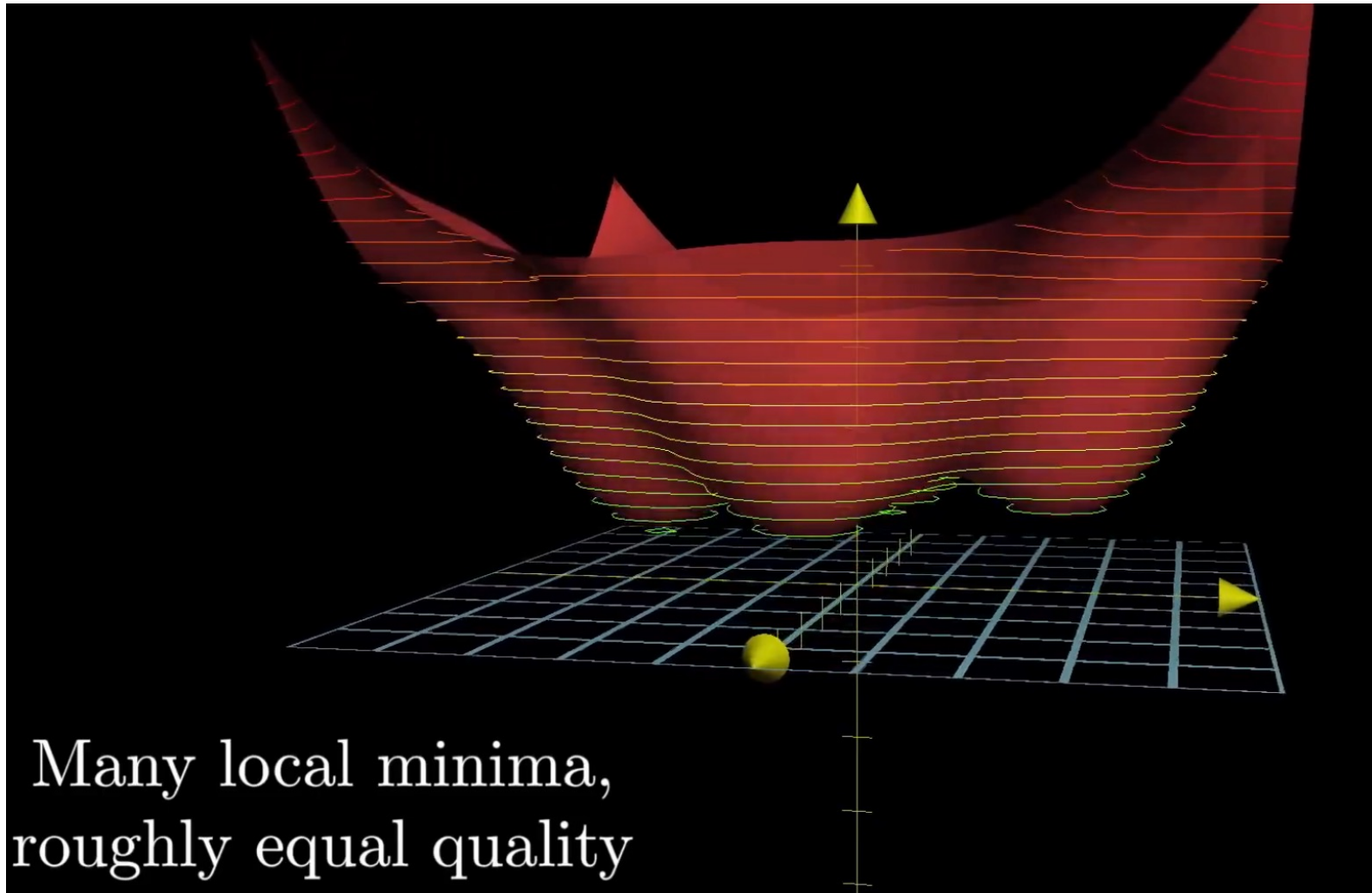
What do you wish in the “neurons”?



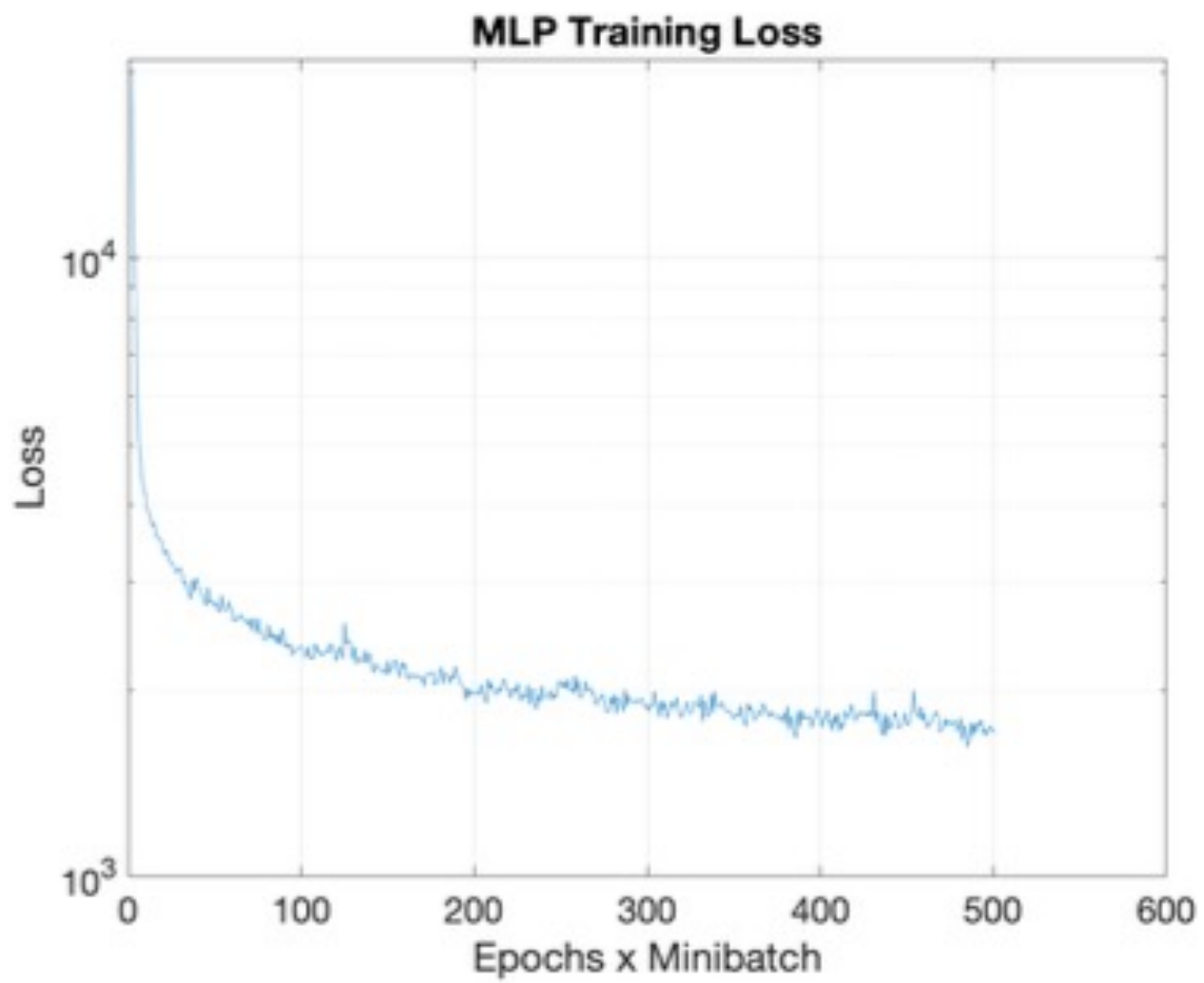
What do we actually get?



MANY local minima of the VERY high dimensional so how important is it to find the best One?

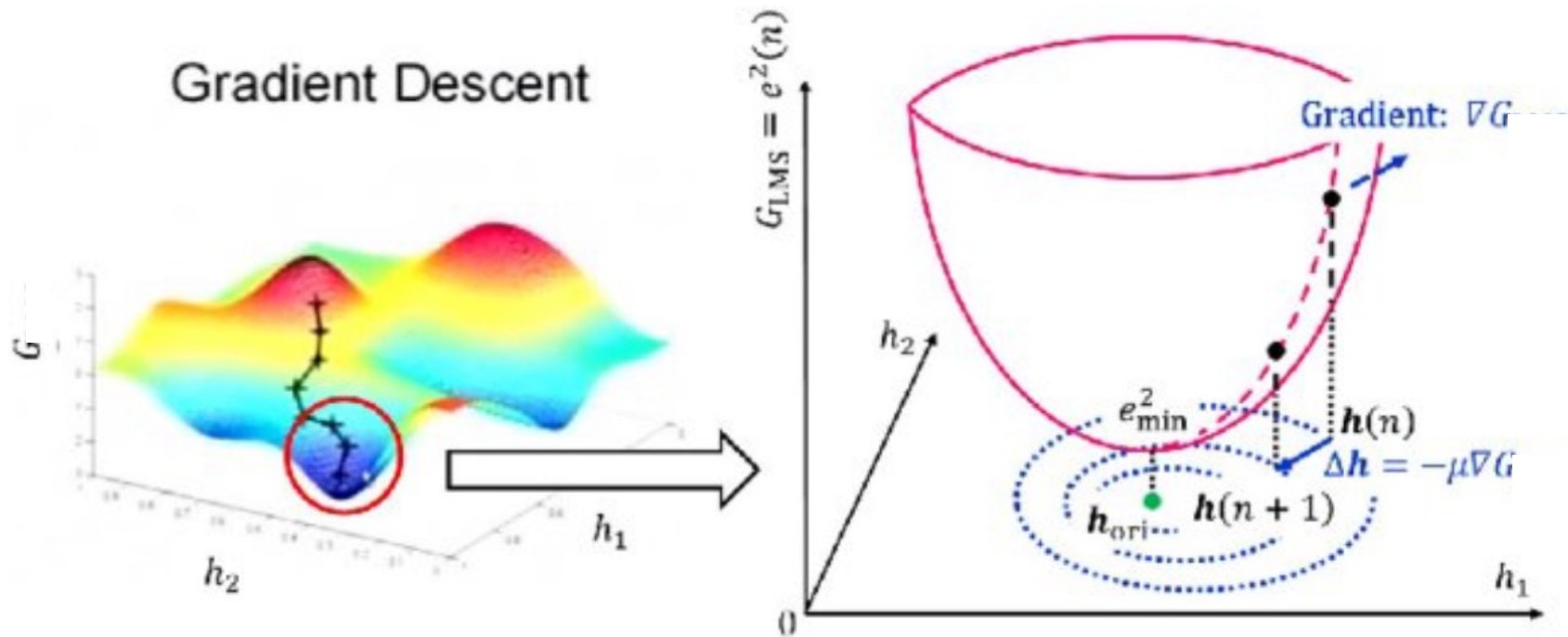


On training

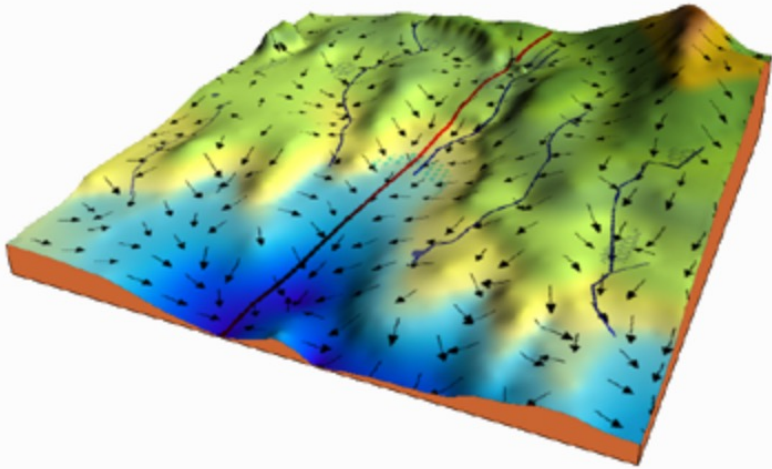


General smooth optimization,  $G(h): \mathbb{R}^d \rightarrow \mathbb{R}$

By general gradient descent



## More on gradient descent



Given the cost function:

$$f(m, b) = \frac{1}{N} \sum_{i=1}^n (y_i - (mx_i + b))^2$$

The gradient can be calculated as:

$$f'(m, b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N} \sum -2(y_i - (mx_i + b)) \end{bmatrix}$$





$$\mathbf{w}_{j+1}(\delta) = \mathbf{w}_j - \delta \nabla f_k(\mathbf{w}_j)$$

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \nabla_{\theta} L(\theta^{(t)}, \mathbf{y})$$

In this equation:

- $\theta^{(t)}$  is our current estimate of  $\theta^*$  at the  $t$ th iteration
- $\alpha$  is the learning rate
- $\nabla_{\theta} L(\theta^{(t)}, \mathbf{y})$  is the gradient of the loss function
- We compute the next estimate  $\theta^{(t+1)}$  by subtracting the product of  $\alpha$  and  $\nabla_{\theta} L(\theta, \mathbf{y})$  computed at  $\theta^{(t)}$

## Stochastic Gradient Descent

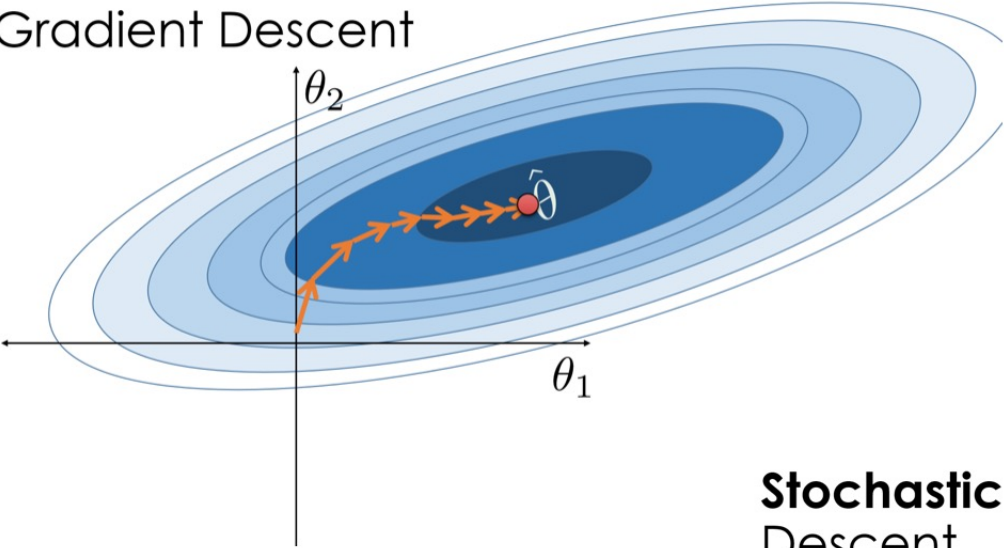
$$\mathbf{w}_{j+1}(\delta) = \mathbf{w}_j - \delta \nabla f_K(\mathbf{w}_j)$$

Batch (several data points)

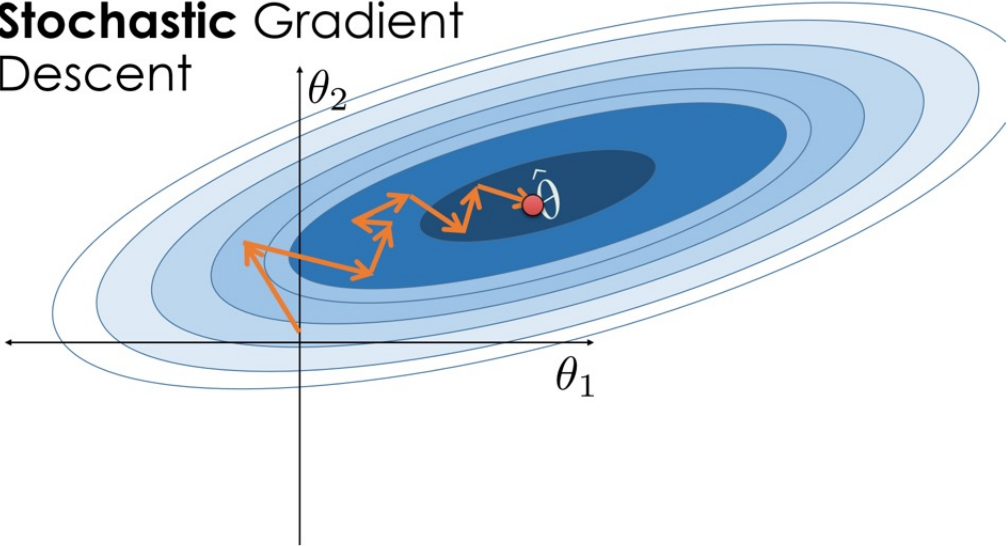
$$K \in [k_1, k_2, \dots, k_p]$$



Gradient Descent



**Stochastic Gradient**  
Descent

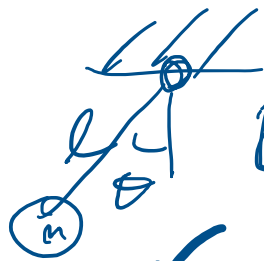


On the ANN for forecasting chaos – first show lots of examples....

-Good for near examples, but does poorly out of sample.

$$\dot{x} = f(x)$$

physics.



$$\frac{d\theta}{dt} = \dot{\theta}$$

$$m\ddot{\theta} + \gamma\dot{\theta} + \frac{g}{l}\theta = 0$$

$$\cancel{k\dot{\theta}^2}$$

$$v = \dot{\theta}$$

$$\dot{v} = \ddot{\theta}$$

$$= -\left(\frac{\gamma}{m}\right)v - \left(\frac{g}{l}\right)\theta$$

$$k_1 = \left(-\frac{\gamma}{m}\right) \quad k_2$$

$$\theta = x_1, \quad v = x_2$$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = k_1 x_2 + k_2 x_1$$

ODEs  
Examples  
Net

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

$$\vec{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

Data  
Examples

$$x = f(x)$$

$$f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

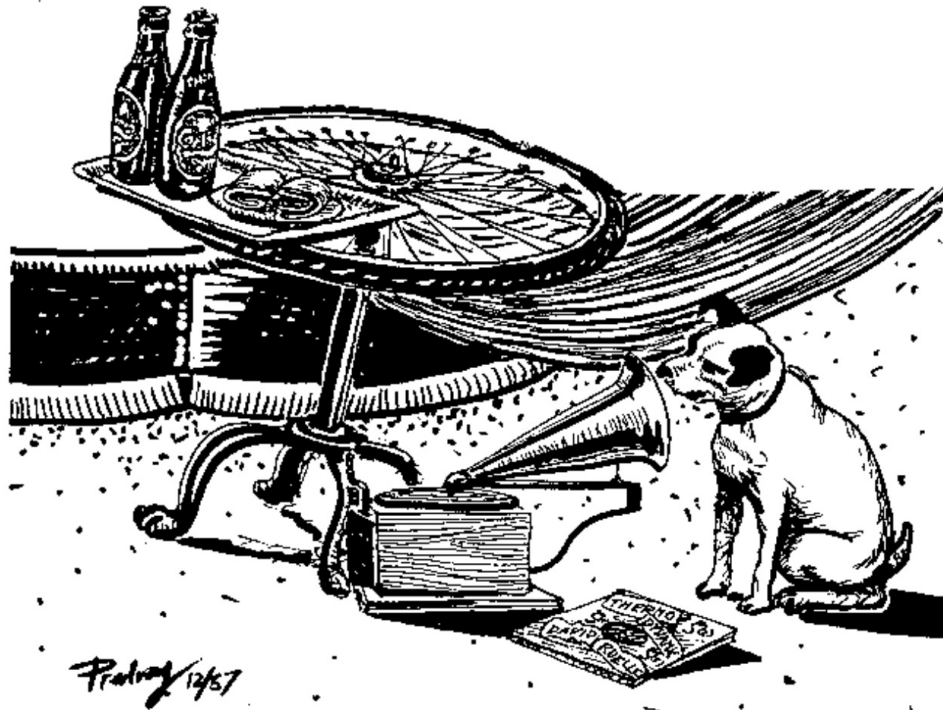
$$f(\vec{x})$$

$$\begin{pmatrix} x_2 \\ k_1 x_2 + k_2 x_1 \end{pmatrix}$$



# Classical and Quantum Chaos

## Part I: Deterministic Chaos



Predrag Cvitanović – Roberto Artuso – Per Dahlqvist – Ronnie Mainieri – Gregor Tanner – Gábor Vattay – Niall Whelan – Andreas Wirzba

## 2.4 Infinite-dimensional flows

Romeo: 'Misshapen chaos of well seeming forms!'  
W. Shakespeare, *Romeo and Juliet*, Act I, Scene I

flows - 10sep2003

version 10.1.6, Oct 25 2003

# The Kuramoto-Sivashinsky equation: A bridge between PDE'S and dynamical systems

## 2.4. INFINITE-DIMENSIONAL FLOWS

39



There is only one honorable cause that would justify sweating through so much formalism - this is but the sharpening of a pencil in order that we may attack the Navier-Stokes equation,

$$\rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \eta \left( \nabla^2 \mathbf{u} + \frac{1}{3} \nabla (\operatorname{div} \mathbf{u}) \right) + \mathbf{f}, \quad (2.13)$$

and solve the problem of turbulence. Being realistic, we are not so foolhardy to immediately plunge into *the* problem – there are too many dimensions and indices. Instead, we start small, in one spatial dimension,  $\mathbf{u} \rightarrow u$ ,  $\mathbf{u} \cdot \nabla \mathbf{u} \rightarrow \frac{1}{2} \partial_x u^2$ , assume constant  $\rho$ , forget about the pressure  $p$ , and so on. This line of reasoning, as well as many other equally sensible threads of thought, such as the amplitude equations obtained via weakly nonlinear stability analysis of steady flows, lead to the essentially same nonlinear PDEs, like the one that we turn to in the next section.

Flows described by partial differential equations are considered infinite dimensional because if one writes them down as a set of ordinary differential equations (ODEs), one needs infinitely many of them to represent the dynamics of one partial differential equation (PDE). Even though their phase space is infinite dimensional, the dynamics of many systems of physical interest converges toward finite-dimensional global attractors. We illustrate how this works with a concrete example, the Kuramoto-Sivashinsky system.

The Kuramoto-Sivashinsky equation: A bridge between PDE'S and dynamical systems

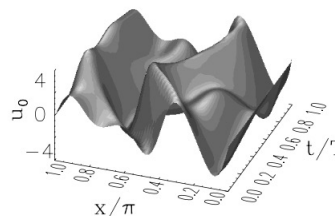
### 2.4.1 Fluttering flame front

The Kuramoto-Sivashinsky system, arising in the description of the flame front flutter of gas burning in a cylindrically symmetric burner on your kitchen stove and many other problems of greater import, is one of the simplest partial differential equations that exhibit chaos. It is a dynamical system extended in one spatial dimension, defined by

$$u_t = (u^2)_x - u_{xx} - \nu u_{xxxx} . \quad (2.14)$$

In this equation  $t \geq 0$  is the time and  $x \in [0, 2\pi]$  is the spatial coordinate. The subscripts  $x$  and  $t$  denote partial derivatives with respect to  $x$  and  $t$ ;  $u_t = du/dt$ ,  $u_{xxxx}$  stands for the 4th spatial derivative of the “height of the flame front” (more correctly the “velocity of the flame front”)  $u = u(x, t)$  at position  $x$  and time  $t$ . The “viscosity” parameter  $\nu$  controls the suppression of solutions with fast spatial variations. We take note, as in the Navier-Stokes equation (2.13), of the  $u\partial_x u$  “inertial” term, the  $\partial_x^2 u$  “diffusive” term (both with a “wrong” sign), etc.

**Figure 2.4:** Spatiotemporally periodic solution  $u_0(x, t)$ . We have divided  $x$  by  $\pi$  and plotted only the  $x > 0$  part, since we work in the subspace of the odd solutions,  $u(x, t) = -u(-x, t)$ .  $N = 16$  Fourier modes truncation with  $\nu = 0.029910$ . (From ref. [2.7])



The term  $(u^2)_x$  makes this a *nonlinear system*. It is one of the simplest conceivable nonlinear PDE, playing the role in the theory of spatially extended systems analogous to the role that the  $x^2$  nonlinearity plays in the dynamics of iterated mappings. The time evolution of a solution of the Kuramoto-Sivashinsky system is illustrated by figure 2.4. How are such solutions computed? The salient feature of such partial differential equations is a theorem saying that for any finite value of the phase-space contraction parameter  $\nu$ , the asymptotic dynamics is describable by a *finite* set of “inertial manifold” ordinary differential equations.

We are studying a “flame front”  $u(x, t) = u(x + 2\pi, t)$  periodic on the  $x \in [0, 2\pi]$  interval, so a reasonable strategy (but by no means the only one) is to expand it in a discrete spatial Fourier series:

$$u(x, t) = \sum_{k=-\infty}^{+\infty} b_k(t) e^{ikx}. \quad (2.15)$$



Since  $u(x, t)$  is real,  $b_k = b_{-k}^*$ . Substituting (2.15) into (2.14) yields the infinite ladder of evolution equations for the Fourier coefficients  $b_k$ :

$$\dot{b}_k = (k^2 - \nu k^4)b_k + ik \sum_{m=-\infty}^{\infty} b_m b_{k-m}. \quad (2.16)$$

As it follows from this equation that  $\dot{b}_0 = 0$ , the solution integrated over space is constant in time. We shall consider only the case of this average - the mean value of  $u$  - equal to zero,  $b_0 = \int dx u(x, t) = 0$ .

The coefficients  $b_k$  are in general complex functions of time  $t$ . We can isolate a smaller subspace of the system (2.16) further by considering the case of  $b_k$  pure imaginary,  $b_k = ia_k$ , where  $a_k$  are real, with the evolution equations

$$\dot{a}_k = (k^2 - \nu k^4)a_k - k \sum_{m=-\infty}^{\infty} a_m a_{k-m}. \quad (2.17)$$

This picks out the subspace of odd solutions  $u(x, t) = -u(-x, t)$ , so  $a_{-k} = -a_k$ . By picking this subspace we eliminate the continuous translational



symmetry from our consideration; that is probably not an option for an experimentalist, but will do for our purposes.

That is the infinite set of ordinary differential equations promised at the beginning of the section.

The trivial solution  $u(x, t) = 0$  is an equilibrium point of (2.14), but that is basically all we know as far as analytical solutions are concerned. You can integrate numerically the Fourier modes (2.17), truncating the ladder of equations to a finite number of modes  $N$ , that is, set  $a_k = 0$  for  $k > N$ . In the applied mathematics literature such a truncation is called a *Galerkin truncation*, or a *Galerkin projection*. For the parameter values explored below,  $N \leq 16$  truncations were deemed sufficiently accurate. In other words, even though our starting point (2.14) is an infinite-dimensional dynamical system, the asymptotic dynamics unfolds on a finite-dimensional attracting manifold, and so we are back on the familiar territory of sect. 2.2: the theory of a finite number of ODEs applies to this infinite-dimensional PDE as well.

Once the trajectory is computed in Fourier space, we can recover and plot the corresponding spatiotemporal pattern  $u(x, t)$  over the configuration space using (2.15), as in figure 2.4.

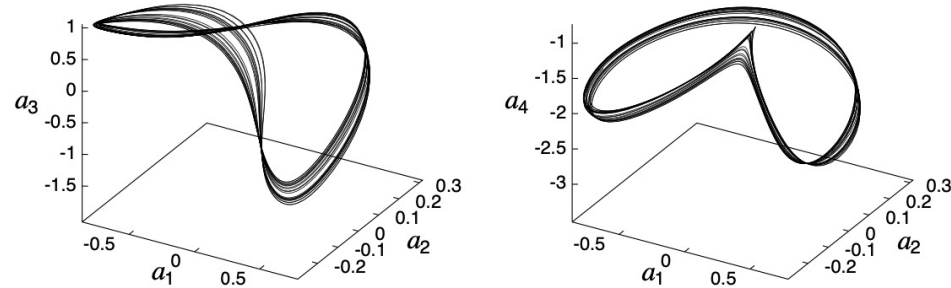
## 2.4.2 Fourier modes truncations

Thinking is extra price.

Fernando Solla

Consider now the case of initial  $a_k$  sufficiently small that the bilinear  $a_m a_{k-m}$  terms in (2.17) can be neglected. Then we have a set of decoupled linear equations for  $a_k$  whose solutions are exponentials, at most a finite number for which  $k^2 > \nu k^4$  is growing with time, and infinitely many with  $\nu k^4 > k^2$  decaying in time. The growth of the unstable long wavelengths (low  $|k|$ ) excites the short wavelengths through the nonlinear term in (2.17). The excitations thus transferred are dissipated by the strongly damped short wavelengths, and a “chaotic equilibrium” can emerge. The very short wavelengths  $|k| \gg 1/\sqrt{\nu}$  remain small for all times, but the intermediate wavelengths of order  $|k| \sim 1/\sqrt{\nu}$  play an important role in maintaining the dynamical equilibrium. As the damping parameter decreases, the solutions increasingly take on shock front character poorly represented by the Fourier basis, and many higher harmonics may need to be kept in truncations of (2.17).

Hence, while one may truncate the high modes in the expansion (2.17), care has to be exercised to ensure that no modes essential to the dynamics are chopped away. In practice one does this by repeating the same calculation at different truncation cutoffs  $N$ , and making sure that the inclusion of additional modes has no effect within the desired accuracy. For the figures given here, the numerical calculations were performed taking  $N = 16$  and



**Figure 2.5:** Projections of a typical 16-dimensional trajectory onto different 3-dimensional subspaces, coordinates (a)  $\{a_1, a_2, a_3\}$ , (b)  $\{a_1, a_2, a_4\}$ .  $N = 16$  Fourier modes truncation with  $\nu = 0.029910$ . (From ref. [2.7].)

the damping parameter value  $\nu = 0.029910$ , for which the system is chaotic (as far as can be determined numerically).

The problem with such high-dimensional truncations of the infinite tower of equations (2.17) is that the dynamics is difficult to visualize. The best we can do without much programming is to examine the trajectory's projections onto any three axes  $a_i, a_j, a_k$ , as in figure 2.5.

Examination of numerical plots such as figure 2.5 suggests that a more thoughtful approach would be to find a coordinate transformation  $y = h(x)$  to a “center manifold”, such that in the new, curvilinear coordinates large-scale dynamics takes place in  $(y_1, y_2)$  coordinates, with exponentially small dynamics in  $y_3, y_4 \dots$ . But - thinking is extra price - we do not know how to actually accomplish this.

We can now start to understand the remark on page 35 that for infinite dimensional systems time reversability is not an option: evolution forward in time strongly damps the higher Fourier modes. There is no turning back: if we reverse the time, the infinity of high modes that contract strongly forward in time now explodes, instantly rendering evolution backward in time meaningless. As everything in dynamics, this claim is also wrong, in a subtle way: if the initial  $u(x, 0)$  is *in* the non-wandering set (2.2), the trajectory is well defined both forward and backward in time. For practical purposes, this subtlety is not of much use, as any time-reversed numerical trajectory in a finite-mode truncation will explode very quickly, unless special precautions are taken.

## Commentary

**Remark 2.1** Model ODE and PDE systems. Rössler system was introduced in ref. [2.2], as a simplified set of equations describing time evolution of concentrations of chemical reagents. The Duffing system (2.6) arises in the study of electronic circuits. The theorem on finite dimensionality of inertial manifolds of phase-space contracting PDE flows is proven in ref. [2.4]. The Kuramoto-Sivashinsky equation



was introduced in refs. [2.5, 2.6]; sect. 2.4 is based on V. Putkaradze's term project (see [wwcb/extras](#)), and on the Christiansen *et al.* article [2.7]. How good a description of a flame front this equation is need not concern us here; suffice it to say that such model amplitude equations for interfacial instabilities arise in a variety of contexts - see e.g. ref. [2.8] - and this one is perhaps the simplest physically interesting spatially extended nonlinear system.

**Remark 2.2** Diagnosing chaos. In sect. 1.3.1 we have stated that a deterministic system exhibits “chaos” if its dynamics is locally unstable (positive Lyapunov exponent) and globally mixing (positive entropy). In sect. 8.3 we shall define Lyapunov exponents, and discuss their evaluation, but already at this point it would be handy to have a few quick numerical methods to diagnose chaotic dynamics. Laskar's *frequency analysis* method [2.10] is useful for extracting quasi-periodic and weakly chaotic regions of phase space in Hamiltonian dynamics with many degrees of freedom. For references to several other numerical methods, see ref. [2.11].

[2.5] Kuramoto Y and Tsuzuki T Persistent propagation of concentration waves in dissipative media far from thermal equilibrium *Progr. Theor. Physics* **55** 365, (1976).

[2.6] Sivashinsky G I Nonlinear analysis of hydrodynamical instability in laminar flames - I. Derivation of basic equations *Acta Astr.* **4** 1177, (1977).

## Kuramoto-Sivashinsky Equations

The Kuramoto-Sivashinsky (KS) equation is a nonlinear PDE which arises as a description of flame front flutter of gas burning in a cylindrically symmetric burner. We will consider Kuramoto-Sivashinsky system in the following form,

$$u_t = -\nu u_{xxxx} - u_{xx} + 2uu_x$$

where  $(t, x) \in [0, \infty) \times (0, L)$  in periodic domain,  $u(t, x) = u(t, x + L)$ , and we restrict our solution to the subspace of odd solutions  $u(t, -x) = -u(t, x)$ .

To develop the Galerkin projection, we expand a periodic solution  $u(x, t)$  using a discrete spatial Fourier series,

$$u(x, t) = \sum_{-\infty}^{\infty} b_k(t) e^{ikqx}$$

where  $q = \frac{2\pi}{L}$ . For simplicity, consider  $L = 2\pi$ , then  $q = 1$ . Then, by substituting, we produce the infinitely many evolution equations for the Fourier coefficient,

$$\dot{b}_k = (k^2 - \nu k^4) b_k + ik \sum_{m=-M}^M b_m b_{k-m}$$

where  $2M + 1$  is the selected finite number of modes. The coefficients  $b_k$  are complex functions of time  $t$ . However, by symmetry, we can reduce to a subspace by considering the special symmetry case that  $b_k$  is pure imaginary,  $b_k = ia_k$  and  $a_k \in \mathbb{R}$ . Then,

$$\dot{a}_k = (k^2 - \nu k^4) a_k - k \sum_{m=-M}^M a_m a_{k-m}$$



and we can take  $N_m = 2M$ , and the  $k$  can take the values  $k = 1, 2, \dots, N_m$  representing  $N_m$ -dimensional ODE, which is implemented in function file KSEode.m.

In order to solve the KSE ode for a finite number of modes, we first define:

```
clearvars; close all; clc;

K = 16;                %Number of modes N_m
a0 = 0.05*rand(K,1);  %Initial condition (small value)

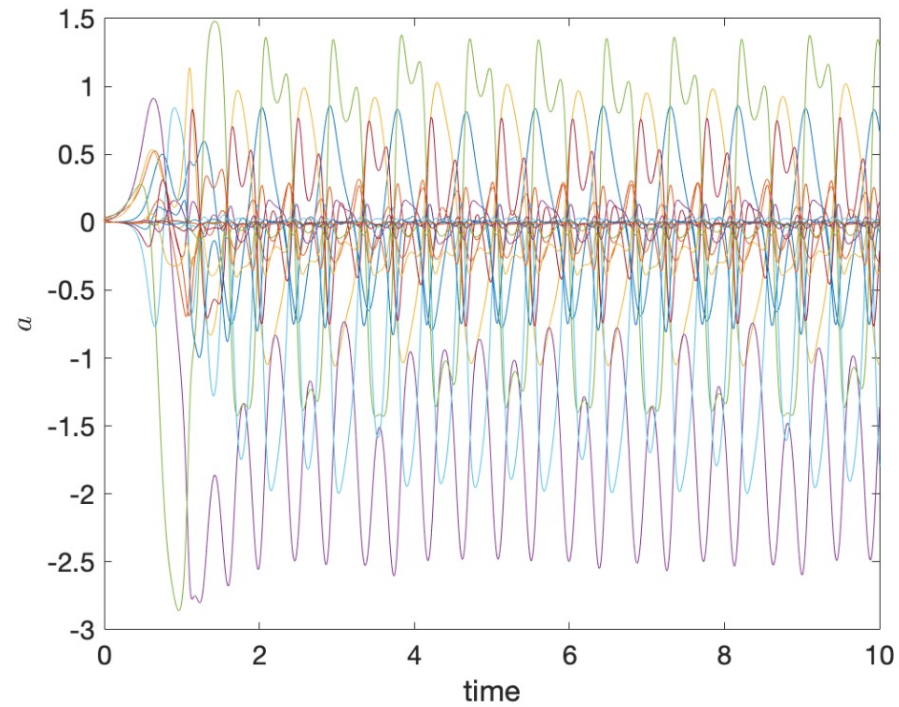
t0 = 0; tfinal = 10; %Initial and final time
h = 0.001; %time step

% Set the ode solver options
options = odeset('RelTol',1e-10,'AbsTol',1e-10);

% Use ode solver to solve the ode defined
% in the function KSEode.m
[t, a] = ode45(@KSEode,(t0:h:tfinal),a0,options);
```

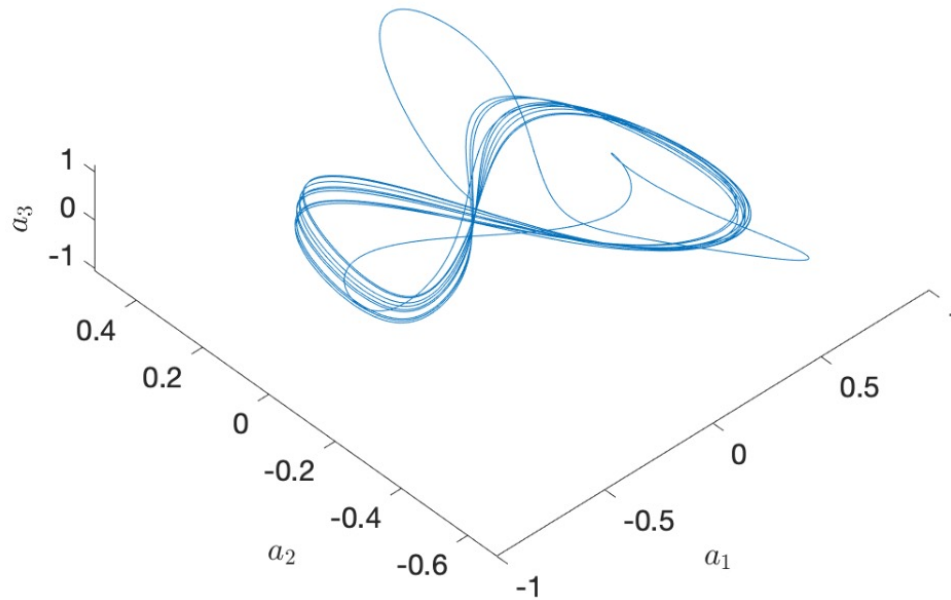
Now, Fouries coefficients is stored in the matrix  $a$ . To visualize the data:

```
figure
plot(t,a)
xlabel('time')
ylabel('$a$', "Interpreter", "latex")
set(gca, 'FontSize', 15)
```



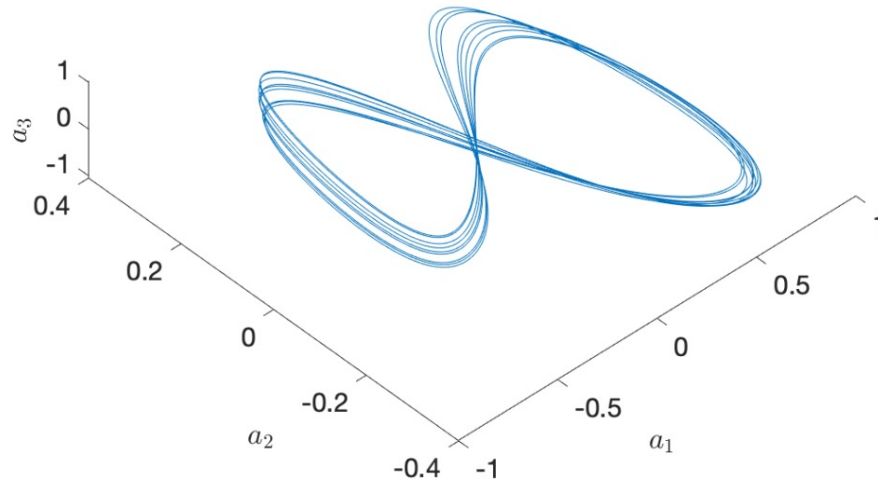
For a more clear view of the dynamic, we can plot sequential modes as a 3D plot. For example, to plot the first three modes we can write:

```
figure
plot3(a(:,1),a(:,2),a(:,3))
view([-43.02 74.81])
xlabel("$a_1$","Interpreter","latex")
ylabel("$a_2$","Interpreter","latex")
zlabel("$a_3$","Interpreter","latex")
set(gca,'FontSize',15)
```



Note that you can skip the transient part of the trajectories for more clear view:

```
figure
% plot the last 80% of points
N = round(0.8*length(t));
plot3(a(end-N:end,1),a(end-N:end,2),a(end-N:end,3))
view([-43.02 74.81])
xlabel("$a_1$","Interpreter","latex")
ylabel("$a_2$","Interpreter","latex")
zlabel("$a_3$","Interpreter","latex")
set(gca,'FontSize',15)
```



Now, we can get the time domain solution from the equation (see main text):

$$u(t, x) = \sum_{-\infty}^{\infty} b_k(t) e^{ikqx}$$

as follows:

```

u = zeros(length(t),K); %Initialize time domain solution
x = linspace(0,2*pi,K); % partition spatial domain for K measuring po.
for n = 1:length(t) %loop over all trajectories
    for k = 1:length(x) % loop through each spatial point
        u(n,k) = 1.i .* sum(a(n,:).*exp(1.i .* k .* x));
    end
end
end

```

To show the time domain solution as surface plot:

```
figure
h = surf(abs(u)');
pbaspect([2 1 1])
colorbar('southoutside')
ylim([1 size(a,2)])
xlim([1 length(t)])
shading interp

set(gca,'ytick',[0 4 8 12 16])
set(gca,'yticklabels',{'0','\pi/4','\pi/2','3\pi/4','2\pi'})

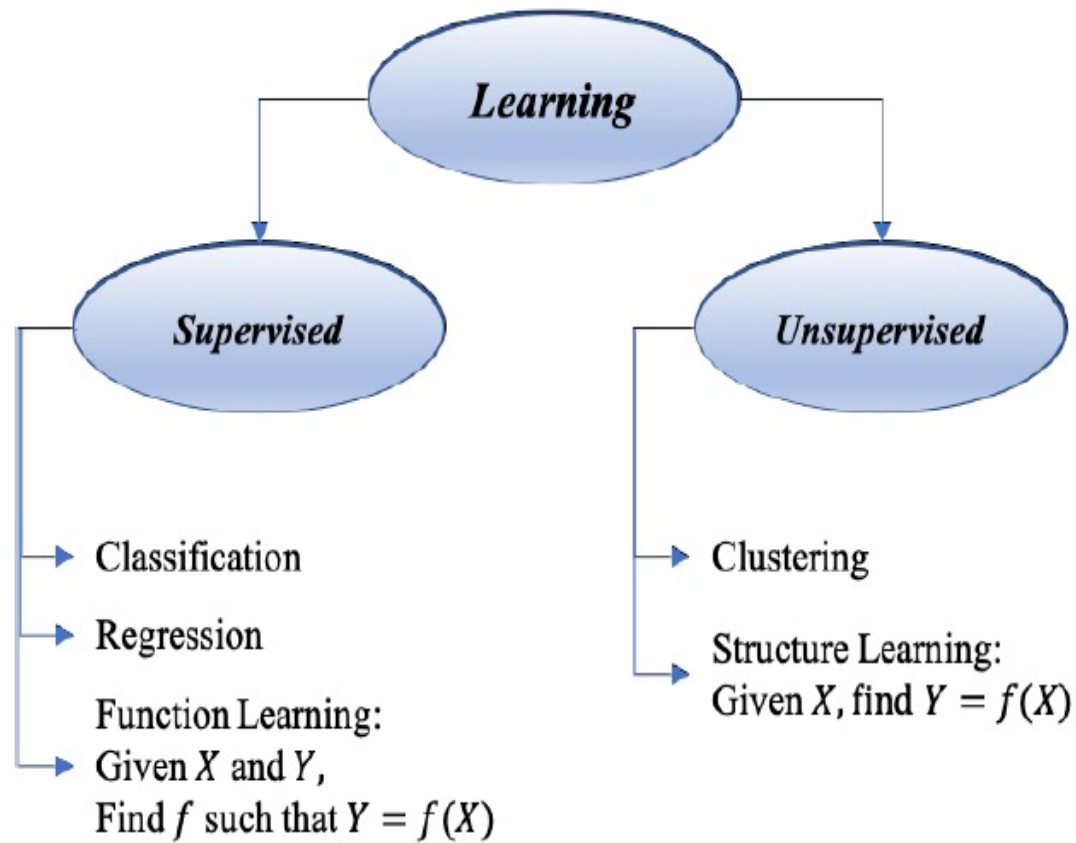
ylabel('$x$', 'Interpreter', 'latex')
xlabel('time steps index', 'Interpreter', 'latex')
title('$u(x,t)$ by true solution', 'Interpreter', 'latex', 'FontSize'
set(gca, 'FontSize', 15)

view([0 90])
```





Learning Dichotomy

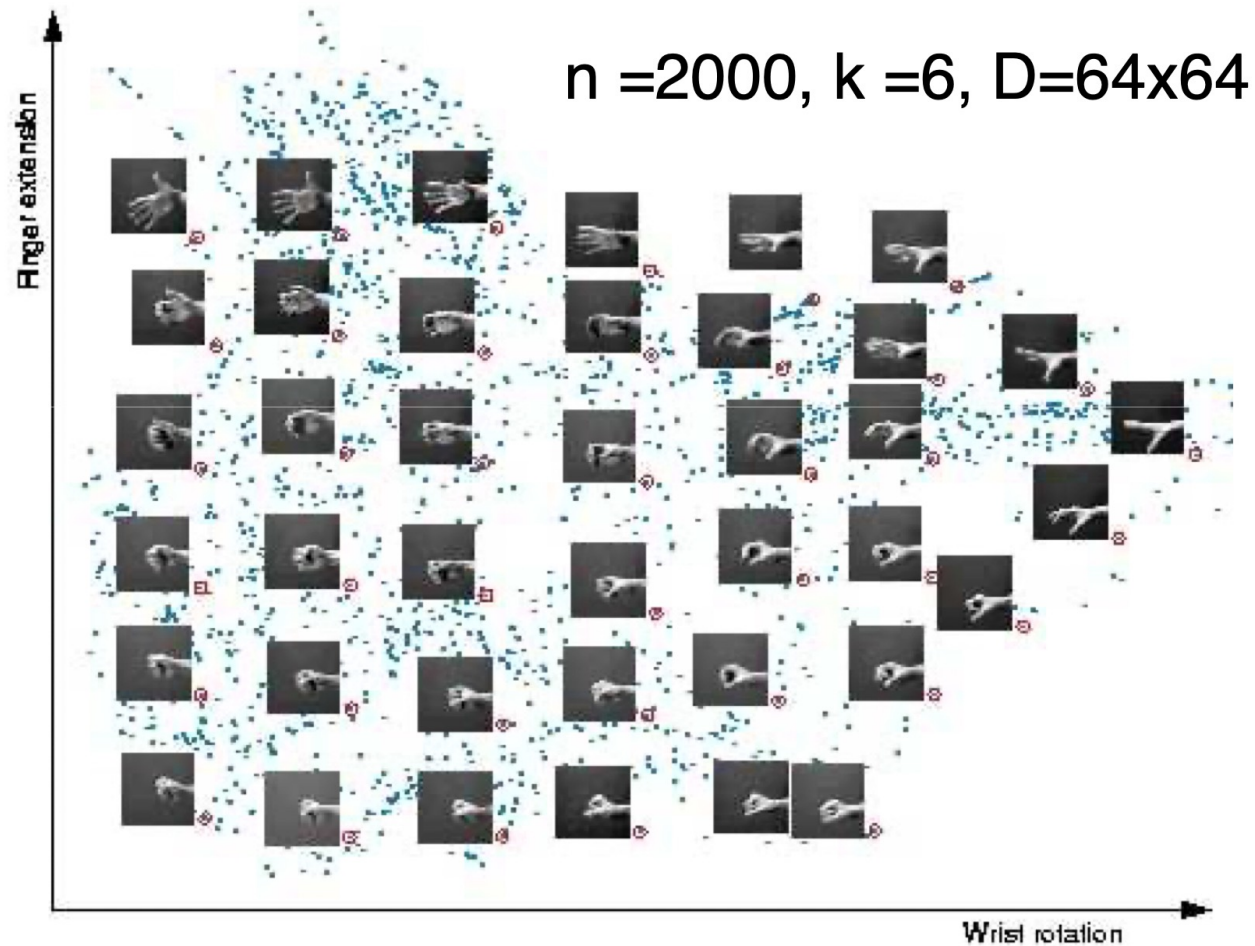


Labelled data  
Learn a function

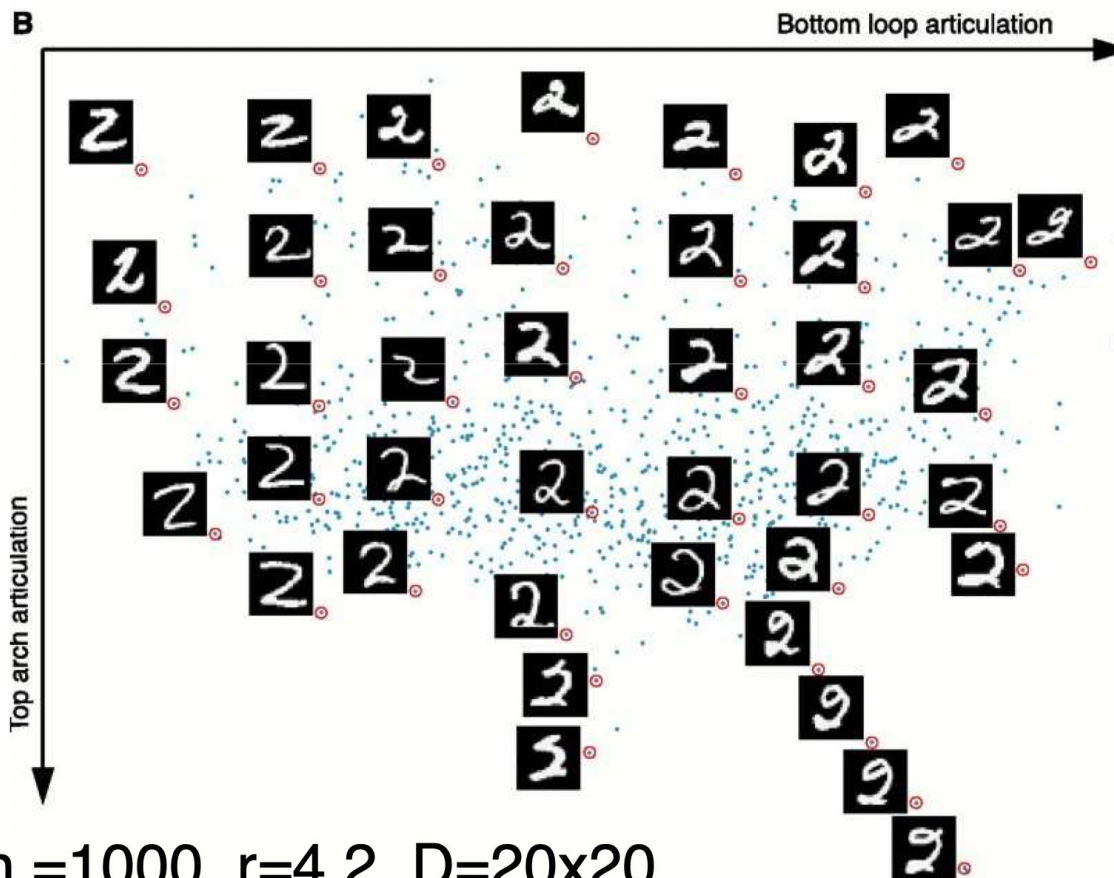
Pattern  
Structure

Well...at least one more  
- re-enforcement learning

# Isomap: Two-dimensional embedding of hand images (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)



Isomap: two-dimensional embedding of hand-written '2' (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)



Isomap: three-dimensional embedding of faces (from Josh. Tenenbaum, Vin de Silva, John Langford 2000)

$n = 698, k = 6$

