The Geometry of Deep Learning. Lecture 2: Deep Learning

Rita Fioresi

Course for PhD Program of Unimore, Ferrara and Parma

February 1, 2023

Deep Learning: CS231 Stanford

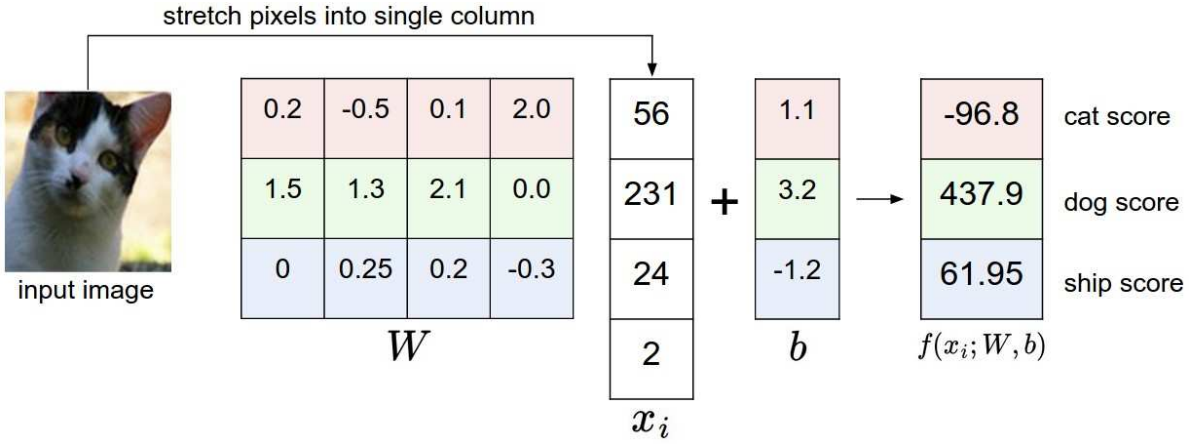
CS231n: Convolutional Neural Networks for Visual Recognition

Spring 2020

Previous Years: [Winter 2015] [Winter 2016] [Spring 2017] [Spring 2018] [Spring 2019]

Ingredients for Deep Learning

- **Score function**: it is a function of the weights w (es. linear classifier)



- **Loss function**: measures error (L_i loss of datum i)

$$L_i = -log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} = -f_{y_i} + log \sum_j e^{f_j}, \qquad L = \sum_i L_i$$

- **Optimizer**: for weights update "minimizes" the Loss (via stochastic gradient):

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \nabla L_{\text{stoc}}, \qquad \nabla L_{\text{stoc}} = \sum_{i=1}^{32} \nabla L_{\text{rand(i)}}$$

Training

Divide the dataset (ex. CIFAR10):
80% Data for **training**
10% Data for **validation**
10% Data for **test** (ONCE)

1. **Learning**: determine weights **parameters**
2. **Validation**: determine net structure.
   Example: choose loss function, number of layers, learning rate etc.
   Goal: find best hyperparameters.
3. **Test**: once at the end.

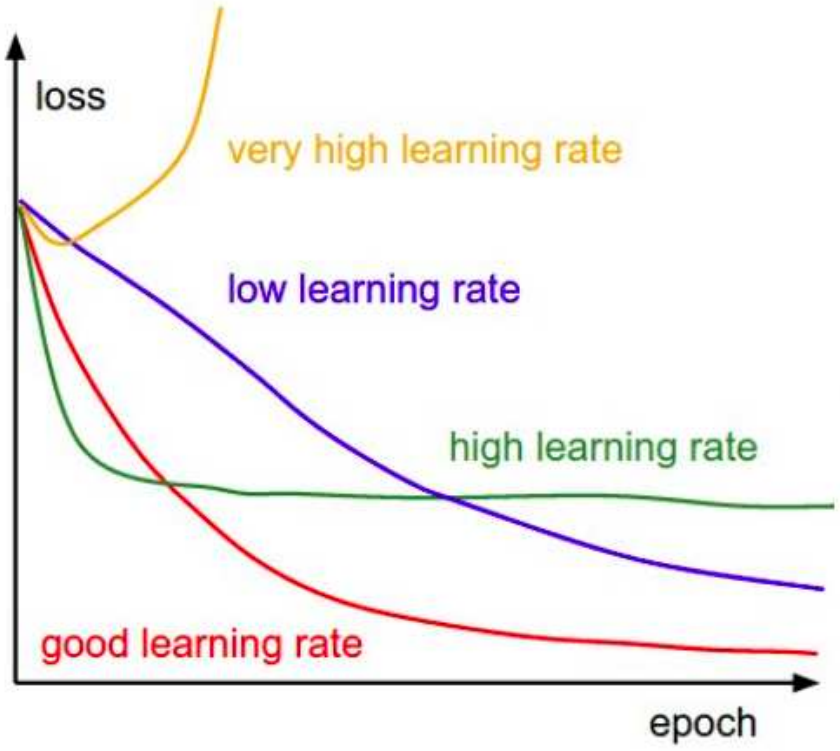
Accuracy: percentage of accurate predictions on tests set.

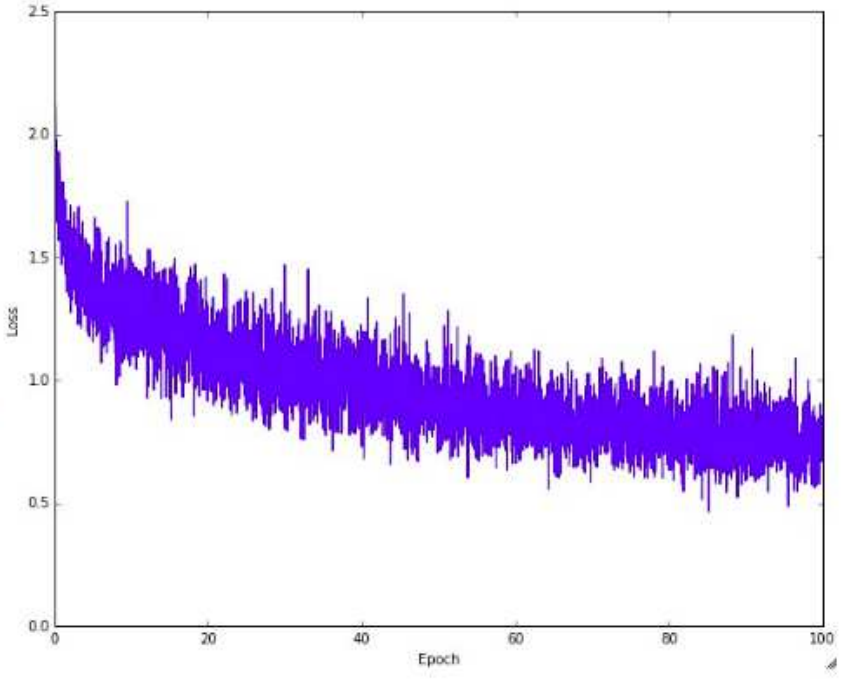
1. Learning process

- **Step 1**: Compute score of images in training set (**Forward pass**) The weights are inizialized randomly.
- **Step 2**: Compute the loss (it measure the "difference" between given label and correct label for each datum in training set).
- **Step 3**: Compute **Stochastic Gradient**. (**Backpropagation**)
- **Step 4**: update weights.
- **Step 5**: Repeat Step 1-2-3 up to an epoch.
- **Step 6**: After 150-200 epochs reduce learning rate and repeat all steps 1-5.

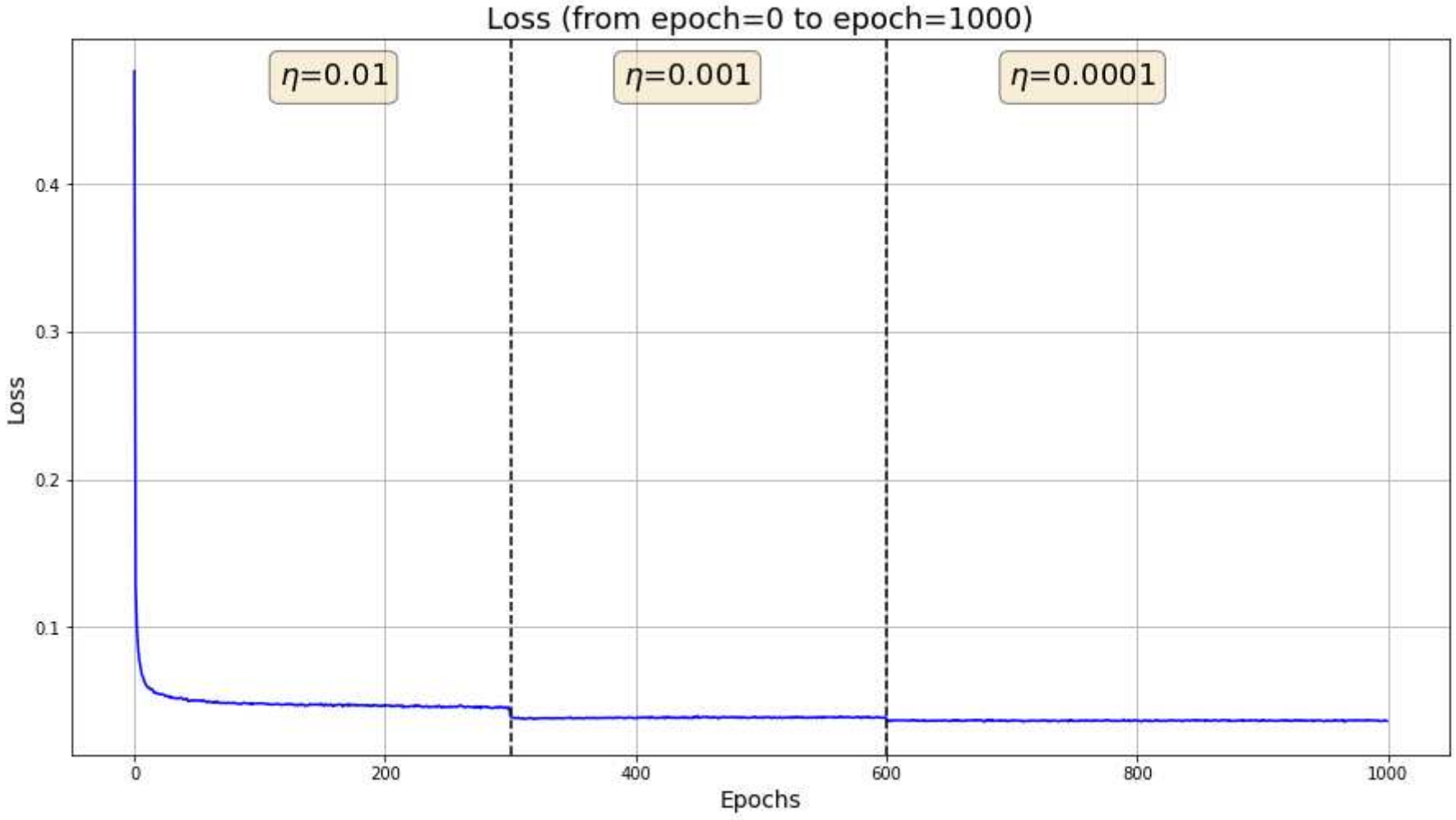
Epoch= ||Training set||/||minibatch size||.
NOTE: measure accuracy every 10-20 epochs.
Example: 40000 training set (CIFAR10), 32 images in minibatch, 1 epoch=40000/32 updates.

Loss accuracy in epochs: CIFAR10





Loss accuracy in epochs: MNIST



2. Validation process

Purpose of validation: Determine **hyperparameters**:

- α learning rate,
- $\mathcal{B}$ minibatch size,
- optimizer (SGD, Adam),
- net structure (e.g. how many layers, parameters)
- training (e.g. number of epochs)

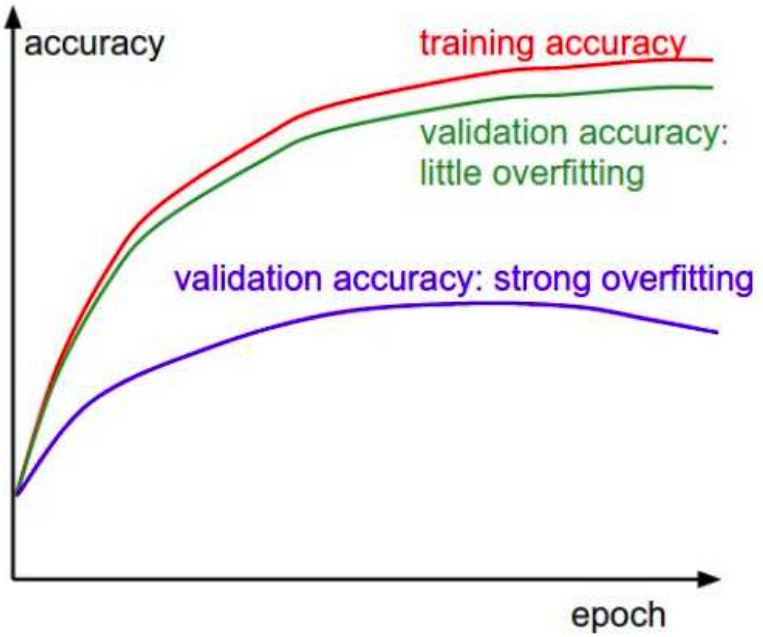
We vary hyperparameters giving some values:

- e.g. $\alpha = 0.1, 00.1$ etc
- e.g. $\mathcal{B} = 8, 16, 32$

We use the **validation set** to test accuracy, while searching for best hyperparameters.

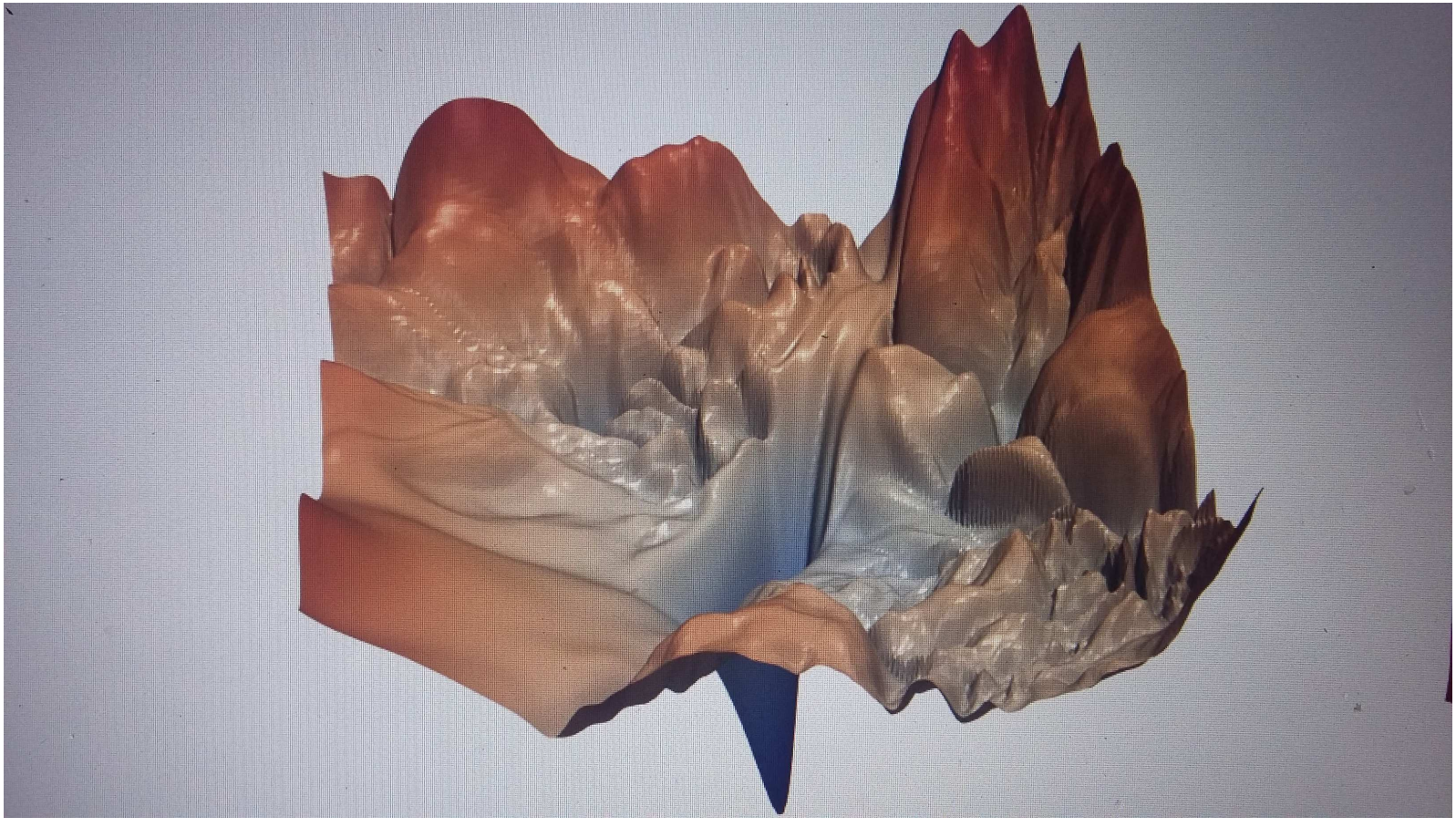
3. Test

ATTENTION!: use test set ONCE to avoid **overfitting**!



Validation technique: cross validation=rotation of the training set.

Loss Landscape



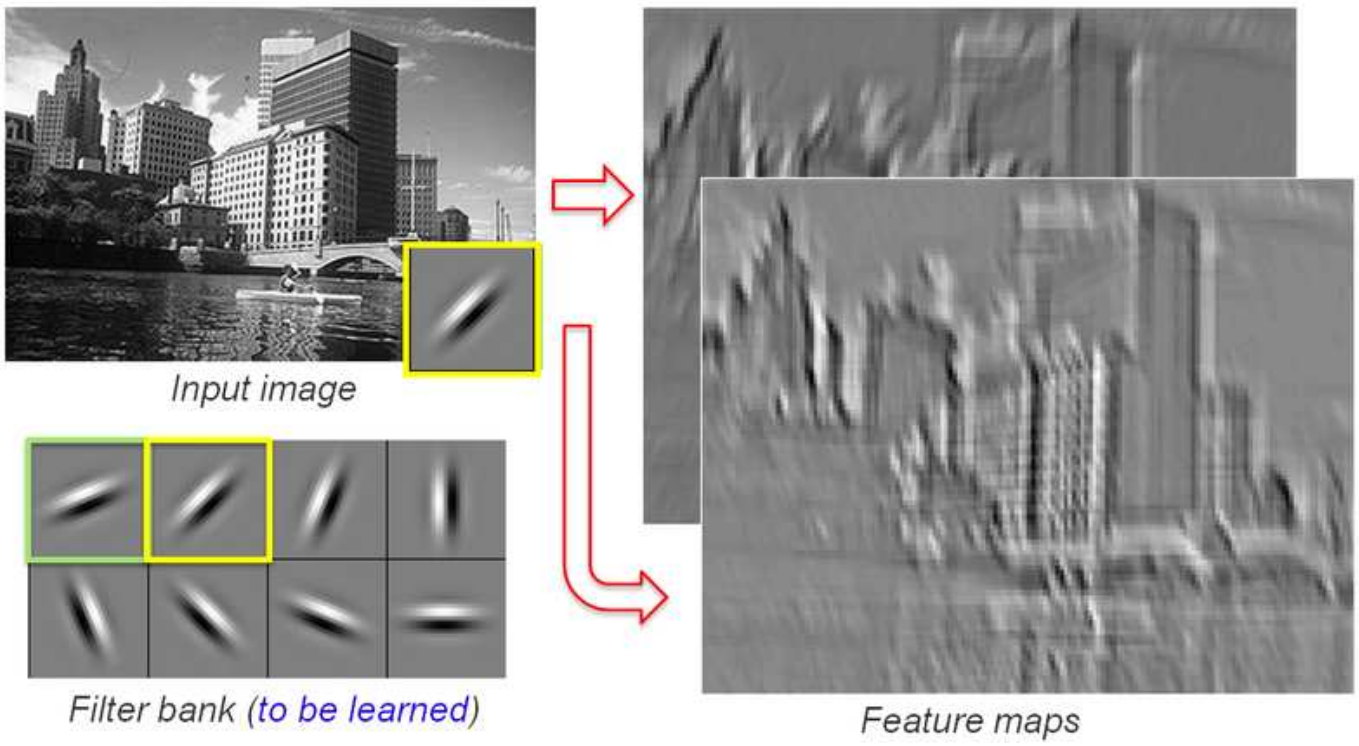
Loss (projection) as function of weights.

The Deep Learning Algorithm in more detail

1. **Score function**
   (convolutions)
2. **Loss function**
   (ex.: cross entropy with softmax)
3. **Optimizer**
   (example: stochastic gradient)

1. SCORE function: Convolutional Neural Networks

Convolutions: extract *features* from images.
Represent the mathematical operation of discrete convolutions via kernels (here called filters).



Convolutions and "edge detection"

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6 x 6

*

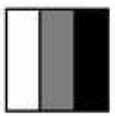
1	0	-1
1	0	-1
1	0	-1

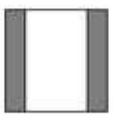
3 x 3

=

-0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

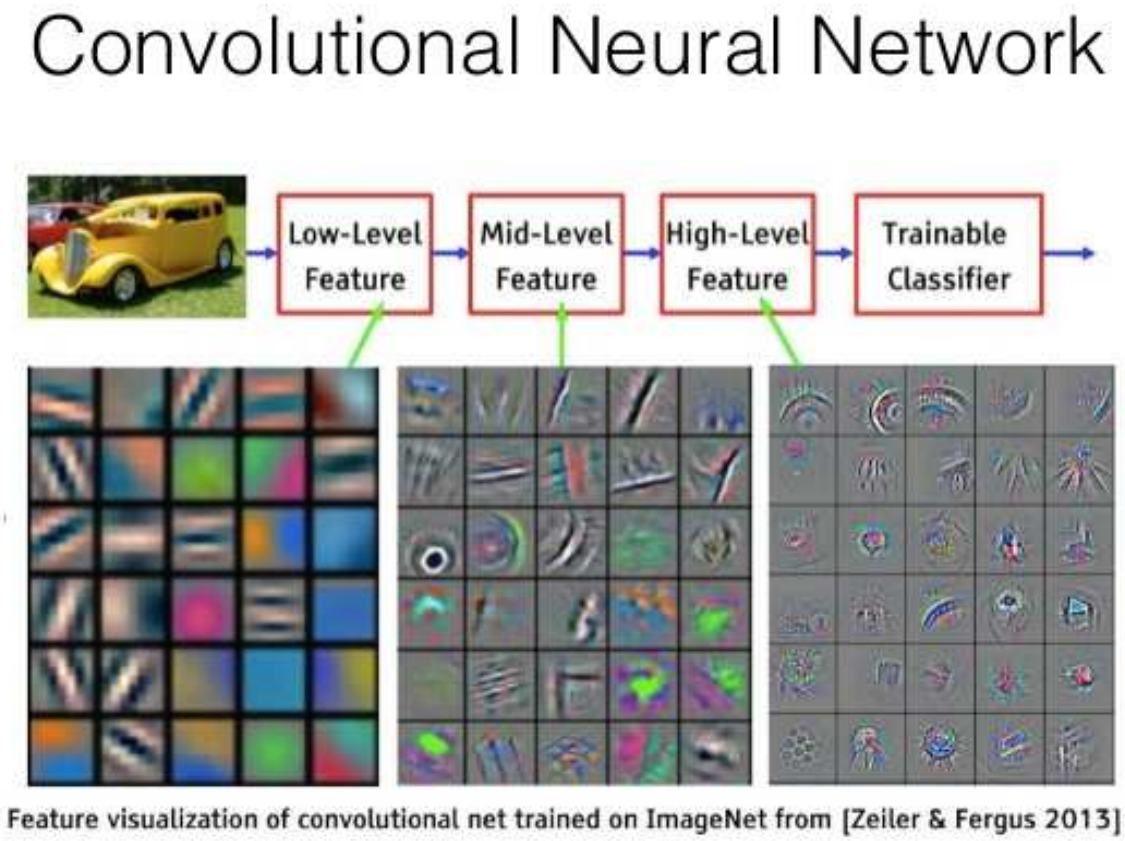
4 x 4





Convolutions

Filters



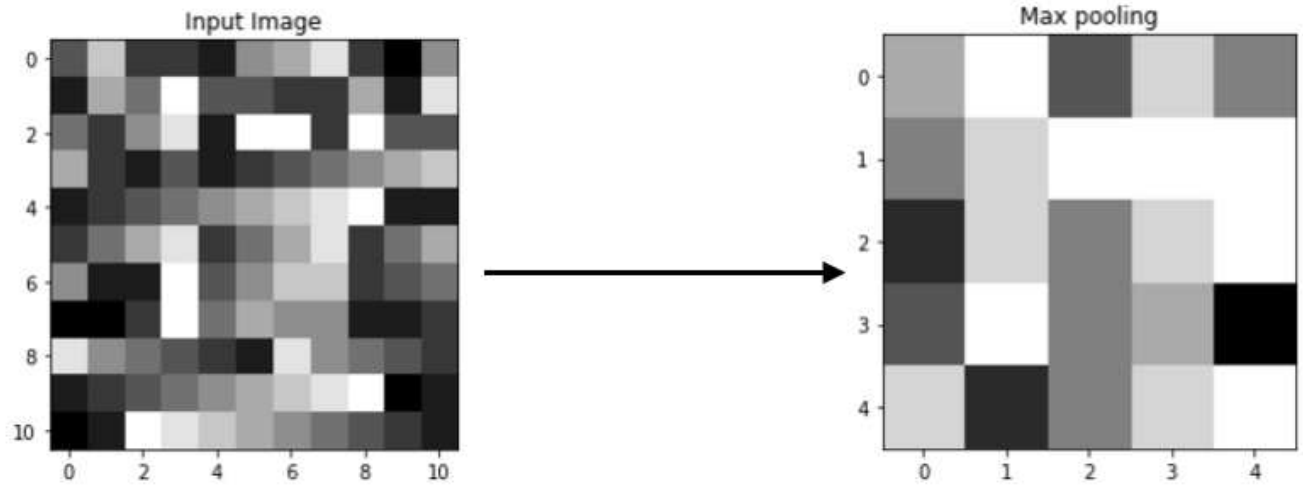
The algorithm **learns** filters!
https://arxiv.org/abs/1711.08856

Maxpool filter

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

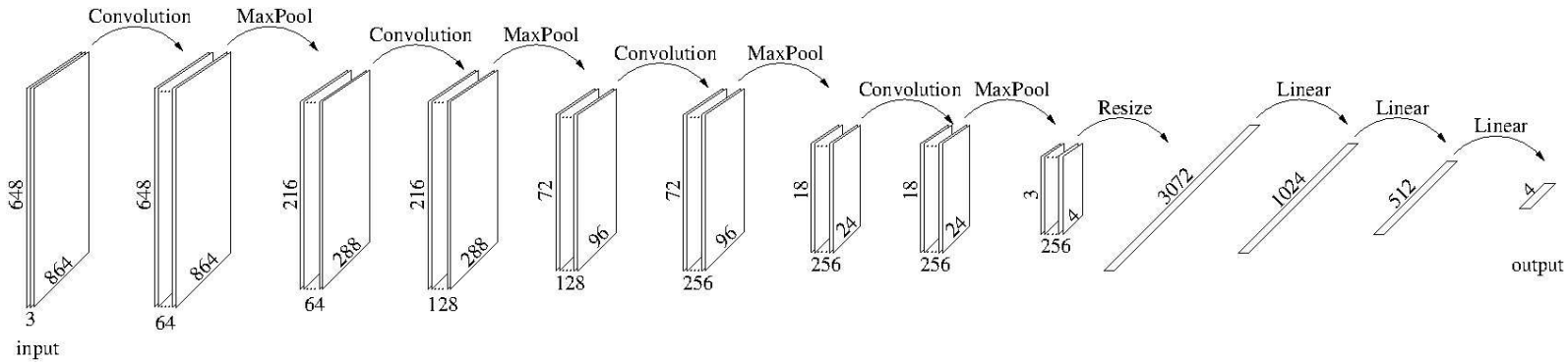
2×2 Max-Pool

20	30
112	37



Example of Deep Learning Neural Network

A typical network structure (alexnet):

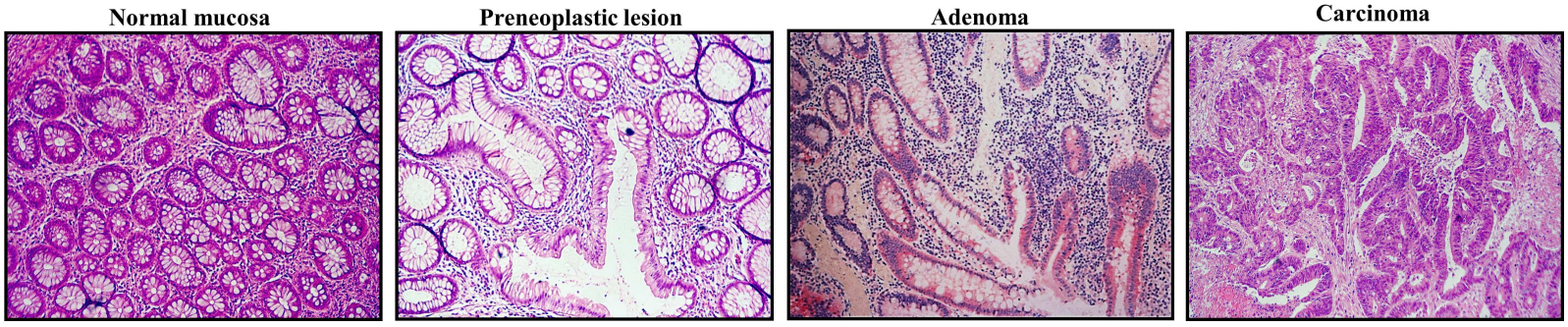


Classification accuracy:

Training	Exact match	Nearest match
Short	$93.79 \pm 0.76\%$	$99.85 \pm 0.11\%$
Long	$95.10 \pm 0.19\%$	$99.84 \pm 0.05\%$

Practical use: Colon Cancer Detection

Optical images with Leica IRBE, CCD Rising Tech, 20X magnification. 200 patients (1998-2008).



Data	Normal	Preneo	Adenoma	Cancer	Total
Train	1616	1628	2736	2968	8048
Validate	200	204	344	256	1004
Test	200	204	340	256	1000

2. LOSS function

Deep Learning uses the Cross Entropy Loss:

$$L(w) = \sum_x L(x, w) = -\sum_x \log[e^{s_{\text{label}(x)}}(x)/(e^{s_1(x)} + \cdots + e^{s_N(x)})]$$

$L(x)$: loss for image x.
More generally:

- **Amari loss**: $l(x, w) = -\log(p(y|x, w))$
- **Empirical loss**: $L(x, w) = \mathbb{E}_{y\sim p}[-\log(p(y|x, w))]$, where

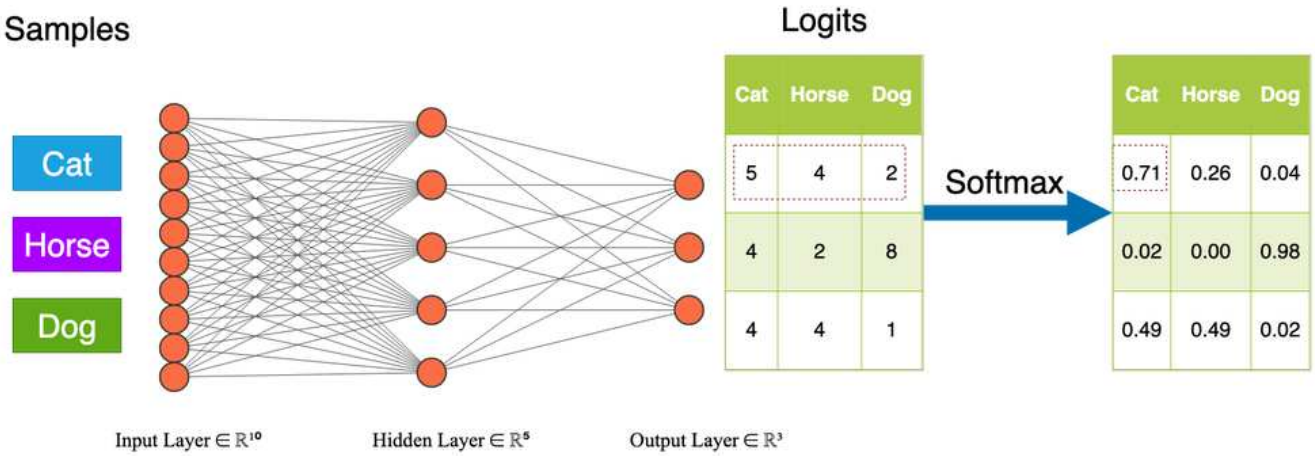
$$L(x, w) = \mathbb{E}_{y\sim p}[-\log(p(y|x, w))] = -\sum_i p_i(y|x, w) \log(p_i(y|x, w))$$

In Deep Learning $p(y|x, w)$ is obtained via Softmax:

$$p(y|x, w) = (p_i(y|x, w)) = \left(\frac{e^{s_i(x)}}{\sum_j e^{s_j(x)}}\right)$$

Softmax and Cross Entropy Loss/1

Softmax



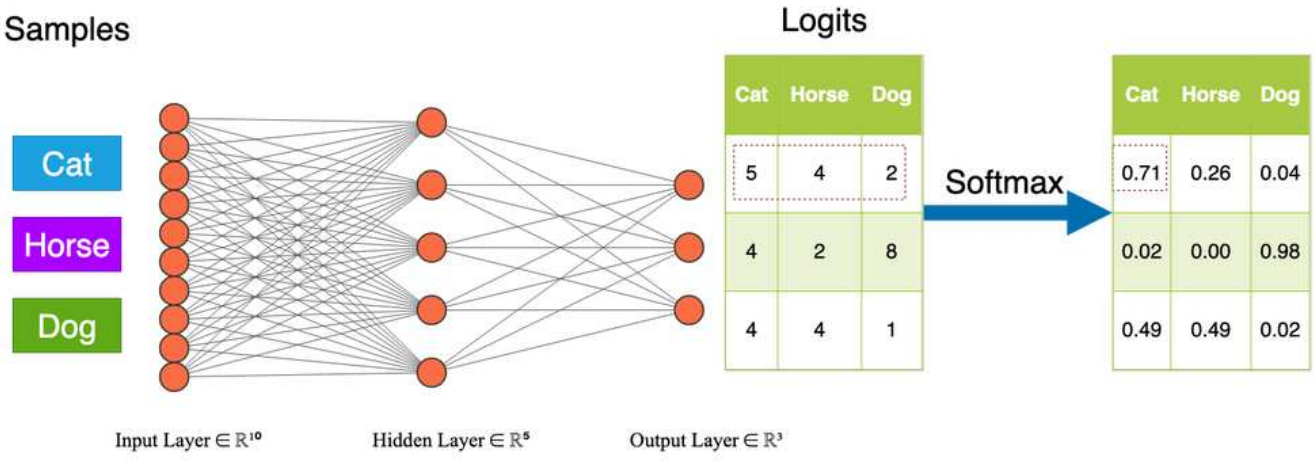
Hence:

$$p(\text{cat}) = \left(\frac{e^5}{e^5+e^4+e^2}, \frac{e^4}{e^5+e^4+e^2}, \frac{e^2}{e^5+e^4+e^2}\right)$$

$$= (0.71, 0.26, 0.04)$$

$Loss(\text{cat}) = -\log(p(cat)) = 0.34$ choose correct label (here "cat")!

Softmax and Cross Entropy Loss/2



Loss of one image:
$L(x) = -\log[e^{s_{\text{label}(x)}}(x)/(e^{s_1(x)} + \cdots + e^{s_N(x)})]$: loss for image x.
Total loss: sum over all images

$$Loss = -\log(p(cat)) - \log(p(horse)) - \log(p(dog)) =$$

$$= -\log(0.71) - \log(0.002) - \log(0.02) = 0.34 + 6 + 3.91 = 10.25$$

KL Divergence

Kullback-Leibler divergence measures the "difference" between the correct distribution $q(x)$ and the empirical one $p(x)$:

$$KL(p(x)\|q(x)) = \mathbf{E}_q[p] = \sum q_i(x)\log(p_i(x)) =$$

$$= q_1(x)\log(p_1(x)) + q_2(x)\log(p_2(x)) + \cdots + q_n(x)\log(p_n(x))$$

Previous example:

$$x = \text{cat} \longrightarrow p(x) = (0.71, 0.26, 0.04), \qquad q(x) = (1, 0, 0)$$

$$KL(p(x)\|q(x)) = 1 \cdot \log(0.71) + 0 \cdot \log(0.002) + 0 \cdot \log(0.02) = 0.34$$

This is the loss function (for image="cat")!

Loss and KL Divergence

The loss function is the Kullback-Leibler divergence up to a constant:

$$
\begin{aligned}
L(x, w) &= \mathbb{E}_{y\sim q}[-\log(p(y|x, w))] = \\
&= \sum_{i=1}^{C} q_i(y|x) \log \frac{q_i(y|x)}{p_i(y|x, w)} - \sum_{i=1}^{C} q_i(y|x) \log q_i(y|x) = \\
&= \mathrm{KL}(q(y|x)\|p(y|x, w)) - \sum_{i=1}^{C} q_i(y|x) \log q_i(y|x). \qquad (1)
\end{aligned}
$$

$q(y|x)$: true distribution (ex.: mass density distribution)
$p(y|x, w)$: empirical distribution

Notice: $L(x, w) = \mathrm{KL}(q(y|x)\|p(y|x, w))$,
when $q(y, x)$ is the mass density distribution i.e.
$q(y|x) = (0, 0, \ldots, 1, \ldots, 0, 0)$

3. OPTIMIZER

Principal optimizers:

- Stochastic Gradient
- SGD with Nesterov momentum
- ADAM (more popular)

Careful: read all options and make sure you understand them!

Regularization: add regularization to the loss to keep your parameters from getting too large (default is zero!).
Example: L^2 regularization, λ regularization parameter.

$$L(w) = \frac{1}{N} \sum_i L(x_i, w) + \lambda \sum_j |w_j|^2$$

The Fisher Information matrix

Information Geometry: use of differential geometry to study probability and statistics.

Key idea: the parameters of a probability distribution are a manifold and possess a natural metric (Fisher).

$$F(x, w) = \mathbb{E}_{y \sim p}[\nabla_w \log p(y|x, w) \cdot (\nabla_w \log p(y|x, w))^T]$$

References

- Shun-ichi Amari, *Natural Gradient Works Efficiently in Learning*, 1998.
- F. Nielsen, *An Elementary Introduction to Information Geometry*, Entropy, 2020.
- J. Martens. *New insights and perspectives on the natural gradient method.* Journal of Machine Learning Research, 21(146):1-76, 2020.

Properties of the Fisher matrix

Proposition. F is the covariance matrix of the gradient of the Amari loss.

Proof. In fact, the Amari loss is $l(x, w) = -\log p(y|x, w)$, its gradient is

$$\nabla_w l(x, w) = -\frac{\nabla_w p(y|x, w)}{p(y|x, w)}$$

Properties of the Fisher matrix

Proposition. F is the covariance matrix of the gradient of the Amari loss.

Proof. In fact, the Amari loss is $l(x, w) = -\log p(y|x, w)$, its gradient is

$$\nabla_w l(x, w) = -\frac{\nabla_w p(y|x, w)}{p(y|x, w)}$$

Notice:

$$\mathbb{E}_{y\sim p}(\nabla_w l) = \sum p_i \frac{\nabla_w p_i}{p_i} =$$

$$= \sum_i \nabla_w p_i = \nabla_w(\sum_i p_i) = 0$$

Properties of the Fisher matrix

Proposition. F is the covariance matrix of the gradient of the Amari loss.

Proof. In fact, the Amari loss is $l(x, w) = -\log p(y|x, w)$, its gradient is

$$\nabla_w l(x, w) = -\frac{\nabla_w p(y|x, w)}{p(y|x, w)}$$

Notice:

$$\mathbb{E}_{y\sim p}(\nabla_w l) = \sum p_i \frac{\nabla_w p_i}{p_i} =$$

$$= \sum_i \nabla_w p_i = \nabla_w(\sum_i p_i) = 0$$

The covariance matrix of $\nabla_w l(x, w)$ is (by definition):

$$\mathrm{Cov}(l) = \mathbb{E}_{y\sim p}[(\nabla_w l - \mathbb{E}_{y\sim p}(\nabla_w l))^t(\nabla_w l - \mathbb{E}_{y\sim p}(\nabla_w l))] =$$

$$= \mathbb{E}_{y\sim p}[(\nabla_w l)^t(\nabla_w l)] = F(x, w)$$

Hessian Geometry

Proposition. $F = \mathbb{E}_{y\sim p}[\mathbb{H}(l)]$, $l = -\log p(y|x,w)$.

Proof. In fact (write $p = p(y|x,w)$):

$$\mathbb{H}[l] = -\mathrm{Jac}\left[\frac{\nabla_w p}{p}\right] = -\left[\mathbb{H}(p)\cdot p + \nabla_w p \cdot \nabla_w p\right]\frac{1}{p^2}$$

Take the expected value:

Hessian Geometry

Proposition. $F = \mathbb{E}_{y\sim p}[\mathbb{H}(l)]$, $l = -\log p(y|x,w)$.

Proof. In fact (write $p = p(y|x,w)$):

$$\mathbb{H}[l] = -\mathrm{Jac}\left[\frac{\nabla_w p}{p}\right] = -\left[\mathbb{H}(p)\cdot p + \nabla_w p \cdot \nabla_w p\right]\frac{1}{p^2}$$

Take the expected value:

$$\mathbb{E}_{y\sim p}[\mathbb{H}[l]] = -\sum_i p_i \frac{\mathbb{H}(p_i)}{p_i} + \mathbb{E}_{y\sim p}\left[\frac{\nabla_w p}{p}\cdot\frac{\nabla_w p}{p}\right] = F$$

where $\sum_i \mathbb{H}(p_i) = \mathbb{H}(\sum_i p_i) = 0$.

The Fisher matrix

The Fisher matrix for a minibatch of ONE sample image:

$$F(x,w) \quad = \mathbb{E}_{y\sim p}[\nabla_w \log p(y|x,w) \cdot (\nabla_w \log p(y|x,w))^T]$$

Key Facts:

$$\mathrm{KL}(p(y|x,w+\delta w)||p(y|x,w)) \quad \cong \tfrac{1}{2}(\delta w)^T F(x,w)(\delta w) + \mathcal{O}(||\delta w||^3)$$

The Fisher matrix F provides a natural metric on the **parameter space** during dynamics of the stochastic gradient descent.

$$\mathrm{rank}(F) < \text{number of classes}$$

The metric is neither Riemannian nor subriemannian.
Not constant rank either!

Data matrix

Idea. Treat weights w and images x on the same ground:

$$F(x,w) = \mathbb{E}_{y\sim p}[\nabla_w \log p(y|x,w)\cdot(\nabla_w \log p(y|x,w))^T]$$

$$G(x,w) = \mathbb{E}_{y\sim p}[\nabla_x \log p(y|x,w)\cdot(\nabla_x \log p(y|x,w))^T].$$

Key Facts:

$$\mathrm{KL}(p(y|x,w+\delta w)||p(y|x,w)) \cong \tfrac{1}{2}(\delta w)^T F(x,w)(\delta w) + \mathcal{O}(||\delta w||^3)$$

$$\mathrm{KL}(p(y|x+\delta x,w)||p(y|x,w)) \cong \tfrac{1}{2}(\delta x)^T G(x,w)(\delta x) + \mathcal{O}(||\delta x||^3)$$

Properties of the Fisher matrix F and local data matrix G

1. $F(x, w)$ and $G(x, w)$ is a positive semidefinite symmetric matrix.
2. $\ker F(x, w) = (\mathrm{span}_{i=1,\ldots,C}\{\nabla_w \log p_i(y|x, w)\})^{\perp}$;
3. $\ker G(x, w) = (\mathrm{span}_{i=1,\ldots,C}\{\nabla_x \log p_i(y|x, w)\})^{\perp}$.
4. $\mathrm{rank}\ F(x, w) < C, \quad \mathrm{rank}\ G(x, w) < C.$

Dataset	$G(x, w)$ size	$\mathrm{rank}\ G(x, w)$ bound
MNIST	784	10
CIFAR-10	3072	10
CIFAR-100	3072	100
ImageNet	150528	1000

For the Fisher the difference in size and rank is larger!

Data manifold

Result (F.-Grementieri, 2021). Let w be the weights of a deep ReLU neural network classifier, p given by softmax, $G(x, w)$ the local data matrix.
The distribution in the data domain:

$$x \mapsto \mathcal{D}_x = (\ker G(x, w))^{\perp}$$

is involutive i.e.

$$[X, Y] \in \mathcal{D}, \qquad \forall\, X, Y \in \mathcal{D}.$$

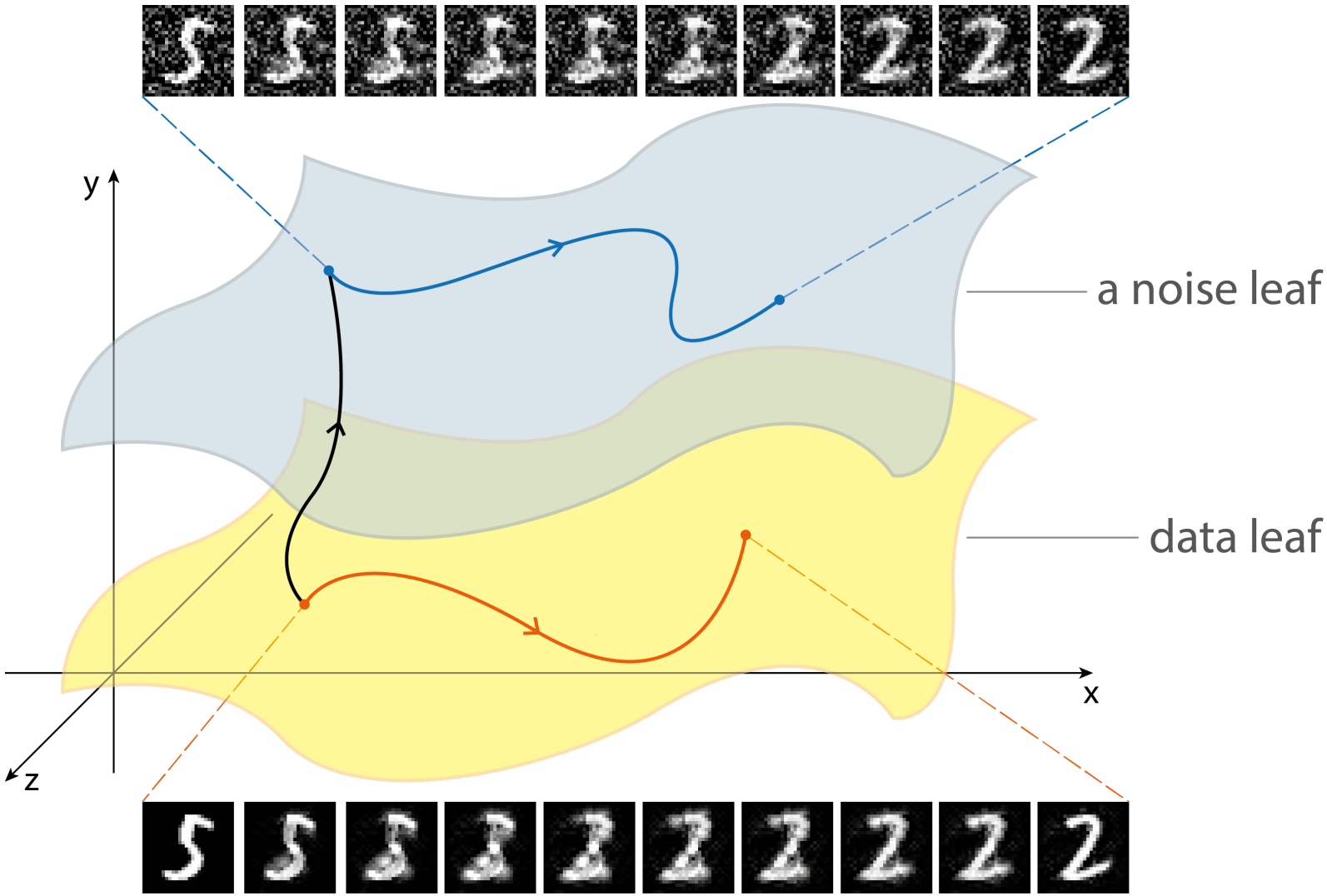
1. At each point in the dataset in $\mathbb{R}^n$, $\ker G(x, w)^{\perp}$ is tangent to a submanifold (**data leaf**) of dimension $\operatorname{rank} G(x, w) < C$
2. G defines a foliation on $\mathbb{R}^n$ of rank at most $C - 1$ (**Frobenius Thm**).

Remark: This is not true for the distribution via the Fisher matrix!

$$w \mapsto \mathcal{D}'_w := (\ker F(w))^{\perp}$$

is **not** involutive (e.g. MNIST, lenet).

Data manifold



Riemannian Structure on the Data Manifold

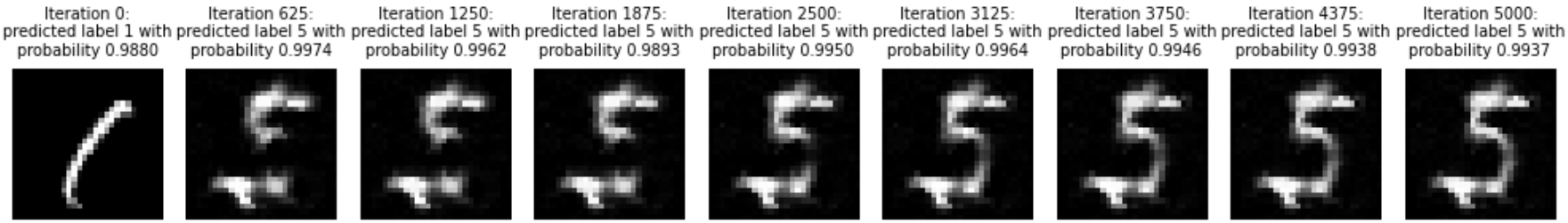
Facts

- The matrix $G(x, w)$, restricted to the subspace $(\ker G(x, w))^{\perp}$ gives a Riemannian metric to each leaf of the foliation.
- All the dataset is on one leaf: the **data leaf**
  We perform dimensionality reduction!
- We move from a point x in our dataset to any other point x' in the dataset with an with an *horizontal* path, that is a path on the data leaf.
- Not all points on the data leaf are in the data set, but they represent *symbols*.

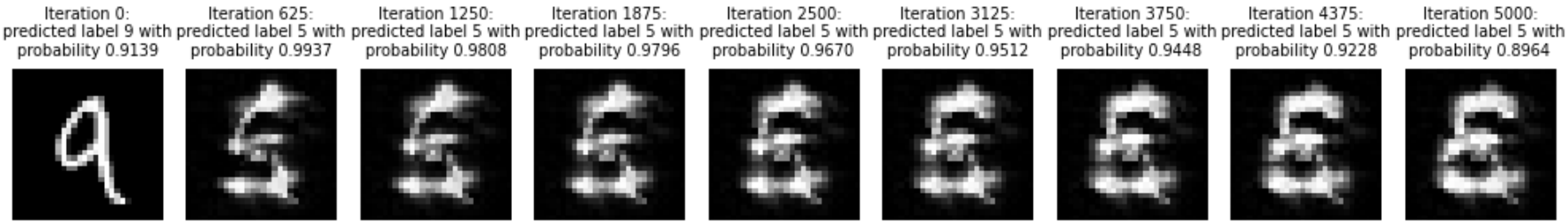
Moving on the data leaf: MNIST

Moving around in **on the data leaf**:

- We can connect any two data=images.
- Any path starting from one image and going to another goes through data with the same level of noise.

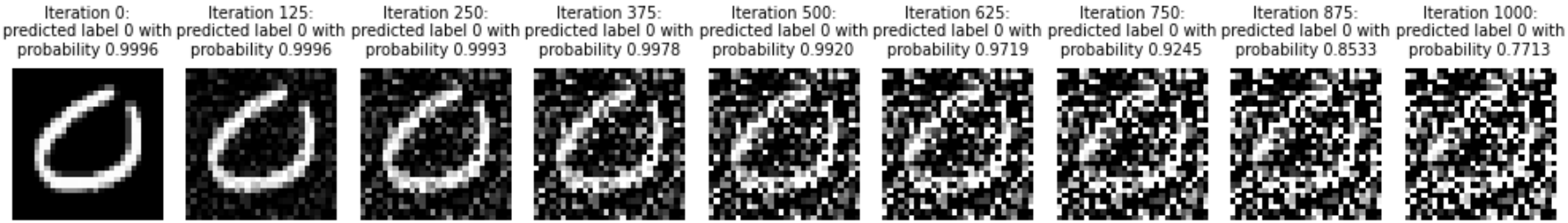


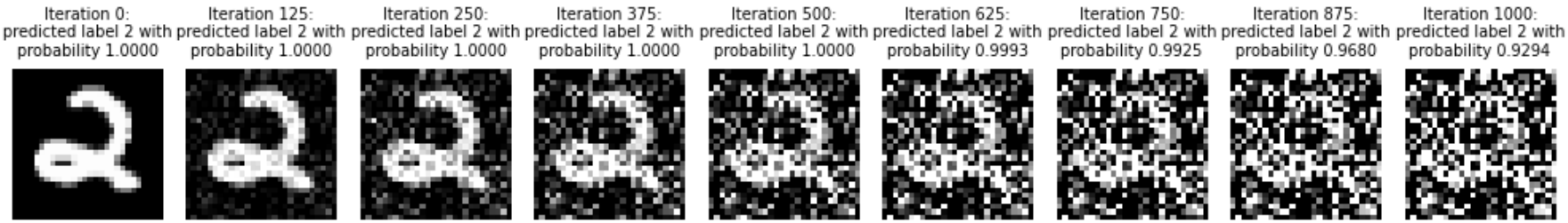
We can connect a digit from MNIST to a symbol **not** in MNIST moving on the **data leaf**:



Moving away from the data leaf: MNIST

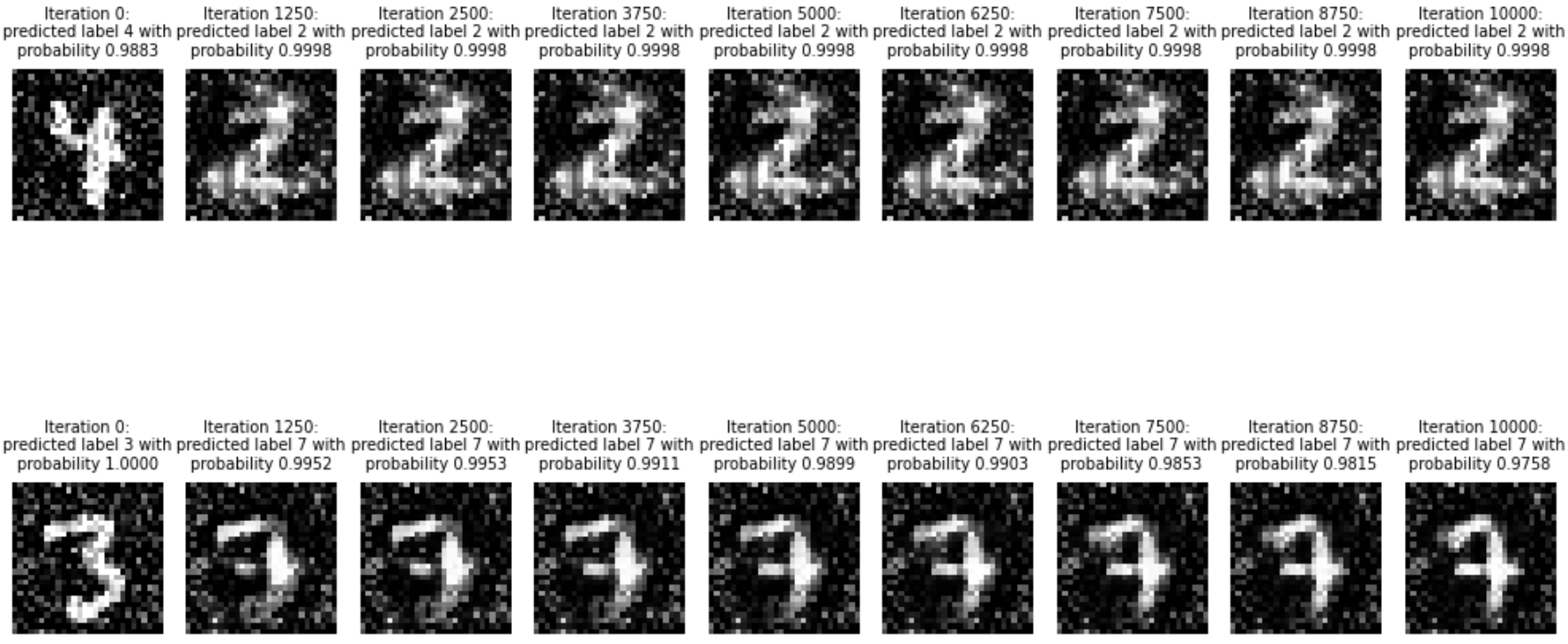
When moving **away** from a given data leaf, noise is added, but the accuracy is high.



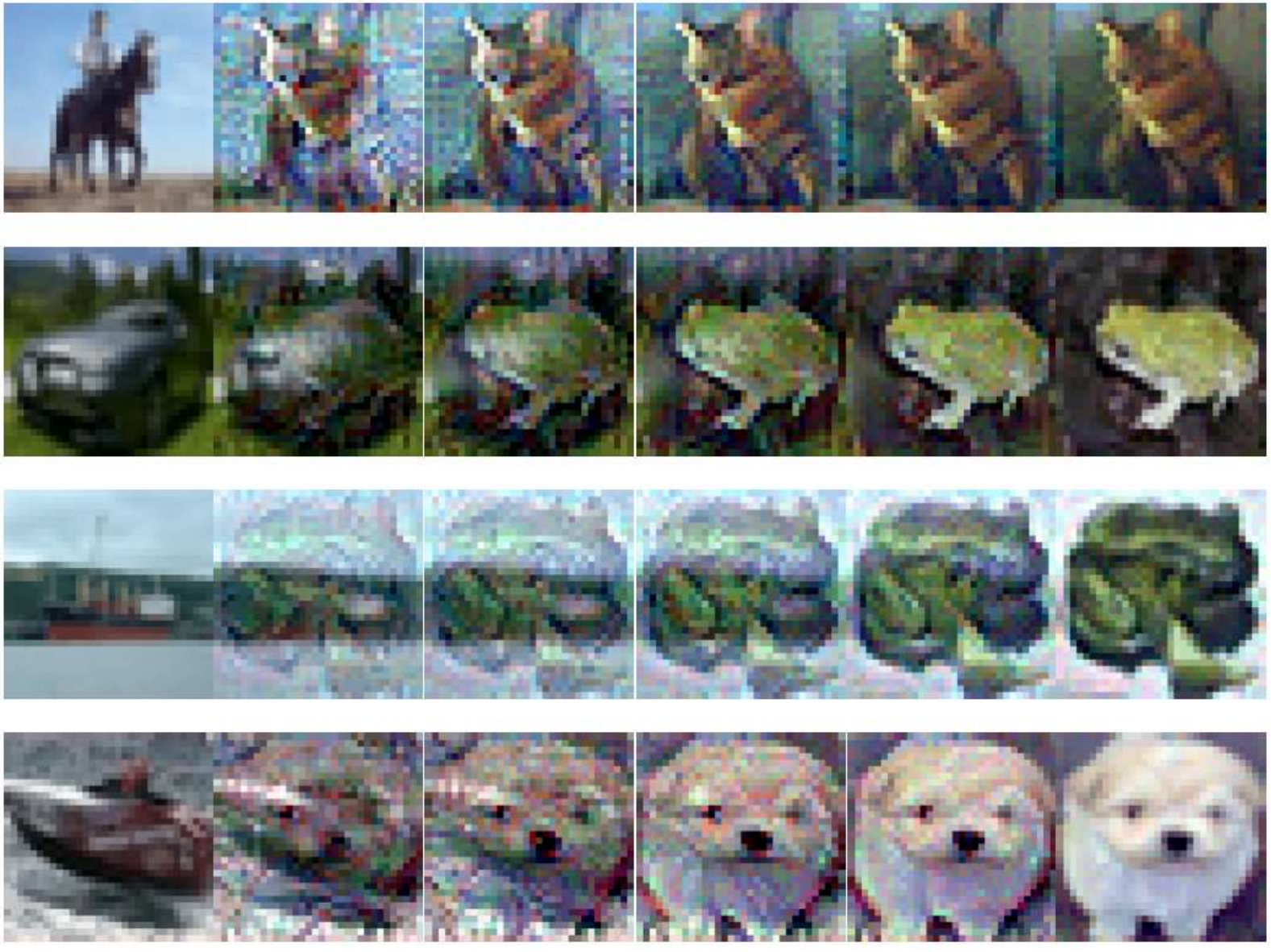


Moving on a noisy leaf: MNIST

We can connect a noisy datum with any other datum with the **same** level of noise:



Moving on the data manifold: CIFAR10



Neurogeometry

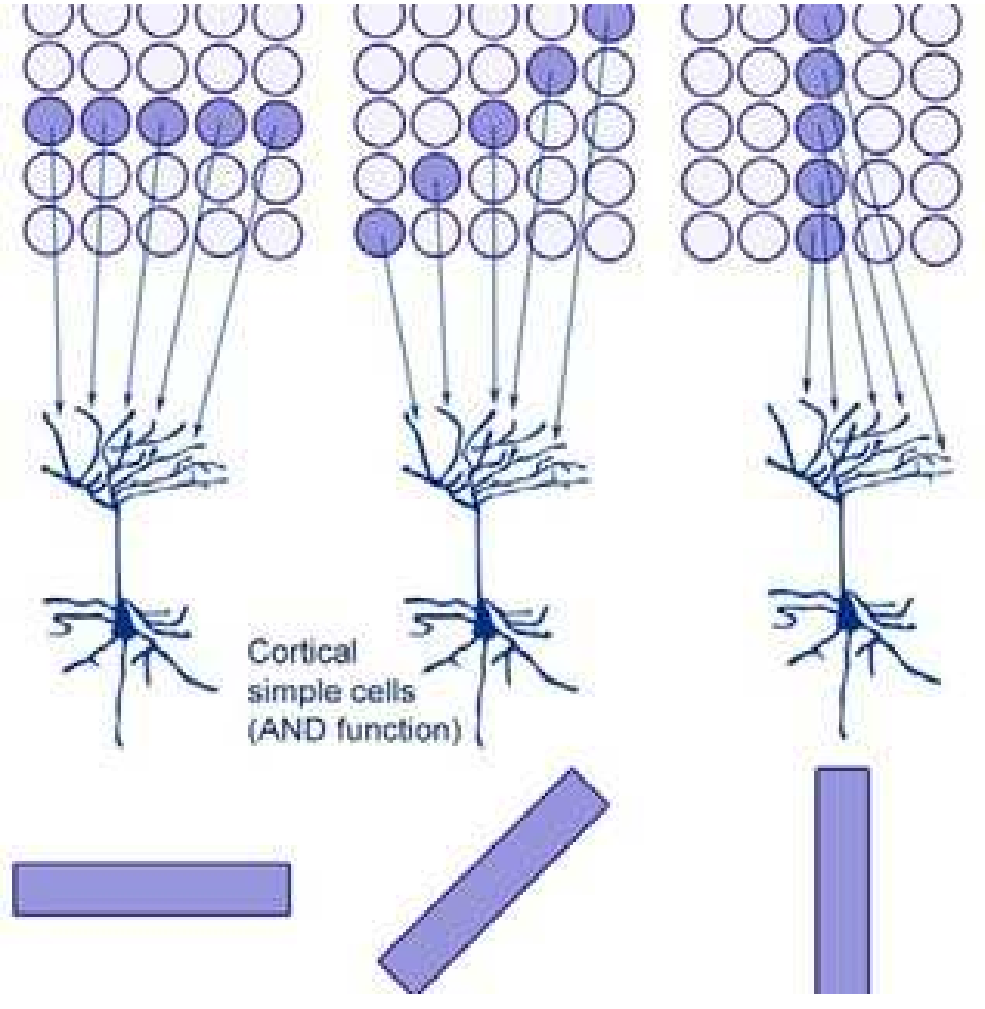
Modelling the human visual cortex using mathematics.

1) **Hoffman W.C.** - *The visual cortex is a contact bundle*, 1989
2) **Mumford D.** - *Elastica and Computer vision*, 1994

3) **Petitot J., Tondut Y.** - *Vers une Neurogeometrie. Fibrations corticales, structures de contactet contours subjectifs modaux*, 1999
4) **Citti G., Sarti A.** - *Cortical based model of perceptual completion in the Roto -Translation space*, 2006

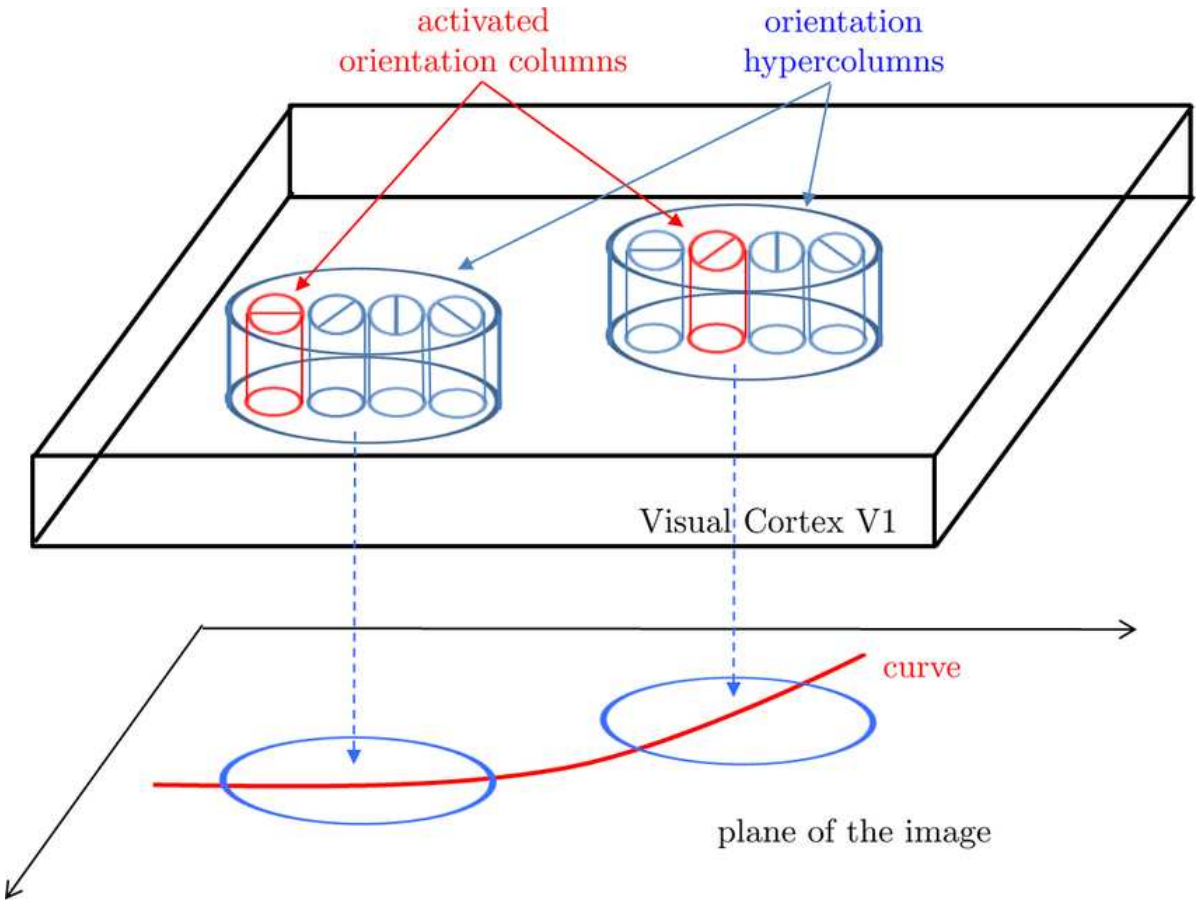
Receptive Fields

Simple and complex cells perceive directions:



Notice: simple and complex cells behave as filters in deep learning!

Orientation Hypercolumn



Orientation Hypercolumn: ice cube model

Hypercolumns

- Together, orientation columns and ocular dominance columns form hypercolumns.
- 18-20 columns to represent all orientations for both eyes
  - about 1 mm square

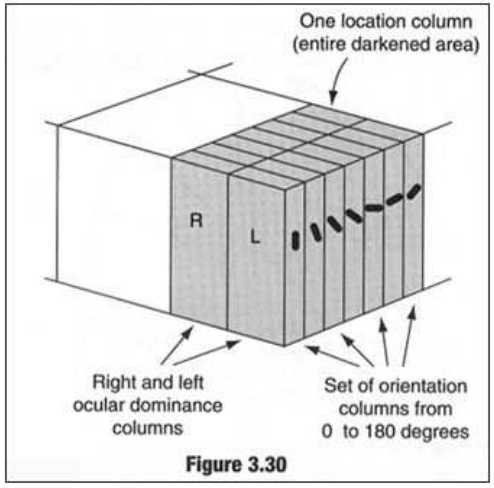
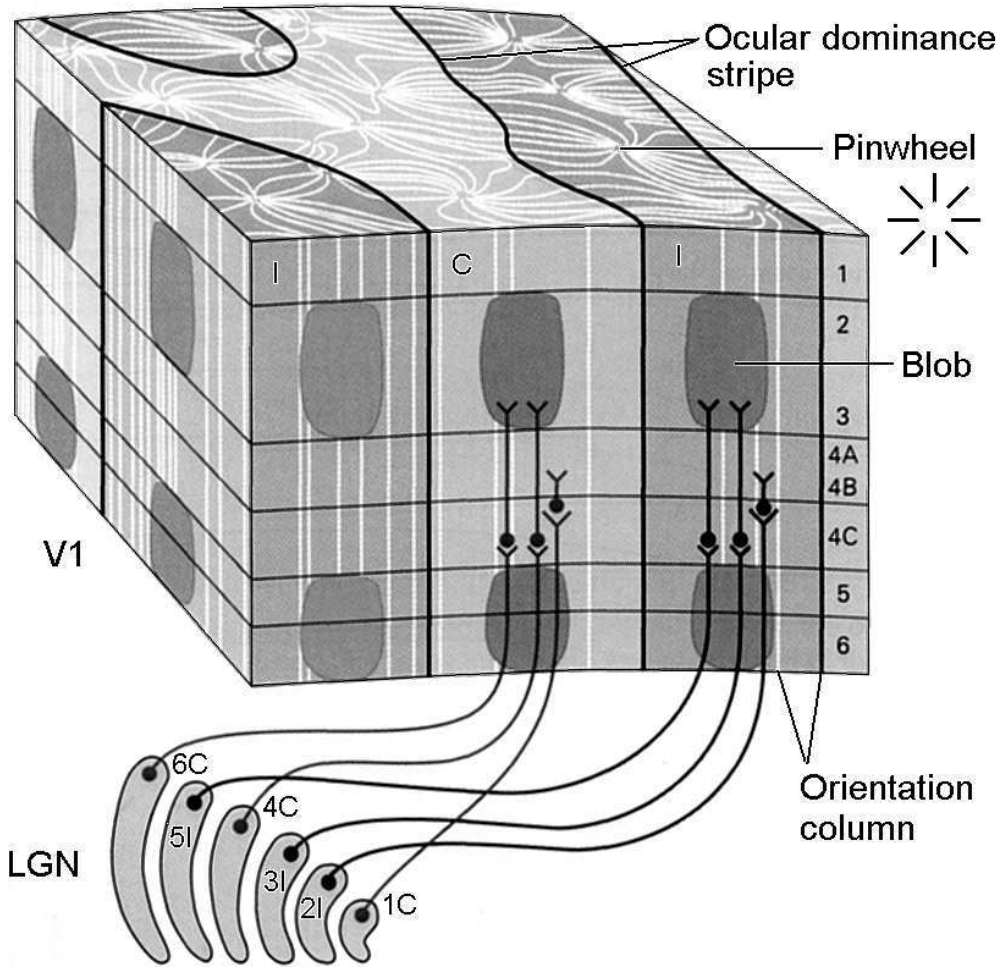


Figure 3.30

At each point we see information regarding **all** possible directions.

Orientation Hypercolumn: V1 structure/1



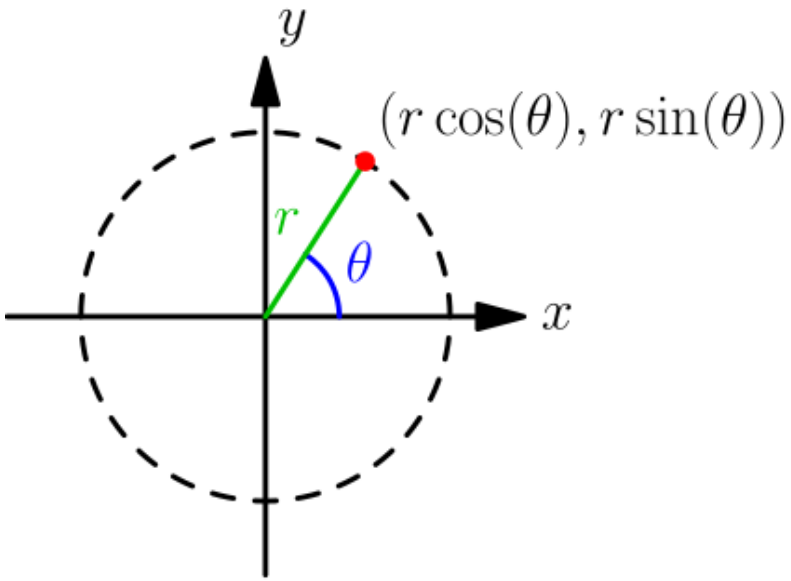
Mathematical model: fiber bundles

Edges as **critical paths** in the S^1-bundle structure

Model for the visual cortex V as lattice of hypercolumns: we take $V \subset \mathbb{R}^2$. But **at any point we have all directions**!

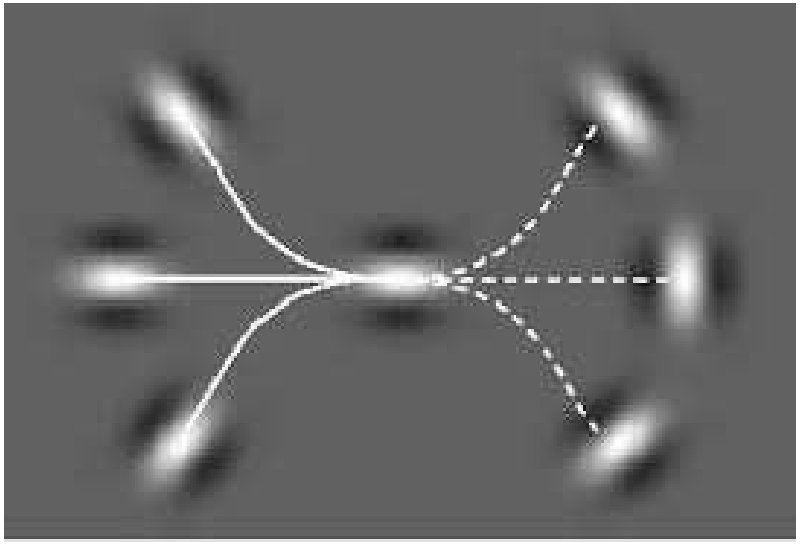
$$\mathbb{R}^2 \times S^1 \longrightarrow \mathbb{R}^2, \qquad (x, y), \theta \mapsto (x, y)$$

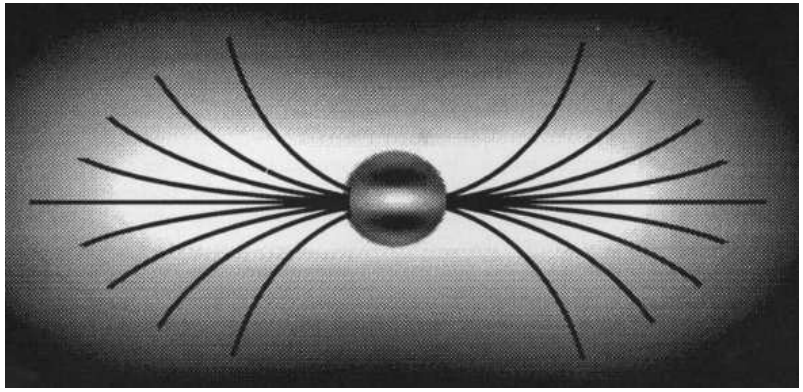
S^1 is the circle in the plane: it contains all the directions and it is parametrized by the angle θ.



Boundary completion: association fields

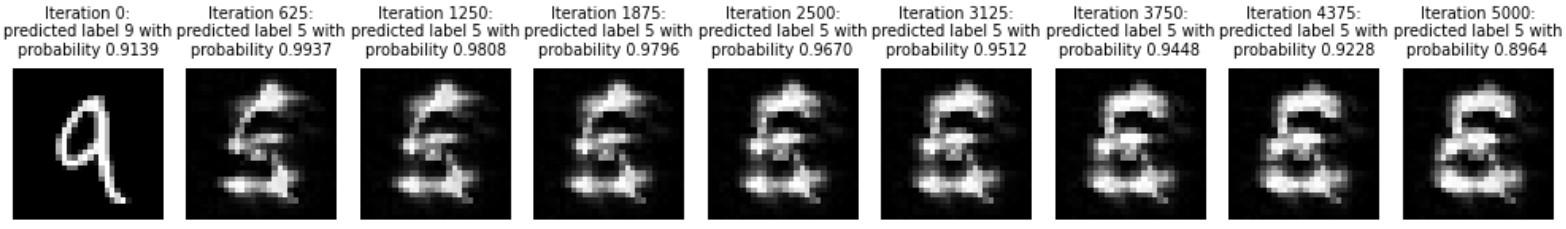
Functional association fields Field, Heyes, Hess - *Contour integration by the human visual system*, 1993





Moving around on the data leaf: MNIST

We can connect a digit from MNIST to a symbol **not** in MNIST moving on the **same** data leaf:



Model for V1 via S^1-fiber bundle

Global S^1 fiber bundle on $\mathbb{R}^2$:

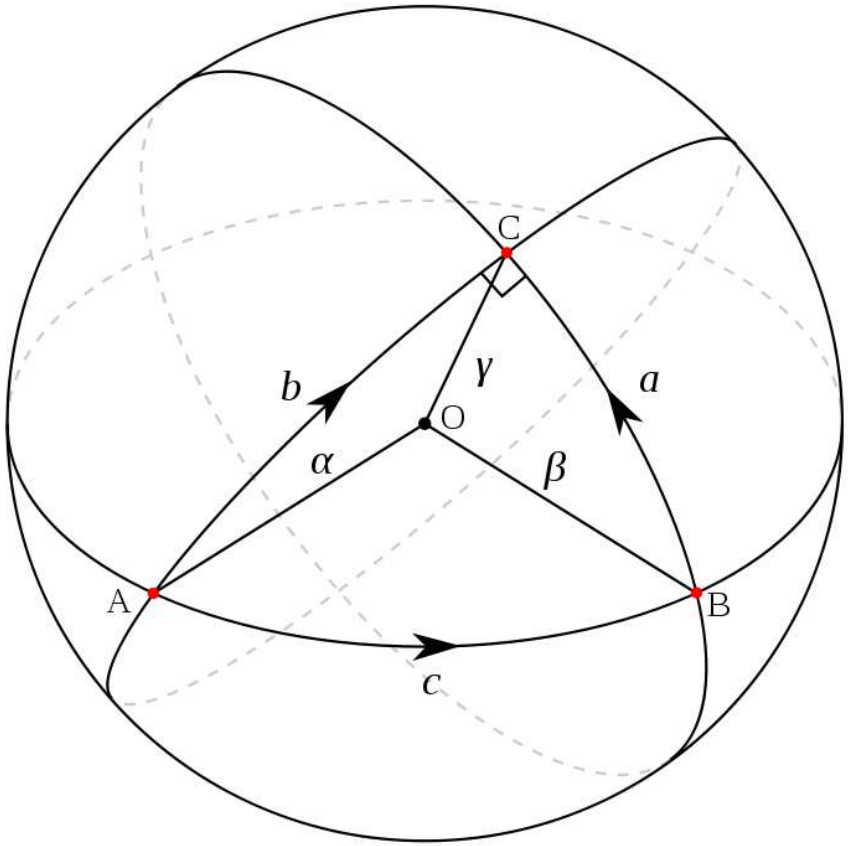
$$\mathbb{R}^2 \times S^1 \longrightarrow \mathbb{R}^2, \qquad (x, y), \theta \mapsto (x, y)$$

At each point (x, y) of the visual cortex V1 we have three main info:

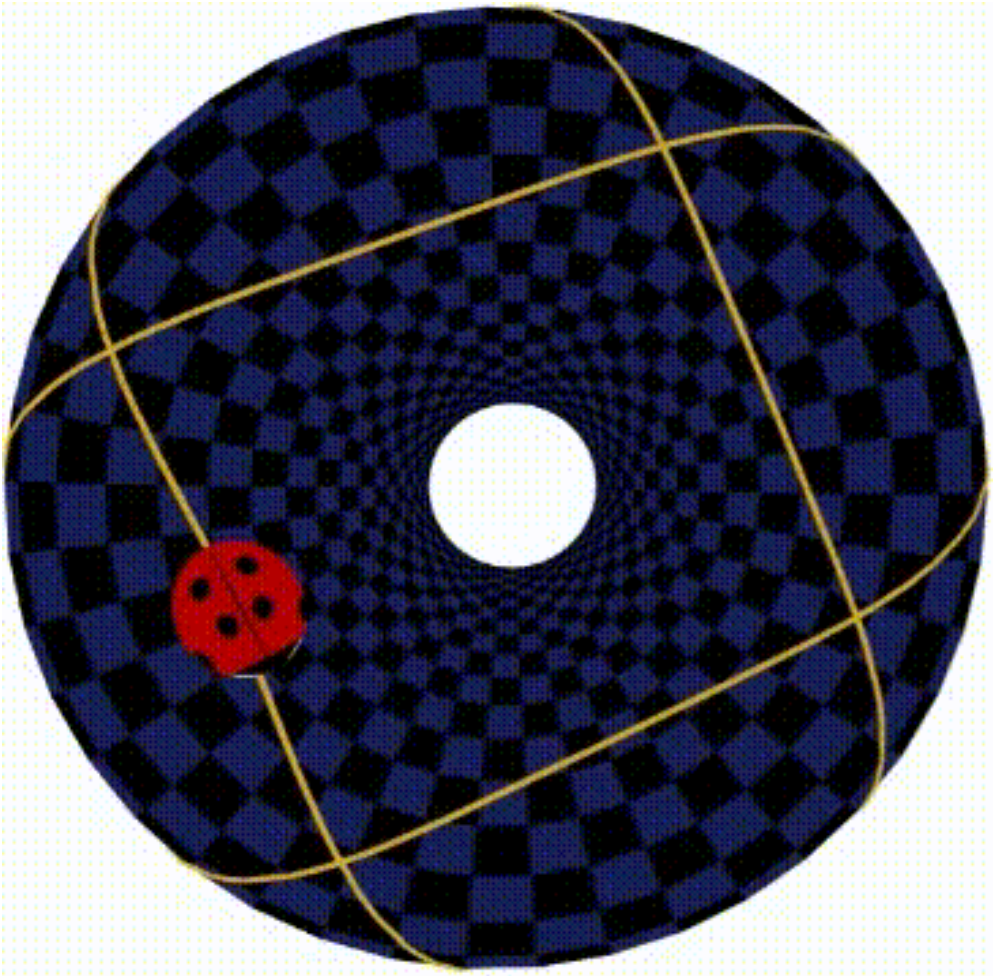
- absolute position of the correspondent of the point (x, y) in the retina;
- orientation θ of some edge at (x, y) (simple and complex cells);
- curvature k of some edge at (x, y) (hypercomplex cells).

Geodesics in Riemannian geometry/1

The geodesic problem: find shortest path on a surface (or manifold).



Geodesics in Riemannian geometry/2

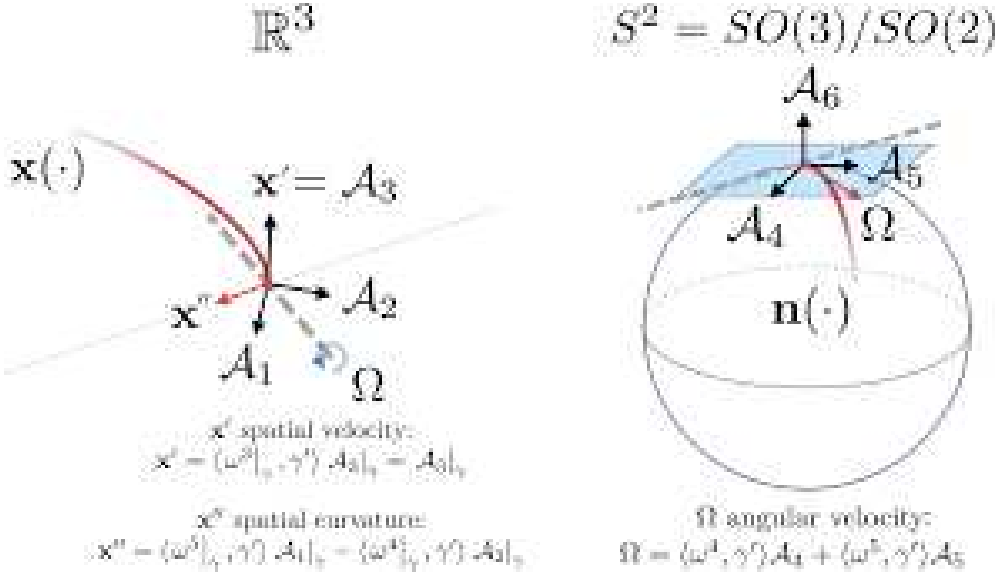


Geodesic equation (via Euler-Lagrange equations):

$$\frac{d^2x^\lambda}{dt^2} + \Gamma^\lambda_{\mu\nu}\frac{dx^\mu}{dt}\frac{dx^\nu}{dt} = 0,$$

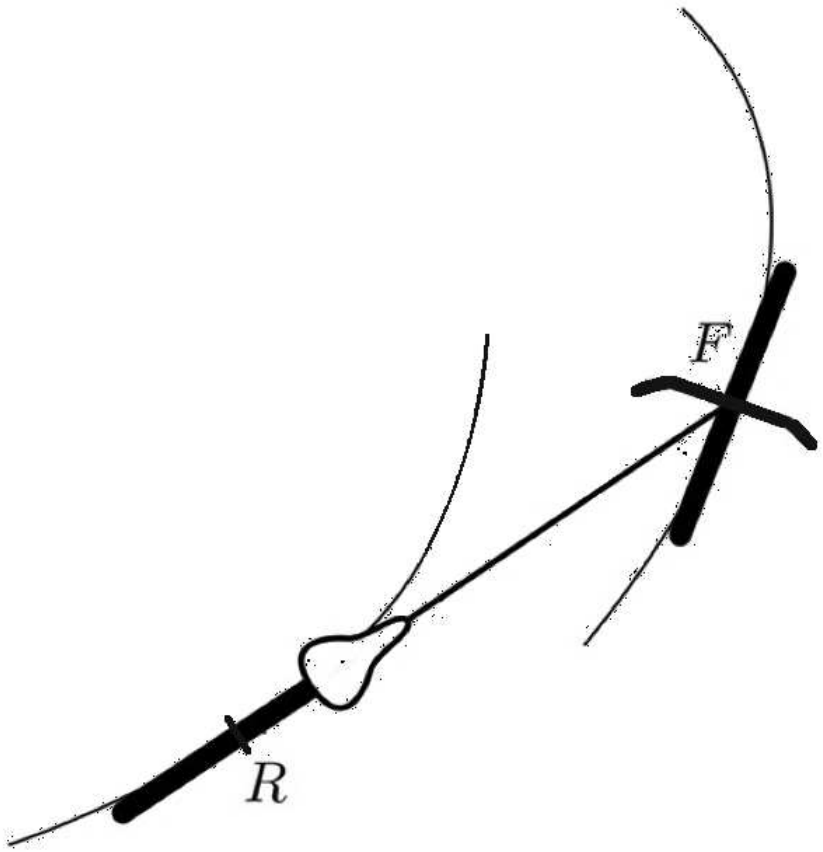
Geodesics in SUB-Riemannian geometry

The geodesic problem: find shortest path on a surface (or manifold) with velocity belonging to a certain subspace.



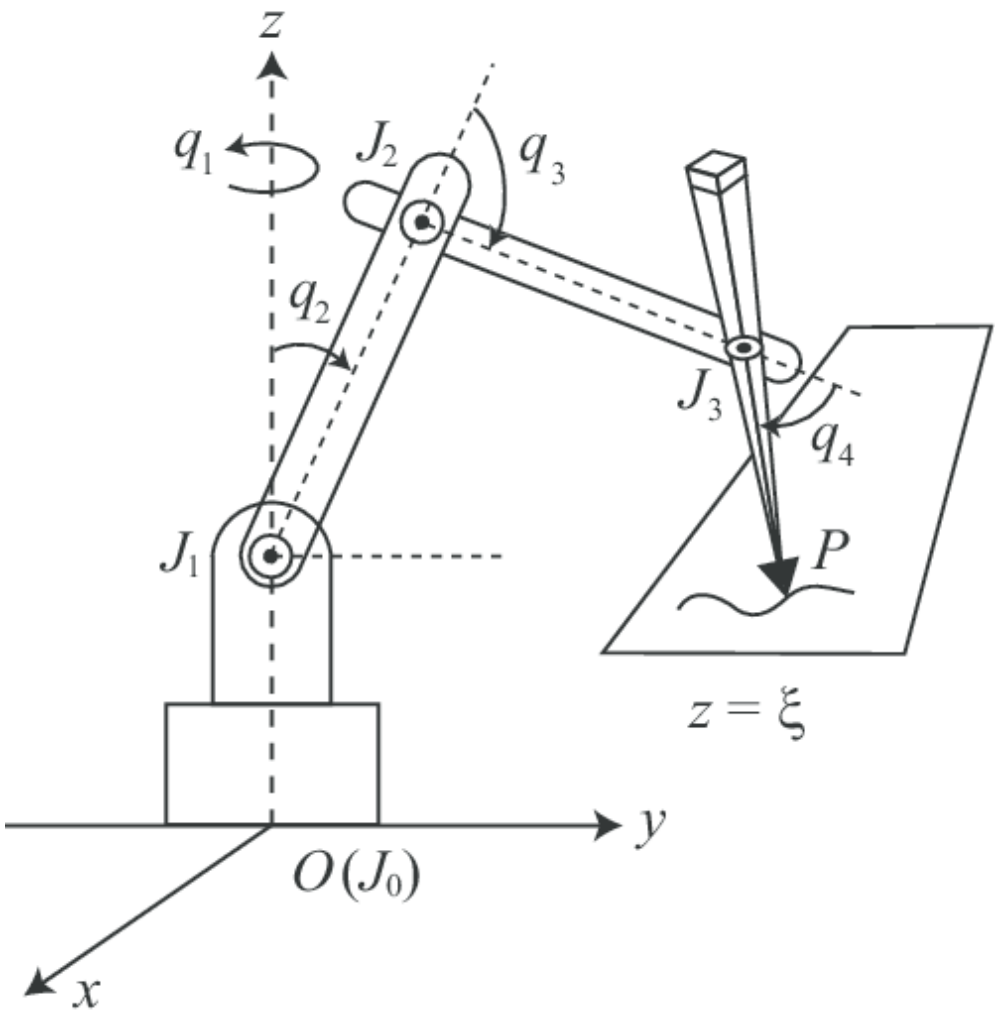
Sub-Riemannian metric: examples/1

On a bicicle we have a constraint on the direction to take **not** the path.



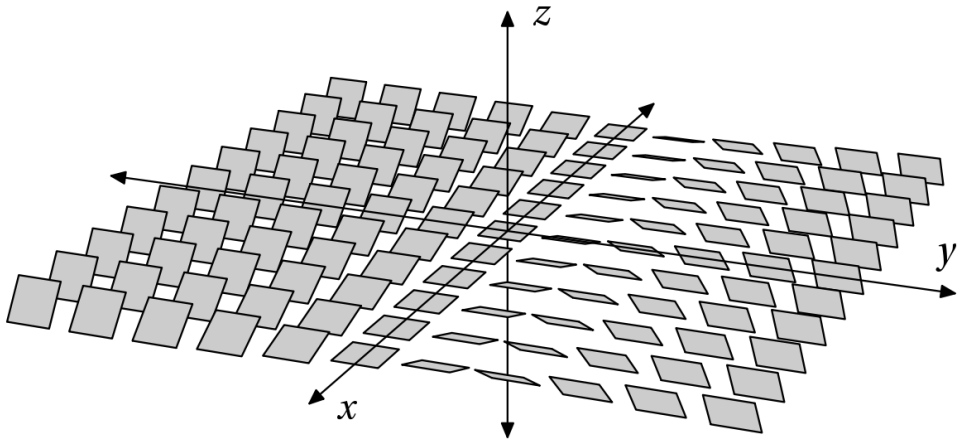
Sub-Riemannian metric: examples/2

A robotic arm can draw curves, but it has constraints on the directions (tangent) of movement.



Distribution

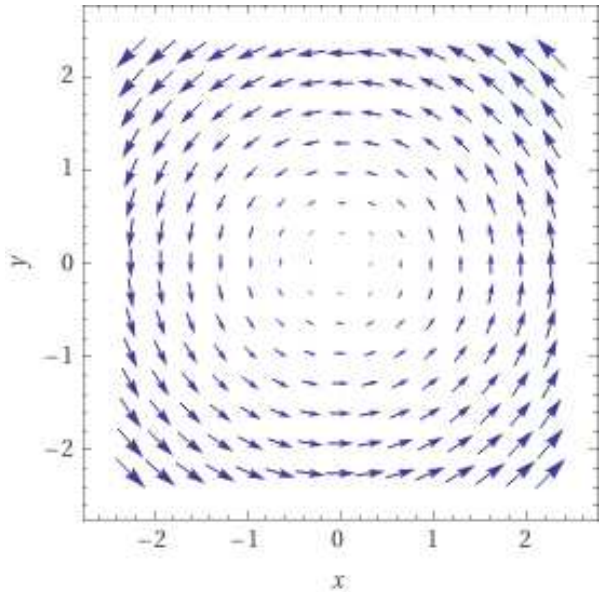
Distribution: $(x, y, z) \mapsto \mathcal{D}_{(x,y,z)}$

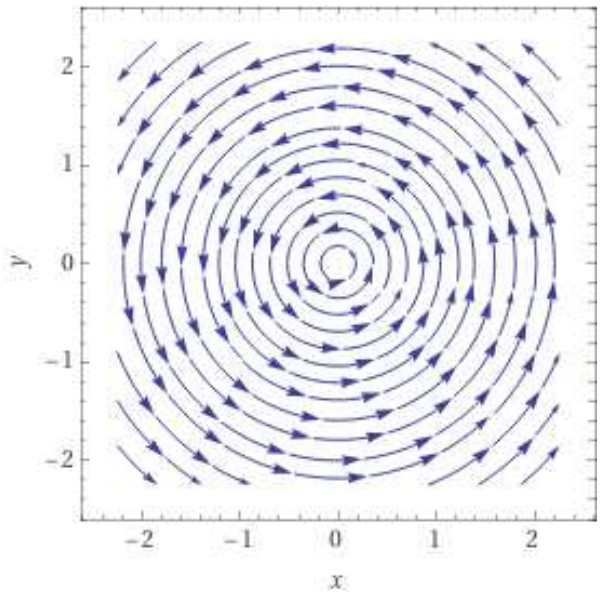


$\mathcal{D}_{(x,y,z)}$ is a subspace in $\mathbb{R}^3$: line or plane.
A subriemannian metric expresses distances ON the distribution only not in the whole space.

Distribution and vector fields

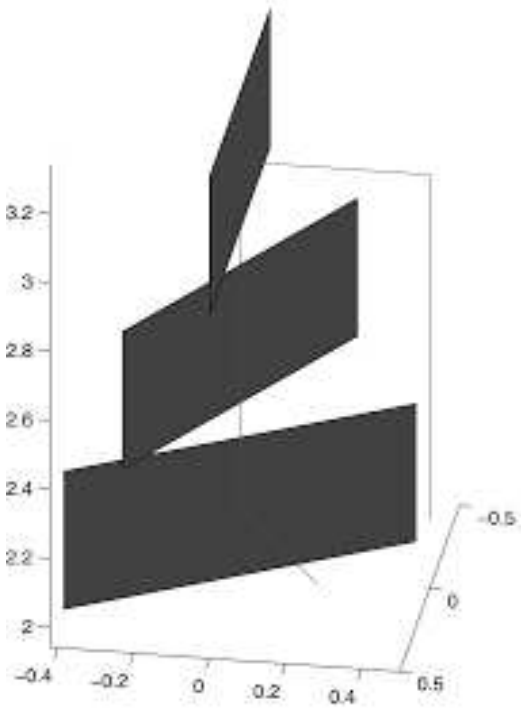
Vector fields in R^2: ad each point (x, y) we have a vector $X_{(x,y)}$.
Example: $(x, y) \mapsto -y\partial_x + x\partial_y$

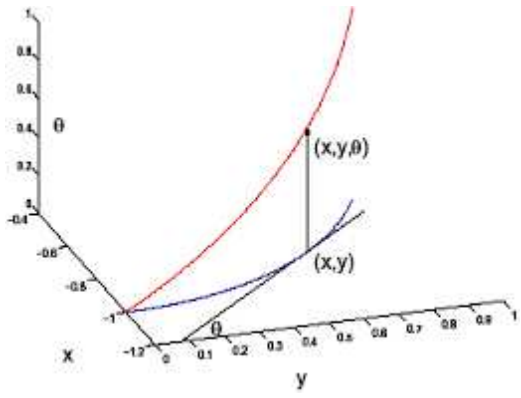




Sub-Riemannian metric

Idea: we build a geodesic on the whole space according to some metric and then we **project** it on the distribution.





How do we find the distribution?

Anatomical model: orientation percept construction

1. **Orientation column activation:** local function with fixed orientation preference θ

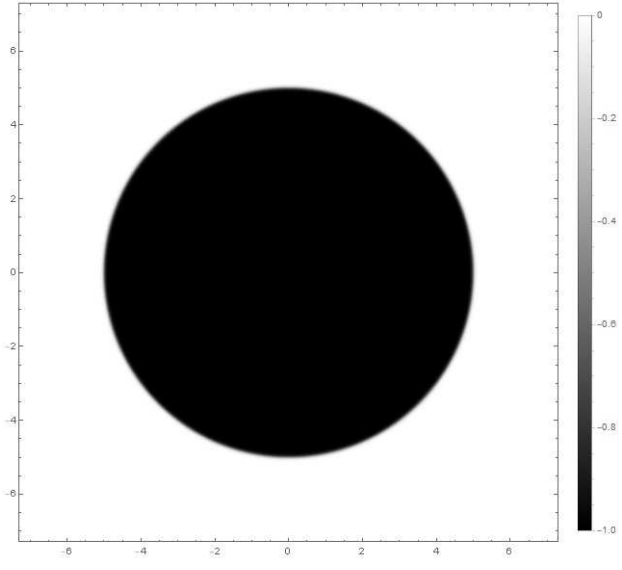
$$\begin{array}{rccc} X_3(\theta)\,\mathcal{R}: & V & \longrightarrow & \mathbb{R} \\ & (x,y) & \longmapsto & [-\sin\theta\,\partial_x + \cos\theta\,\partial_y]\,\mathcal{R}(x,y) \end{array}$$

   where we identify retina=gangli pointwise=V1 domain in $\mathbb{R}^2$.

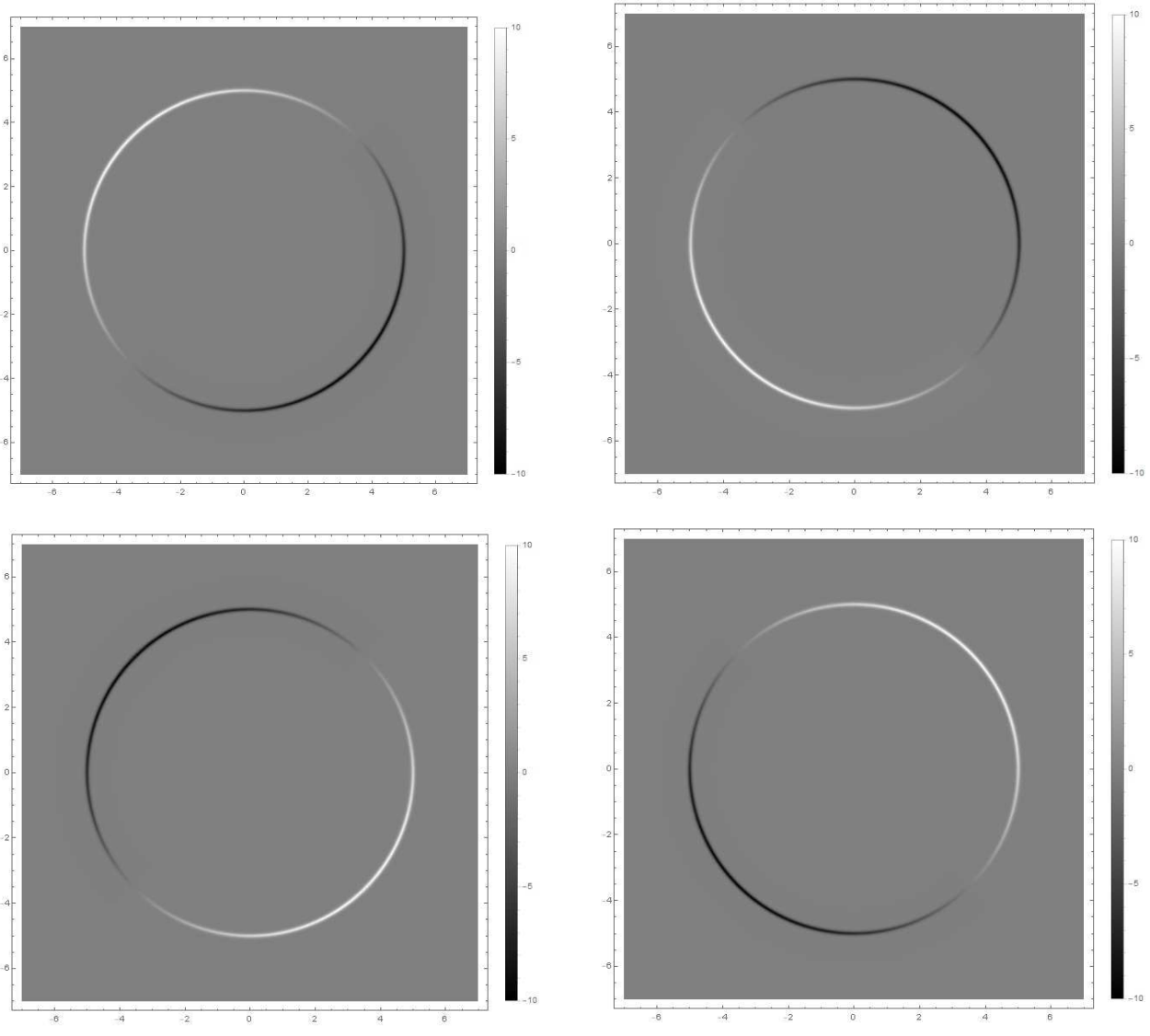
2. **Hypercolumnar orientation detection**: scalar function

$$\begin{array}{rccc} \Theta: & V & \longrightarrow & \mathbb{R} \\ & (x,y) & \longmapsto & \Theta(x,y) := \mathrm{argmax}_{\theta\in[0,2\pi]}\{X_3(\theta)\mathcal{R}(x,y)\} \end{array}$$

Anatomical model: orientation percept construction



$\mathcal{R}(x, y)$



$X_3(\theta)\mathcal{R}(x, y), \quad \theta = \frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}$

Anatomical model: orientation percept construction

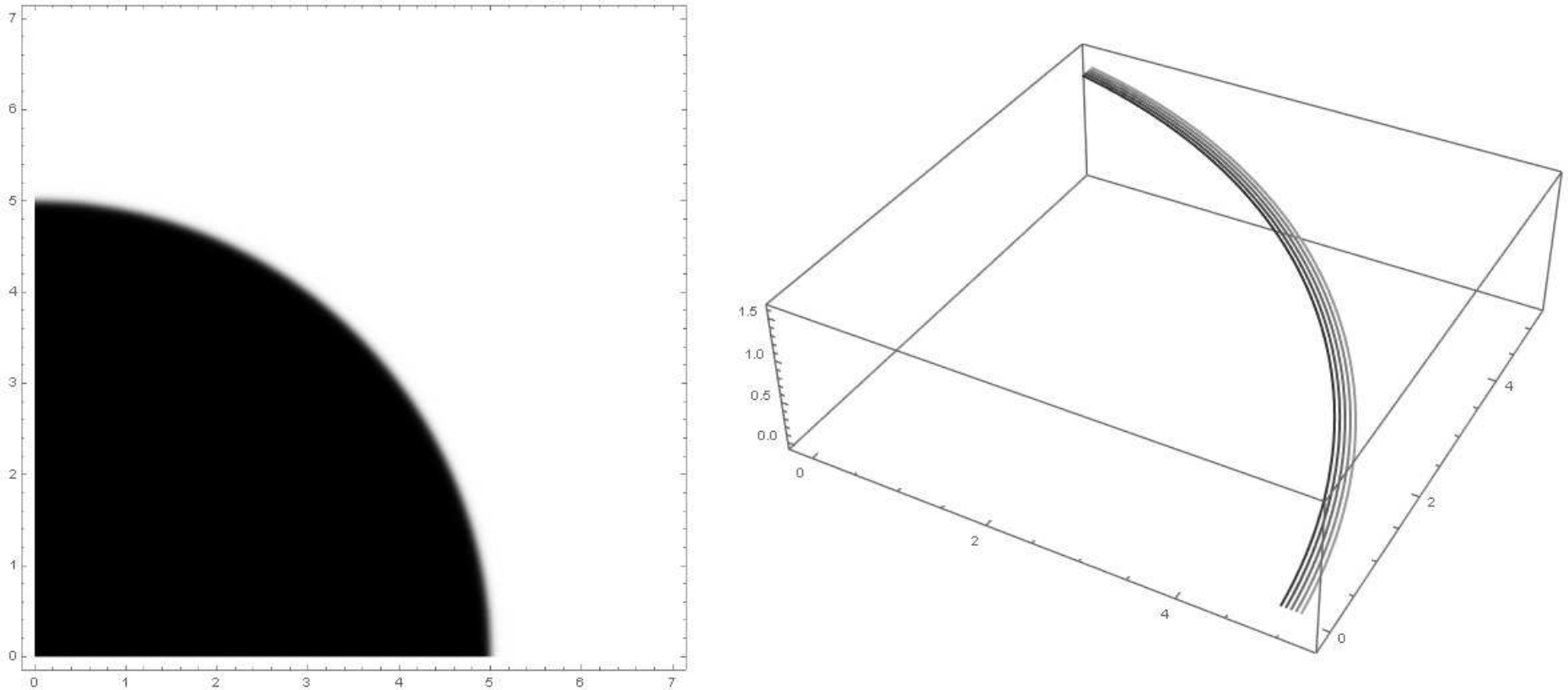


Figure: Lift of the $\mathcal{R}$ regular sets into $\mathbb{R}^2 \times [0, 2\pi]$

Distribution in $R^2 \times S^1 \subset \mathbb{R}^3$

$$(x, y, \theta) \mapsto \mathcal{D}_{(x,y,\theta)} = \operatorname{span} \left\{ \begin{array}{l} X_1 = \cos\,\theta\,\partial_x + \sin\,\theta\,\partial_y \\ X_2 = \partial_\theta \end{array} \right\}$$

We want to find curves $\gamma(t)$ that are **tangent** to the distribution:

$$\gamma'(t) \in \operatorname{span} \left\{ \begin{array}{l} X_1 = \cos\,\theta\,\partial_x + \sin\,\theta\,\partial_y \\ X_2 = \partial_\theta \end{array} \right\}$$

They will be the geodesics in a subriemannian metric!
How to find them: Hamilton equations!

Sub-riemannian geodesics

Hamilton Equations for Subriemannian geodesics:

$$\begin{cases} \dot{x} = \cos\theta\, p_1 & \dot{p}_1 = p_3\, p_1 \\ \dot{y} = \sin\theta\, p_1 & \dot{p}_2 = -p_3\, p_1 \\ \dot{\theta} = p_2 & \dot{p}_3 = 0 \end{cases}$$

Geodesic solutions, with 6 parameters to be determined from the initial conditions

$$x(t) = \int_0^t v\cos(\omega s\phi)\cos(\theta(s))\,ds + x_0$$

$$y(t) = \pm\int_0^t v\cos(\omega s\phi)\sin(\theta(s))\,ds + y_0$$

$$\theta(t) = \mp\tfrac{v}{\omega}\cos(\omega s\phi) + \theta_0$$

Compatibility with visive association fields