Perceptron Algorithm

- Assume for simplicity: all x_i has length 1

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.

Why is this a reasonable strategy?

Intuition: correct the current mistake

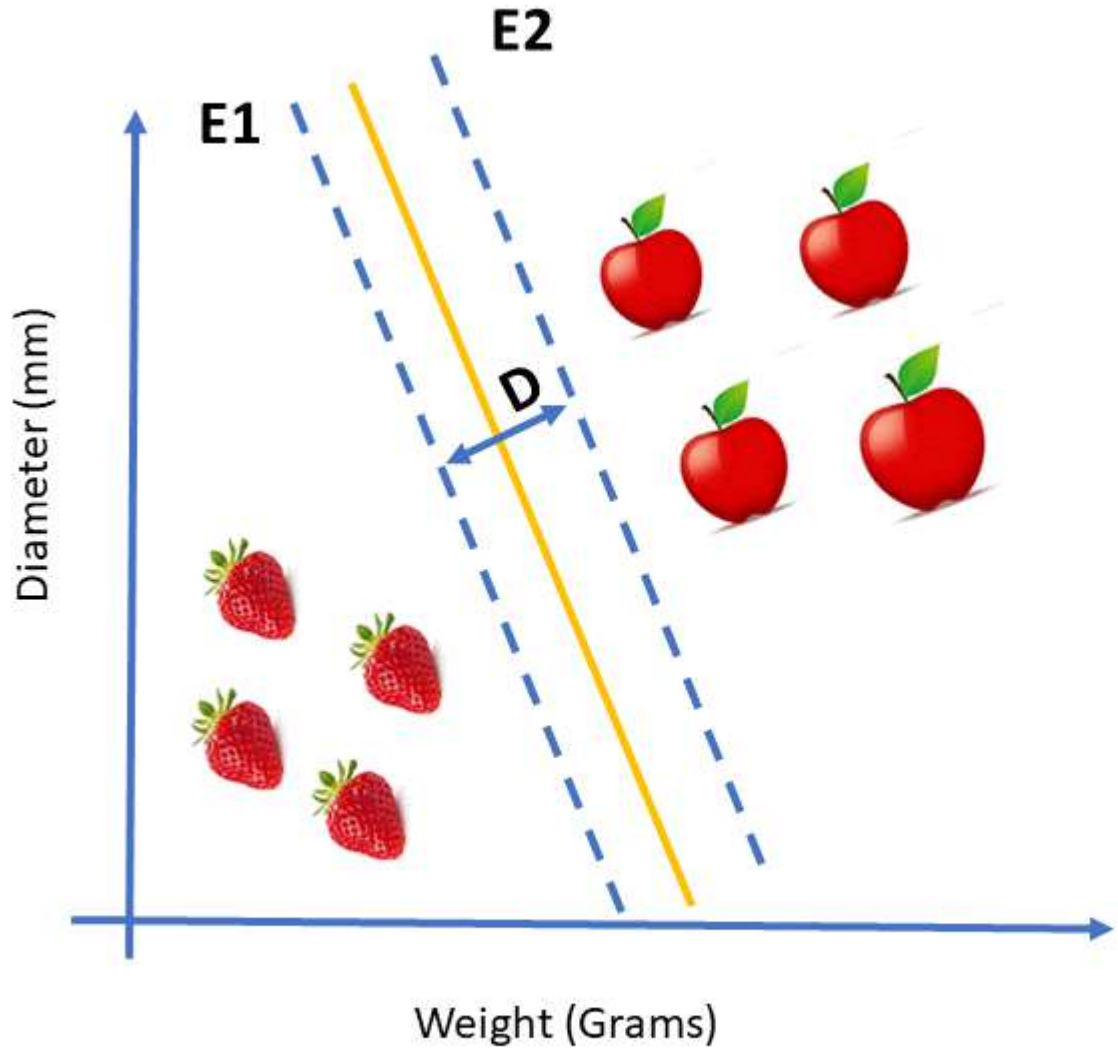
- If mistake on a positive example

$$w_{t+1}^T x = (w_t + x)^T x = w_t^T x + x^T x = w_t^T x + 1$$

- If mistake on a negative example

$$w_{t+1}^T x = (w_t - x)^T x = w_t^T x - x^T x = w_t^T x - 1$$

Beyond the perceptron: Maximum margin



The one that maximizes the distances to the closest data points from both classes.
This is the hyperplane with maximum margin
Let's define more precisely the concept of margin

What is the distance of a point $\mathbf{x}$ to the hyperplane $\mathcal{H}$?

Consider some point $\mathbf{x}$. Let $\mathbf{d}$ be the vector from $\mathcal{H}$ to $\mathbf{x}$ of minimum length. Let $\mathbf{x}^P$ be the projection of $\mathbf{x}$ onto $\mathcal{H}$. It follows then that:

$\mathbf{x}^P = \mathbf{x} - \mathbf{d}$.

$\mathbf{d}$ is parallel to $\mathbf{w}$, so $\mathbf{d} = \alpha\mathbf{w}$ for some $\alpha \in \mathbb{R}$.

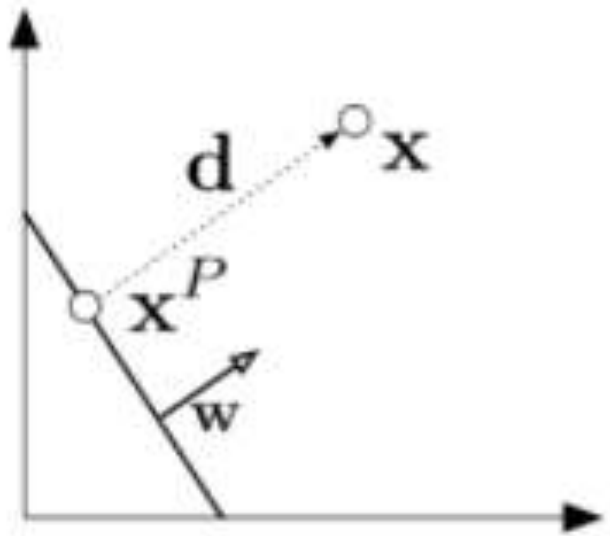
$\mathbf{x}^P \in \mathcal{H}$ which implies $\mathbf{w}^T\mathbf{x}^P + b = 0$

therefore $\mathbf{w}^T\mathbf{x}^P + b = \mathbf{w}^T(\mathbf{x} - \mathbf{d}) + b = \mathbf{w}^T(\mathbf{x} - \alpha\mathbf{w}) + b = 0$

which implies $\alpha = \frac{\mathbf{w}^T\mathbf{x}+b}{\mathbf{w}^T\mathbf{w}}$

The length of $\mathbf{d}$:

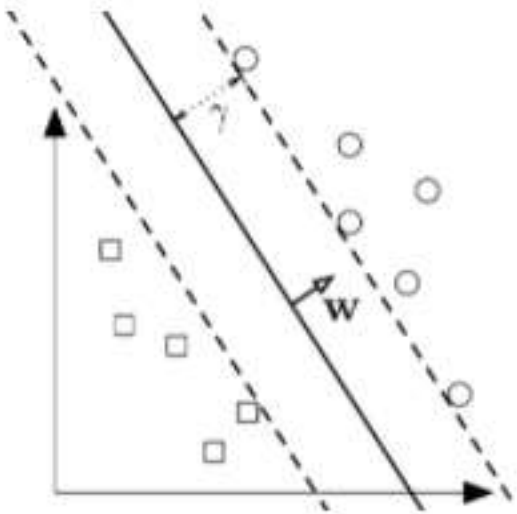
$$\|\mathbf{d}\|_2 = \sqrt{\mathbf{d}^T\mathbf{d}} = \sqrt{\alpha^2\mathbf{w}^T\mathbf{w}} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{\sqrt{\mathbf{w}^T\mathbf{w}}} = \frac{|\mathbf{w}^T\mathbf{x}+b|}{\|\mathbf{w}\|_2}$$



Margin of $\mathcal{H}$ with respect to D: $\gamma(\mathbf{w}, b) = \min_{\mathbf{x}\in D} \frac{|\mathbf{w}^T\mathbf{x}+b|}{\|\mathbf{w}\|_2}$

We can formulate our search for the maximum margin separating hyperplane as a constrained optimization problem. The objective is to maximize the margin under the constraints that all data points must lie on the correct side of the hyperplane:

$$\underbrace{\max_{\mathbf{w},b} \gamma(\mathbf{w}, b)}_{\textit{maximize margin}} \text{ such that } \underbrace{\forall i \; y_i(\mathbf{w}^T x_i + b) \geq 0}_{\textit{separating hyperplane}}$$



If we plug in the definition of γ we obtain:

$$\underbrace{\max_{\mathbf{w},b} \underbrace{\frac{1}{\|\mathbf{w}\|_2} \min_{\mathbf{x}_i \in D} \left|\mathbf{w}^T \mathbf{x}_i + b\right|}_{\gamma(\mathbf{w},b)}}_{\textit{maximize margin}} \quad s.t. \quad \underbrace{\forall i \; y_i(\mathbf{w}^T x_i + b) \geq 0}_{\textit{separating hyperplane}}$$

Because the hyperplane is scale invariant, we can fix the scale of $\mathbf{w}, b$ anyway we want. Let's be clever about it, and choose it such that

$$\min_{\mathbf{x} \in D} \left|\mathbf{w}^T \mathbf{x} + b\right| = 1.$$

We can add this re-scaling as an equality constraint. Then our objective becomes:

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2} \cdot 1 = \min_{\mathbf{w},b} \|\mathbf{w}\|_2 = \min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w}$$

(Where we made use of the fact $f(z) = z^2$ is a monotonically increasing function for $z \geq 0$ and $\|\mathbf{w}\| \geq 0$; i.e. the $\mathbf{w}$ that maximizes $\|\mathbf{w}\|_2$ also maximizes $\mathbf{w}^\top \mathbf{w}$.)

The new optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{w},b} \mathbf{w}^\top \mathbf{w} \\ \text{s.t.} \quad \begin{aligned} \forall i,\ y_i(\mathbf{w}^T\mathbf{x}_i + b) &\geq 0 \\ \min_i \left|\mathbf{w}^T\mathbf{x}_i + b\right| &= 1 \end{aligned} \end{aligned}$$

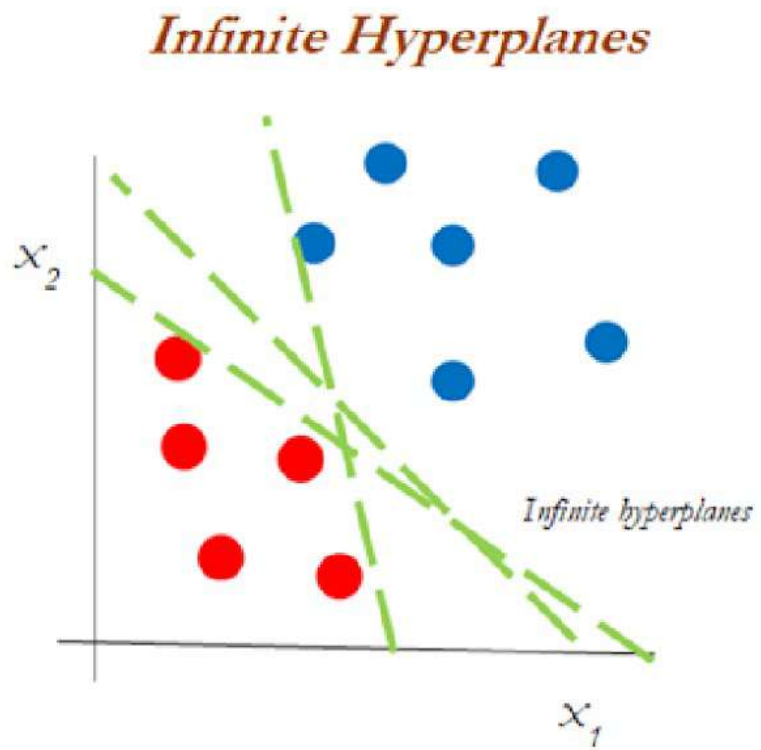
These constraints are still hard to deal with, however luckily we can show that (for the optimal solution) they are equivalent to a much simpler formulation. (Makes sure you know how to prove that the two sets of constraints are equivalent.)

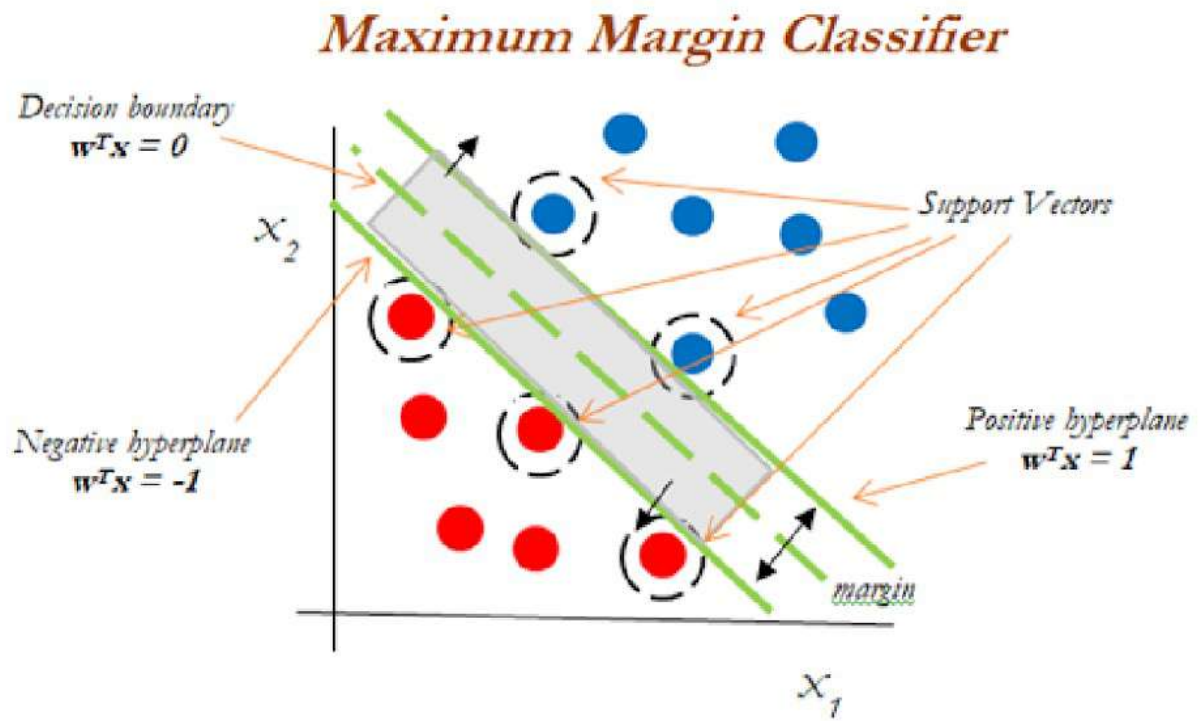
$$\begin{aligned} &\min_{\mathbf{w},b} \mathbf{w}^T\mathbf{w} \\ &\text{s.t.} \quad \forall i\ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{aligned}$$

Support vectors

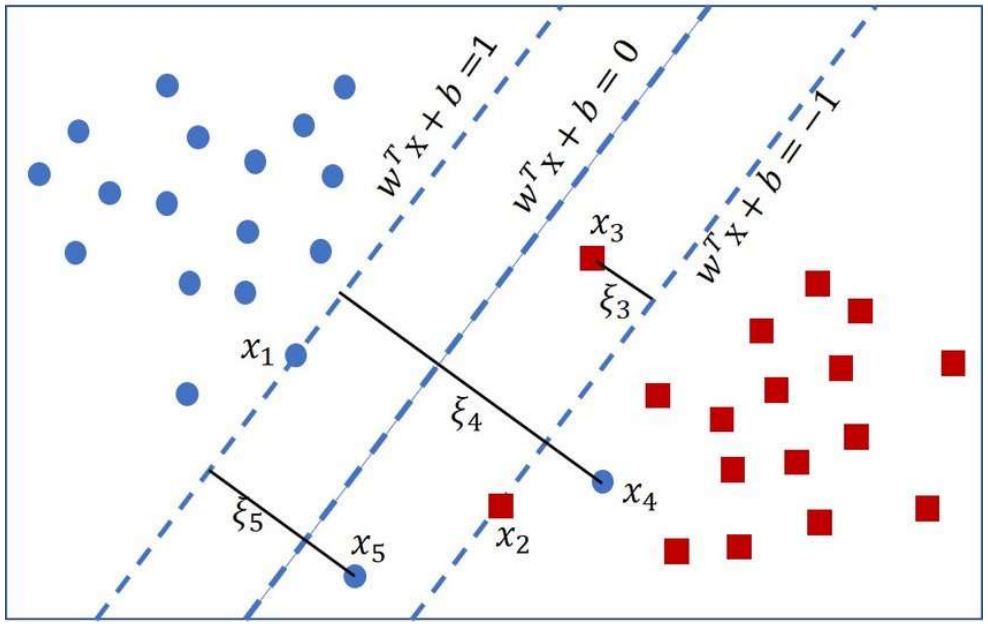
For the optimal $\mathbf{w}, b$ pair, some training points will have tight constraints, i.e.

$$y_i(\mathbf{w}^T\mathbf{x_i} + b) = 1.$$





Maximal margin classification and limitations



If the data is low dimensional it is often the case that there is no separating hyperplane between the two classes. In this case, there is no solution to the optimization problems stated above. We can fix this by allowing the constraints to be violated ever so slight with the introduction of slack variables:

$$\min_{\mathbf{w},b} \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i$$
$$s.t.\ \forall i\ y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$$
$$\forall i\ \xi_i \geq 0$$

The slack variable ξ_i allows the input $\mathbf{x_i}$ to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack". If C is very large, the SVM becomes very strict and tries to get all points to be on the right side of the hyperplane. If C is very small, the SVM becomes very loose and may "sacrifice" some points to obtain a simpler (i.e. lower $\|\mathbf{w}\|_2^2$) solution.

New optimization problem

Let us consider the value of ξ_i for the case of $C \neq 0$. Because the objective will always try to minimize ξ_i as much as possible, the equation must hold as an *equality* and we have:

$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) & \text{if } y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1 \\ 0 & \text{if } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{cases}$$

This is equivalent to the following closed form:

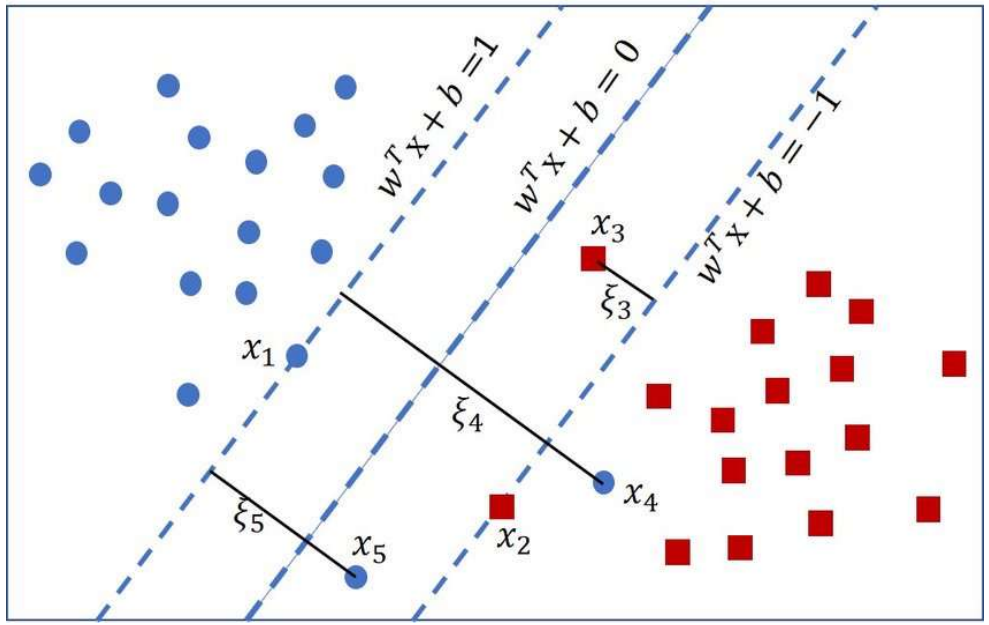
$$\xi_i = \max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0).$$

If we plug this closed form into the objective of our SVM optimization problem, we obtain the following *unconstrained* version as loss function and regularizer:

$$\min_{\mathbf{w},b} \underbrace{\mathbf{w}^T\mathbf{w}}_{l_2-regularizer} + C \sum_{i=1}^{n} \underbrace{\max\left[1 - y_i(\mathbf{w}^T\mathbf{x} + b), 0\right]}_{hinge-loss}$$

This formulation allows us to optimize the SVM paramters $(\mathbf{w}, b)$ just like logistic regression (e.g. through gradient descent). The only difference is that we have the **hinge-loss** instead of the **logistic loss**.

Solving for the new optimization problem

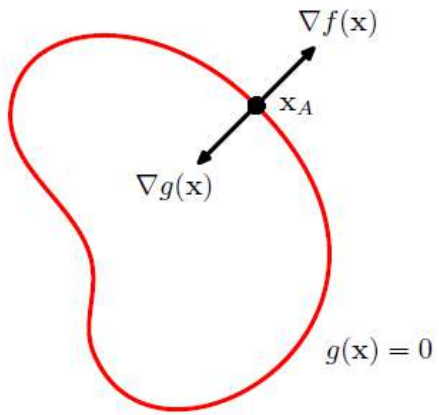


slack penalty

$$\min_{\substack{w \in \mathbb{R}^d \\ \xi \geq 0}} \quad \frac{1}{2} w^\top w + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i \qquad \forall i \in \{1, \ldots, n\}$$

Lagrange Multipliers



Consider the problem:

$$\max_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$s.t. \quad g(\boldsymbol{x}) = 0$$

- Points on $g(\boldsymbol{x}) = 0$ must have $\nabla g(\boldsymbol{x})$ normal to surface
- A stationary point must have no change in f in the direction of the constraint surface, so $\nabla f(\boldsymbol{x})$ must also be normal to the surface.
  - So there must be some $\lambda \neq 0$ such that $\nabla f(\boldsymbol{x}) + \lambda \nabla g(\boldsymbol{x}) = 0$
- Define Lagrangian:

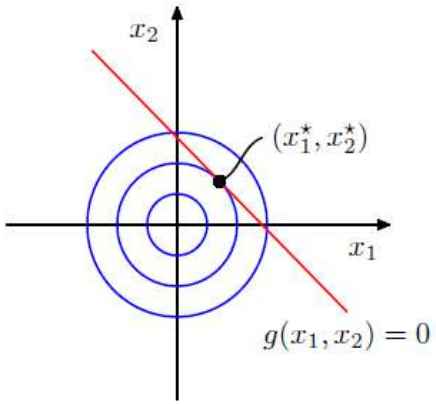
$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$$

- Stationary points of $L(\boldsymbol{x}, \lambda)$ have $\nabla_x L(\boldsymbol{x}, \lambda) = \nabla f(\boldsymbol{x}) + \lambda \nabla g(\boldsymbol{x}) = 0$ and $\nabla_\lambda L(\boldsymbol{x}, \lambda) = g(\boldsymbol{x}) = 0$

This is because on the curves g is constant

This is because they are the same vector up to a multiplicative constant

Lagrange Multipliers Example



- Consider the problem

$$\max_{\boldsymbol{x}} f(x_1, x_2) = 1 - x_1^2 - x_2^2$$
$$s.t. \quad g(x_1, x_2) = x_1 + x_2 - 1 = 0$$

- Lagrangian:

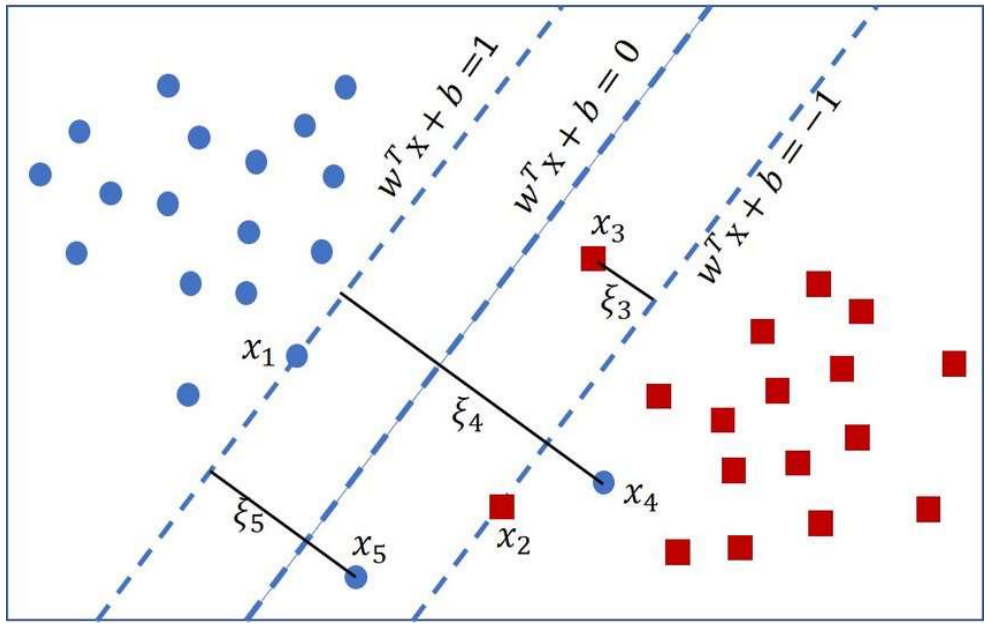
$$L(\boldsymbol{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$$

- Stationary points require:

$$\begin{aligned} \partial L/\partial x_1 &= -2x_1 + \lambda = 0 \\ \partial L/\partial x_2 &= -2x_2 + \lambda = 0 \\ \partial L/\partial \lambda &= x_1 + x_2 - 1 = 0 \end{aligned}$$

- So stationary point is $(x_1^*, x_2^*) = (\frac{1}{2}, \frac{1}{2})$, $\lambda = 1$

Solving for the new optimization problem



slack penalty

$$\min_{\substack{w \in \mathbb{R}^d \\ \xi \geq 0}} \quad \frac{1}{2} w^\top w + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i \qquad \forall i \in \{1, \ldots, n\}$$

The Lagrangian: $L(w, b, \xi, \alpha, \lambda)$

$$= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(w^\top x_i + b\right) - \xi_i\right) + \sum_{i=1}^{n}\lambda_i(-\xi_i)$$

with dual variable constraints

$$\alpha_i \geq 0, \qquad \lambda_i \geq 0.$$

Minimize wrt the primal variables w, b, and ξ.

Derivative wrt w:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \qquad w = \sum_{i=1}^{n} \alpha_i y_i x_i.$$

Derivative wrt b:

$$\frac{\partial L}{\partial b} = \sum_{i} y_i \alpha_i = 0.$$

Derivative wrt ξ_i:

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0 \qquad \alpha_i = C - \lambda_i.$$

Noting that $\lambda_i \geq 0$,

$$\alpha_i \leq C.$$

$$L(w, b, \xi, \alpha, \beta) = \frac{1}{2} w^\top w + C \sum_i \xi_i - \sum_i \alpha_i (y_i (w^\top x_i + b) - 1 + \xi_i) - \sum_i \beta_i \xi_i$$

Gradients

$$\nabla_w L = w - \sum_i \alpha_i y_i x_i = 0$$

$$\nabla_b L = -\sum_i \alpha_i y_i = 0$$

$$\nabla_\xi L = C - \alpha - \beta = 0$$

Consequences

$$w = \sum_i \alpha_i y_i x_i$$

$$\sum_i \alpha_i y_i = 0$$

$$\alpha = C - \beta$$

$$\beta = C - \alpha$$

w is a linear combination of the training points!

Rewriting the Lagrangian…

$$
\begin{aligned}
L(\alpha,\lambda) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(w^\top x_i + b\right) - \xi_i\right) \\
&\quad + \sum_{i=1}^{n}\lambda_i(-\xi_i) \\
&= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^\top x_j + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^\top x_j \\
&\quad - b\underbrace{\sum_{i=1}^{m}\alpha_i y_i}_{0} + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i\xi_i - \sum_{i=1}^{m}\underbrace{(C-\alpha_i)}_{\lambda_i}\xi_i \\
&= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^\top x_j.
\end{aligned}
$$

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^\top x_j,$$

subject to the constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

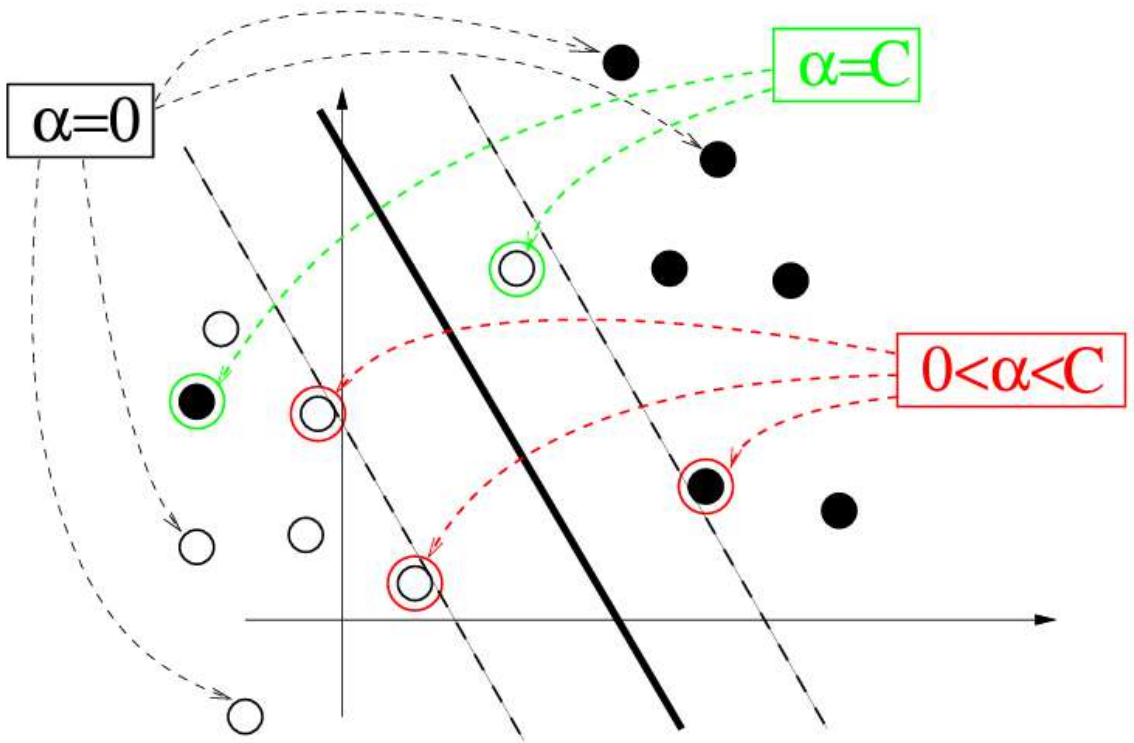
Support vectors

- The solution w can be written as a linear combination of the data
- Only those data for which $0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$ are counted for the solution

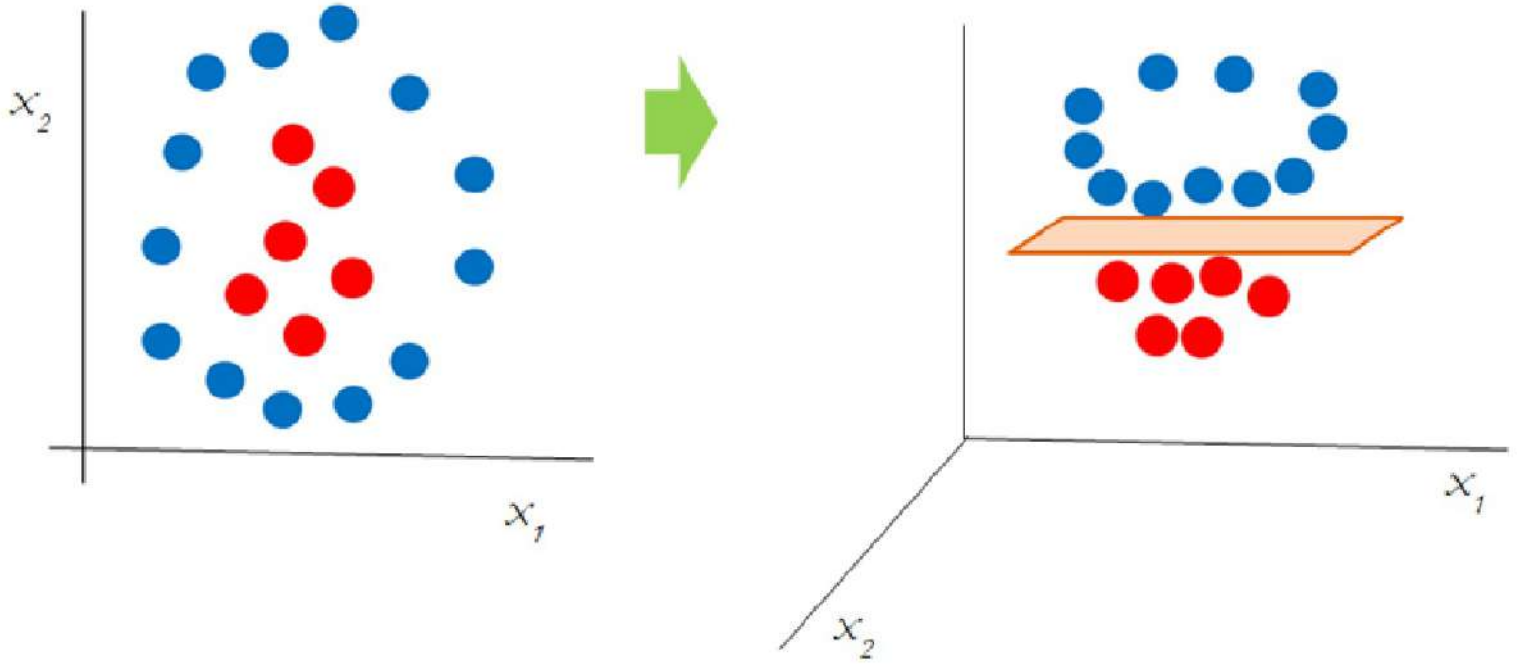
Support vectors

$$\min_{\substack{w \in \mathbb{R}^d \\ \xi \geq 0}} \quad \frac{1}{2} w^\top w + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i$$



w is given by a linear combination only of the points that satisfy the inequality!



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix}$$

$$f(x) = w^T x \quad \rightarrow \quad f(x) = w^T \phi(x)$$

- A problem that is not linearly separable in 2 dim becomes separable in 3 dim.
- We know a good coordinate change; what's the problem with this?

Polynomial kernel

For $x, z \in [0,1]$ and $0 < \alpha < 1$

$$k(x,z) = \frac{1}{1-\alpha^2 xz}$$

Proof

$$\frac{1}{1-\alpha^2 xz} = \sum_{s=0}^{\infty} (\alpha^2 xz)^s = \Phi(x)^\top \Phi(z)$$

with

$$\Phi(x) = (1, \alpha x, \alpha^2 x^2, \alpha^3 x^3, \ldots)$$

Gaussian kernel

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}||x_i - x_j||^2\right) \qquad \sigma = 1/\sqrt{2}$$

$$= \exp\left(-||x_i - x_j||^2\right)$$

$$= \exp\left(-x_i^\top x_i\right) \exp\left(-x_j^\top x_j\right) \exp(2x_i^\top x_j) \qquad \exp(z) = \sum_{n=0}^{\infty} \frac{z^n}{n!}$$

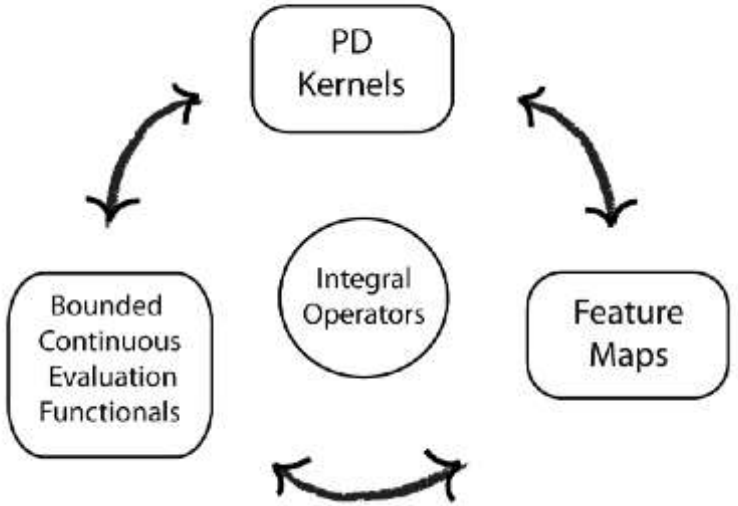
$$= \exp\left(-x_i^\top x_i\right) \exp\left(-x_j^\top x_j\right) \sum_{n=0}^{\infty} \frac{2^n (x_i^\top x_j)^n}{n!}$$

order-n polynomial kernel* $\Phi^n(x)$

$$\Phi^{\text{rbf}} = \exp(-x^\top x)\left[\Phi^1(x)^\top, \Phi^2(x)^\top, \ldots, \Phi^\infty(x)^\top\right]^\top$$

Class 3: Math of kernel methods

- Aronszajn theorem (positive definite maps and features)
- Reproducing property. Reproducing kernel Hilbert spaces (RKHS)
- Pointwise continuity



Some slides from : Julien Mairal

https://members.cbio.mines-paristech.fr/~jvert/svn/kernelcourse/slides/master2017/master2017.pdf

Can we start from the kernel instead of features? What defines a good kernel?

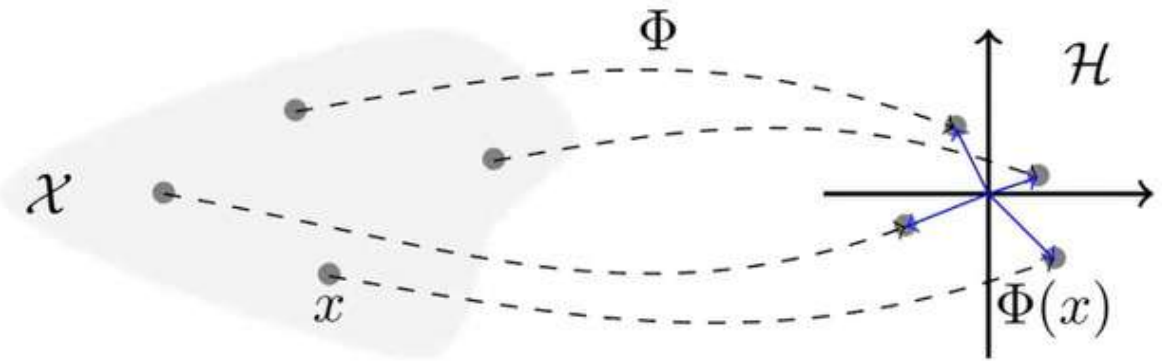
Theorem (Aronszajn, 1950)

K is a p.d. kernel on the set $\mathcal{X}$ if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H}$$

such that, for any $\mathbf{x}, \mathbf{x}'$ in $\mathcal{X}$:

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}} .$$



Def1: Hilbert space

- An inner product on an $\mathbb{R}$-vector space $\mathcal{H}$ is a mapping $(f, g) \mapsto \langle f, g \rangle_{\mathcal{H}}$ from $\mathcal{H}^2$ to $\mathbb{R}$ that is bilinear, symmetric and such that $\langle f, f \rangle_{\mathcal{H}} > 0$ for all $f \in \mathcal{H} \backslash \{0\}$.
- A vector space endowed with an inner product is called pre-Hilbert. It is endowed with a norm defined as $\| f \|_{\mathcal{H}} = \langle f, f \rangle_{\mathcal{H}}^{\frac{1}{2}}$.
- A Cauchy sequence $(f_n)_{n \geq 0}$ is a sequence whose elements become progressively arbitrarily close to each other:

$$\lim_{N \to +\infty} \sup_{n,m \geq N} \|f_n - f_m\|_{\mathcal{H}} = 0.$$

- A Hilbert space is a pre-Hilbert space complete for the norm $\|.\|_{\mathcal{H}}$. That is, any Cauchy sequence in $\mathcal{H}$ converges in $\mathcal{H}$.

Completeness is necessary to keep "good" convergence properties of Euclidean spaces in an infinite-dimensional context.

Def2: Positive definite kernel

Definition. Ω: set. $k: \Omega \times \Omega \to \mathbf{R}$ is a positive definite kernel if

1) (symmetry) $k(x, y) = k(y, x)$

2) (positivity) for arbitrary $x_1, \ldots, x_n \in \Omega$

$$\begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$ is positive semidefinite,

(Gram matrix)

i.e., $\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$ for any $c_i \in \mathbf{R}$ $\qquad c^T K c$

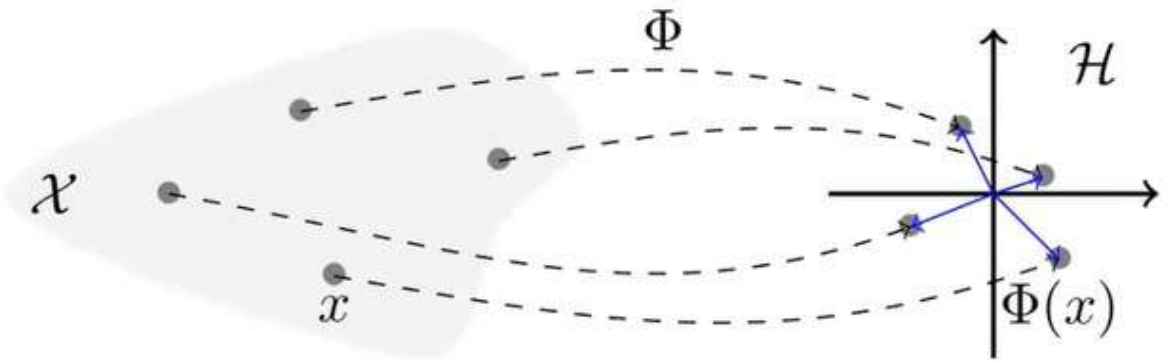
Theorem (Aronszajn, 1950)

K is a p.d. kernel on the set $\mathcal{X}$ if and only if there exists a Hilbert space $\mathcal{H}$ and a mapping

$$\Phi : \mathcal{X} \mapsto \mathcal{H}$$

such that, for any $\mathbf{x}, \mathbf{x}'$ in $\mathcal{X}$:

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}} .$$



The mapping is not unique!

Example 37. Consider $\mathcal{X} = \mathbb{R}$, and

$$k(x,y) = xy = \begin{bmatrix} \frac{x}{\sqrt{2}} & \frac{x}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{y}{\sqrt{2}} \\ \frac{y}{\sqrt{2}} \end{bmatrix},$$

where we defined the feature maps $\phi(x) = x$ and $\tilde{\phi}(x) = \begin{bmatrix} \frac{x}{\sqrt{2}} & \frac{x}{\sqrt{2}} \end{bmatrix}$, and where the feature spaces are respectively, $\mathcal{H} = \mathbb{R}$, and $\tilde{\mathcal{H}} = \mathbb{R}^2$.

What is a good definition of the restriction of the H space?

The reproducing property guarantees the uniqueness

Reproducing kernels Hilbert spaces (RKHS)

$$f(x) = \int f(y)K(x,y)dy$$

Definition

Let $\mathcal{X}$ be a set and $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ be a class of functions forming a (real) Hilbert space with inner product $\langle .,. \rangle_{\mathcal{H}}$. The function $K : \mathcal{X}^2 \mapsto \mathbb{R}$ is called a reproducing kernel (r.k.) of $\mathcal{H}$ if

1. $\mathcal{H}$ contains all functions of the form

$$\forall \mathbf{x} \in \mathcal{X}, \quad K_{\mathbf{x}} : \mathbf{t} \mapsto K(\mathbf{x}, \mathbf{t}) .$$

2. For every $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$ the reproducing property holds:

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}} .$$

If a r.k. exists, then $\mathcal{H}$ is called a reproducing kernel Hilbert space (RKHS).

First a definition (from matrices to operators):

Fix a symmetric function $k : \mathcal{X}^2 \to \mathbb{R}$ on a compact set $\mathcal{X} \subset \mathbb{R}^d$, and consider the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$ defined as

$$T_k f(\cdot) = \int_{\mathcal{X}} k(\cdot, x) f(x)\, dx.$$

We say T_k is positive semidefinite if, for all $f \in L_2(\mathcal{X})$, $\langle f, T_k f \rangle_{L_2(\mathcal{X})} \geq 0$, that is,

$$\int_{\mathcal{X}^2} k(u, v) f(u) f(v)\, du\, dv \geq 0.$$

Mercer's Theorem

Theorem: If k is continuous and T_k is positive semidefinite, then T_k has eigenfunctions $\psi_i \in L_2(\mathcal{X})$ (say $\|\psi_i\|_{L_2} = 1$) with eigenvalues $\lambda_i \geq 0$, and for all $u, v \in \mathcal{X}$, we can write

$$k(u, v) = \sum_{i=1}^{\infty} \lambda_i \psi_i(u) \psi_i(v).$$

Furthermore, this series converges uniformly.

- A positive semidefinite operator defines a kernel
- Eigenfunctions play the role of features/basis (not normalized)

We skip the demonstration (Spectral Theorem)

Reproducing property

Under certain conditions (Mercer's theorem and extensions), we can write

$$k(x,x') = \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(x'), \qquad \int_{\mathcal{X}} e_i(x) e_j(x) d\mu(x) = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

where this sum is guaranteed to converge whatever the x and x'.

Infinite dimensional feature map: $\phi(x) = \begin{bmatrix} \vdots \\ \sqrt{\lambda_i} e_i(x) \\ \vdots \end{bmatrix} \in \ell_2.$

Reproducing property when we have pd: more concrete

$$f(x) = \sum_i \alpha_i e_i(x)$$

$$K(x, x') = \sum_j \lambda_j e_j(x) e_j(x')$$

$$\begin{aligned}
\langle f, K_x \rangle &= \int K(x, x') f(x') dx' \\
&= \int \Big(\sum_i \lambda_i e_i(x) e_i(x') \Big) f(x') dx' \\
&= \sum_i \Big(\int \sqrt{\lambda_i} e_i(x') f(x') dx' \Big) \sqrt{\lambda_i} e_i(x) \\
&= \sum_i \alpha_i e_i(x) = f(x)
\end{aligned}$$

How does the reproducing property guarantees uniqueness?
We divide the demonstration into steps

$$\text{Kernel} \leftrightarrow \text{PD} \leftrightarrow \text{r.k.} \leftrightarrow \text{RKHS}$$

Kernel ↔ PD ↔ r.k. ↔ RKHS

Mercer theorem

PD ↔ r.k. ↔ RKHS

A function $k : X \times X \to \mathbb{R}$ defines a positive definite kernel if and only if there exists a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ of functions on X such that:

1. For every $x \in X$, the function $k_x(\cdot) = k(x, \cdot)$ belongs to $\mathcal{H}$.
2. The reproducing property holds: for every $f \in \mathcal{H}$ and every $x \in X$,

$$f(x) = \langle f, k_x \rangle_{\mathcal{H}}.$$

(⇒) If k is a positive definite kernel, then it induces an RKHS with the reproducing property.

Construct an inner product on the space of functions generated by the kernel k and show that this construction leads to an RKHS that inherently satisfies the reproducing property.

1. **Construct the RKHS:** Given a positive definite kernel $k : X \times X \to \mathbb{R}$, we define the linear space $\mathcal{H}_0$ consisting of finite linear combinations of kernel functions:

$$\mathcal{H}_0 = \left\{ f = \sum_{i=1}^{n} c_i k_{x_i} : c_i \in \mathbb{R}, x_i \in X, n \in \mathbb{N} \right\}.$$

Define an inner product on this space by:

$$\left\langle \sum_{i=1}^{n} c_i k_{x_i}, \sum_{j=1}^{m} d_j k_{y_j} \right\rangle_{\mathcal{H}_0} = \sum_{i=1}^{n} \sum_{j=1}^{m} c_i d_j k(x_i, y_j).$$

2. **Completion to an RKHS:** The space $\mathcal{H}_0$ can be completed to form a Hilbert space $\mathcal{H}$. By construction, every function in this Hilbert space can be expressed (or approximated arbitrarily closely) as a linear combination of the kernel functions k_x.

3. **Reproducing Property:** For each $x \in X$ and any function $f = \sum_{i=1}^{n} c_i k_{x_i} \in \mathcal{H}_0$,

$$f(x) = \sum_{i=1}^{n} c_i k(x_i, x) = \left\langle \sum_{i=1}^{n} c_i k_{x_i}, k_x \right\rangle_{\mathcal{H}_0}.$$

Since $k_x \in \mathcal{H}$ by construction, this property extends to the entire RKHS $\mathcal{H}$. Therefore, k satisfies the reproducing property.

The reproducing property **emerges** naturally from how the inner product was defined, due to the kernel's structure. This is the key of the proof: defining the space and inner product using the kernel, then completing this space to obtain the RKHS where the reproducing property holds.

(⇐) If there exists an RKHS with the reproducing property, then k is a positive definite kernel.

1. **Existence of the Kernel Function:** Suppose $\mathcal{H}$ is an RKHS of functions on X with the reproducing property. For every $x \in X$, there exists a function $k_x \in \mathcal{H}$ such that for any $f \in \mathcal{H}$,

$$f(x) = \langle f, k_x \rangle_{\mathcal{H}}.$$

2. **Define the Kernel:** Define the function $k : X \times X \to \mathbb{R}$ by

$$k(x, y) = \langle k_y, k_x \rangle_{\mathcal{H}}.$$

3. **Positive Definiteness:** To show that k is a positive definite kernel, take any finite set of points $\{x_1, x_2, \ldots, x_n\} \subset X$ and any real coefficients $c_1, c_2, \ldots, c_n$. Consider the linear combination in the RKHS:

$$f = \sum_{i=1}^{n} c_i k_{x_i}.$$

Then, the norm squared of f in the RKHS is given by:

$$\|f\|_{\mathcal{H}}^2 = \left\langle \sum_{i=1}^{n} c_i k_{x_i}, \sum_{j=1}^{n} c_j k_{x_j} \right\rangle_{\mathcal{H}} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \langle k_{x_i}, k_{x_j} \rangle_{\mathcal{H}} = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j).$$

Since $\|f\|_{\mathcal{H}}^2 \geq 0$ (as it is the squared norm of a function in a Hilbert space), it follows that

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0. \qquad c^T K c$$

Therefore, k is a positive definite kernel.

Uniqueness of RKHS and RK

Theorem

- If $\mathcal{H}$ is a RKHS, then it has a unique r.k.
- Conversely, a function K can be the r.k. of at most one RKHS.

$\Longrightarrow$

If a r.k. exists then it is unique

Let K and K' be two r.k. of a RKHS $\mathcal{H}$. Then for any $\mathbf{x} \in \mathcal{X}$:

$$\begin{aligned} \| K_{\mathbf{x}} - K'_{\mathbf{x}} \|^2_{\mathcal{H}} &= \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K_{\mathbf{x}} - K'_{\mathbf{x}} \rangle_{\mathcal{H}} \\ &= \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} - \langle K_{\mathbf{x}} - K'_{\mathbf{x}}, K'_{\mathbf{x}} \rangle_{\mathcal{H}} \\ &= K_{\mathbf{x}}(\mathbf{x}) - K'_{\mathbf{x}}(\mathbf{x}) - K_{\mathbf{x}}(\mathbf{x}) + K'_{\mathbf{x}}(\mathbf{x}) \\ &= 0 . \end{aligned}$$

This shows that $K_{\mathbf{x}} = K'_{\mathbf{x}}$ as functions, i.e., $K_{\mathbf{x}}(\mathbf{y}) = K'_{\mathbf{x}}(\mathbf{y})$ for any $\mathbf{y} \in \mathcal{X}$. In other words, K=K'. □

Another way of characterizing RKHS is:
reproducing property and pointwise continuity

If $\mathcal{H}$ is a RKHS then $f \mapsto f(\mathbf{x})$ is continuous

If a r.k. K exists, then for any $(\mathbf{x}, f) \in \mathcal{X} \times \mathcal{H}$:

$$
\begin{aligned}
|f(\mathbf{x})| &= |\langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}}| \\
&\leq \|f\|_{\mathcal{H}} . \|K_{\mathbf{x}}\|_{\mathcal{H}} \\
&\leq \|f\|_{\mathcal{H}} . K(\mathbf{x}, \mathbf{x})^{\frac{1}{2}} ,
\end{aligned}
$$

$$|\langle \mathbf{u}, \mathbf{v} \rangle|^2 \leq \langle \mathbf{u}, \mathbf{u} \rangle \cdot \langle \mathbf{v}, \mathbf{v} \rangle$$

(Cauchy-Schwarz)

because $\|K_{\mathbf{x}}\|^2_{\mathcal{H}} = \langle K_{\mathbf{x}}, K_{\mathbf{x}} \rangle_{\mathcal{H}} = K(\mathbf{x}, \mathbf{x})$. Therefore $f \in \mathcal{H} \mapsto f(\mathbf{x}) \in \mathbb{R}$ is a continuous linear mapping. □

Reproducing property implies the fact that is pointwise continuous

Reproducing property and pointwise continuity

$\Longleftarrow$

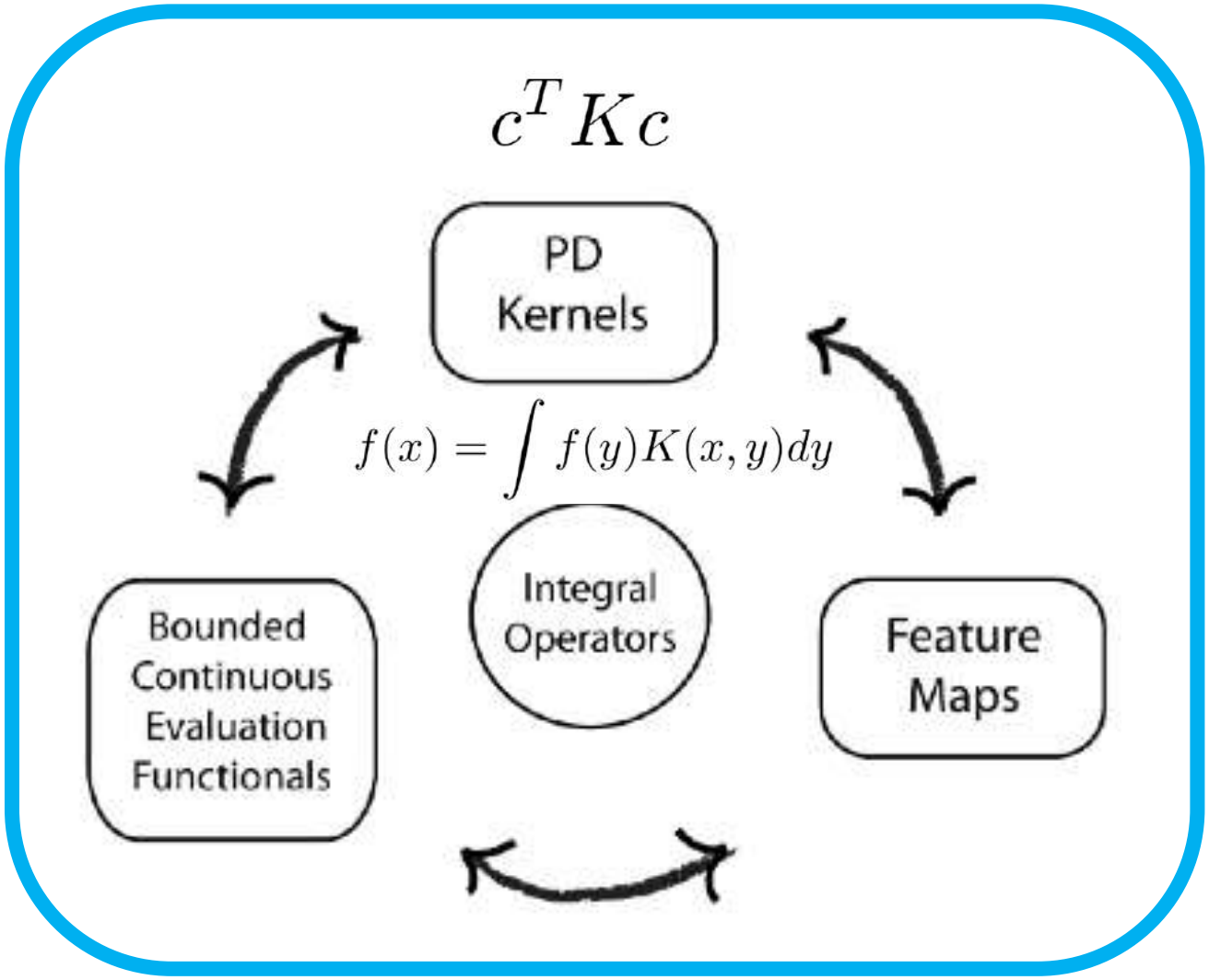
If $f \mapsto f(\mathbf{x})$ is continuous then $\mathcal{H}$ is a RKHS

Conversely, let us assume that for any $\mathbf{x} \in \mathcal{X}$ the linear form $f \in \mathcal{H} \mapsto f(\mathbf{x})$ is continuous.
Then by Riesz representation theorem (general property of Hilbert spaces) there exists a unique $g_{\mathbf{x}} \in \mathcal{H}$ such that:

$$f(\mathbf{x}) = \langle f, g_{\mathbf{x}} \rangle_{\mathcal{H}} .$$

The function $K(\mathbf{x}, \mathbf{y}) = g_{\mathbf{x}}(\mathbf{y})$ is then a r.k. for $\mathcal{H}$. $\square$

The picture is:



Summarizing

- Feature maps define kernels and PD operators but the relation is not one to one
- Feature maps defined by the reproducing kernel Hilbert spaces property are one to one with kernels and pd
- To have the reproducing property we consider operators that satisfy the Mercer theorem
- Pointwise continuity is one to one with RKHS

Why do we care?

(Representer Theorem): *The optimal solution to any vector valued function learning problem of the form.*

$$f^* = \text{argmin}_{f \in \mathcal{H}_K} L(f(\mathbf{x}_1) \dots f(\mathbf{x}_l)) + \lambda \|f\|^2_{\mathcal{H}_K}$$

is a sum of matrix-vector products of the form

$$f^*(\mathbf{x}) = \sum_{i=1}^{l} K(\mathbf{x}, \mathbf{x}_i)\boldsymbol{\alpha}_i$$

where $\boldsymbol{\alpha}_i \in \mathbb{R}^n, \quad i = 1 \dots l$, *L is an arbitrary loss function (which can also be an indicator function encoding arbitrary constraints) and* $\lambda > 0$ *is a regularization parameter.*

- **We reduced an infinite dimensional non-linear optimization to a linear finite one!!!**
- **Choosing a kernel we choose a regularization!!!**

Motivation

- An RKHS is a space of (potentially nonlinear) functions, and $\| f \|_{\mathcal{H}}$ measures the smoothness of f.
- Given a set of data $(\mathbf{x}_i \in \mathcal{X}, y_i \in \mathbb{R})_{i=1,\dots,n}$, a natural way to estimate a regression function $f : \mathcal{X} \to \mathbb{R}$ is to solve something like:

$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\lambda \|f\|_{\mathcal{H}}^2}_{\text{regularization}} . \tag{1}$$

  for a loss function ℓ such as $\ell(y, t) = (y - t)^2$.
- How to solve in practice this problem, potentially in infinite dimension?

Proof

f in the subspace spanned by the kernel → The loss is minimized

Suppose we project f onto the subspace:

$$\text{span}\{k(x_i, \cdot)\colon\ 1 \leq i \leq n\}$$

obtaining f_s (the component along the subspace) and $f_\perp$ (the component perpendicular to the subspace). We have:

$$f = f_s + f_\perp \Rightarrow ||f||^2 = ||f_s||^2 + ||f_\perp||^2 \geq ||f_s||^2$$

Since Ω is non-decreasing,

$$\Omega(||f||_H^2) \geq \Omega(||f_s||_H^2)$$

implying that $\Omega(\cdots)$ is minimized if f lies in the subspace. Furthermore, since the kernel k has the reproducing property, we have:

$$f(x_i) = \langle f, k(x_i, \cdot)\rangle = \langle f_s, k(x_i, \cdot)\rangle + \langle f_\perp, k(x_i, \cdot)\rangle = \langle f_s, k(x_i, \cdot)\rangle = f_s(x_i)$$

Implying that:

$$L(f(x_1), \ldots, f(x_n)) = L(f_s(x_1), \ldots, f_s(x_n))$$

Hence, $L(\cdots)$ depends only on the component of f lying in the subspace: $\text{span}\{k(x_i, \cdot)\colon 1 \leq i \leq n\}$, and $\Omega(\cdots)$ is minimized if f lies in that subspace. Hence, $J(f)$ is minimized if f lies in that subspace, and we can express the minimizer as:

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$$

Practical use of the representer theorem (1/2)

- When the representer theorem holds, we know that we can look for a solution of the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) , \quad \text{for some } \boldsymbol{\alpha} \in \mathbb{R}^n.$$

- For any $j = 1, \ldots, n$, we have

$$f(\mathbf{x}_j) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{K}\boldsymbol{\alpha}]_j.$$

- Furthermore,

$$\| f \|_{\mathcal{H}}^2 = \left\| \sum_{i=1}^{n} \alpha_i K_{\mathbf{x}_i} \right\|_{\mathcal{H}}^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}.$$

Practical use of the representer theorem (2/2)

- Therefore, a problem of the form

$$\min_{f \in \mathcal{H}} \Psi \left(f\left(\mathbf{x}_1\right), \cdots, f\left(\mathbf{x}_n\right), \| f \|_{\mathcal{H}}^2 \right)$$

is equivalent to the following n-dimensional optimization problem:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \Psi \left([\mathbf{K}\boldsymbol{\alpha}]_1, \cdots, [\mathbf{K}\boldsymbol{\alpha}]_n, \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \right).$$

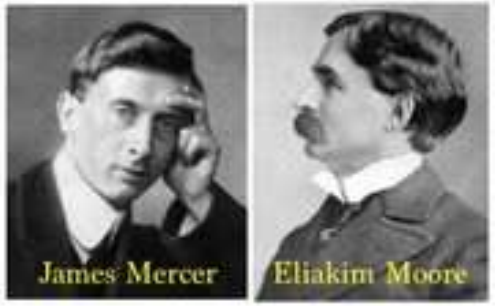
Summary

Feature map: $\varphi : x \in \mathbb{R}^d \longmapsto \mathcal{H}$ Hilbert space.

Reproducing Hilbert space: $f : \mathbb{R}^d \to \mathbb{R}$

$$\|f\|_R = \min \{\|h\|_{\mathcal{H}} \; ; \; h \in \mathcal{H}, f = \langle h, \varphi(\cdot)\rangle_{\mathcal{H}}\}$$

Reproducing kernel: $k(x,y) \overset{\text{def.}}{=} \langle \varphi(x), \varphi(y)\rangle_{\mathcal{H}}$



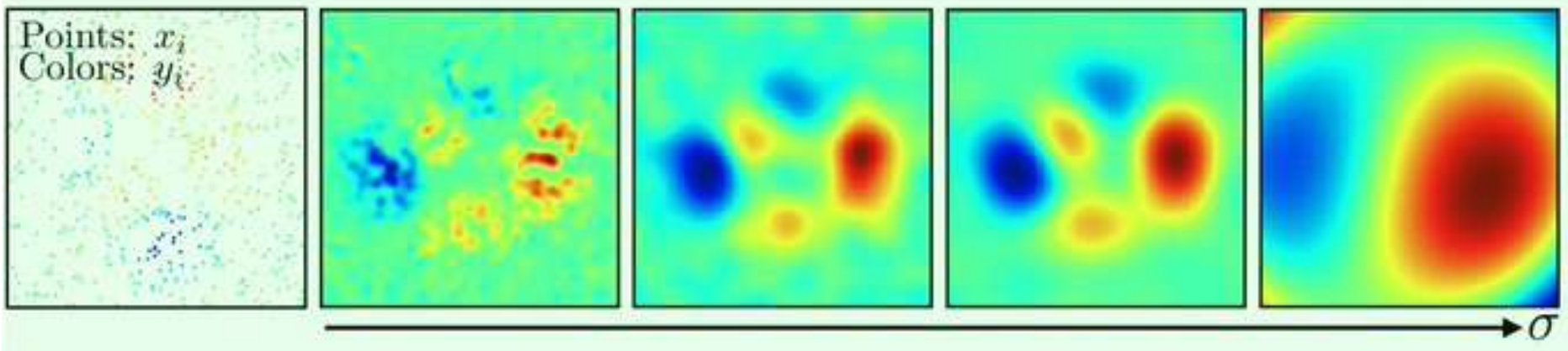
Representer theorem: the solution of $\min_{f:\mathbb{R}^d \to \mathbb{R}} \mathcal{E}((f(x_i))_i) + \|f\|_R^2$

satisfies $f(x) = \sum_i w_i K(x_i, x)$.

Example: regression

$$\mathcal{E}((f_i)_i) = \frac{1}{n}\sum_{i=1}^n |f_i - y_i|^2$$

$$k(x,y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$



Kernel Ridge Regression (KRR)

- Let us now consider a RKHS $\mathcal{H}$, associated to a p.d. kernel K on $\mathcal{X}$.
- KRR is obtained by regularizing the MSE criterion by the RKHS norm:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f\left(\mathbf{x}_i\right)\right)^2 + \lambda \| f \|_{\mathcal{H}}^2 \qquad (2)$$

- *1st effect = prevent overfitting by penalizing non-smooth functions.*
- By the representer theorem, any solution of (2) can be expanded as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K\left(\mathbf{x}_i, \mathbf{x}\right).$$

- *2nd effect = simplifying the solution.*

Solving KRR

- Let $\mathbf{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$
- Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \mathbb{R}^n$
- Let $\mathbf{K}$ be the $n \times n$ Gram matrix: $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$
- We can then write:

$$\left(\hat{f}(\mathbf{x}_1), \ldots, \hat{f}(\mathbf{x}_n)\right)^\top = \mathbf{K}\boldsymbol{\alpha}$$

- The following holds as usual:

$$\| \hat{f} \|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$$

- The KRR problem (2) is therefore equivalent to:

$$\arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{n} (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})^\top (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$$

Solving KRR

$$\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{n}\left(\mathbf{K}\boldsymbol{\alpha}-\mathbf{y}\right)^\top\left(\mathbf{K}\boldsymbol{\alpha}-\mathbf{y}\right)+\lambda\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$$

- This is a convex and differentiable function of $\boldsymbol{\alpha}$. Its minimum can therefore be found by setting the gradient in $\boldsymbol{\alpha}$ to zero:

$$\begin{aligned} 0 &= \frac{2}{n}\mathbf{K}\left(\mathbf{K}\boldsymbol{\alpha}-\mathbf{y}\right)+2\lambda\mathbf{K}\boldsymbol{\alpha} \\ &= \mathbf{K}\left[\left(\mathbf{K}+\lambda n\mathbf{I}\right)\boldsymbol{\alpha}-\mathbf{y}\right] \end{aligned}$$

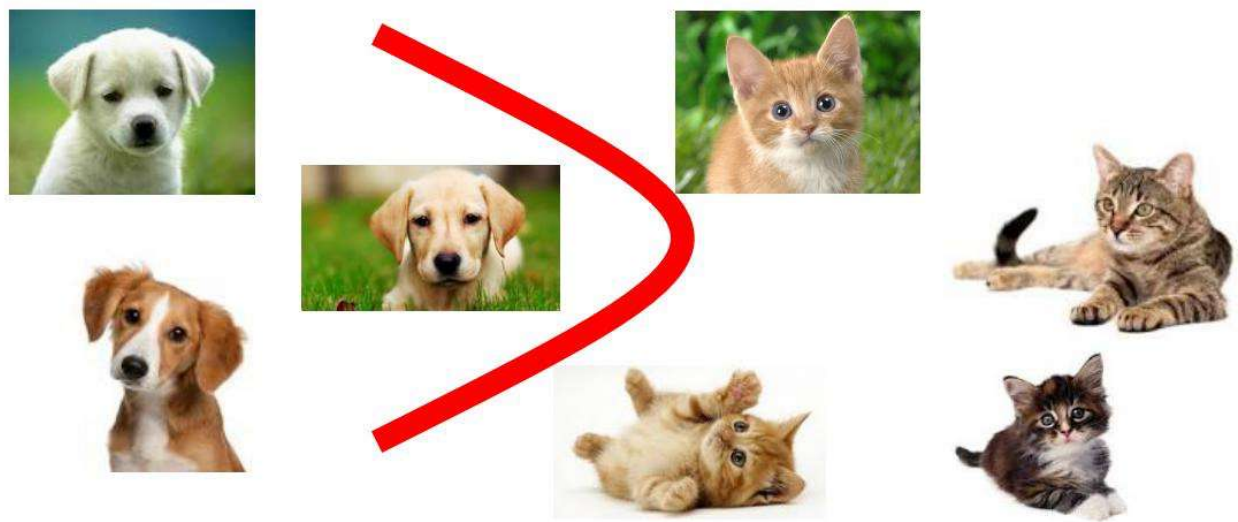
- For $\lambda > 0$, $\mathbf{K}+\lambda n\mathbf{I}$ is invertible (because $\mathbf{K}$ is positive semidefinite) so one solution is to take:

$$\boldsymbol{\alpha} = \left(\mathbf{K}+\lambda n\mathbf{I}\right)^{-1}\mathbf{y}.$$

Binary classification

Setup

- $\mathcal{X}$ set of inputs
- $\mathcal{Y} = \{-1, 1\}$ binary outputs
- $\mathcal{S}_n = (\mathbf{x}_i, y_i)_{i=1,\dots,n} \in (\mathcal{X} \times \mathcal{Y})^n$ a training set of n pairs
- Goal = find a function $f : \mathcal{X} \to \mathbb{R}$ to predict y by $sign(f(\mathbf{x}))$



The 0/1 loss

- The 0/1 loss measures if a prediction is correct or not:

$$\ell_{0/1}\left(f(\mathbf{x}), y\right)) = \mathbf{1}\left(yf(\mathbf{x}) < 0\right) = \begin{cases} 0 & \text{if } y = sign\left(f(\mathbf{x})\right) \\ 1 & \text{otherwise.} \end{cases}$$

- It is then tempting to learn f by solving:

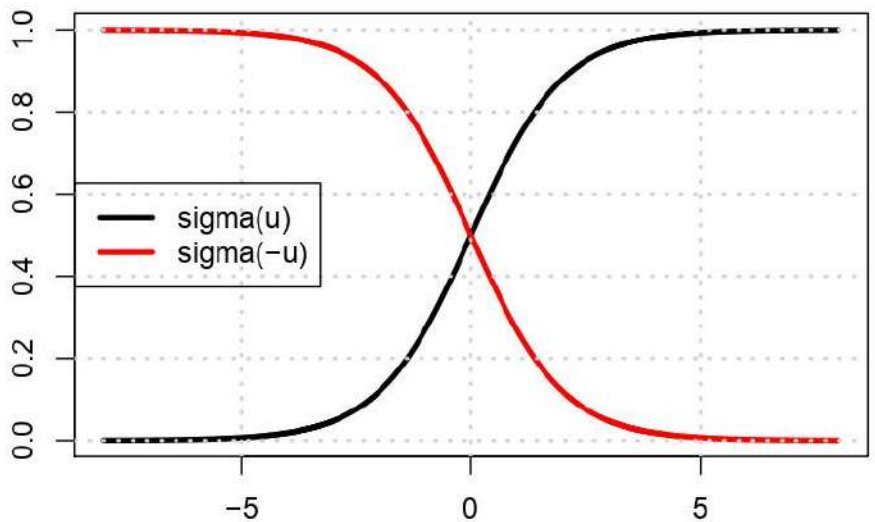
$$\min_{f \in \mathcal{H}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}\left(f\left(\mathbf{x}_i\right), y_i\right)}_{\text{misclassification rate}} + \underbrace{\lambda \| f \|_{\mathcal{H}}^2}_{\text{regularization}}$$

- However:
  - The problem is non-smooth, and typically NP-hard to solve

The logistic loss

- An alternative is to define a probabilistic model of y parametrized by $f(\mathbf{x})$, e.g.:

$$\forall \mathbf{y} \in \{-1, 1\}, \quad p(y \mid f(\mathbf{x})) = \frac{1}{1 + e^{-yf(\mathbf{x})}} = \sigma(yf(\mathbf{x}))$$



- The logistic loss is the negative conditional likelihood:

$$\ell_{logistic}(f(\mathbf{x}), y) = -\ln p(y \mid f(\mathbf{x})) = \ln\left(1 + e^{-yf(\mathbf{x})}\right)$$

Kernel logistic regression (KLR)

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{logistic}\left(f(\mathbf{x}_i), y_i\right) + \frac{\lambda}{2} \| f \|_{\mathcal{H}}^2$$

$$= \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + e^{-y_i f(\mathbf{x}_i)}\right) + \frac{\lambda}{2} \| f \|_{\mathcal{H}}^2$$

- Can be interpreted as a regularized conditional maximum likelihood estimator
- No explicit solution, but smooth convex optimization problem that can be solved numerically

Solving KLR

- By the representer theorem, any solution of KLR can be expanded as

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x})$$

and as always we have:

$$\left(\hat{f}(\mathbf{x}_1), \ldots, \hat{f}(\mathbf{x}_n)\right)^\top = \mathbf{K}\boldsymbol{\alpha} \quad \text{and} \quad \| \hat{f} \|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$$

- To find $\boldsymbol{\alpha}$ we therefore need to solve:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \ln\left(1 + e^{-y_i [\mathbf{K}\boldsymbol{\alpha}]_i}\right) + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$$

Given:

- $f(x_i) = \sum_{j=1}^{n} \alpha_j K(x_i, x_j)$, where $K(x_i, x_j)$ is the kernel function.
- The probability in logistic regression is $\sigma(f(x_i)) = \frac{1}{1+e^{-\sum_{j=1}^{n} \alpha_j K(x_i, x_j)}}$.

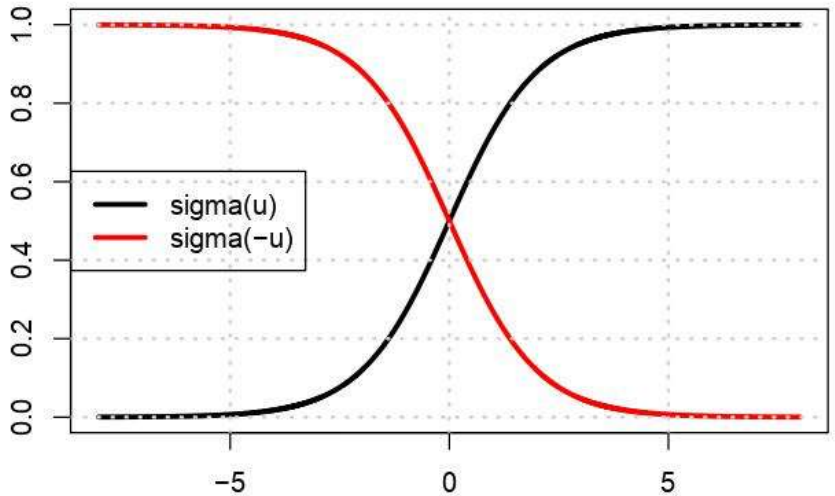
1. Logistic Regression Cost Function

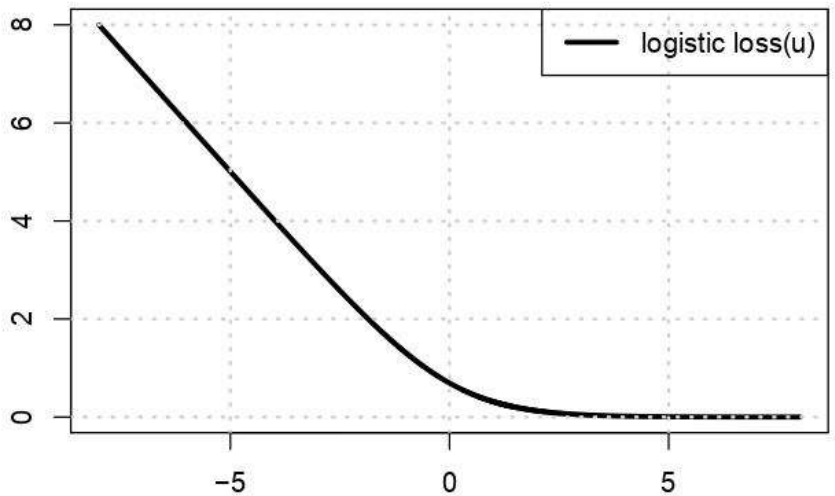
The cost function for logistic regression is:

$$J(\alpha) = -\sum_{i=1}^{n} [y_i \log(\sigma(f(x_i))) + (1 - y_i) \log(1 - \sigma(f(x_i)))] + \frac{\lambda}{2} \sum_{j,k=1}^{n} \alpha_j \alpha_k K(x_j, x_k).$$

Here, λ is the regularization parameter.

Technical facts





Sigmoid:

- $\sigma(u) = \frac{1}{1+e^{-u}}$
- $\sigma(-u) = 1 - \sigma(u)$
- $\sigma'(u) = \sigma(u)\sigma(-u) \geq 0$

Logistic loss:

- $\ell_{logistic}(u) = \ln\left(1 + e^{-u}\right)$
- $\ell'_{logistic}(u) = -\sigma(-u)$
- $\ell''_{logistic}(u) = \sigma(u)\sigma(-u) \geq 0$

Step 1: Derivative of the Logistic Loss

The logistic loss for a single sample x_i is:

$$L_i = -\left[y_i \log(\sigma(f(x_i))) + (1 - y_i) \log(1 - \sigma(f(x_i)))\right].$$

Since $f(x_i) = \sum_{j=1}^{n} \alpha_j K(x_i, x_j)$, the derivative of the logistic loss with respect to $f(x_i)$ is:

$$\frac{\partial L_i}{\partial f(x_i)} = \sigma(f(x_i)) - y_i.$$

Now, we need to express this in terms of α_j. The derivative of $f(x_i)$ with respect to α_j is:

$$\frac{\partial f(x_i)}{\partial \alpha_j} = K(x_i, x_j).$$

Using the chain rule, the derivative of L_i with respect to α_j is:

$$\frac{\partial L_i}{\partial \alpha_j} = \frac{\partial L_i}{\partial f(x_i)} \frac{\partial f(x_i)}{\partial \alpha_j} = (\sigma(f(x_i)) - y_i) K(x_i, x_j).$$

Step 2: Derivative of the Regularization Term

The regularization term is:

$$\frac{\lambda}{2} \sum_{j,k=1}^{n} \alpha_j \alpha_k K(x_j, x_k).$$

The derivative of this term with respect to α_j is:

$$\frac{\partial}{\partial \alpha_j} \left(\frac{\lambda}{2} \sum_{k=1}^{n} \alpha_k K(x_j, x_k) \right) = \lambda \sum_{k=1}^{n} \alpha_k K(x_j, x_k).$$

3. Combining Both Parts

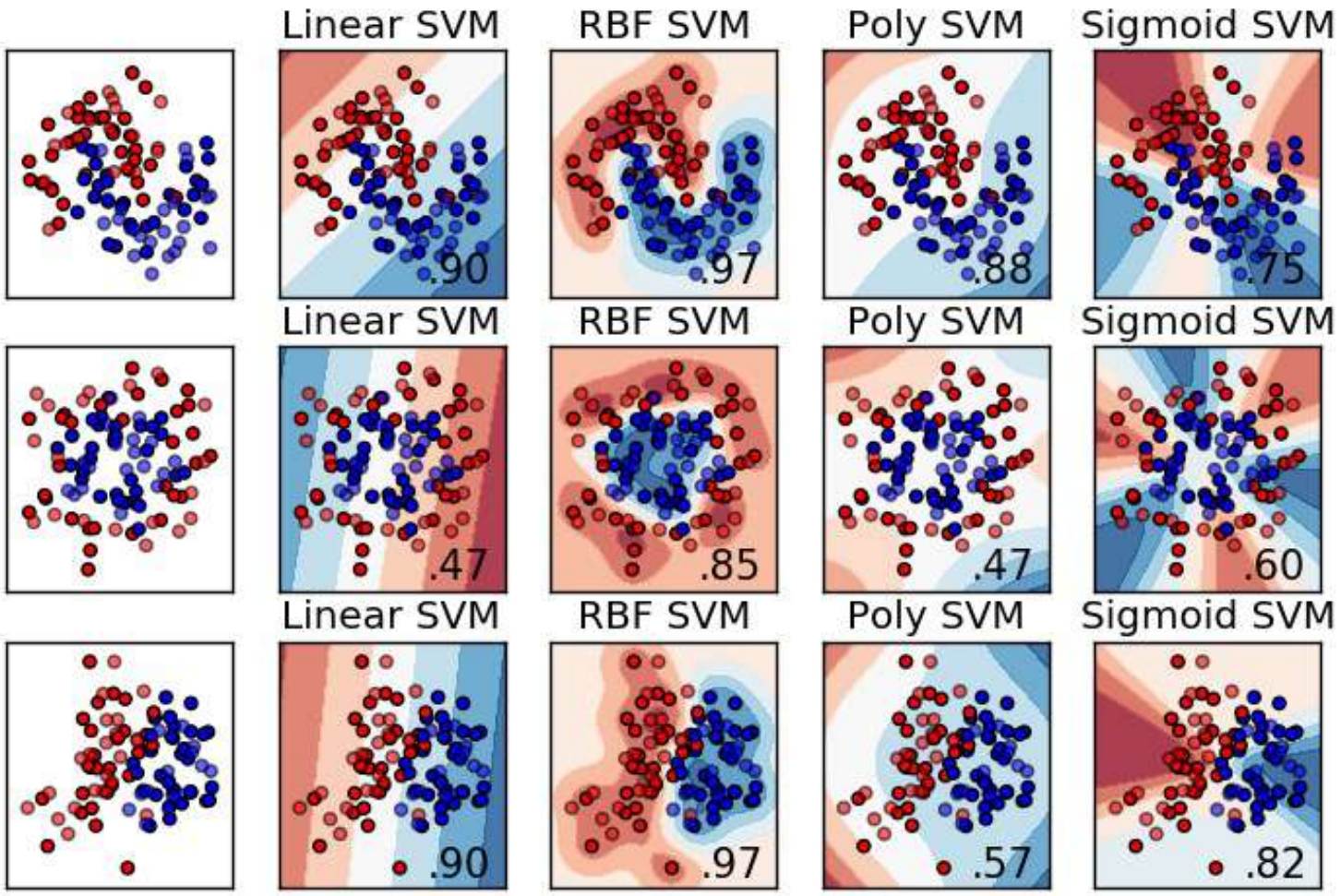
Now, we combine the derivatives from both parts (logistic loss and regularization):

$$\frac{\partial J(\alpha)}{\partial \alpha_j} = \sum_{i=1}^{n} (\sigma(f(x_i)) - y_i) K(x_i, x_j) + \lambda \sum_{k=1}^{n} \alpha_k K(x_j, x_k).$$

Final Expression

The derivative of the logistic regression cost function in the kernelized form with respect to the coefficient α_j is:

$$\frac{\partial J(\alpha)}{\partial \alpha_j} = \sum_{i=1}^{n} (\sigma(\sum_{k=1}^{n} \alpha_k K(x_i, x_k)) - y_i) K(x_i, x_j) + \lambda \sum_{k=1}^{n} \alpha_k K(x_j, x_k).$$



Kernel SVM example

Different kernels induce different regularizations

The norm depends on the kernel

Changing the kernel we change the regularization

Since $f = \sum_{j=1}^{n} c_j K_{x_j}$, then

$$\begin{aligned} \|f\|_{\mathcal{H}}^2 &= \langle f, f \rangle_{\mathcal{H}} \\ &= \langle \sum_{i=1}^{n} c_i K_{x_i}, \sum_{j=1}^{n} c_j K_{x_j} \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \langle K_{x_i}, K_{x_j} \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) = c^t \mathbf{K} c \end{aligned}$$

$$\|f\|_{\mathcal{H}} = c^T K c = c^T U^T \Lambda U c = \tilde{c}^T \Lambda \tilde{c}$$

Smoothness functional

A simple inequality

- By Cauchy-Schwarz we have, for any function $f \in \mathcal{H}$ and any two points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

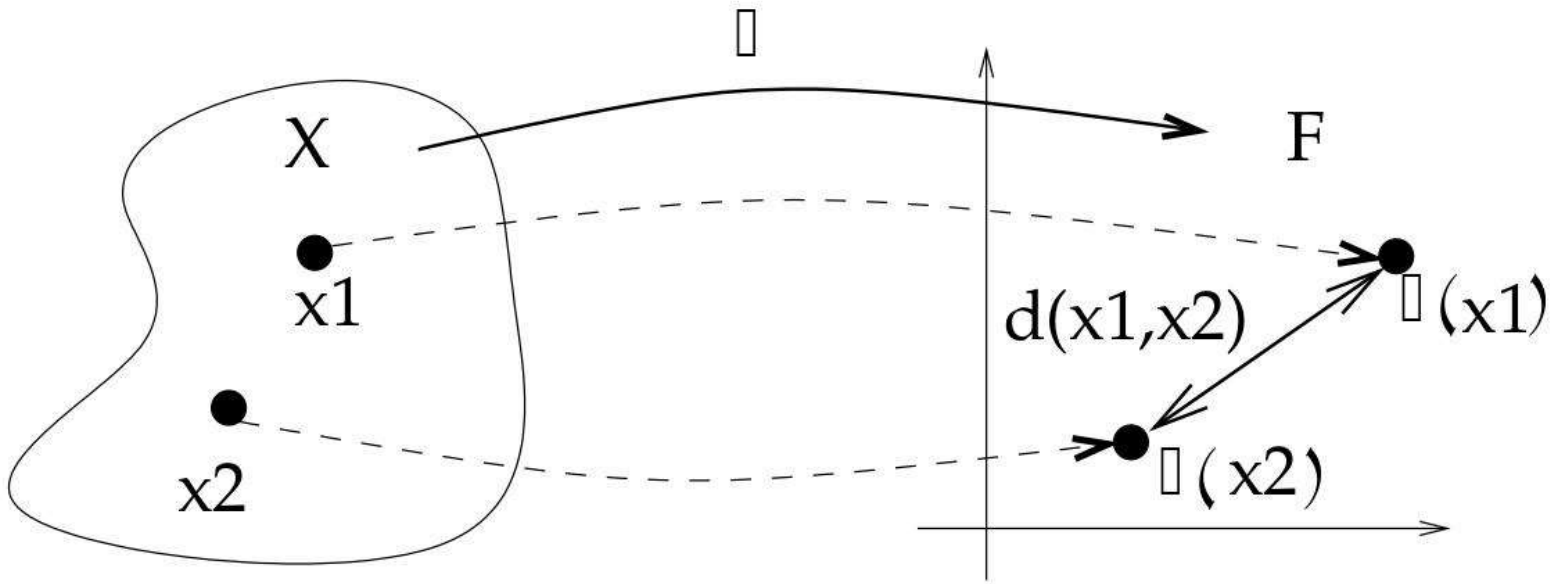
$$\begin{aligned} \left| f(\mathbf{x}) - f(\mathbf{x}') \right| &= \left| \langle f, K_{\mathbf{x}} - K_{\mathbf{x}'} \rangle_{\mathcal{H}} \right| \\ &\leq \| f \|_{\mathcal{H}} \times \| K_{\mathbf{x}} - K_{\mathbf{x}'} \|_{\mathcal{H}} \\ &= \| f \|_{\mathcal{H}} \times d_K(\mathbf{x}, \mathbf{x}') \ . \end{aligned}$$

- The norm of a function in the RKHS controls how fast the function varies over $\mathcal{X}$ with respect to the geometry defined by the kernel (Lipschitz with constant $\| f \|_{\mathcal{H}}$).

Important message

Small norm $\Longrightarrow$ slow variations.

Example 1: computing distances in the feature space



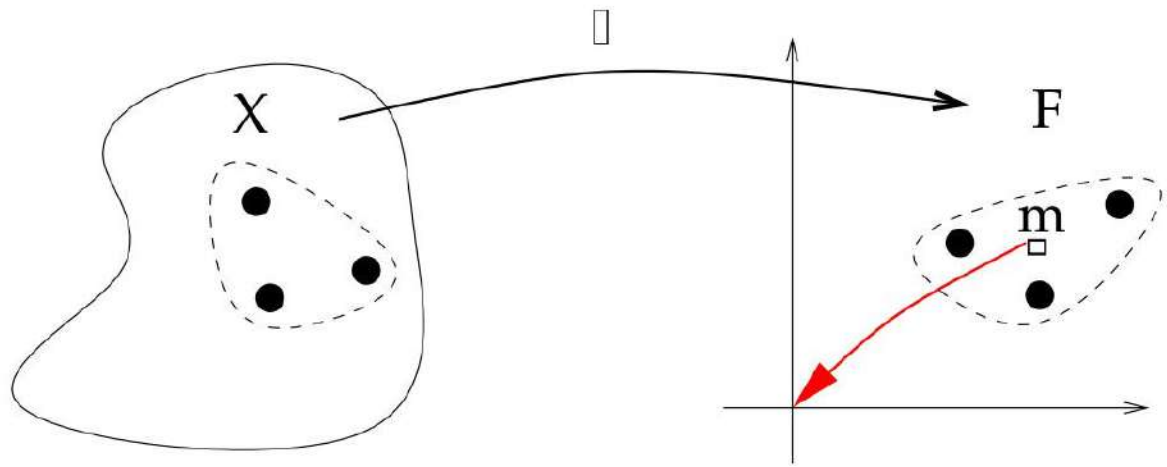
$$d_K(\mathbf{x}_1, \mathbf{x}_2)^2 = \| \Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2) \|^2_{\mathcal{H}}$$
$$= \langle \Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2), \Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}}$$
$$= \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_1) \rangle_{\mathcal{H}} + \langle \Phi(\mathbf{x}_2), \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}} - 2 \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}}$$

$$d_K(\mathbf{x}_1, \mathbf{x}_2)^2 = K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)$$

Example 3: Centering data in the feature space

Problem

- Let $\mathcal{S} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ be a finite set of points in $\mathcal{X}$ endowed with a p.d. kernel K. Let $\mathbf{K}$ be their $n \times n$ Gram matrix: $[\mathbf{K}]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.
- Let $\boldsymbol{\mu} = 1/n \sum_{i=1}^{n} \Phi(\mathbf{x}_i)$ their barycenter, and $\mathbf{u}_i = \Phi(\mathbf{x}_i) - \boldsymbol{\mu}$ for $i = 1, \ldots, n$ be centered data in $\mathcal{H}$.
- How to compute the centered Gram matrix $[\mathbf{K}^c]_{i,j} = \langle \mathbf{u}_i, \mathbf{u}_j \rangle_{\mathcal{H}}$?



Computation

- A direct computation gives, for $0 \leq i, j \leq n$:

$$\begin{aligned}
\mathbf{K}^c_{i,j} &= \langle \Phi(\mathbf{x}_i) - \boldsymbol{\mu}, \Phi(\mathbf{x}_j) - \boldsymbol{\mu} \rangle_{\mathcal{H}} \\
&= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}} - \langle \boldsymbol{\mu}, \Phi(\mathbf{x}_i) + \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}} + \langle \boldsymbol{\mu}, \boldsymbol{\mu} \rangle_{\mathcal{H}} \\
&= \mathbf{K}_{i,j} - \frac{1}{n} \sum_{k=1}^{n} (\mathbf{K}_{i,k} + \mathbf{K}_{j,k}) + \frac{1}{n^2} \sum_{k,l=1}^{n} \mathbf{K}_{k,l} \,.
\end{aligned}$$

- This can be rewritten in matricial form:

$$\mathbf{K}^c = \mathbf{K} - \mathbf{U}\mathbf{K} - \mathbf{K}\mathbf{U} + \mathbf{U}\mathbf{K}\mathbf{U} = (\mathbf{I} - \mathbf{U})\,\mathbf{K}\,(\mathbf{I} - \mathbf{U}) \,,$$

where $\mathbf{U}_{i,j} = 1/n$ for $1 \leq i, j \leq n$.

Kernel PCA

We want to extend the PCA in the features space.

- Calculate the covariance matrix in the feats space:

$$C_F = \frac{1}{N} \sum_{i}^{N} \phi(x_i) \phi^T(x_i)$$

- We want to solve $C_F v = \lambda v$

Eigenvectors can be expressed as a linear combination of the features i.e.

$$v = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

- We have

$$C_F v = \lambda v = \frac{1}{N} \sum_{i}^{N} \phi(x_i) \phi^T(x_i) v$$

- Thus

$$v = \frac{1}{N\lambda} \sum_{i}^{N} \phi(x_i) \phi^T(x_i) v = \sum_{i}^{N} \phi(x_i) \frac{\beta_i}{N\lambda} = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

Let's multiply both sides by $\phi^T(x_i)$

- Expanding

$$\lambda \phi^T(x_k) v = \phi^T(x_k) C_F v$$

$$\lambda \sum_i \alpha_i \phi^T(x_k) \phi(x_i) = \frac{1}{N} \sum_i^N \alpha_i (\phi^T(x_k) \sum_j^N \phi(x_j) (\phi^T(x_j) \phi(x_i))$$

- Define $K_{ij} = \phi^T(x_j)\phi(x_i)$ we have

$$N\lambda K\alpha = K^2\alpha \quad \rightarrow \quad N\lambda\alpha = K\alpha$$

Normalization

$$\hat{\phi}(x_i) = \phi(x_i) - \frac{1}{N}\sum_{j=1}^{N}\phi(x_k)$$

$$\begin{aligned}
\hat{K}_{ij} &= \left(\phi(x_i) - \frac{1}{N}\sum_{k=1}^{N}\phi(x_k)\right)\left(\phi(x_j) - \frac{1}{N}\sum_{l=1}^{N}\phi(x_l)\right) \\
&= K_{ij} - \frac{1}{N}\sum_{k=1}^{N}\phi^T(x_i)\phi(x_k) - \frac{1}{N}\sum_{l=1}^{N}\phi^T(x_j)\phi(x_l) + \frac{1}{N^2}\sum_{kl=1}^{N}\phi^T(x_k)\phi(x_l) \\
&= K_{ij} - \frac{1}{N}\sum_{k=1}^{N}K_{i,k} - \frac{1}{N}\sum_{k=1}^{N}K_{j,k} + \frac{1}{N^2}\sum_{k,l=1}^{N}K_{k,l}
\end{aligned}$$

Different kernels different regularizations

The norm depends on the kernel

Changing the kernel we change the regularization

Since $f = \sum_{j=1}^{n} c_j K_{x_j}$, then

$$\begin{aligned} \|f\|_{\mathcal{H}}^2 &= \langle f, f \rangle_{\mathcal{H}} \\ &= \langle \sum_{i=1}^{n} c_i K_{x_i}, \sum_{j=1}^{n} c_j K_{x_j} \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \langle K_{x_i}, K_{x_j} \rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) = c^t \mathbf{K} c \end{aligned}$$

$$\|f\|_{\mathcal{H}} = c^T K c = c^T U^T \Lambda U c = \tilde{c}^T \Lambda \tilde{c}$$

Regularization due to kernel choice

Gaussian kernel

$$K(x, y) = e^{-\frac{(x-y)^2}{2\sigma^2}}$$

corresponds to:

$$\varphi(t) = e^{-\frac{t^2}{2\sigma^2}}$$

$$\hat{\varphi}(\omega) = e^{-\frac{\sigma^2\omega^2}{2}}$$

and

$$\mathcal{H} = \left\{ f : \int \left| \hat{f}(\omega) \right|^2 e^{\frac{\sigma^2\omega^2}{2}} d\omega < \infty \right\}.$$

In particular, all functions in $\mathcal{H}$ are infinitely differentiable with all derivatives in L^2.

Laplace kernel

$$K(x,y) = \frac{1}{2} e^{-\gamma|x-y|}$$

corresponds to:

$$\varphi(to) = \frac{1}{2} e^{-\gamma|t|}$$

$$\hat{\varphi}(\omega) = \frac{\gamma}{\gamma^2 + \omega^2}$$

and

$$\mathcal{H} = \left\{ f : \int \left| \hat{f}(\omega) \right|^2 \frac{(\gamma^2 + \omega^2)}{\gamma} d\omega < \infty \right\},$$

the set of functions L^2 differentiable with derivatives in L^2 (Sobolev norm).

Compare with Gaussian Kernel

Low-frequency filter

$$K(x,y) = \frac{\sin(\Omega(x-y))}{\pi(x-y)}$$

corresponds to:

$$\varphi(t) = \frac{\sin(\Omega t)}{\pi t}$$
$$\hat{\varphi}(\omega) = 1_{[-\Omega,\Omega]}(\omega)$$

and

$$\mathcal{H} = \left\{ f : \int_{|\omega|>\Omega} \left| \hat{f}(\omega) \right|^2 d\omega = 0 \right\},$$

the set of functions whose spectrum is included in $[-\Omega, \Omega]$.

Compare with Gaussian Kernel

Hypothesis space

- Linear separable

$$f(x) = w^T x$$

We learn the separator on data

- Non-linear separable

$$f(x) = w^T \phi(x)$$

We learn the separator on a priori hand-crafted features

Models

- Linear

$$f(x) = w^\top x.$$

- Features

$$f(x) = w^\top \Phi(x) = \sum_{j=1}^{p} w^j \varphi_j(x).$$

- Kernels

$$f(x) = \sum_{i=1}^{n} K(x, x_i)\, c_i.$$

- Can we learn the features together with the separator?
- Which features? We choose this parametrization:

$$f(x) = w^T \sigma(W^T x)$$

$$w \in \mathbb{R}^k, \quad W \in \mathbb{R}^{d \times k}, \quad \sigma : \mathbb{R} \to \mathbb{R}$$

(1):Linear filtering

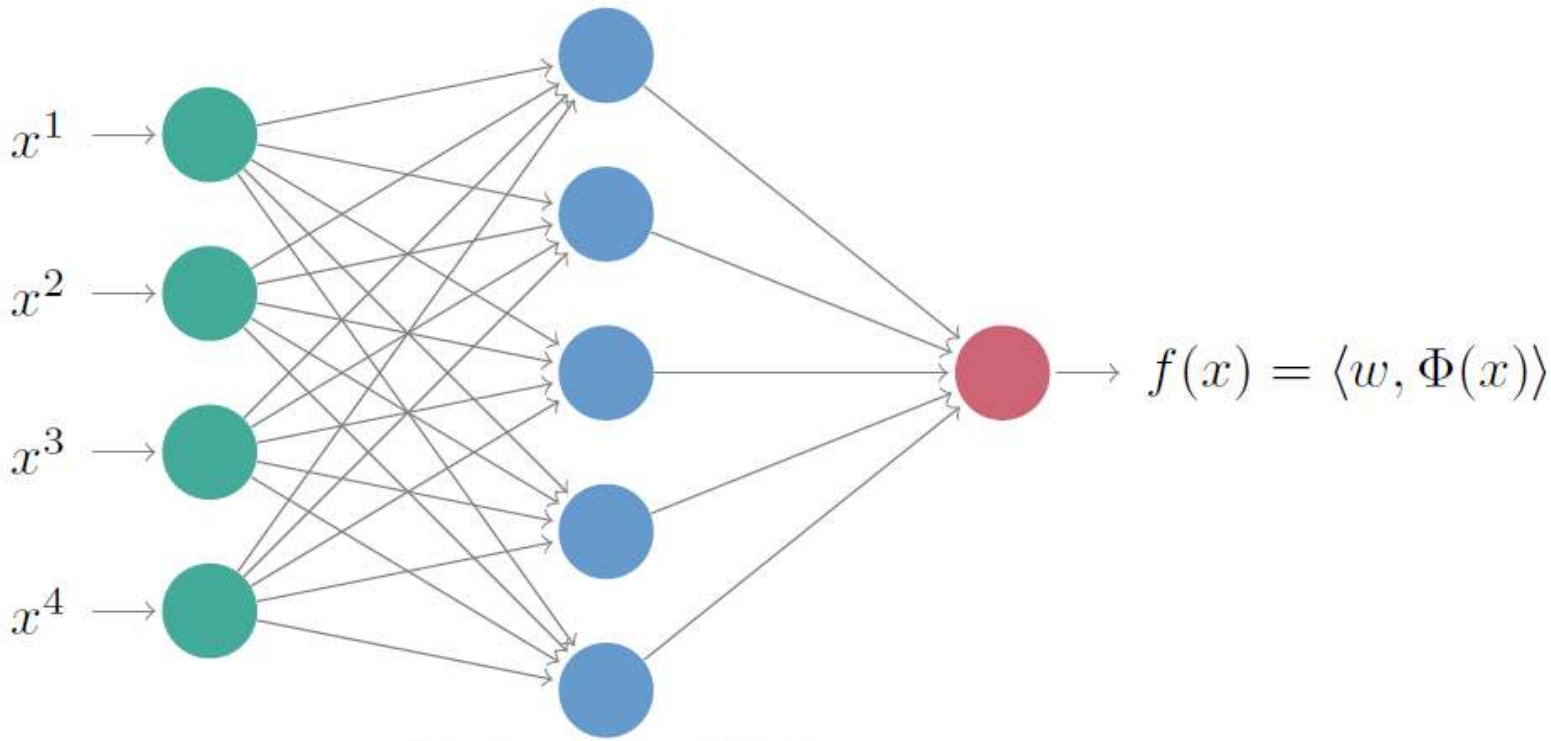
$$y = W^T x + b = \begin{bmatrix} w_1^T x + b_1 \\ w_2^T x + b_2 \\ \vdots \\ w_M^T x + b_M \end{bmatrix}$$

(2):Non-Linear pointwise filtering

$$y = \sigma(W^T x + b) = \begin{bmatrix} \sigma(w_1^T x + b_1) \\ \sigma(w_2^T x + b_2) \\ \vdots \\ \sigma(w_M^T x + b_M) \end{bmatrix}$$

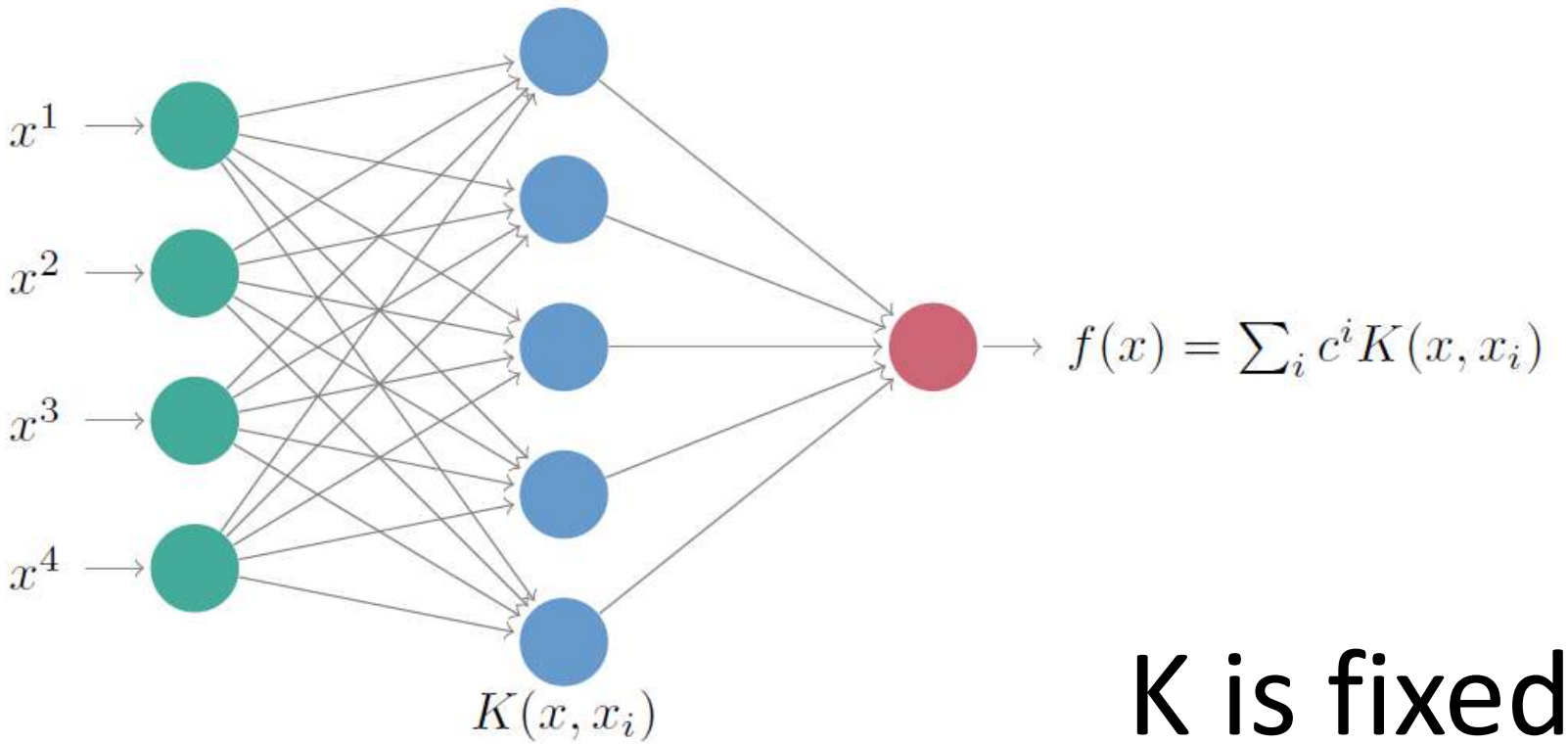
Shallow Neural Network Representations

l_0, input l_1, hidden layer l_L, output layer



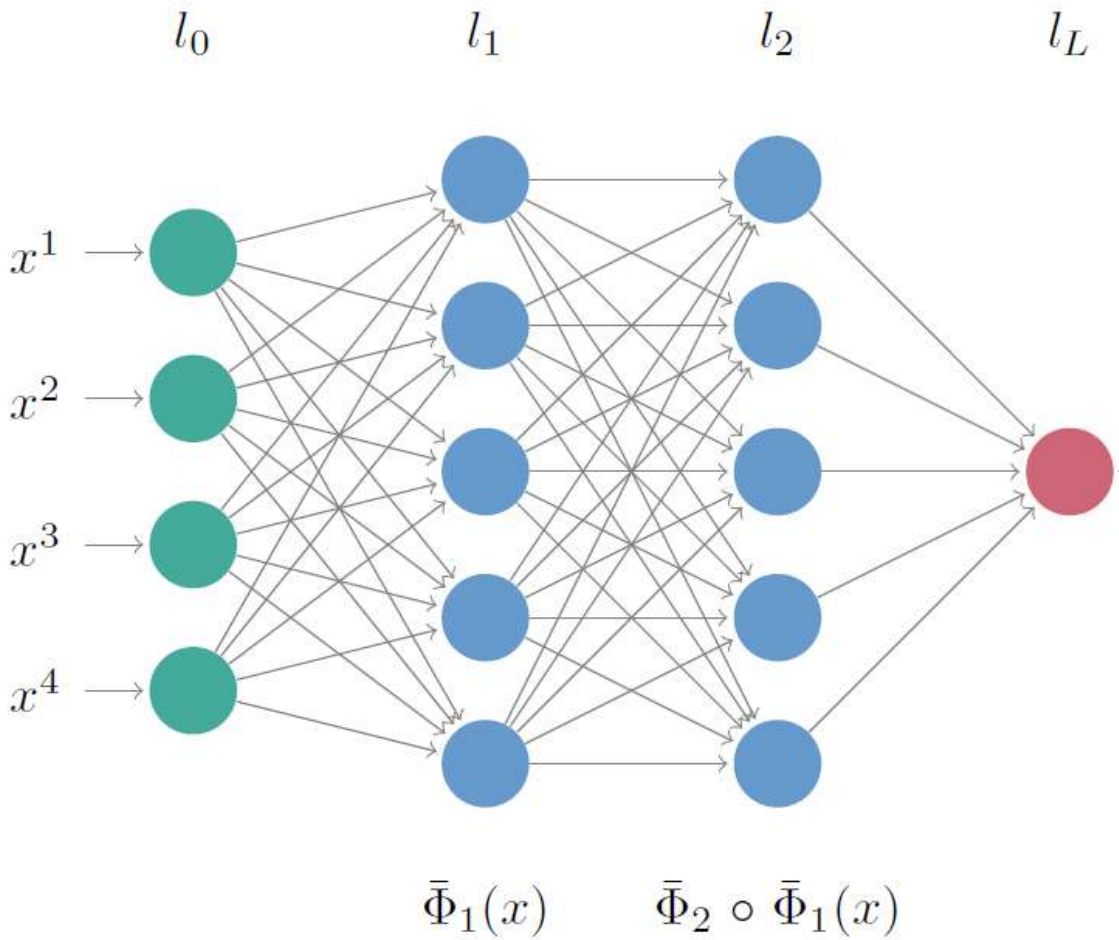
Kernel Machine Representations (e.g. RBF)

l_0, input l_1, hidden layer l_L, output layer



K is fixed!

Deep Neural Network Representations



$$f(x) = \langle w, \Phi_L(x) \rangle$$

$$f(x) = w^T \sigma(W_L^T(\sigma(W_{L-1}^T \cdots \sigma(W_1^T x) \cdots)$$

$$w \in \mathbb{R}^{k_L}, \quad W \in \mathbb{R}^{d \times k_\ell}, \quad \sigma : \mathbb{R} \to \mathbb{R}$$

Link with Kernels: Neural Tangent Kernel

Neural Tangent Kernel:
Convergence and Generalization in Neural Networks

Arthur Jacot
École Polytechnique Fédérale de Lausanne
arthur.jacot@netopera.net

Franck Gabriel
Imperial College London and École Polytechnique Fédérale de Lausanne
franckrgabriel@gmail.com

Clément Hongler
École Polytechnique Fédérale de Lausanne
clement.hongler@gmail.com

Slides by Nolan Dey

https://arxiv.org/abs/1806.07572

Setup

- We have some neural network: $f(x, w)$
  - Input data: $x \in \mathbb{R}^{n \times d}$
  - Network parameters: $w \in \mathbb{R}^{p}$
- Loss: $L(f(x, w), y) = \frac{1}{2}(f(x, w) - y)^2$
- Optimize using full-batch gradient descent

What is the Neural Tangent Kernel (NTK)?

- Infinitely wide neural networks can fit any function

Universal approximation theorem

- Infinite width networks trained to convergence can be described by the NTK
- NTK describes training dynamics: $\frac{df(x,w)}{dt} = -NTK(w_0)(f(x,w) - y)$

Is the kernel associated to the network features

Universal approximation theorem

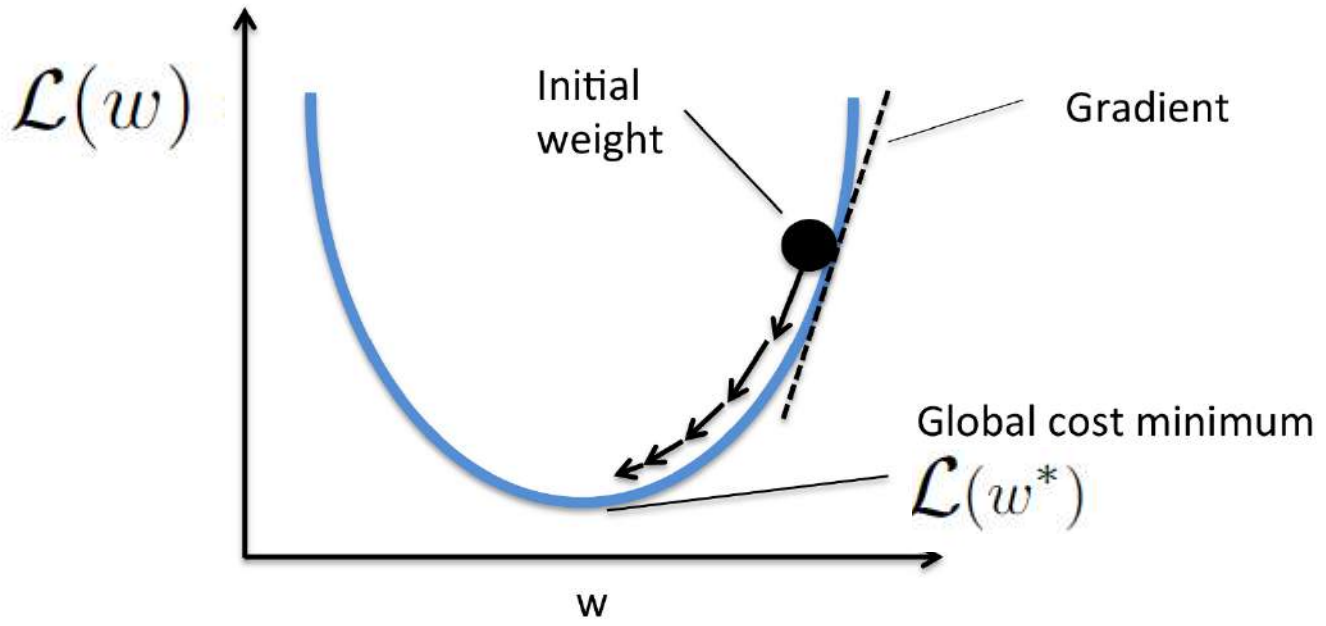
Any **continuous function** defined in the **n-dimensional unit hypercube** may be approximated by a **finite sum** of the type:

$$\sum_{j=1}^{N} v_j \, \varphi \left(\vec{\omega}^{(j)} \cdot \vec{x} + b_j \right),$$

wherein $v_j, b_j \in \mathbb{R}$, $\vec{\omega}^{(j)} \in \mathbb{R}^n$, and φ is a **continuous discriminatory function**.

How big is N?

Finding minima with gradient descent



$$w_{t+1} = w_t - \gamma \frac{d\mathcal{L}(w)}{dw}$$

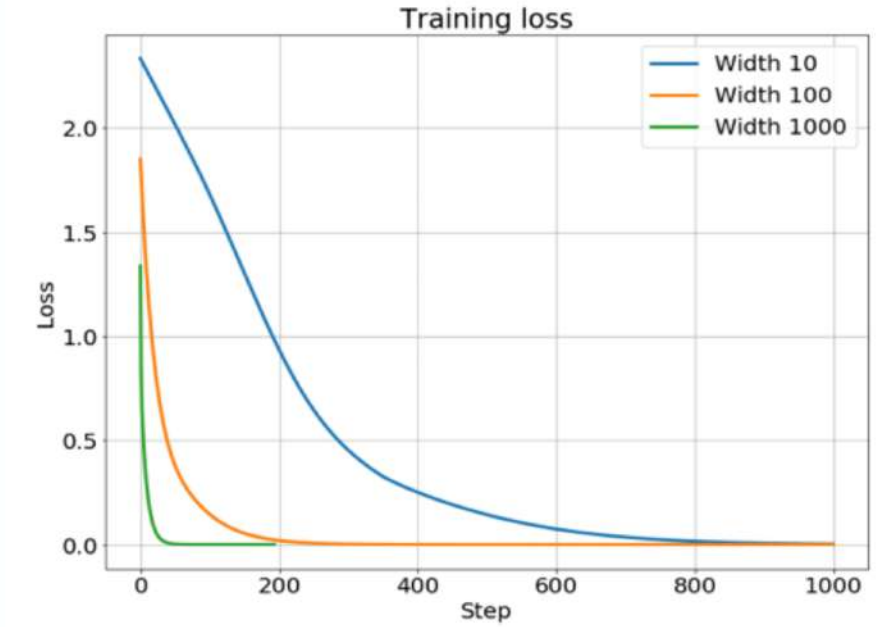
Taylor expansion of neural network

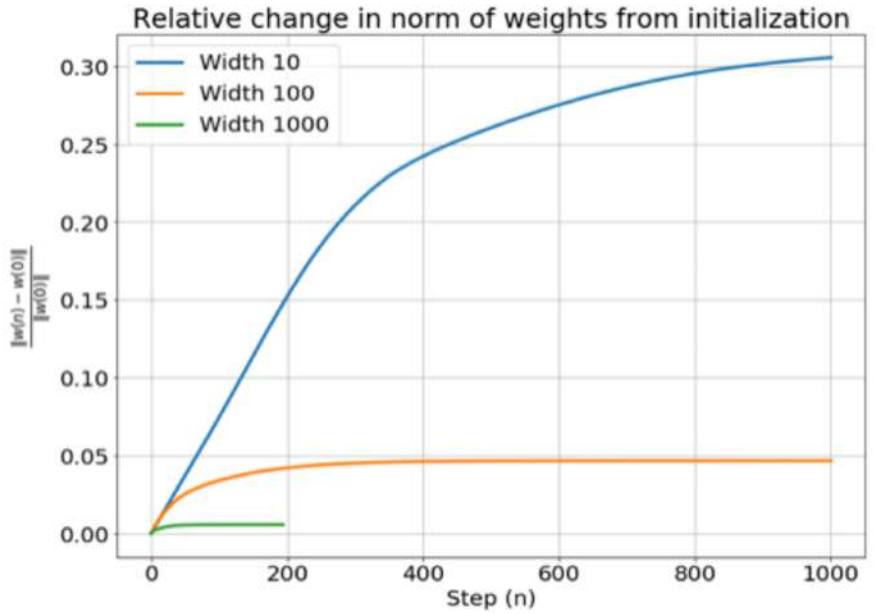
- Taylor expansion of neural network with respect to initialization weights w_0
  - $f(x, w) \approx f(x, w_0) + \nabla_w f(x, w_0)^T (w - w_0)$
- This Taylor expansion is only accurate when weights remain close to initialization

Small learning!

When is Taylor expansion accurate?

- Taylor expansion is only accurate when the weights don't change much during training
  - Weights don't change much when network is sufficiently wide
  - Lazy training = weights don't need to change





This is true when the width of the network is very large. Why?

Gradient Flow

- Gradient descent: $w_{t+1} = w_t - \eta \nabla_w L(w_t)$
- Rewrite as: $\frac{w_{t+1} - w_t}{\eta} = -\nabla_w L(w_t)$
  - Resembles finite difference^
- Take infinitesimally small learning rate η
- Gradient flow! —> $\frac{dw(t)}{dt} = -\nabla_w L(w(t))$

$$
\begin{aligned}
\frac{dw(t)}{dt} &= -\nabla_w \mathcal{L}(w(t), x) \\
&= \nabla_w \frac{1}{N} \sum_i (f(w(t), x_i) - y_i)^2 \\
&= -\frac{2}{N} \sum_i (f(w(t), x_i) - y_i) \nabla_w f(w(t), x_i)
\end{aligned}
$$

This is not static!!!!

$$
\begin{aligned}
\frac{df(w(t),x)}{dt} &= \nabla_w f(w(t),x)\frac{dw(t)}{dt} \\
&= \nabla_w^T f(w(t),x)\frac{-2}{N}\sum_i (f(w(t),x_i)-y_i)\nabla_w f(w(t),x_i) \\
&= -\frac{2}{N}\sum_i (f(w(t),x_i)-y_i)\nabla_w^T f(w(t),x)\nabla_w f(w(t),x_i) \\
&= -\frac{2}{N}\sum_i (f(w(t),x_i)-y_i)K(x,x_i)
\end{aligned}
$$

Feature $\Phi(x_i)$

Supposing quasi-static

If the width is infinite the change in w is very small and K is the kernel associated to the dynamic of the output

Calculate the kernel for

$$f(w,x) = \sum_i v_i \sigma(w_i^T x)$$

What does this mean?

- NTK is a useful tool for studying the dynamics of infinitely wide neural networks
- Successful nets in practice DO NOT OPERATE IN THE NTK REGIME
  - Finite nets still outperform their exactly computed infinite width counterparts
  - SGD vs full-batch gradient descent

This analysis is useful to show:

- Kernel models are not good enough to capture NNs
- Implicit Bias/Regularization (next)

Implicit regularization:NTK solution

$$
\begin{aligned}
\frac{df(\theta)}{dt} &= -\eta \nabla_\theta f(\theta)^\top \nabla_\theta f(\theta) \nabla_f \mathcal{L} \\
&= -\eta \nabla_\theta f(\theta)^\top \nabla_\theta f(\theta) \nabla_f \mathcal{L} \\
&= -\eta K(\theta) \nabla_f \mathcal{L} \\
&= -\eta K_\infty \nabla_f \mathcal{L}
\end{aligned}
$$

NTK solution

$$\frac{df(\theta)}{dt} = -\eta K_\infty (f(\theta) - \mathcal{Y})$$

$$\frac{dg(\theta)}{dt} = -\eta K_\infty g(\theta) \qquad ; \text{let } g(\theta) = f(\theta) - \mathcal{Y}$$

$$\int \frac{dg(\theta)}{g(\theta)} = -\eta \int K_\infty dt$$

$$g(\theta) = Ce^{-\eta K_\infty t}$$

When $t = 0$, we have $C = f(\theta(0)) - \mathcal{Y}$ and therefore,

$$f(\theta) = (f(\theta(0)) - \mathcal{Y})e^{-\eta K_\infty t} + \mathcal{Y} = f(\theta(0))e^{-K_\infty t} + (I - e^{-\eta K_\infty t})\mathcal{Y}$$

The expression dictates how the error changes in time!

Implicit regularization

Solving for f and using the fact that in the large-width limit (1-layer shallow network with many hidden units) the kernel is approximately stationary:

$$f(X, W(t)) \approx (I - e^{-Kt}) \cdot Y.$$

Now, since the kernel K is positive semi-definite, we can take its spectral decomposition $K = Q\Lambda Q^T$ where Q is an orthogonal matrix whose i-th column is the eigenvector q_i of K and Λ is a diagonal matrix whose diagonal entries λ_i are the corresponding eigenvalues. Since $e^{-Kt} = Qe^{-\Lambda t}Q^T$ we have

$$Q^T \left(f(X, W(t)) - Y\right) = -e^{-\Lambda t} Q^T Y,$$

i.e.,

$$\begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ q_N^T \end{bmatrix} \left(f(X, W(t)) - Y\right) = \begin{bmatrix} e^{-\lambda_1 t} & & & \\ & e^{-\lambda_2 t} & \cdots & \\ & & \ddots & \\ & & & e^{-\lambda_N t} \end{bmatrix} \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ q_N^T \end{bmatrix} Y.$$

Implicit regularization

The above equation shows that the convergence rate of $q_i^T(f(X, W(t)) - Y)$, *the error in the* X *predictions*, is determined by the i-th eigenvalue λ_i. Moreover, we can decompose the training error into the eigenspace of the NTK as

$$f(X, W(t)) - Y = \sum_{i=1}^{N} (f(X, W(t)) - Y, q_i) q_i$$

$$= \sum_{i=1}^{N} q_i^T (f(X, W(t)) - Y) q_i$$

$$= \sum_{i=1}^{N} \left(e^{-\lambda_i t} q_i^T Y\right) q_i.$$

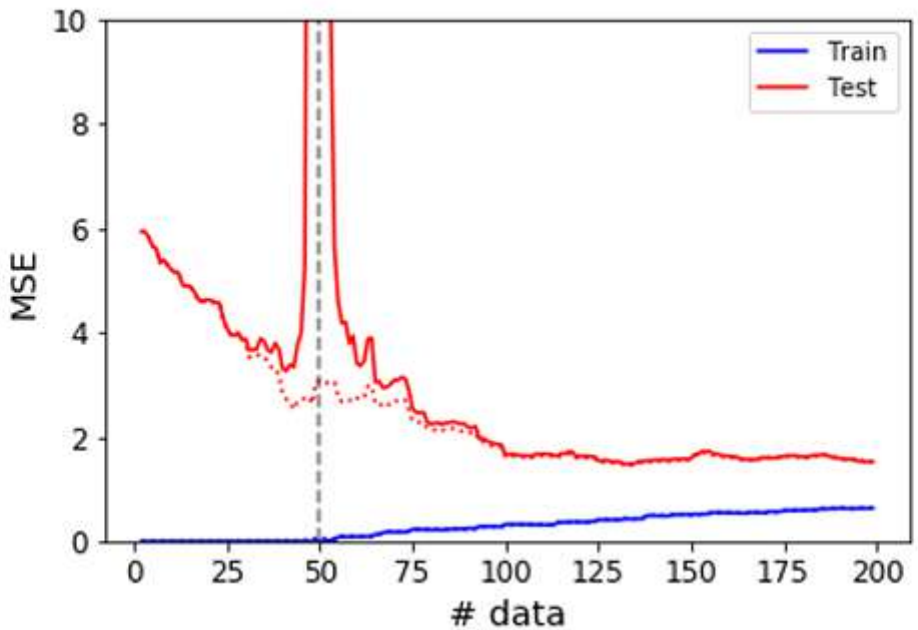
Clearly, the network is biased to first learn the target function along the eigen-directions of the neural tangent kernel with larger eigenvalues, and then the rest components corresponding to smaller eigenvalues.

Double Descent Phenomenon

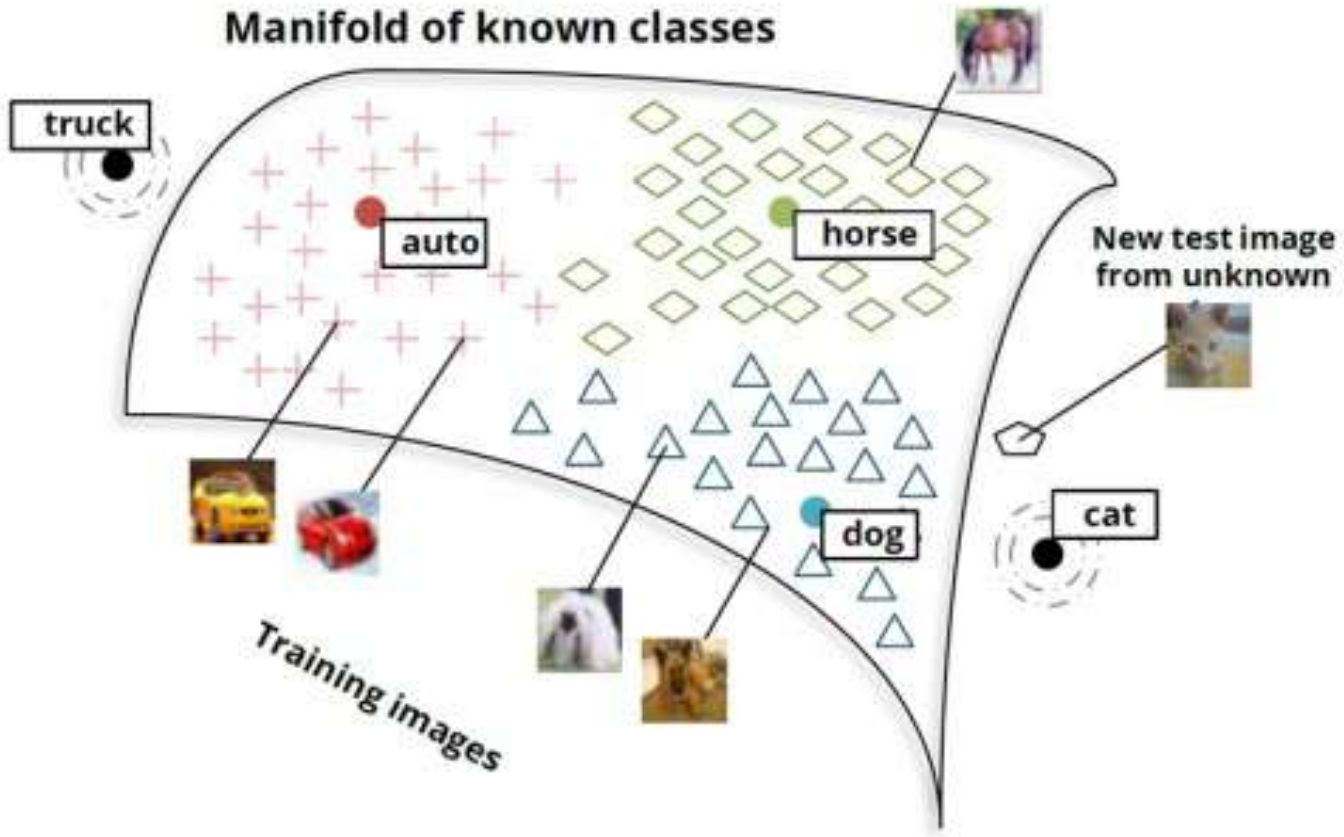
https://openai.com/blog/deep-double-descent/

Intuition:

- **Case 1:** $N \gg D$. There's way more than enough data to pin down the optimal parameters, so it generalizes well.
- **Case 2:** $N \approx D$. It can memorize the training set, but just barely. It might need a large $\|\mathbf{w}\|$ to do so.
- **Case 3:** $N \ll D$. It can fit the training set easily. The implicit regularization of gradient descent makes it do so with a small $\|\mathbf{w}\|$.



Intuition: data manifold hypothesis

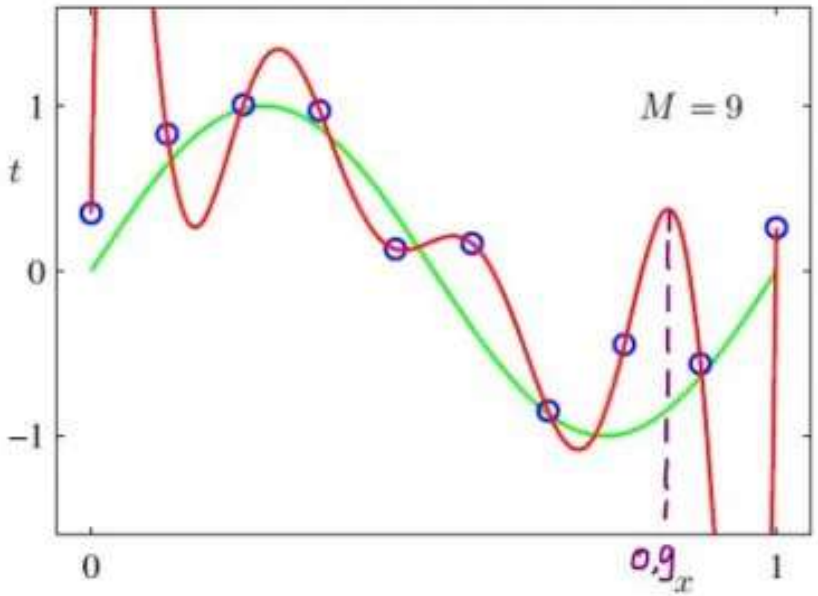


How can we understand if the network separator makes sense?

The network is capable to grasp the geometry of the manifold without overfitting.
How is this possible?

Ansuini et al.

Explicit regularization: lp norm



$$h_a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

$$\mathcal{L}(w) = \sum_{i=1}^{N} (y_i - h(x_i))^2 + \lambda \|a\|_p$$

What is the effect of the regularization on the loss surface?

Weight decay and hessian interpretation

- We often refer to $\mathbf{H}$ as the curvature of a function.
- Suppose you move along a line defined by $\boldsymbol{\theta} + t\mathbf{v}$ for some vector $\mathbf{v}$.
- Second-order Taylor approximation:

$$\mathcal{J}(\boldsymbol{\theta} + t\mathbf{v}) \approx \mathcal{J}(\boldsymbol{\theta}) + t\nabla\mathcal{J}(\boldsymbol{\theta})^\top\mathbf{v} + \frac{t^2}{2}\mathbf{v}^\top\mathbf{H}(\boldsymbol{\theta})\mathbf{v}$$

- Hence, in a direction where $\mathbf{v}^\top\mathbf{H}\mathbf{v} > 0$, the cost function curves upwards, i.e. has positive curvature. Where $\mathbf{v}^\top\mathbf{H}\mathbf{v} < 0$, it has negative curvature.

Hessian matrix

- The Hessian matrix, denoted $\mathbf{H}$, or $\nabla^2 \mathcal{J}$ is the matrix of second derivatives:

$$\mathbf{H} = \nabla^2 \mathcal{J} = \begin{pmatrix} \frac{\partial^2 \mathcal{J}}{\partial \theta_1^2} & \frac{\partial^2 \mathcal{J}}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 \mathcal{J}}{\partial \theta_1 \partial \theta_D} \\ \frac{\partial^2 \mathcal{J}}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \mathcal{J}}{\partial \theta_2^2} & \cdots & \frac{\partial^2 \mathcal{J}}{\partial \theta_2 \partial \theta_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{J}}{\partial \theta_D \partial \theta_1} & \frac{\partial^2 \mathcal{J}}{\partial \theta_D \partial \theta_2} & \cdots & \frac{\partial^2 \mathcal{J}}{\partial \theta_D^2} \end{pmatrix}$$

- It's a symmetric matrix because $\frac{\partial^2 \mathcal{J}}{\partial \theta_i \partial \theta_j} = \frac{\partial^2 \mathcal{J}}{\partial \theta_j \partial \theta_i}$.

Assuming no bias parameter, the objective function is:

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^\top w + J(w; X, y)$$

with the gradient:

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y).$$

A single gradient update step is:

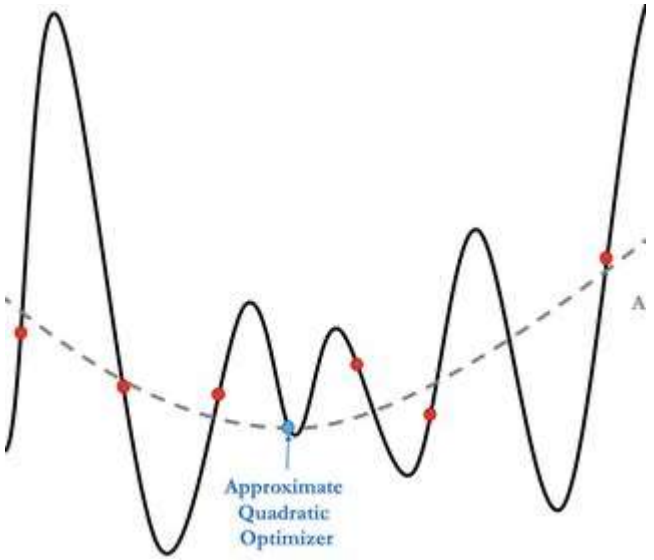
$$w \leftarrow w - \epsilon(\alpha w + \nabla_w J(w; X, y)),$$

or equivalently:

$$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y).$$

This shows that weight decay modifies the learning rule by shrinking the weights by a factor of $(1 - \epsilon\alpha)$ before the gradient update. Further analysis considers a quadratic approximation around the unregularized minimum w^*.

A simplification



The approximation of the objective function $\tilde{J}$ is given by:

$$\tilde{J}(\theta) = J(w^*) + \frac{1}{2}(w - w^*)^\top H(w - w^*),$$

where H is the Hessian matrix of J with respect to w, evaluated at w^*. Since w^* is defined to be a minimum, the first-order term vanishes. The Hessian H is positive semidefinite because w^* is a minimum.

The minimum of $\tilde{J}$ occurs where its gradient:

$$\nabla_w \tilde{J}(w) = H(w - w^*) \qquad *$$

is equal to zero.

To study the effect of weight decay, we modify equation (*) by adding the weight decay gradient and solve for the minimum of the regularized objective function. The location of the minimum is represented by $\tilde{w}$. (the new minimum!)

Starting from:

$$\alpha\tilde{w} + H(\tilde{w} - w^*) = 0,$$

we can rewrite this as:

$$(H + \alpha I)\tilde{w} = Hw^*,$$

leading to:

$$\tilde{w} = (H + \alpha I)^{-1} Hw^*.$$

As $\alpha \to 0$, $\tilde{w}$ approaches w^*. To explore what happens as α increases, we decompose H into $Q\Lambda Q^\top$, where Q is an orthonormal matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues. Applying this decomposition:

$$\tilde{w} = (Q\Lambda Q^\top + \alpha I)^{-1} Q\Lambda Q^\top w^*,$$

simplifying to:

$$\tilde{w} = Q(\Lambda + \alpha I)^{-1} \Lambda Q^\top w^*.$$

$$\tilde{w} = Q(\Lambda + \alpha I)^{-1}\Lambda Q^\top w^*$$

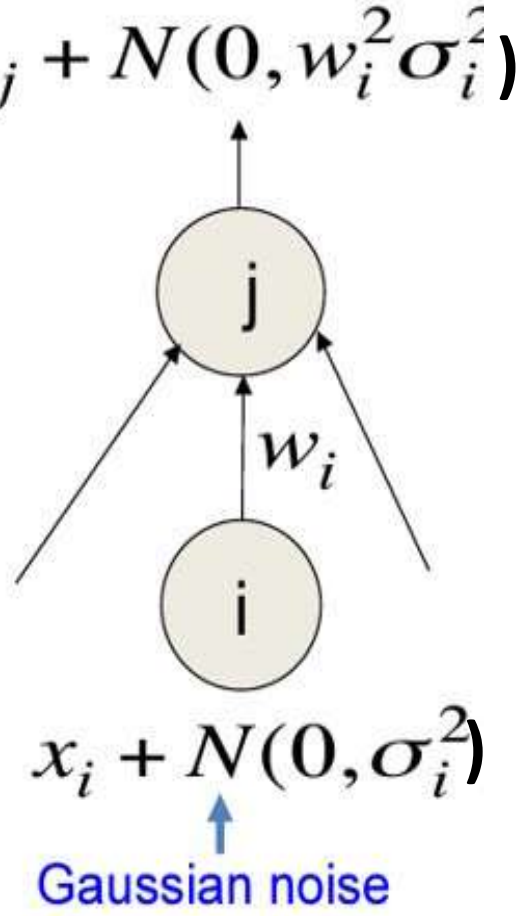
This shows that weight decay scales w^* along the eigenvector directions of H. Each component along the i-th eigenvector of H is rescaled by a factor of $\frac{\lambda_i}{\lambda_i+\alpha}$.

- If $\lambda_i \gg \alpha$, regularization has little effect.
- If $\lambda_i \ll \alpha$, components are shrunk significantly, approaching zero.

This explains how regularization influences directions with small eigenvalues, shrinking their magnitudes more aggressively.

L2 regularization via noisy inputs

- Suppose we add Gaussian noise to the inputs.
  - The variance of the noise is amplified by the squared weight before going into the next layer.
- In a simple net with a linear output unit directly connected to the inputs, the amplified noise gets added to the output.
- This makes an additive contribution to the squared error.
  - So minimizing the squared error tends to minimize the squared weights when the inputs are noisy.



output on one case → $y^{noisy} = \sum_i w_i x_i + \sum_i w_i \varepsilon_i$ *where* ε_i *is sampled from* $N(0, \sigma_i^2)$

$$E\left[(y^{noisy} - t)^2\right] = E\left[\left(y + \sum_i w_i \varepsilon_i - t\right)^2\right] = E\left[\left((y - t) + \sum_i w_i \varepsilon_i\right)^2\right]$$

$$= (y - t)^2 + E\left[2(y - t)\sum_i w_i \varepsilon_i\right] + E\left[\left(\sum_i w_i \varepsilon_i\right)^2\right]$$

$$= (y - t)^2 + E\left[\sum_i w_i^2 \varepsilon_i^2\right]$$

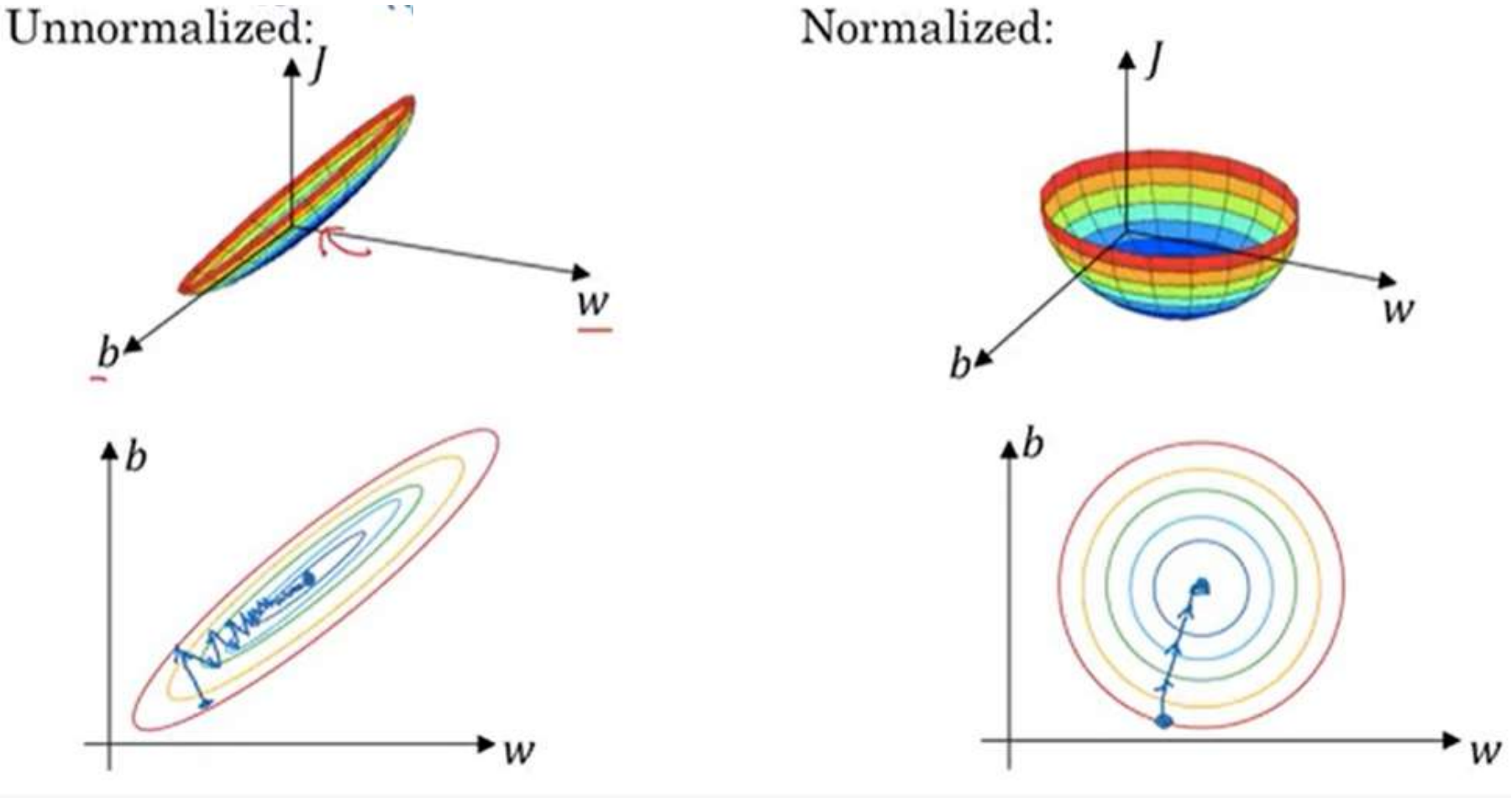
because ε_i *is independent of* ε_j
and ε_i *is independent of* $(y - t)$

$$= (y - t)^2 + \sum_i w_i^2 \sigma_i^2$$

So σ_i^2 is equivalent to an L2 penalty

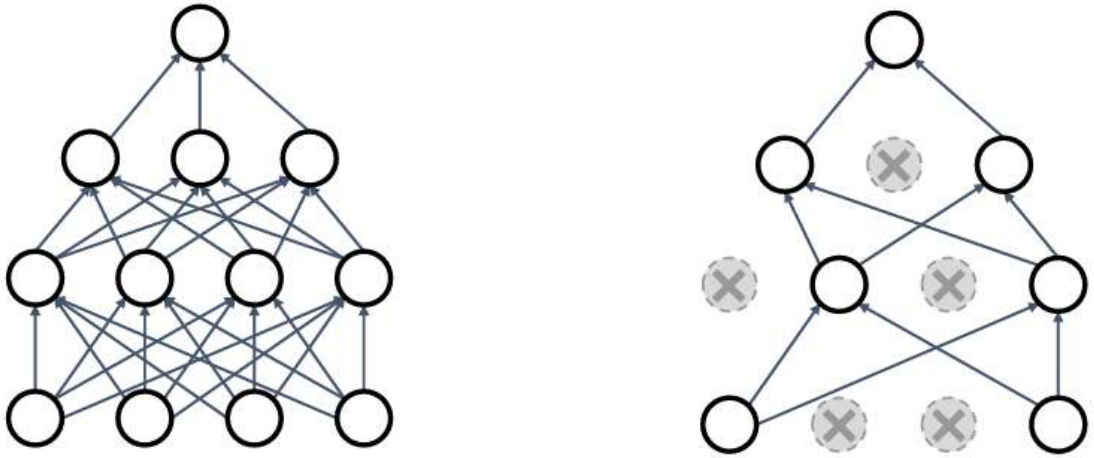
Is this true for more complex nets?

Data normalization changes the geometry of the loss

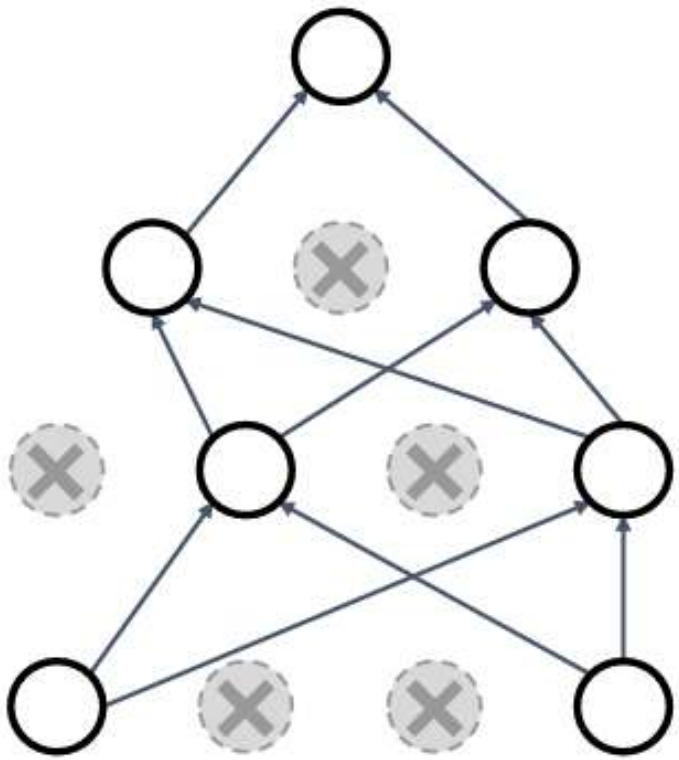


Another example of explicit NNs regularization: Dropout

In each forward pass, randomly set some neurons to zero
Probability of dropping is a hyperparameter; 0.5 is common



Regularization: Dropout



Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

An FC layer with 4096 units has
$2^{4096} \sim 10^{1233}$ possible masks!
Only $\sim 10^{82}$ atoms in the universe...

Example with regression: minimum norm solution

$A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $m \gg n$

$Ax = y$

Parameters are more than data

one particular solution is

$$x_{\ln} = A^T(AA^T)^{-1}y$$

Moore-Penrose pseudo-inverse
(a possible algorithm)

(AA^T is invertible since A full rank)

in fact, $x_{\ln}$ is the solution of $y = Ax$ that minimizes $\|x\|$

i.e., $x_{\ln}$ is solution of optimization problem

$$\begin{array}{ll} \text{minimize} & \|x\| \\ \text{subject to} & Ax = y \end{array}$$

(with variable $x \in \mathbf{R}^n$)

Example with **regression**: close form is minimum norm solution

suppose $Ax = y$, so $A(x - x_{\ln}) = 0$ and

$$\begin{aligned}(x - x_{\ln})^T x_{\ln} &= (x - x_{\ln})^T A^T (AA^T)^{-1} y \\ &= (A(x - x_{\ln}))^T (AA^T)^{-1} y \\ &= 0\end{aligned}$$

i.e., $(x - x_{\ln}) \perp x_{\ln}$, so

$$\|x\|^2 = \|x_{\ln} + x - x_{\ln}\|^2 = \|x_{\ln}\|^2 + \|x - x_{\ln}\|^2 \geq \|x_{\ln}\|^2$$

i.e., $x_{\ln}$ has smallest norm of any solution

Regression: minimum norm solution with GD

Let cost function $f : \mathbb{R}^n \to \mathbb{R}$ be defined by

$$f(\mathrm{x}) := \frac{1}{2}\|\mathrm{Ax} - \mathrm{b}\|_2^2$$

$$A \in \mathbb{R}^{m \times n}, \; x \in \mathbb{R}^n, \quad m \gg n$$

whose gradient is

$$\nabla f(\mathrm{x}) = \mathrm{A}^\top (\mathrm{Ax} - \mathrm{b})$$

Using gradient descent with step $\mu > 0$,

$$\begin{aligned} \mathrm{x}_{k+1} &= \mathrm{x}_k - \mu \nabla f(\mathrm{x}_k) \\ &= \left(\mathrm{I} - \mu \mathrm{A}^\top \mathrm{A}\right) \mathrm{x}_k + \mu \mathrm{A}^\top \mathrm{b} \end{aligned}$$

Hence,

$$\mathrm{x}_k = \left(\mathrm{I} - \mu \mathrm{A}^\top \mathrm{A}\right)^k \mathrm{x}_0 + \mu \sum_{\ell=0}^{k-1} \left(\mathrm{I} - \mu \mathrm{A}^\top \mathrm{A}\right)^\ell \mathrm{A}^\top \mathrm{b}$$

$$\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^\top = \mathrm{U}\begin{bmatrix}\Sigma_1 & \mathrm{O}\end{bmatrix}\begin{bmatrix}\mathrm{V}_1^\top \\ \mathrm{V}_2^\top\end{bmatrix} = \mathrm{U}\Sigma_1\mathrm{V}_1^\top$$

Letting $\mathrm{y} := \mathrm{V}^\top \mathrm{x}$, we rewrite

$$\begin{aligned} \mathrm{y}_k &= \left(\mathrm{I} - \mu \Sigma^\top \Sigma\right)^k \mathrm{y}_0 + \mu \sum_{\ell=0}^{k-1} \left(\mathrm{I} - \mu \Sigma^\top \Sigma\right)^\ell \Sigma^\top \mathrm{U}^\top \mathrm{b} \\ &= \begin{bmatrix} \left(\mathrm{I} - \mu \Sigma_1^2\right)^k & \mathrm{O} \\ \mathrm{O} & \mathrm{I} \end{bmatrix} \mathrm{y}_0 + \mu \sum_{\ell=0}^{k-1} \begin{bmatrix} \left(\mathrm{I} - \mu \Sigma_1^2\right)^\ell & \mathrm{O} \\ \mathrm{O} & \mathrm{I} \end{bmatrix} \begin{bmatrix} \Sigma_1 \\ \mathrm{O} \end{bmatrix} \mathrm{U}^\top \mathrm{b} \\ &= \begin{bmatrix} \left(\mathrm{I} - \mu \Sigma_1^2\right)^k & \mathrm{O} \\ \mathrm{O} & \mathrm{I} \end{bmatrix} \mathrm{y}_0 + \mu \sum_{\ell=0}^{k-1} \begin{bmatrix} \left(\mathrm{I} - \mu \Sigma_1^2\right)^\ell \Sigma_1 \\ \mathrm{O} \end{bmatrix} \mathrm{U}^\top \mathrm{b} \end{aligned}$$

Example with **regression**: minimum norm solution with GD

Choosing $\mu > 0$ such that all eigenvalues of $\mathrm{I} - \mu\Sigma_1^2$ are strictly inside the unit circle, then $\mathrm{y}_k \to \mathrm{y}_\infty$, where

$$\mathrm{y}_\infty = \begin{bmatrix} \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{I} \end{bmatrix} \mathrm{y}_0 + \mu \sum_{\ell=0}^{\infty} \begin{bmatrix} \left(\mathrm{I} - \mu\Sigma_1^2\right)^\ell \Sigma_1 \\ \mathrm{O} \end{bmatrix} \mathrm{U}^\top \mathrm{b}$$

where

$$\mu \sum_{\ell=0}^{\infty} \left(\mathrm{I} - \mu\Sigma_1^2\right)^\ell \Sigma_1 = \mu\left(\mathrm{I} - \mathrm{I} + \mu\Sigma_1^2\right)^{-1} \Sigma_1 = \Sigma_1^{-1}$$

and, thus,

$$\mathrm{y}_\infty = \begin{bmatrix} \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{I} \end{bmatrix} \mathrm{y}_0 + \begin{bmatrix} \Sigma_1^{-1} \\ \mathrm{O} \end{bmatrix} \mathrm{U}^\top \mathrm{b}$$

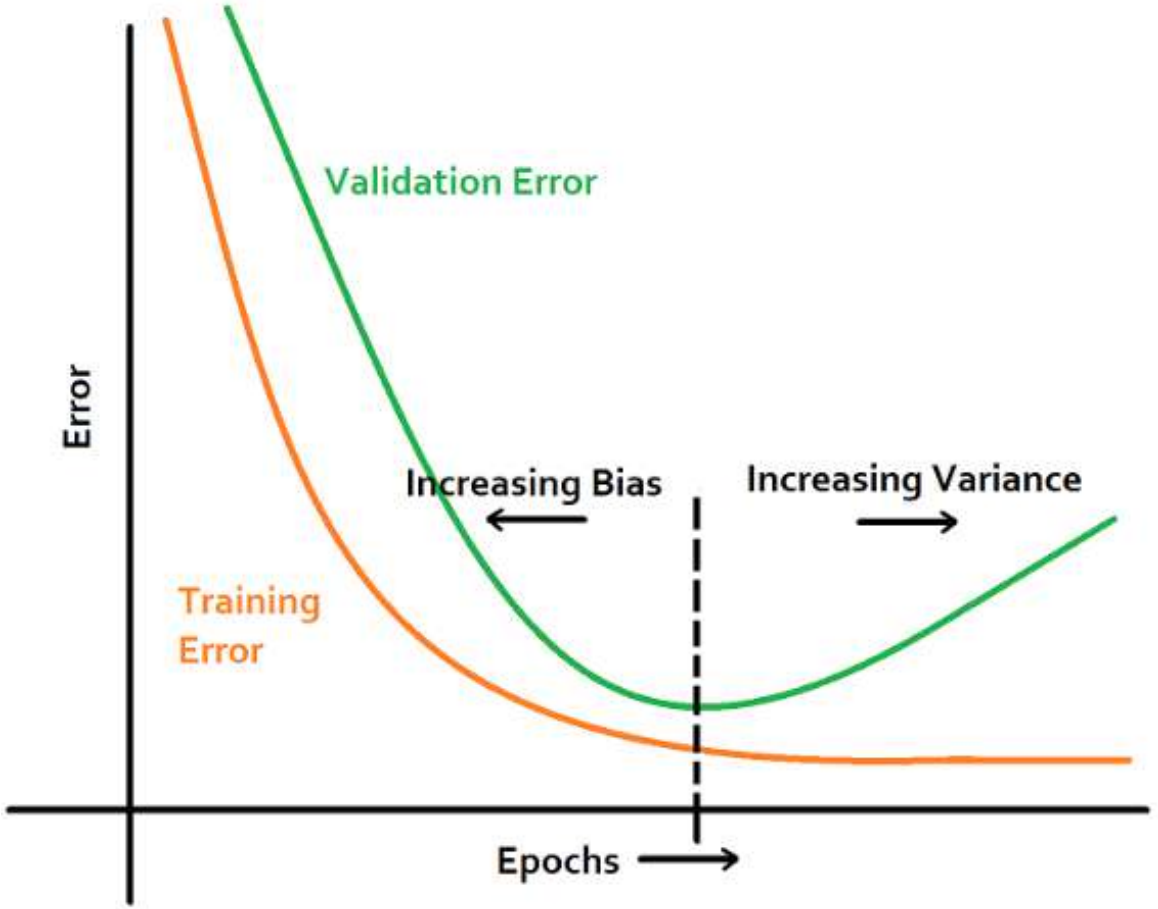
Since $\mathrm{x} := \mathrm{V}\mathrm{y}$,

$$\boxed{\mathrm{x}_\infty = \mathrm{V}_2 \mathrm{V}_2^\top \mathrm{x}_0 + \underbrace{\mathrm{V}_1 \Sigma_1^{-1} \mathrm{U}^\top \mathrm{b}}_{=\mathrm{x}_{\mathrm{LN}}}}$$

This is the pseudo-inverse

Therefore, we conclude that if x_0 is orthogonal to the null space of A, then gradient descent will converge to the least-norm solution.

Early stopping, L2 norm and learning rate



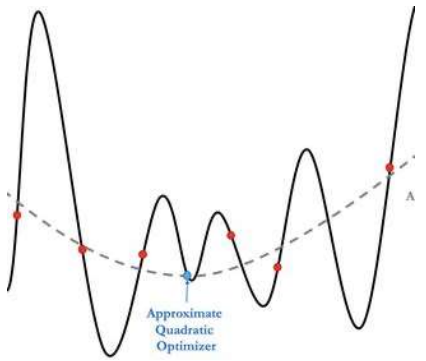
From Taylor series expansion, we will make a quadratic approximation around the empirically optimal value of the weights $\boldsymbol{w}^*$. The matrix $\mathbf{H}$ is the hessian matrix.

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*),$$

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*).$$

Since $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \Big|_{\boldsymbol{w}^*} = 0$$



The gradient of this approximation comes out to be a linear function of H and w. Now, we can calculate the weights at each step of the gradient descent.

$$\begin{aligned} \boldsymbol{w}^{(\tau)} &= \boldsymbol{w}^{(\tau-1)} - \epsilon \nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}^{(\tau-1)}) \\ &= \boldsymbol{w}^{(\tau-1)} - \epsilon \boldsymbol{H}(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \\ \boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* &= (\boldsymbol{I} - \epsilon \boldsymbol{H})(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*). \end{aligned}$$

Since the Hessian matrix is real, symmetric and positive semi-definite. By eigenvalue decomposition. matrix H can be written as,

$$\begin{aligned} \boldsymbol{H} &= \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \\ \boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* &= (\boldsymbol{I} - \epsilon \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top)(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \\ \boldsymbol{Q}^\top(\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^*) &= (\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})\boldsymbol{Q}^\top (\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \end{aligned}$$

Assuming ε is chosen to be small and $w(0) = 0$. The weights after τ **iterations** are:

$$\boldsymbol{Q}^\top \boldsymbol{w}^{(\tau)} = [\boldsymbol{I} - (\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})^\tau] \boldsymbol{Q}^\top \boldsymbol{w}^* \qquad (1)$$

The optimal weights obtained on imposing L2 norm penalty on the objective function is given by (refer):

$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^*$$
$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = [\boldsymbol{I} - (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \alpha] \boldsymbol{Q}^\top \boldsymbol{w}^* \qquad (2)$$

Comparing Eq(1) and Eq(2), we derive the following relation between τ, **w, and** ε**.** The equivalence can be seen between the L2 regularization and early stopping.

$$(\boldsymbol{I} - \epsilon \boldsymbol{\Lambda})^\tau = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \alpha,$$

Taking log both sides and using the series approximation of log(1+x), we can conclude that if all λi are small (that is, ελi << 1 and λi/α << 1) then the following equation holds.

$$\tau \approx \frac{1}{\epsilon \alpha},$$
$$\alpha \approx \frac{1}{\tau \epsilon}.$$

Here, α is the regularization constant, τ is no. of iterations, and ε is the learning rate.

$$1 - \frac{\alpha}{\Lambda + \alpha I} = \frac{\Lambda + \alpha I - \alpha I}{\Lambda + \alpha I} = \frac{\Lambda}{\Lambda + \alpha I}$$

$$\log(1 - x) \approx x$$

$$\tau \log(I - \epsilon \Lambda) = \log(\alpha) - \log(\Lambda + \alpha I)$$
$$\tau \log(I - \epsilon \Lambda) = \log(\alpha) - \log(\alpha(\frac{\Lambda}{\alpha} + I))$$
$$\tau \log(I - \epsilon \Lambda) = -\log(\frac{\Lambda}{\alpha} + I)$$
$$\tau \epsilon \Lambda = \frac{\lambda}{\alpha}$$

Increasing no. of epochs/iterations τ is equivalent to reducing the regularization constant. Similarly, early stopping the model, i.e. reducing the no. of iterations is similar to L2 regularization with large α. Thus, we can say that early stopping regularizes the model.

References:

Deep Learning (Adaptive Computation and Machine Learning series) — By Ian Goodfellow, Yoshua Bengio, Aaron Courville

Weights initialization

Weight Initialization: Xavier Initialization

"Xavier" initialization:
std = 1/sqrt(Din)

Derivation: Variance of output = Variance of input

$y = Wx$ $\qquad y_i = \sum_{j=1}^{Din} x_j w_j$

$\text{Var}(y_i) = Din * \text{Var}(x_i w_i)$ [Assume x, w are iid]

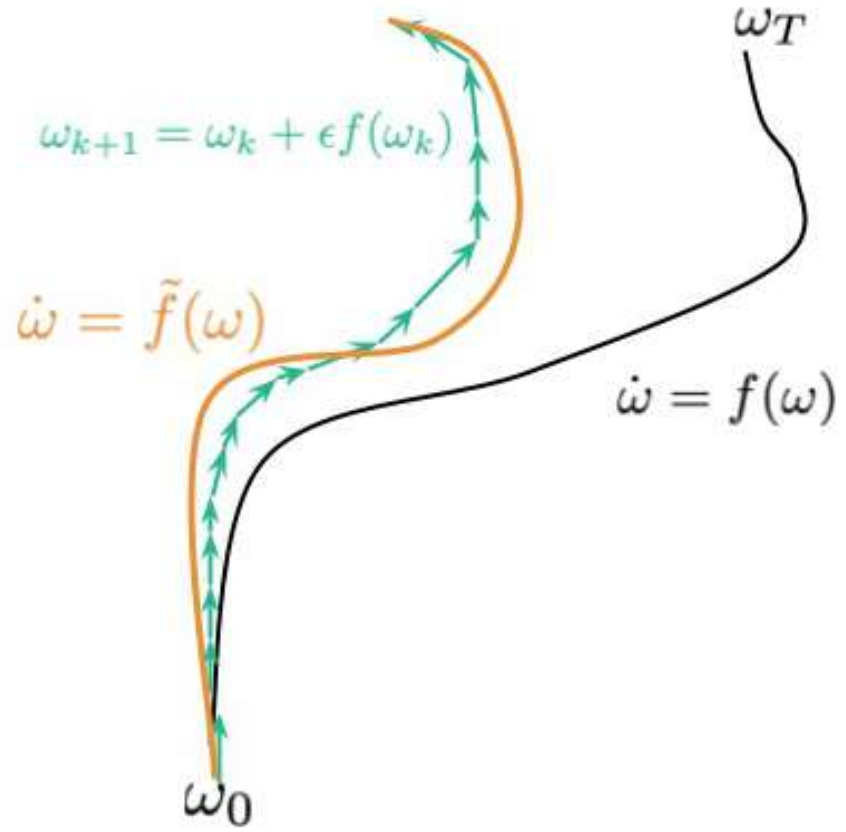
$= Din * (E[x_i^2]E[w_i^2] - E[x_i]^2 E[w_i]^2)$ [Assume x, w independent]

$= Din * \text{Var}(x_i) * \text{Var}(w_i)$ [Assume x, w are zero-mean]

If $\text{Var}(w_i) = 1/Din$ then $\text{Var}(y_i) = \text{Var}(x_i)$

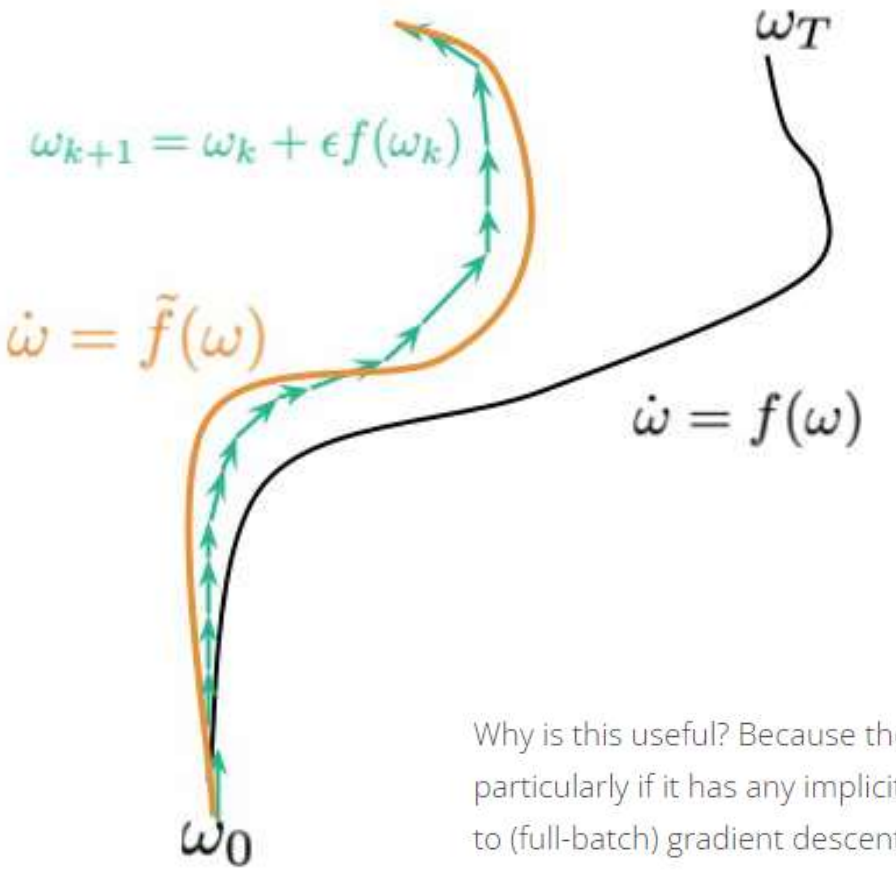
Effectively reduce the complexity of the model!!

Discretization in GD and implicit regularization



Let's say we have a differential equation $\dot{\omega} = f(\omega)$. The solution to this ODE with initial condition ω_0 is a continuous trajectory ω_t, shown in the image in black. We usually can't compute this solution in closed form, and instead simulate the ODE using the Euler's method, $\omega_{k+1} = \omega_k + \epsilon f(\omega_k)$. This results in a discrete trajectory shown in teal. Due to discretization error, for finite stepsize ϵ, this discrete path may not lie exactly where the continuous black path lies. Errors accumulate over time, as shown in this illustration. The goal of backward error analysis is to find a different ODE, $\dot{\omega} = \tilde{f}(\omega)$ such that the approximate discrete path we got from Euler's method lieas near the the continuous path which solves this new ODE. Our goal is to reverse engineer a modified $\tilde{f}$ such that the discrete iteration can be well-modelled by an ODE.

See also: https://arxiv.org/pdf/2101.12176.pdf , https://www.inference.vc/notes-on-the-origin-of-implicit-regularization-in-stochastic-gradient-descent/



IMPLICIT GRADIE REGULARIZATION DUE TO DISCRETIZATION

Why is this useful? Because the form $\tilde{f}$ takes can reveal interesting aspects of the behaviour of the discrete algorithm, particularly if it has any implicit bias towards moving into different areas of the space. When the authors apply this technique to (full-batch) gradient descent, it already suggests the kind of implicit regularization bias gradient descent has.

In Gradient descent with a cost function C, the original ODE is $f(\omega) = -\nabla C(\omega)$. The modified ODE which corresponds to a finite stepsize ϵ takes the form $\dot{\omega} = -\nabla \tilde{C}_{GD}(\omega)$ where

$$\tilde{C}_{GD}(\omega) = C(\omega) + \frac{\epsilon}{4}\|\nabla C(\omega)\|^2$$

Problem Setup

Given a continuous flow represented by an ordinary differential equation (ODE) of the form:

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

we are interested in finding a modified loss function that accounts for the discretization error when integrating this ODE using a discrete method, like Euler's method.

Discretization Error

For simplicity, let's assume that the ODE is solved using a discrete update rule, such as the explicit Euler method:

$$\omega_{i+1} = \omega_i + \epsilon f(\omega_i)$$

where ϵ is the step size (e.g., the learning rate in gradient descent). This discrete step introduces an approximation error because the true solution of the continuous flow is not exactly reached by the discrete step.

Backward Error Analysis

The goal is to find a modified flow (and hence, a modified loss function) such that the continuous flow with this modified loss matches the discrete flow to a given order in ϵ. This means we want to express the continuous flow as:

$$\dot{\omega} = f_{\text{mod}}(\omega) = f(\omega) + \epsilon f_1(\omega) + \epsilon^2 f_2(\omega) + \dots$$

where the correction terms $f_1(\omega), f_2(\omega), \dots$ account for the discretization error at different orders of ϵ.

Derivation of the First Correction Term

We start by expanding the discrete step ω_{i+1} in a Taylor series about ω_i:

$$\omega_{i+1} = \omega_i + \epsilon\dot{\omega}_i + \frac{\epsilon^2}{2}\ddot{\omega}_i + \mathcal{O}(\epsilon^3)$$

where $\dot{\omega}_i = f(\omega_i)$ and $\ddot{\omega}_i$ is the time derivative of $\dot{\omega}_i$. Using the chain rule, $\ddot{\omega}_i$ is:

$$\ddot{\omega}_i = \frac{d}{dt}f(\omega) = \nabla f(\omega) \cdot \dot{\omega} = \nabla f(\omega) f(\omega)$$

Now we match this Taylor expansion to the discrete update $\omega_{i+1} = \omega_i + \epsilon f(\omega_i)$, which only has terms of order ϵ. To ensure consistency between the continuous and discrete flows, we need to add correction terms to the continuous flow at higher orders in ϵ.

At order ϵ^2, the correction term $f_1(\omega)$ is derived by requiring that the Taylor expansion of the continuous flow matches the discrete update. This leads to the condition:

$$f_1(\omega) + \frac{1}{2}\nabla f(\omega) f(\omega) = 0$$

Solving for $f_1(\omega)$, we obtain:

$$f_1(\omega) = -\frac{1}{2}\nabla f(\omega) f(\omega)$$

General Correction to the Loss

To interpret this result in terms of a correction to the **loss function**, we note that the flow $\dot{\omega} = f(\omega)$ is typically derived as the gradient of a loss function $C(\omega)$:

$$f(\omega) = -\nabla C(\omega)$$

Substituting this into the expression for $f_1(\omega)$, we find that the correction term becomes:

$$f_1(\omega) = -\frac{1}{2}\nabla\nabla C(\omega)\nabla C(\omega)$$

This is exactly the gradient of $\frac{1}{4}||\nabla C(\omega)||^2$, meaning the corrected flow is:

$$\dot{\omega} = -\nabla C(\omega) - \frac{\epsilon}{4}\nabla||\nabla C(\omega)||^2$$

Thus, the **corrected loss function** to match the discrete flow to the continuous flow up to order ϵ is:

$$C_{\text{mod}}(\omega) = C(\omega) + \frac{\epsilon}{4}||\nabla C(\omega)||^2$$

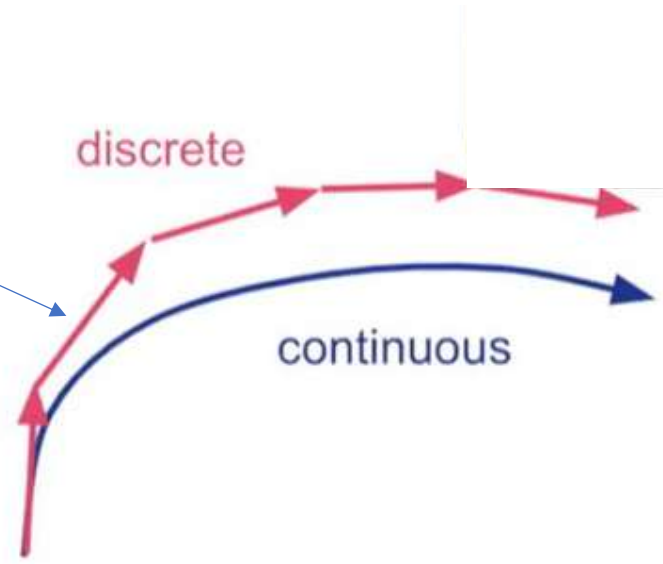
Discretization in GD and implicit regularization

Intuition:

On average, the gradient is half a step stale

Taylor expansion

$$\nabla C(\omega + (\epsilon/2)\nabla C) \approx \nabla C + (\epsilon/2)\nabla\nabla C \nabla C$$
$$= \nabla C + (\epsilon/4)\nabla\left(||\nabla C||^2\right)$$



For "small finite learning rates":

$$\dot{\omega} = -\nabla \widetilde{C}_{GD}(\omega) + O(\epsilon^2)$$
$$\widetilde{C}_{GD}(\omega) = C(\omega) + (\epsilon/4)||\nabla C(\omega)||^2$$

Original loss

Implicit regularizer

$$\mathcal{L}(w, X) = \sum_i l_i(w, x_i)$$

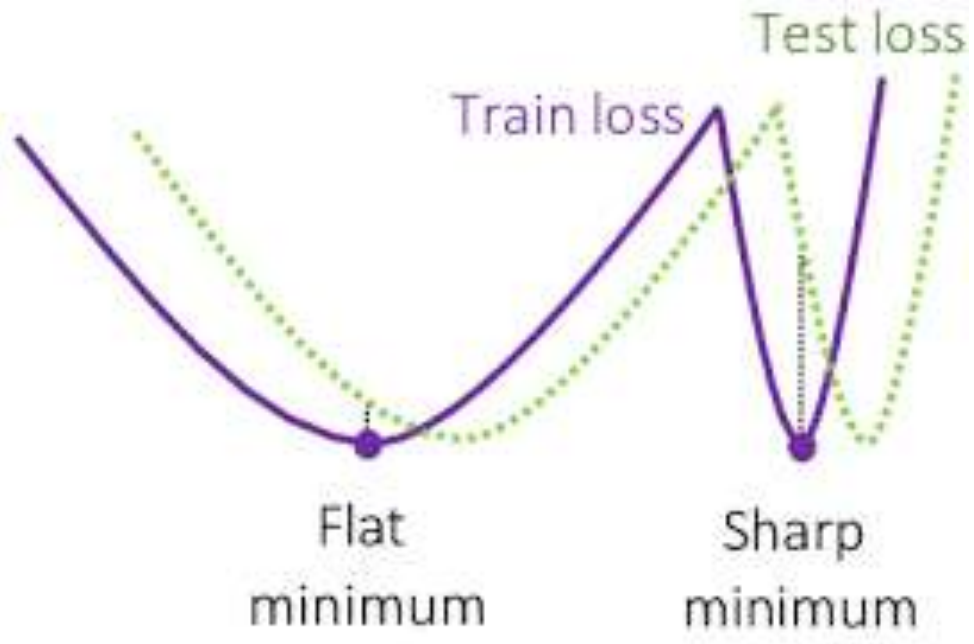
$$w_{t+1} = w_t - \eta \nabla_w \ell_i$$ Stochastic GD

$$w_{t+1} = w_t - \eta \sum_{i \in I} \nabla_w \ell_i$$ Mini-batch GD

$$w_{t+1} = w_t - \eta \sum_i \nabla_w \ell_i$$ Full-batch GD

SGD and generalization

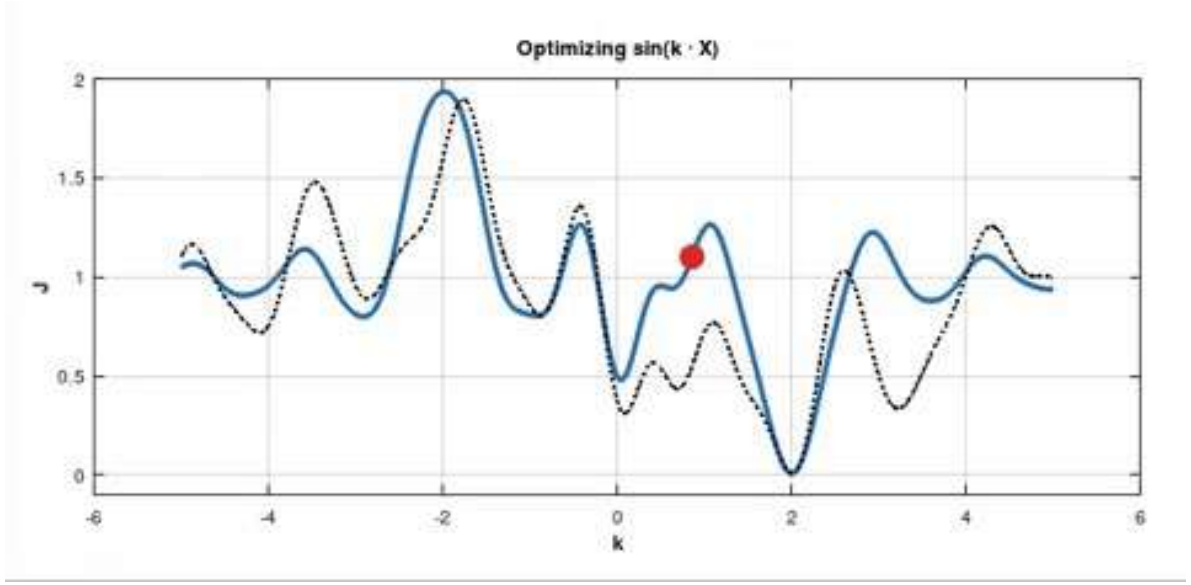
- Skip bad local minima, converge to flat minima (basins of attraction). Good for generalization, robustness.

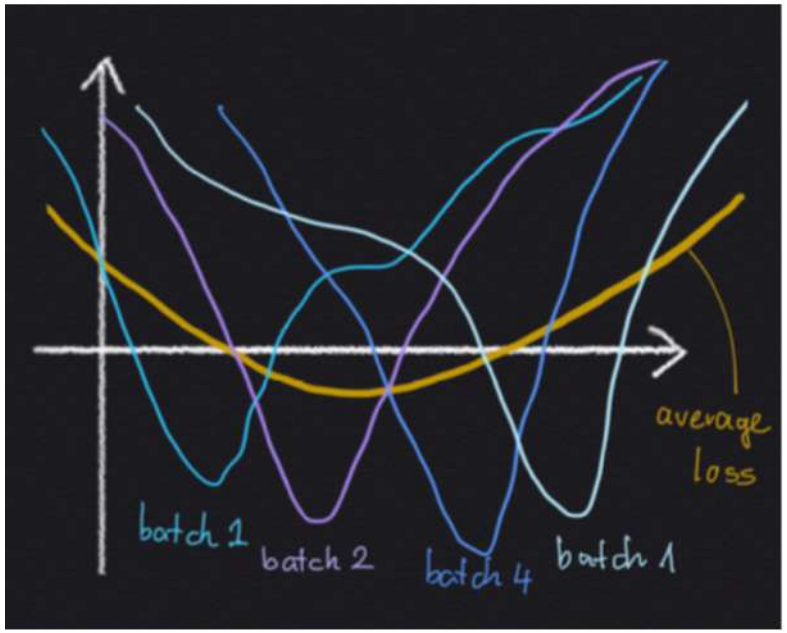


Another advantage?

Can we be more precise?

Intuition

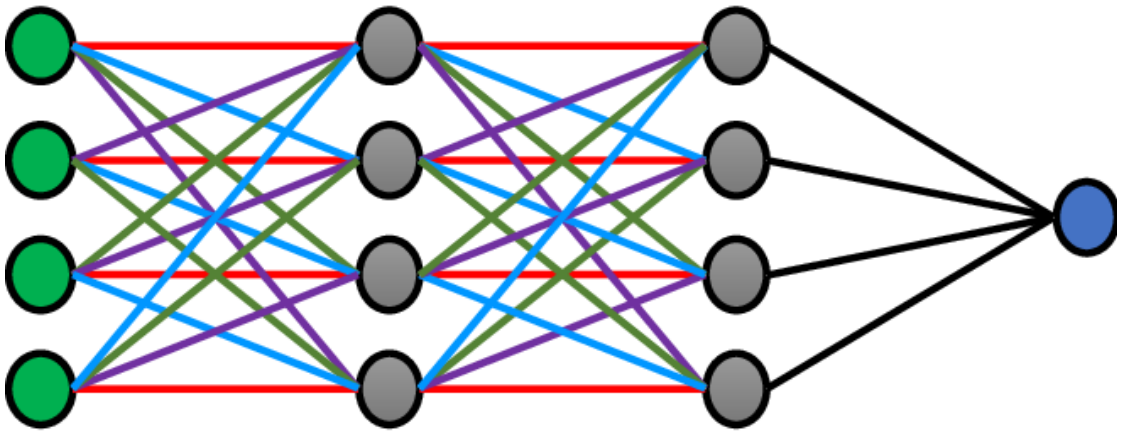




At each step the SGD is looking to a “different, sampled” function

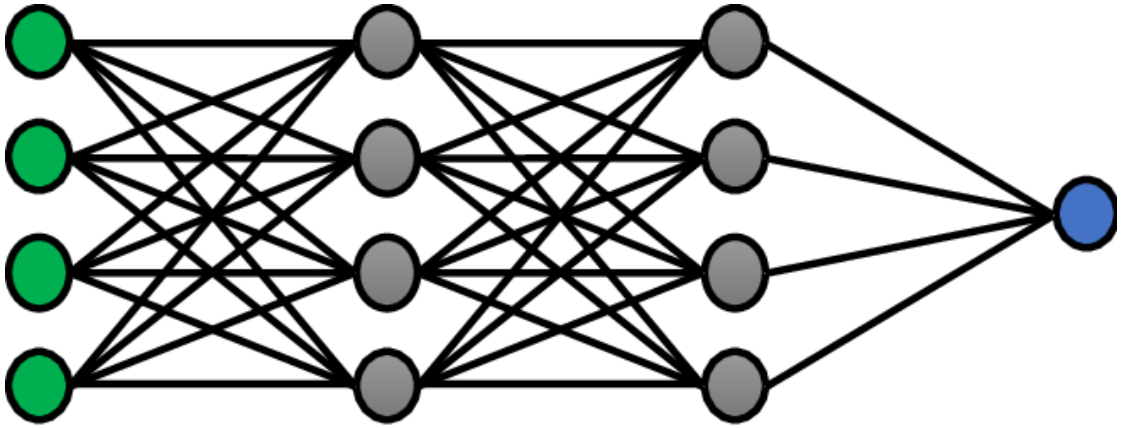
On average the main effect is dominated by big “basins of attraction”

Weights bias for fully connected and convolutions linear networks



(b) Convolutional network of depth L

$\overline{\boldsymbol{\beta}}^{\infty} \propto$ first order stationary point of $\underset{\forall n,\, y_n \langle \mathbf{x}_n, \boldsymbol{\beta} \rangle \geq 1}{\operatorname{argmin}} \|\widehat{\boldsymbol{\beta}}\|_{2/L}$

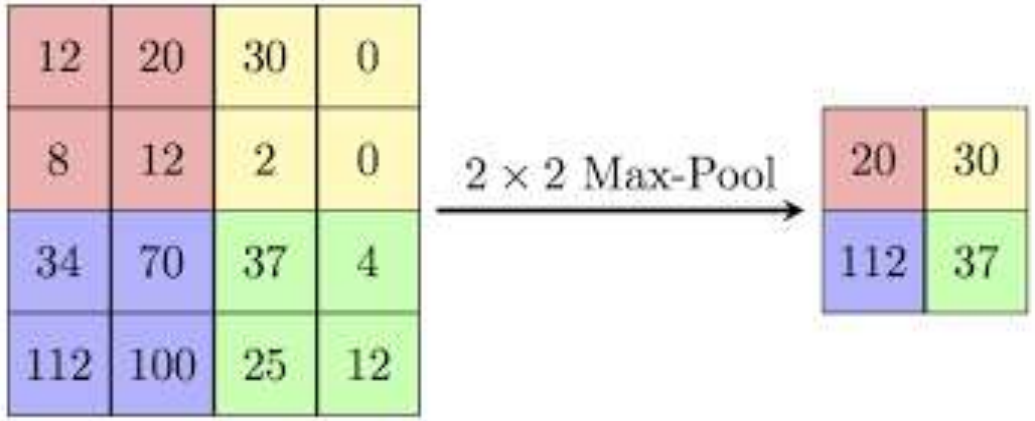


(a) Fully connected network of depth L

$\overline{\boldsymbol{\beta}}^{\infty} \propto \underset{\forall n,\, y_n \langle \mathbf{x}_n, \boldsymbol{\beta} \rangle \geq 1}{\operatorname{argmin}} \|\boldsymbol{\beta}\|_2$ (independent of L)

Gunasekar 2018

Pooling operation and invariance



Convolution operation, pooling and invariance: bias on the target function

$$\langle g_i w, x \rangle = (x * w)_i$$

$$f(x) = \sum_i \langle g_i w, x \rangle$$

$$f(gx) = f(x)$$

$$f(gx) = \sum_i \langle g_i w, gx \rangle = \sum_i \langle g^T g_i w, x \rangle = \sum_i \left\langle g_i^{'} w, x \right\rangle$$

$$g^T g_i = g_i^{'}$$

$$f(gx) = \begin{bmatrix} \sigma\left\langle g_1 w, gx \right\rangle \\ \vdots \\ \sigma\left\langle g_N w, gx \right\rangle \end{bmatrix} = \begin{bmatrix} \sigma\left\langle w, g_1^T gx \right\rangle \\ \vdots \\ \sigma\left\langle w, g_N^T gx \right\rangle \end{bmatrix} = P_g \begin{bmatrix} \sigma\left\langle w, gx \right\rangle \\ \vdots \\ \sigma\left\langle w, gx \right\rangle \end{bmatrix} = P_g f(x)$$

$$f(gx) = \Pi_g f(x)$$

Cnns are translation equivariant

$$\phi(T_\alpha x)(\beta) = \sigma(\int k(\beta - \gamma) T_\alpha x(\gamma) d\gamma) = T_\alpha \phi(x)(\beta)$$

Let’s prove it

Equivariance and convolutionality are one to one:

On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups

Risi Kondor [1] **Shubhendu Trivedi** [2]

Abstract

Convolutional neural networks have been extremely successful in the image recognition domain because they ensure equivariance to translations. There have been many recent attempts to generalize this framework to other domains, including graphs and data lying on manifolds. In this paper we give a rigorous, theoretical treatment of convolution and equivariance in neural networks with respect to not just translations, but the action of any compact group. Our main result is to prove that (given some natural constraints) convolutional structure is not just a sufficient, but also a necessary condition for equivariance to the action of a compact group. Our exposition makes use of concepts from representation theory and noncommutative harmonic analysis and derives new generalized convolution formulae.

Recently, there has been considerable interest in extending neural networks to more exotic types of data, such as graphs or functions on manifolds (Niepert et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Li et al., 2016; Cohen et al., 2018; Monti et al., 2017; Masci et al., 2015). In these domains, equivariance and multiscale structure are just as important as for images, but finding the right notion of convolution is not obvious.

On the other hand, mathematics does offer a sweeping generalization of convolution tied in deeply with some fundamental ideas of abstract algebra: if G is a compact group and f and g are two functions $G \to \mathbb{C}$, then the convolution of f with g is defined

$$(f * g)(u) = \int_G f(uv^{-1})\, g(v)\, d\mu(v). \qquad (1)$$

Note the striking similarity of this formula to the ordinary notion of convolution, except that in the argument of f, $u - v$ has been replaced by the group operation uv^{-1}, and

How do we learn invariance or equivariance w.r.t. generic label-preserving transformations?

Effect of Data Augmentation: invariant loss for one layer network

$$\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N}\int \ell\left(\sigma\langle w, g_\theta x_i\rangle; y_i\right) d\theta$$

$$\begin{aligned}\mathcal{L}(g_{\bar{\theta}}w) &= \frac{1}{N}\sum_{i=1}^{N}\int \ell\left(\sigma\langle w, g^* g_\theta x_i\rangle; y_i\right) d\theta \\ &= \sum_{i=1}^{N}\int \ell\left(\sigma\langle w, g_\theta x_i\rangle; y_i\right) d\theta \\ &= \mathcal{L}(w)\end{aligned}$$

The Loss is an invariant function!

$$g_\theta g_{\theta'} = g_{\theta''}$$

Intuition with translation invariant/robust functions

- **Invariance:** Let $\underline{\mathrm{f}} : R \to R$ s.t. $\underline{\mathrm{f}}(w+q) = \underline{\mathrm{f}}(w), \ \ \forall q \in R.$ $\underline{\mathrm{f}}$ is constant and $\hat{\underline{\mathrm{f}}}(0)$ is the only non zero Fourier transform.

$$\underline{\mathrm{f}}(w) = \int_{-\infty}^{\infty} dt\ f(w-t), \qquad f : R \to R.$$

- **Robustness:** Suppose we relax this invariance properties asking for robustness/approximate invariance. One way is to integrate in the interval $[-a, a]$.

$$\underline{\mathrm{f}}(w) = \int_{-a}^{a} dt\ f(w-t) = \int_{-\infty}^{+\infty} dt\ Ind_{[-a,a]}(t) f(w-t).$$

Taking the Fourier and using $\widehat{f(\cdot - t)} = e^{ikt}\hat{f}(k)$

$$\hat{\underline{\mathrm{f}}}(k) = \Big(\int_{-\infty}^{\infty} dt\ e^{ikt} Ind_{[-a,a]}(t) \Big)\ \hat{f}(k) = 2a\ sinc(2ka)\hat{f}(k).$$

Effect on the gradient descent

From the group structure we have:

$$\mathcal{L}(U_\alpha w; X, y) = \mathcal{L}(w; X, y), \quad \alpha \in R.$$

Then

$$\frac{d}{d\alpha}\mathcal{L}(U_\alpha w) = 0 \;\; \rightarrow \;\; \langle \nabla_w \mathcal{L}(w), \partial_\alpha U_\alpha w \rangle = 0,$$

i.e. **some directions in the gradient are not allowed by the invariances of the Loss**.

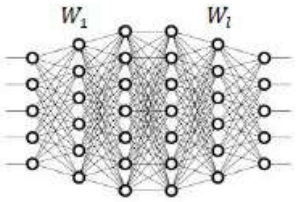
This is important since:

$$\frac{dw}{dt} = \nabla \mathcal{L}(w).$$

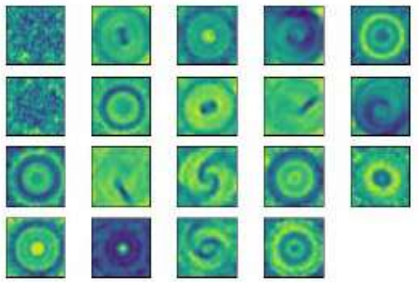
Learned weights reflects the symmetries of the data augmentation

Goal: $y_i = \varphi_W(x_i), \quad \varphi_{\mathbf{W}}(\mathbf{g}\mathbf{x_i}) = \varphi_{\mathbf{W}}(\mathbf{x_i})$

Data augmentation:



Rotation augmented



$$\mathcal{L}(w) = \frac{1}{N}\sum_{i=1}^{N}\int \ell\left(\sigma\langle w, g_\theta x_i\rangle; y_i\right) d\theta$$

Equivariance and learned weights

Theorem 1 (see also, [43]): Consider the Loss in eq. (5). The gradient of the Loss is an equivariant function in the $\mathcal{G}$ transformations, i.e.:

$$\nabla_W \mathcal{L}_a(gW; X, y) = g\nabla_W \mathcal{L}_a(W; X, y), \quad \forall\, g \in \mathcal{G}.$$

Proof 1: Taking the gradient of eq. (5) we have:

$$\begin{aligned}
&\nabla_W \mathcal{L}_a(W; X, y) = \\
&= \frac{1}{Q}\sum_{i=1}^{R}\sum_{j=1}^{|\mathcal{G}|} \ell'(v^T\sigma\left\langle W, g_j x_i\right\rangle, y_i) v^T\sigma'\left\langle W, g_j x_i\right\rangle g_j x_i
\end{aligned}$$

Calculating $\nabla_W \mathcal{L}_a(gW; X, y)$ we have:

$$\begin{aligned}
&\nabla_W \mathcal{L}_a(gW; X, y) = \\
&= \frac{1}{Q}\sum_{i=1}^{R}\sum_{j=1}^{|\mathcal{G}|} \ell'(v^T\sigma\left\langle gW, g_j x_i\right\rangle, y_i) v^T\sigma'\left\langle gW, g_j x_i\right\rangle g_j x_i \\
&= \frac{1}{Q}\sum_{i=1}^{R}\sum_{j=1}^{|\mathcal{G}|} \ell'(v^T\sigma\left\langle W, g^T g_j x_i\right\rangle, y_i) v^T\sigma'\left\langle W, g^T g_j x_i\right\rangle g_j x_i \\
&= \frac{1}{Q}\sum_{i=1}^{R}\sum_{j=1}^{|\mathcal{G}|} \ell'(v^T\sigma\left\langle W, \hat{g}_j x_i\right\rangle, y_i) v^T\sigma'\left\langle W, \hat{g}_j x_i\right\rangle g\hat{g}_j x_i = \\
&= g\nabla_W \mathcal{L}_a(W; X, y)
\end{aligned}$$

where in the fourth line we redefined $g^T g_j = \hat{g}_j$ to get $g_j = g\hat{g}_j$. The key property here is the closure of the group transformations. ■

Equivariance and learned weights

Corollary 1: Consider the Loss in eq. 5 i.e. the case of data augmentation with full orbits, i.e. $gX = X$ for all $g \in \mathcal{G}$, and suppose we initialize the network with symmetric weights i.e. $gW_0 = W_0$ for all $g \in \mathcal{G}$. Then the learned weights are symmetric $W_T = gW_T$.

Proof 2: We first notice that due to the property $gX = X$, $\nabla_W \mathcal{L}_a(W; X, y) = \nabla_W \mathcal{L}_a(W; gX, y)$. Then by the dot product structure of the Loss we can see the transformation as applied to the weights i.e. $\langle W, gx \rangle = \langle g^T W, x \rangle$. Thus, using the equivariance property in theorem 1, we have that for all $g \in \mathcal{G}$:

$$\begin{aligned} \nabla_W \mathcal{L}_a(W; X, y) &= \nabla_W \mathcal{L}_a(W; gX, y) \\ &= \nabla_W \mathcal{L}_a(g^T W; X, y) \\ &= g^T \nabla_W \mathcal{L}_a(W; gX, y). \end{aligned}$$

Therefore each weights update in the gradient descent is group invariant and as a consequence the learned weight at time T is invariant, being:

$$W_T = W_0 - \gamma \sum_{k=1}^{T} \nabla_W \mathcal{L}_a(W_k; X, y).$$

■

Permutation equivariant layer

Our goal is to design neural network layers that are equivariant to permutations of elements in the input $\mathbf{x}$. The function $\mathbf{f} : \mathcal{X}^M \to \mathcal{Y}^M$ is **equivariant** to the permutation of its inputs iff

$$\mathbf{f}(\pi\mathbf{x}) = \pi\mathbf{f}(\mathbf{x}) \quad \forall\pi \in \mathcal{S}_M$$

where the symmetric group $\mathcal{S}_M$ is the set of all permutation of indices $1, \ldots, M$.

Consider the standard neural network layer

$$\mathbf{f}_\Theta(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M\times M} \tag{20}$$

where Θ is the weight vector and $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinearity such as sigmoid function. The following lemma states the necessary and sufficient conditions for permutation-equivariance in this type of function.

Lemma 3 *The function $\mathbf{f}_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ as defined in (20) is permutation equivariant if and only if all the off-diagonal elements of Θ are tied together and all the diagonal elements are equal as well. That is,*

$$\Theta = \lambda\mathbf{I} + \gamma\,(\mathbf{1}\mathbf{1}^\mathsf{T}) \qquad \lambda, \gamma \in \mathbb{R} \quad \mathbf{1} = [1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^M$$

where $\mathbf{I} \in \mathbb{R}^{M\times M}$ is the identity matrix.

Deep Sets , Zaheer et al.
NIPS 2017

1. **Goal**: To show that the matrix $\Theta = \lambda I + \gamma(11^T)$ commutes with all permutation matrices $\pi \in S_M$, ensuring permutation equivariance of the function $f_\Theta(x) = \sigma(\Theta x)$.
2. **Linearity of Commute**:
    - The linear combination $\Theta = \lambda I + \gamma(11^T)$ commutes with any permutation matrix π because both the identity matrix I and the constant matrix 11^T commute with any permutation matrix.
3. **Diagonal Elements are Identical**:
    - For any transposition $\pi_{k,l}$, it is shown that the diagonal elements $\Theta_{i,i}$ must be identical for all i, implying $\Theta_{i,i} = \lambda$.
4. **Off-Diagonal Elements are Identical**:
    - By considering pairs of off-diagonal elements, it is shown that the off-diagonal elements $\Theta_{i,j}$ must be identical, concluding that $\Theta_{i,j} = \gamma$ for all off-diagonal $i \neq j$.
5. **Conclusion**: The necessary and sufficient conditions for permutation equivariance are met, as $\Theta = \lambda I + \gamma(11^T)$ commutes with all permutation matrices $\pi \in S_M$.

From definition of permutation equivariance $\mathbf{f}_\Theta(\pi\mathbf{x}) = \pi\mathbf{f}_\Theta(\mathbf{x})$ and definition of $\mathbf{f}$ in (20), the condition becomes $\boldsymbol{\sigma}(\Theta\pi\mathbf{x}) = \pi\boldsymbol{\sigma}(\Theta\mathbf{x})$, which (assuming sigmoid is a bijection) is equivalent to $\Theta\pi = \pi\Theta$. Therefore we need to show that the necessary and sufficient conditions for the matrix $\Theta \in \mathbb{R}^{M\times M}$ to commute with all permutation matrices $\pi \in \mathcal{S}_M$ is given by this proposition. We prove this in both directions:

- To see why $\Theta = \lambda\mathbf{I} + \gamma\ (\mathbf{1}\mathbf{1}^\mathsf{T})$ commutes with any permutation matrix, first note that commutativity is linear – that is

  $$\Theta_1\pi = \pi\Theta_1 \wedge \Theta_2\pi = \pi\Theta_2 \quad \Rightarrow \quad (a\Theta_1 + b\Theta_2)\pi = \pi(a\Theta_1 + b\Theta_2).$$

  Since both Identity matrix $\mathbf{I}$, and constant matrix $\mathbf{1}\mathbf{1}^\mathsf{T}$, commute with any permutation matrix, so does their linear combination $\Theta = \lambda\mathbf{I} + \gamma\ (\mathbf{1}\mathbf{1}^\mathsf{T})$.

- We need to show that in a matrix Θ that commutes with "all" permutation matrices
  - *All diagonal elements are identical*: Let $\pi_{k,l}$ for $1 \leq k, l \leq M, k \neq l$, be a transposition (*i.e.* a permutation that only swaps two elements). The inverse permutation matrix of $\pi_{k,l}$ is the permutation matrix of $\pi_{l,k} = \pi_{k,l}^\mathsf{T}$. We see that commutativity of Θ with the transposition $\pi_{k,l}$ implies that $\Theta_{k,k} = \Theta_{l,l}$:

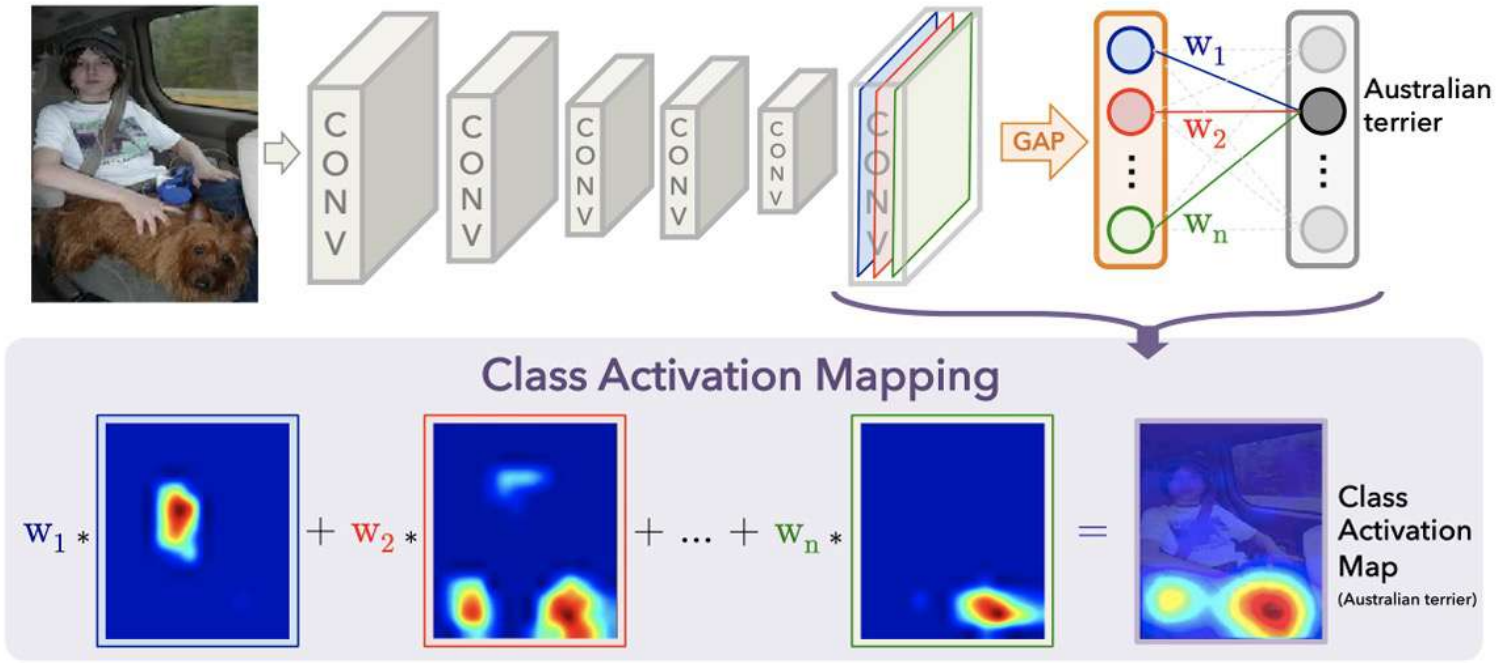
    $$\pi_{k,l}\Theta = \Theta\pi_{k,l} \ \Rightarrow\ \pi_{k,l}\Theta\pi_{l,k} = \Theta \ \Rightarrow\ (\pi_{k,l}\Theta\pi_{l,k})_{l,l} = \Theta_{l,l} \ \Rightarrow\ \Theta_{k,k} = \Theta_{l,l}$$

    Therefore, π and Θ commute for any permutation π, they also commute for any transposition $\pi_{k,l}$ and therefore $\Theta_{i,i} = \lambda\, \forall i$.
  - *All off-diagonal elements are identical*: We show that since Θ commutes with any product of transpositions, any choice two off-diagonal elements should be identical. Let (i, j) and (i', j') be the index of two off-diagonal elements (*i.e.* $i \neq j$ and $i' \neq j'$). Moreover for now assume $i \neq i'$ and $j \neq j'$. Application of the transposition $\pi_{i,i'}\Theta$, swaps the rows i, i' in Θ. Similarly, $\Theta\pi_{j,j'}$ switches the j^{th} column with j'^{th} column. From commutativity property of Θ and $\pi \in \mathcal{S}_n$ we have

    $$\begin{aligned}
    &\pi_{j',j}\pi_{i,i'}\Theta = \Theta\pi_{j',j}\pi_{i,i'} \ \Rightarrow \pi_{j',j}\pi_{i,i'}\Theta(\pi_{j',j}\pi_{i,i'})^{-1} = \Theta \qquad\qquad \Rightarrow \\
    &\pi_{j',j}\pi_{i,i'}\Theta\pi_{i',i}\pi_{j,j'} = \Theta \ \Rightarrow (\pi_{j',j}\pi_{i,i'}\Theta\pi_{i',i}\pi_{j,j'})_{i,j} = \Theta_{i,j} \quad \Rightarrow\ \Theta_{i',j'} = \Theta_{i,j}
    \end{aligned}$$

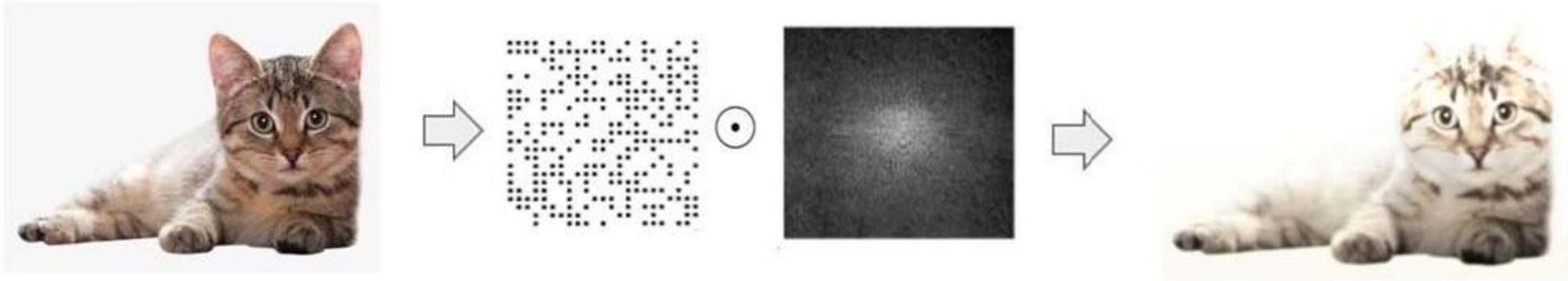
    where in the last step we used our assumptions that $i \neq i'$, $j \neq j'$, $i \neq j$ and $i' \neq j'$. In the cases where either $i = i'$ or $j = j'$, we can use the above to show that $\Theta_{i,j} = \Theta_{i'',j''}$ and $\Theta_{i',j'} = \Theta_{i'',j''}$, for some $i'' \neq i, i'$ and $j'' \neq j, j'$, and conclude $\Theta_{i,j} = \Theta_{i',j'}$.

Where is a ANN looking at? Class activation mapping

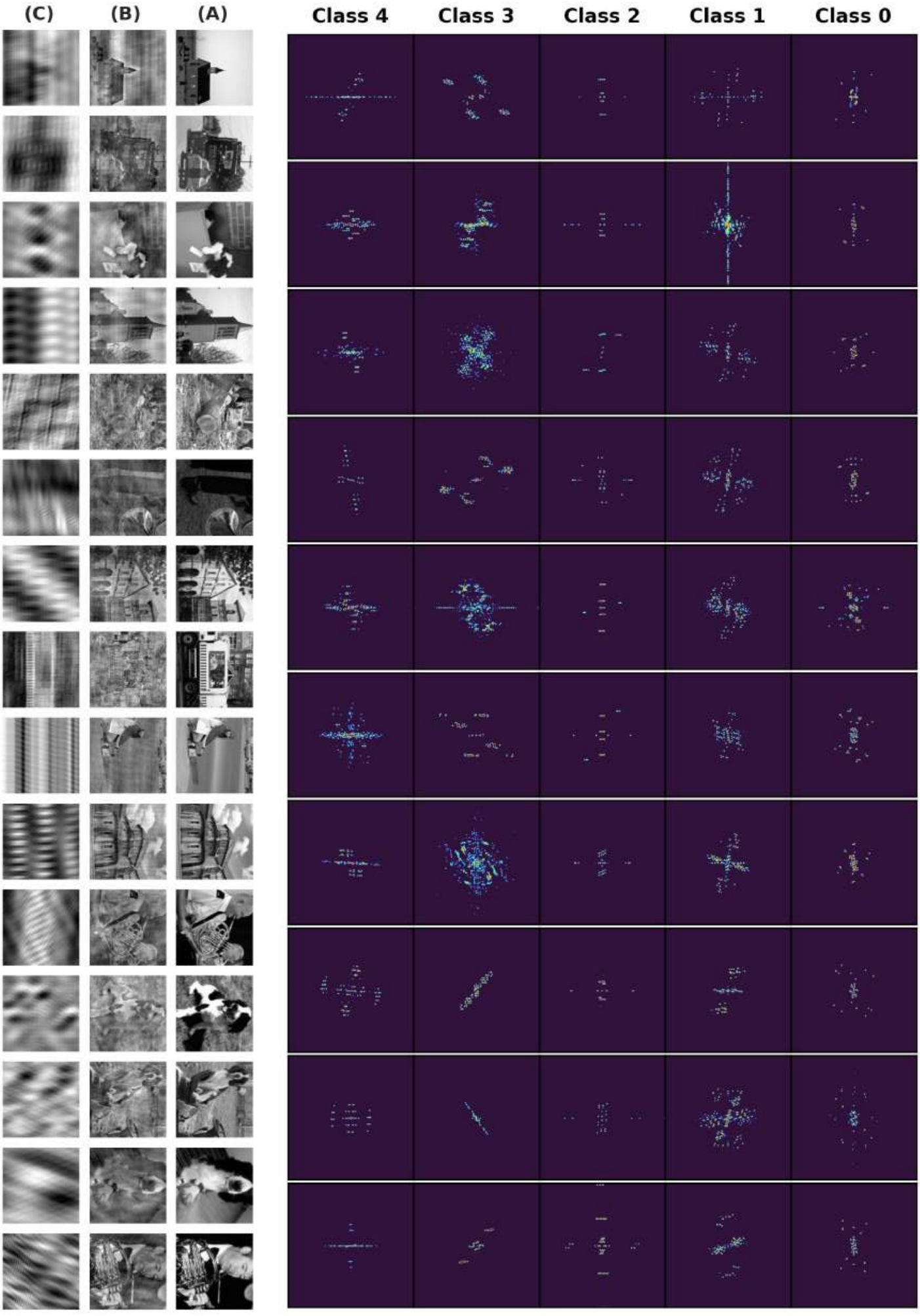


How does a network make a decision? Fourier important features.

x $\quad M_\Phi \odot \mathcal{F}x \quad \mathcal{F}^{-1}(M_\Phi \odot \mathcal{F}x)$



$$M_\Phi(\lambda, p) = \text{argmin}_{M_\Phi} \sum_{x \in \mathcal{X}_V} e^{[\mathcal{L}(\Phi(\bar{x}),y) - \mathcal{L}(\Phi(x),y)]^2} + \lambda \| M_\Phi \|_p, \quad \lambda \in \mathbb{R}_+$$


(C)
(B)
(A)
Class 4
Class 3
Class 2
Class 1
Class 0

Or texture?

IMAGENET-TRAINED CNNS ARE BIASED TOWARDS TEXTURE; INCREASING SHAPE BIAS IMPROVES ACCURACY AND ROBUSTNESS

Robert Geirhos
University of Tübingen & IMPRS-IS
robert.geirhos@bethgelab.org

Patricia Rubisch
University of Tübingen & U. of Edinburgh
p.rubisch@sms.ed.ac.uk

Claudio Michaelis
University of Tübingen & IMPRS-IS
claudio.michaelis@bethgelab.org

Matthias Bethge*
University of Tübingen
matthias.bethge@bethgelab.org

Felix A. Wichmann*
University of Tübingen
felix.wichmann@uni-tuebingen.de

Wieland Brendel*
University of Tübingen
wieland.brendel@bethgelab.org

(a) Texture image

81.4%	**Indian elephant**
10.3%	indri
8.2%	black swan

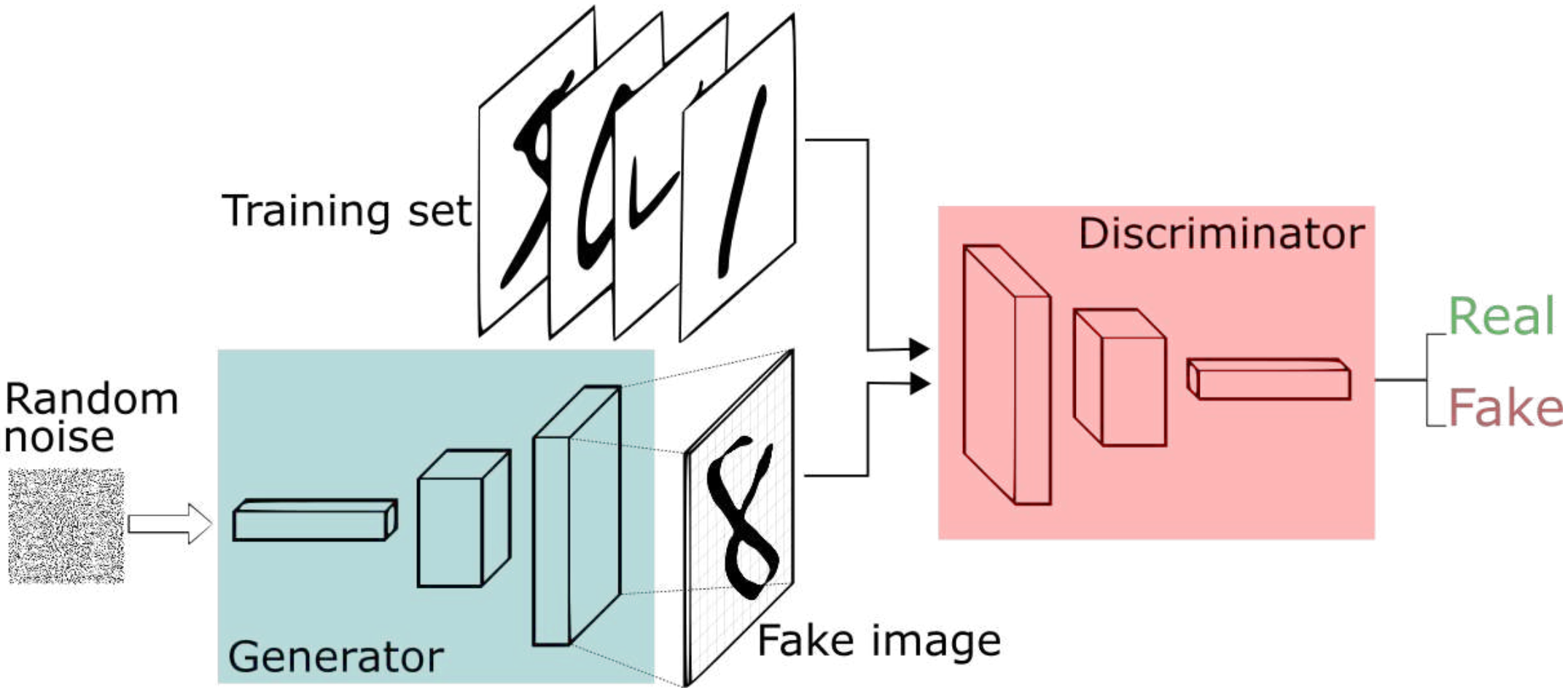
(b) Content image

71.1%	**tabby cat**
17.3%	grey fox
3.3%	Siamese cat

(c) Texture-shape cue conflict

63.9%	**Indian elephant**
26.4%	indri
9.6%	black swan

Generative adversarial networks



Gan Loss

gradient ascent

predict well on real images
=> want probability close to 1

predict well on fake images
=> want probability close to 0

$$\nabla_{\mathbf{w}_D} \frac{1}{n} \sum_{i=1}^{n} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

Discriminator objective in the neg. log-likelihood (binary cross entropy) perspective:

Real images, $y = 1$

$$\mathcal{L}(\mathbf{w}) = -y^{(i)} \log\left(\hat{y}^{(i)}\right) - \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

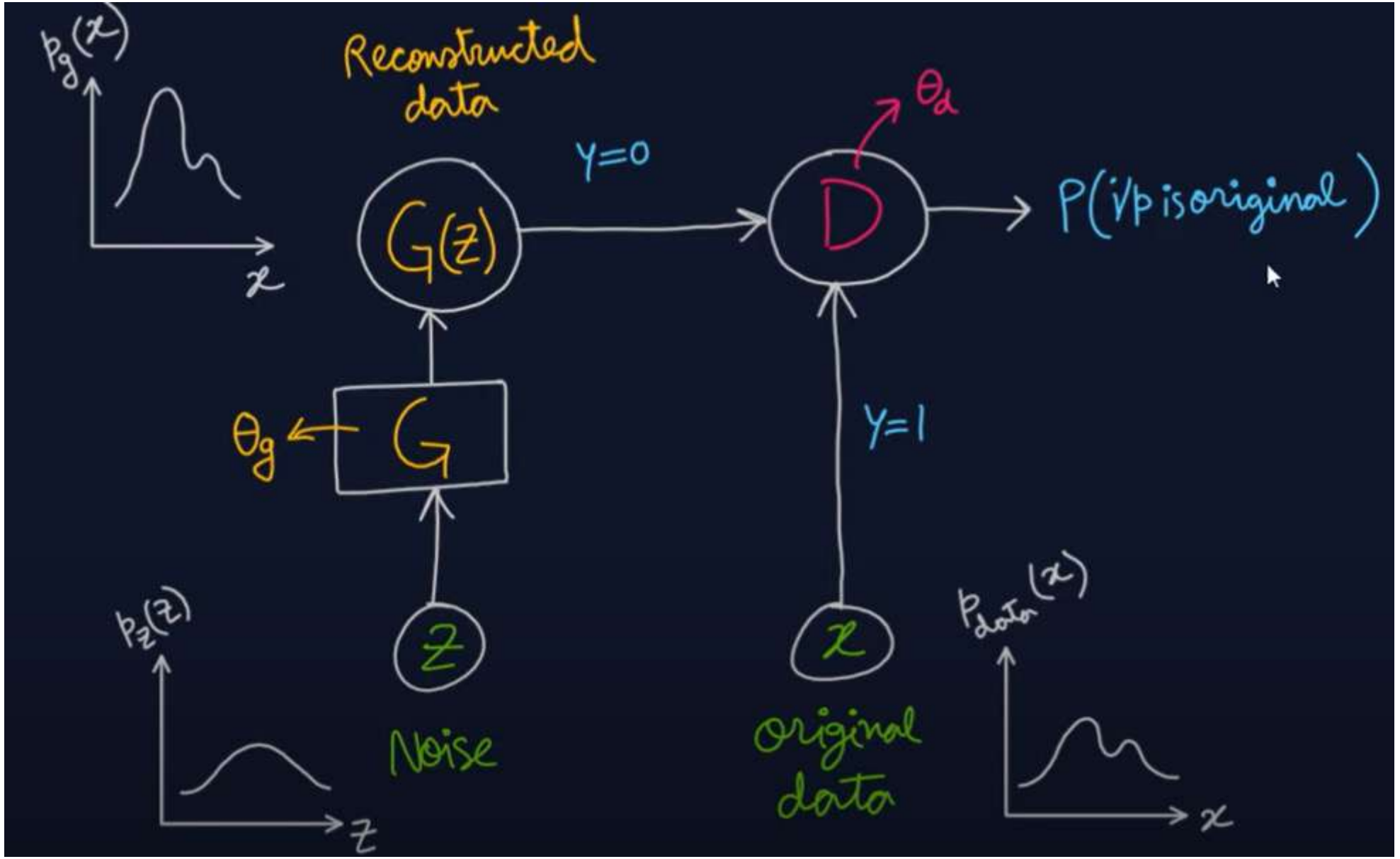
Want, $\hat{y} = 1$

Fake images, $y = 0$

$$\mathcal{L}(\mathbf{w}) = -y^{(i)} \log\left(\hat{y}^{(i)}\right) - \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

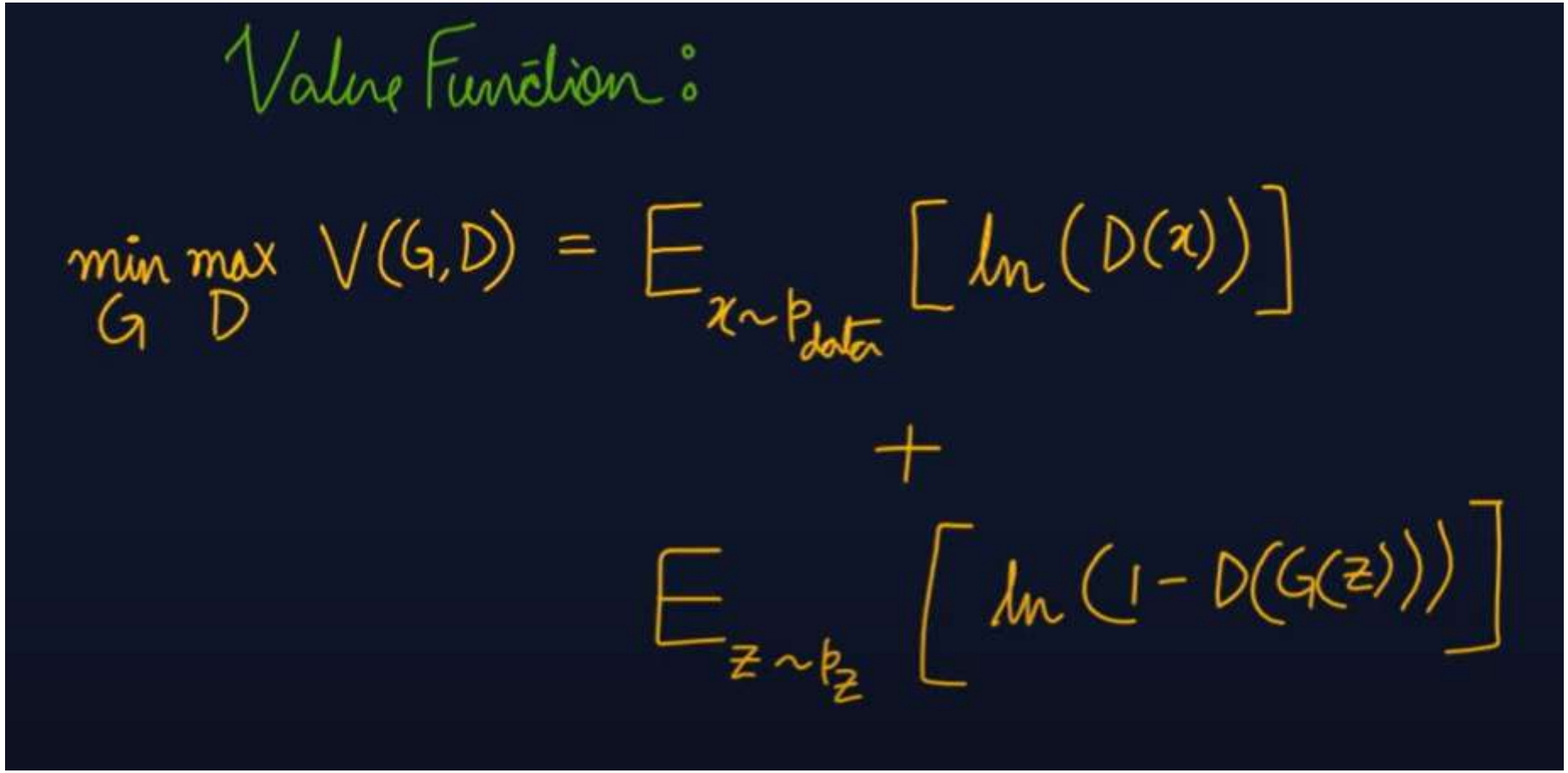
Want, $\hat{y} = 0$

General setting

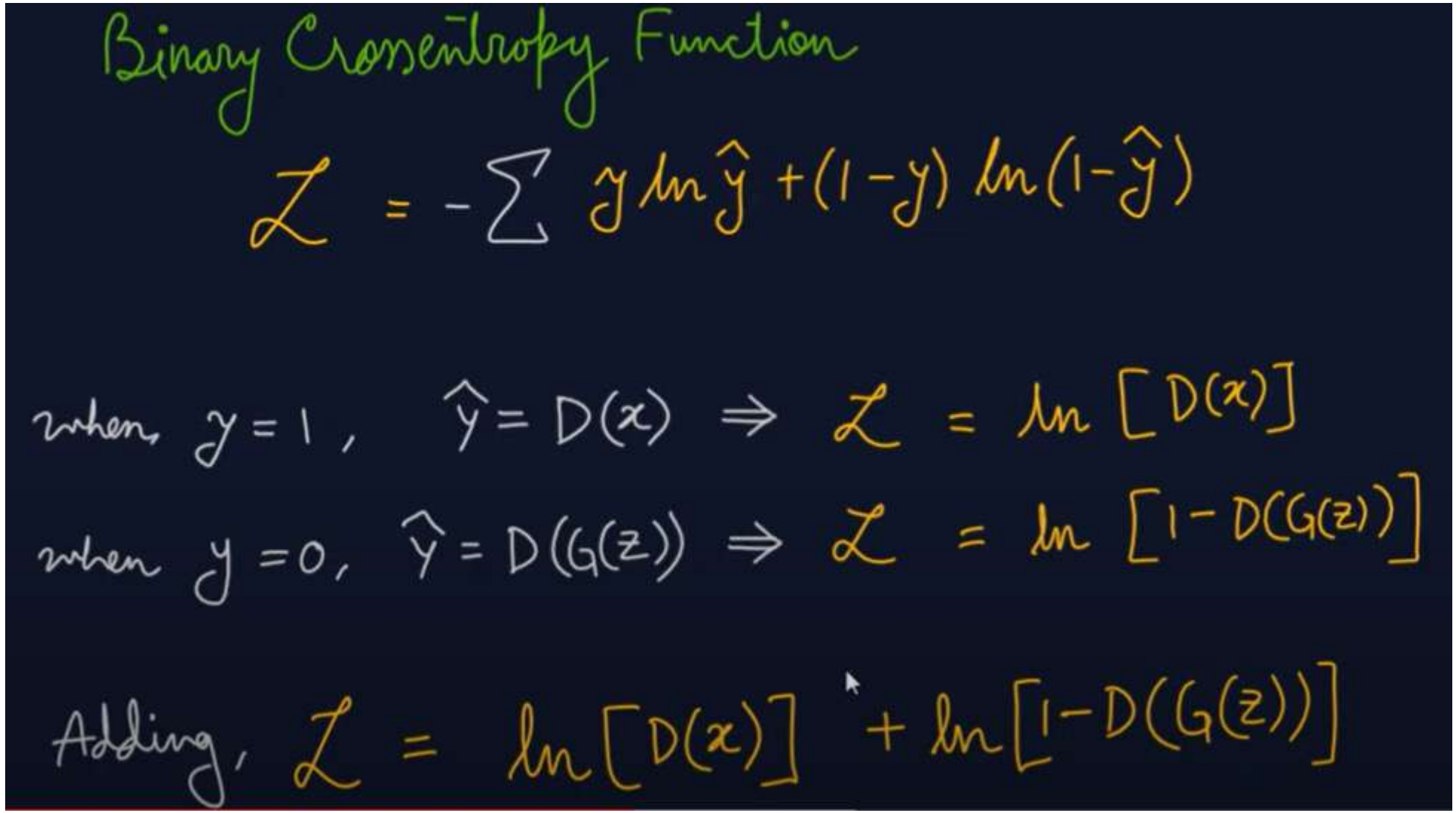


https://www.youtube.com/watch?v=Gib_kiXgnvA

The problem: the discriminator try to mimic data distribution; the generator tries to fool the discriminator



Analogy with Binary Cross-Entropy



Expectation

$$E(\mathcal{L}) = E\left(\ln[D(x)]\right) + E\left(\ln[1 - D(G(z))]\right)$$

$$\sum p_{data}(x) \ln[D(x)] + \sum p_z(z) \ln[1 - D(G(z))]$$

$$\int p_{data}(x) \ln[D(x)]\, dx + \int p_z(z) \ln[1 - D(G(z))]\, dz$$

$$V(G,D) = E_{x \sim p_{data}}\left[\ln(D(x))\right] + E_{z \sim p_z}\left[\ln(1 - D(G(z)))\right]$$

Traning loop:

* fix the learning of G *

Inner loop for D:

- take m data samples & m fake data samples
- update θ_d by grad. ascent

$$\frac{\partial}{\partial \theta_d} \frac{1}{m} \left[\ln [D(x)] + \ln [1 - D(G(z))] \right]$$

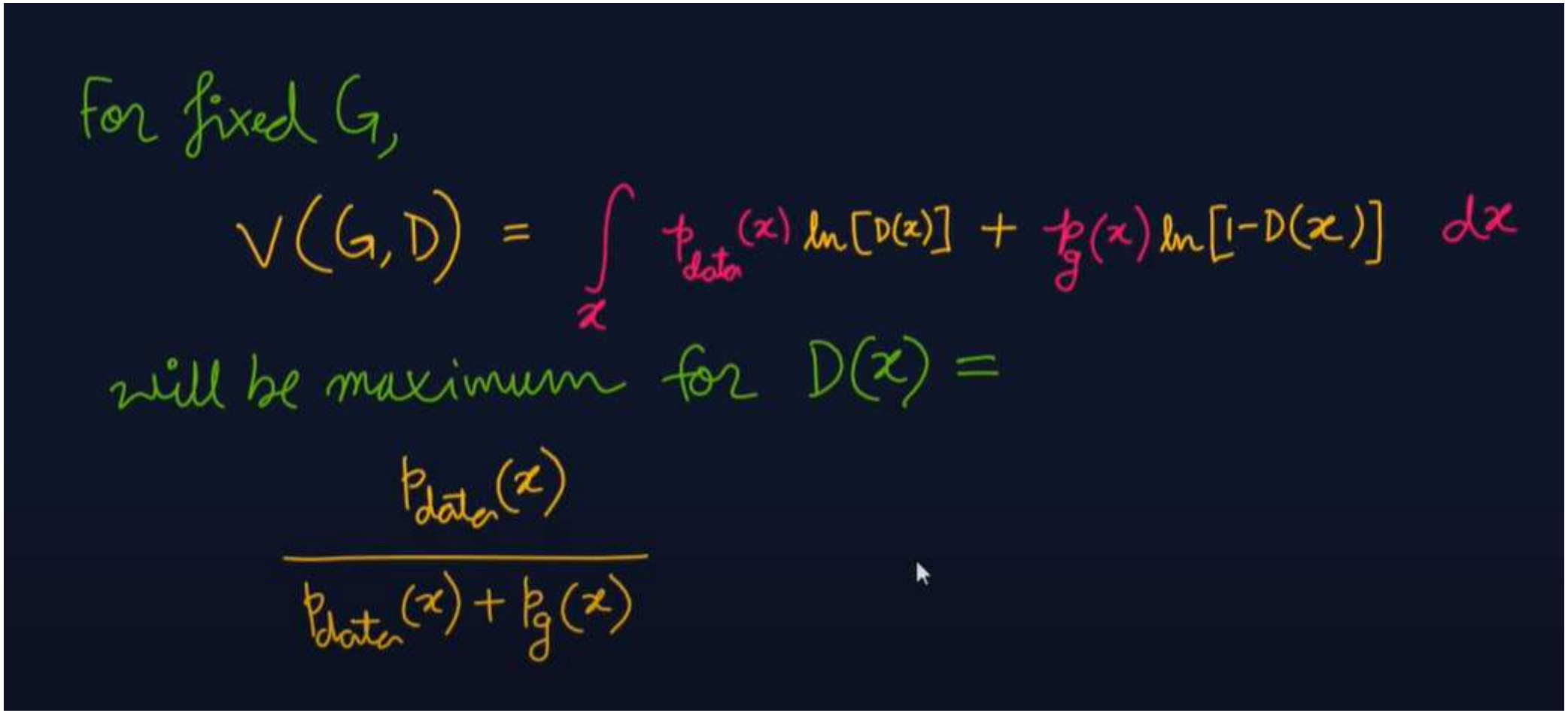
* fix the learning of D *

take m fake data samples

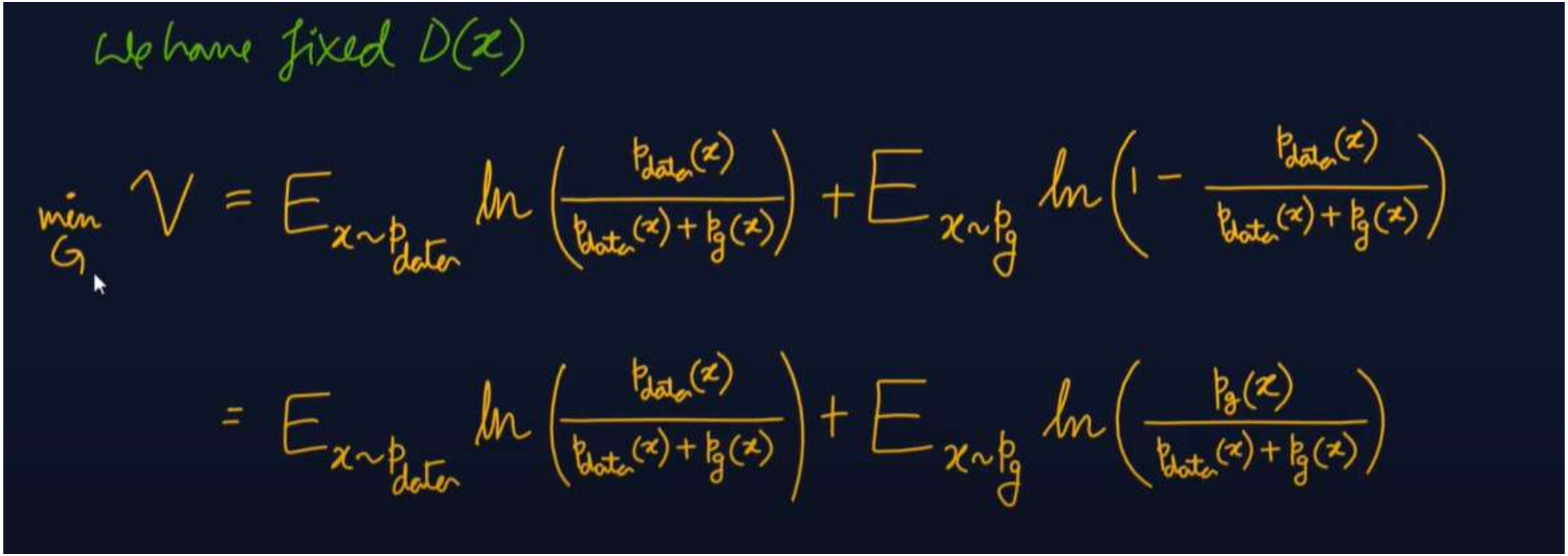
update θ_g by grad. descent

$$\frac{\partial}{\partial \theta_g} \frac{1}{m} \left[\ln [1 - D(G(z))] \right]$$

What is the Optimization doing?



https://www.youtube.com/watch?v=Gib_kiXgnvA



$$= E_{x \sim p_{data}} \ln \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) + E_{x \sim p_g} \ln \left(\frac{p_g(x)}{p_{data}(x) + p_g(x)} \right)$$

JS divergence

$$JS(p_1 \| p_2) = \frac{1}{2} E_{x \sim p_1} \ln \left(\frac{p_1}{\frac{p_1 + p_2}{2}} \right) + \frac{1}{2} E_{x \sim p_2} \left(\frac{p_2}{\frac{p_1 + p_2}{2}} \right)$$

$$\min_G V = E_{x \sim p_{data}} \ln\left(\frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}}\right) + E_{x \sim p_g} \ln\left(\frac{p_g(x)}{\frac{p_{data}(x) + p_g(x)}{2}}\right) - 2\ln 2$$

$$\min_G V = 2\, JS(p_{data} \| p_g) - 2\ln 2$$

The optimization tries to match the training data and the generator data probability distributions

Importance of ANNs in Neuroscience

2021 Special Issue on AI and Brain Science: Perspective

Natural and Artificial Intelligence: A brief introduction to the interplay between AI and neuroscience research

Tom Macpherson [a], Anne Churchland [b], Terry Sejnowski [c,d], James DiCarlo [e], Yukiyasu Kamitani [f,g], Hidehiko Takahashi [h], Takatoshi Hikida [a,*]

[a] Laboratory for Advanced Brain Functions, Institute for Protein Research, Osaka University, Osaka, Japan
[b] Cold Spring Harbor Laboratory, Neuroscience, Cold Spring Harbor, NY, USA
[c] Computational Neurobiology Laboratory, Salk Institute for Biological Studies, CA, USA
[d] Division of Biological Sciences, University of California San Diego, CA, USA
[e] Brain and Cognitive Sciences, Massachusetts Institute of Technology, MA, USA
[f] Department of Neuroinformatics, ATR Computational Neuroscience Laboratories, Kyoto, Japan
[g] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[h] Department of Psychiatry and Behavioral Sciences, Tokyo Medical and Dental University Graduate School, Tokyo, Japan

ARTICLE INFO

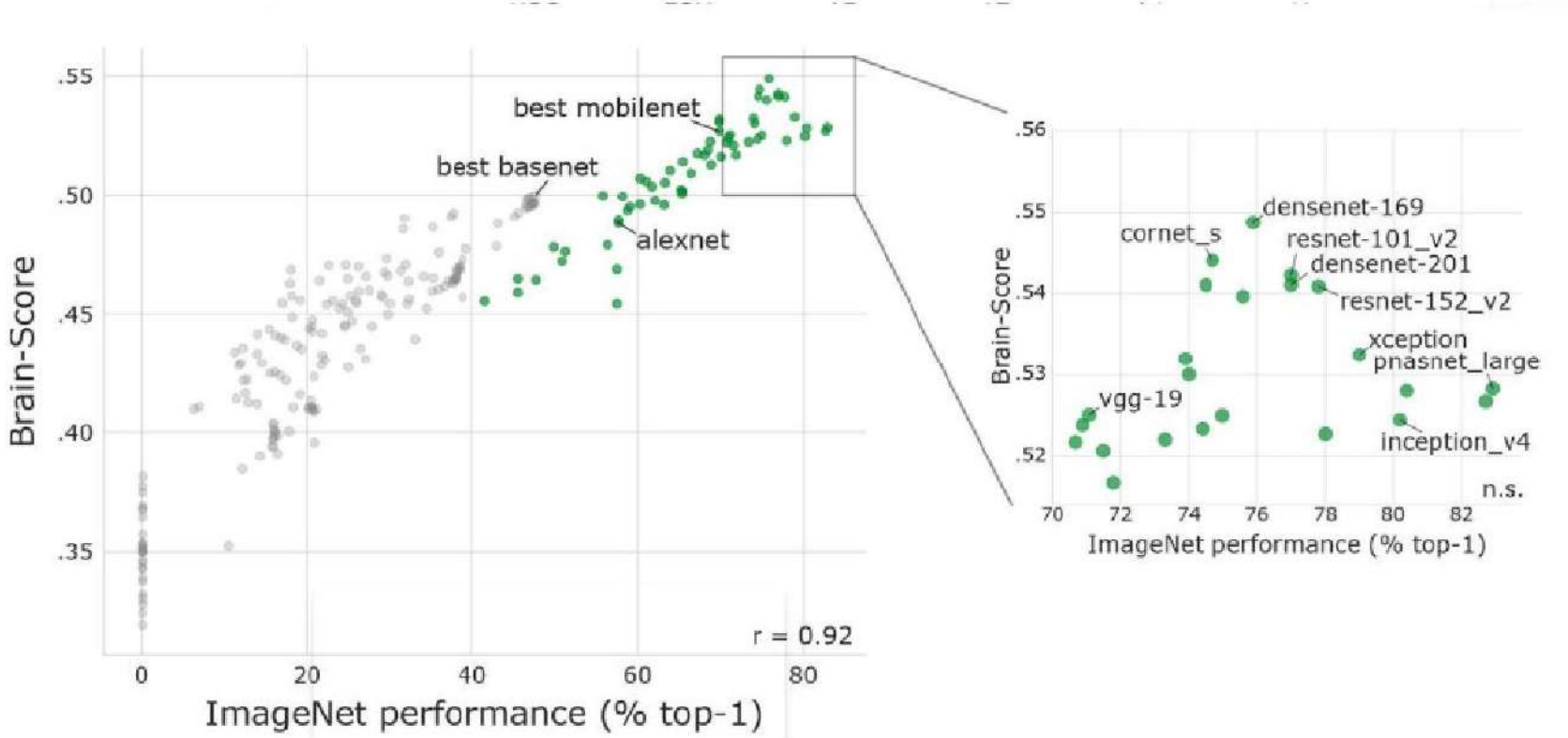
Article history:
Available online 28 September 2021

Keywords:
Artificial intelligence
Neuroscience
Neural imaging
Visual processing
Working memory
Computational psychiatry

ABSTRACT

Neuroscience and artificial intelligence (AI) share a long history of collaboration. Advances in neuroscience, alongside huge leaps in computer processing power over the last few decades, have given rise to a new generation of *in silico* neural networks inspired by the architecture of the brain. These AI systems are now capable of many of the advanced perceptual and cognitive abilities of biological systems, including object recognition and decision making. Moreover, AI is now increasingly being employed as a tool for neuroscience research and is transforming our understanding of brain functions. In particular, deep learning has been used to model how convolutional layers and recurrent connections in the brain's cerebral cortex control important functions, including visual processing, memory, and motor control. Excitingly, the use of neuroscience-inspired AI also holds great promise for understanding how changes in brain networks result in psychopathologies, and could even be utilized in treatment regimes. Here we discuss recent advancements in four areas in which the relationship between neuroscience and AI has led to major advancements in the field; (1) AI models of working memory, (2) AI visual processing, (3) AI analysis of big neuroscience datasets, and (4) computational psychiatry.

Which ANN?: Brain score



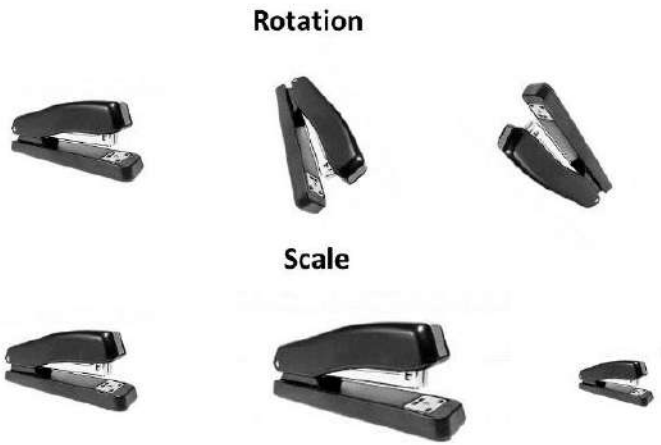
"Brain-Score: Which Artificial Neural Network for Object Recognition is most Brain-Like?" M. Schrimpfet al. 2020, *https://www.brain-score.org/*

Score based on: neural activity matching and behavioural tasks (recognition etc.).

Invariance w.r.t. to what?: group transformations

Group: set $\mathcal{G}$ with composition/multiplication operation satisfying:

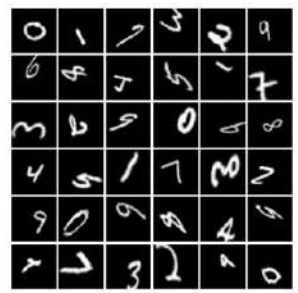
1. closure: $gg^{'} \in \mathcal{G}, \ \forall \ g, g^{'} \in \mathcal{G}$
2. associativity: $(gg^{'})g^{''} = g(g^{'}g^{''}) \in \mathcal{G}, \ \forall \ g, g^{'}, g^{''} \in \mathcal{G}$
3. identity: there exists $Id \in \mathcal{G}$ such that $Idg = gId = g \ \forall \ g \in \mathcal{G}$
4. invertibility: $\forall \, g \in \mathcal{G}$ there exists $g^{-1} \in G \ : \ gg^{-1} = Id.$



Examples of Transformations/Groups

- Compact: rotation
- Locally compact: translation, scaling (multiplication), affine
  e.g., Translation group: linear operator $T_\tau : \mathcal{X} \to \mathcal{X}$

$$T_\tau x(p) = x(p - \tau), \quad \forall p, \tau \in \mathbb{R}, x \in \mathcal{X}$$



- Non-locally compact: diffeomorphisms, local/global deformations
  e.g. given a smooth map $d : \mathbb{R} \to \mathbb{R}$: linear operator $D_d : \mathcal{X} \to \mathcal{X}$

$$D_d x(p) = x(d(p)), \quad \forall p \in \mathbb{R}, x \in \mathcal{X}$$

- Non-group transformations
  - e.g. 3D rotations in depth, illumination.

Defining invariance w.r.t. group transformations of stimuli: equivalence classes

Let $\mathcal{G}$ a group. Let X set is a collection of orbits of stimuli:

$$O_t = \{t' \mid t' = gt,\ g \in \mathcal{G},\ t \in \mathcal{X}\}, \qquad X = (O_{t_1}, \cdots, O_{t_M})$$

The group $\mathcal{G}$ induces a **partition on the data**:

$$\boxed{x' \sim x \ \Leftrightarrow\ x' = gx, \quad \exists g \in \mathcal{G}}$$

Invariant and Selective Representations

Let $\Phi : \mathcal{X} \to \mathcal{F}$.

Invariance:

$$x' \sim x \Rightarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

$$x' = gx \Rightarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

(trivial invariant representations are possible, e.g. constant function)

Selectivity:

$$x' \sim x \Leftarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

$$x = gx' \Leftarrow \Phi(x) = \Phi(x'), \quad \forall x, x' \in \mathcal{X}$$

Invariant representations: Haar measure

Example (Translation group/Lebesgue measure)

$$d(T_{s'}s) = d(s - s') = ds$$

Translation invariance of the integral:

$$\int f(s)ds = \int f(s - s')ds$$

Key observation: a (locally) compact group has associated an **invariant Haar measure** dg i.e.

$$d(gg') = dg, \quad \forall g' \in \mathcal{G}$$

Building invariants by group averaging (1)

Haar invariance allows to build invariants in the form of group averages:

$$I_f = \int dg f(g) = \int dg f(ge) \equiv I_f(e)$$

In fact

$$\begin{aligned} I_f(\bar{g}) = & \int dg f(g\bar{g}) \\ = & \int d(\hat{g}\bar{g}^{-1}) f(\hat{g}) \qquad \hat{g} = g\bar{g}, \quad g = \hat{g}\bar{g}^{-1} \\ = & \int d\hat{g} f(\hat{g}), \quad d(\hat{g}\bar{g}^{-1}) = d\hat{g} \\ = & \int d\hat{g} f(\hat{g}) = I_f(e) \end{aligned}$$

where we used: 1) reparameterization of group elements 2) Haar invariance 3) group closure under composition.

Choosing a specific representation

We choose f as:

$$f_{x,t}(g) = \eta \langle gx, t \rangle$$

where $\eta : \mathbb{R} \to \mathbb{R}$ and $t \in \mathcal{X}$.

This choice is good for:

- Biological plausibility (we will see later)
- Allows to build an invariant representation of x

Invariant representations by group averaging (2)

Theorem

Invariance $\mu_t : \mathcal{X} \to \mathbb{R}$ *defined as*

$$\mu_t(x) = \int dg \;\; \eta \left\langle gx, t \right\rangle, \quad x, t \in \mathcal{X}$$

is invariant w.r.t. transformations $x \to \bar{g}x, \;\; \forall \bar{g} \in \mathcal{G}$

Indeed

$$\begin{aligned}
\mu_t(\bar{g}x) &= \int dg \; \eta \left\langle g\bar{g}x, t \right\rangle \\
&= \int dg \; \eta \left\langle \hat{g}x, t \right\rangle \qquad \hat{g} = g\bar{g}, \quad g = \hat{g}\bar{g}^{-1} \\
&= \int d\hat{g} \; \eta \left\langle \hat{g}x, t \right\rangle, \quad d(\hat{g}\bar{g}^{-1}) = d\hat{g} \\
&= \mu_t(x)
\end{aligned}$$

Enough projections ensure selectivity

If we have enough t-projections we can discriminate signals s.t. $x \not\sim x'$

Define a metric given K templates/projections:

$$D_K(x, x') = \frac{1}{K} \sum_{k=1}^{K} |\mu_{t_k}(x) - \mu_{t_k}(x')|.$$

A simple concentration inequality shows that:

$$|D_K(x, x') - D_\infty(x, x')| \leq \epsilon, \quad \text{with probability } 1 - \delta^2$$

if $K > (\frac{2}{c}\epsilon^{-2}) \ln(\frac{Q}{\delta})$ with Q the number of equivalence classes.

This can be interpreted as a random projections/Johnson-Lindenstrauss type theorem but on 1) On quotient space w.r.t. $\mathcal{G}$. 2) With non-linear projections.

$$\mu_t(x) = \int dg \;\; \eta \langle gx, t \rangle$$

Do we need the full orbit of x?

Transferability and sample complexity

μ_t uses full orbit of x to build an invariant representation. However:

$$\langle gx, t \rangle = \langle x, g^{-1}t \rangle, \forall x, t \in \mathcal{X}$$

Then we have mathematical equivalence of

- μ_t calculated having O_x
- μ_t calculated having O_t

i.e.

$$\mu_t(x) = \int dg\, \eta \langle gx, t \rangle = \int dg\, \eta \langle x, g^{-1}w \rangle = \int dg\, \eta \langle x, gt \rangle$$

We need only one example of x and a set of transformed templates (memorized or learned) to have an invariant representation of x.

Main result

Theorem (Invariance and Selectivity for Finite Templates)

For Q orbits of $\mathcal{G}$ in $\mathcal{X}$, with $K > (\frac{2}{c}\epsilon^{-2}) \ln(\frac{Q}{\delta})$ templates, $\epsilon, \delta > 0$, the representation

$$\Phi(x) = (\mu_1(x), \cdots, \mu_K(x))$$

is

- **Invariant** *i.e.* $\Phi(gx) = \Phi(x),\ \forall g \in G$
- **(Almost) selective** *i.e.* $|D_K(x, x') - D_\infty(x, x')| \leq \epsilon$ *with probability* $1 - \delta^2$.

In other words:

$$\Phi(x) = \Phi(x') \iff x \sim x'$$

Algorithmic remarks

$$\mu_t(x) = \int dg\eta \langle x, gt \rangle$$

- We consider a finite group: $\mathcal{G} = \{g_1, \cdots, g_N\}$.
- Activation function: step functions, sigmoids, max, moments, with different thresholds e.g.
$$\eta_b \langle gx, t \rangle = \sigma(\langle x, gt \rangle - b)$$
- Templates are generic (not reflecting data distribution) and finite $\mathcal{T} = \{t^k\}_{k=1}^K$.

$$\mu_b^k(x) = \sum_i \sigma(\langle g_i x, t_k \rangle - b)$$

Algorithmic steps

Template sampling $t \in \mathcal{T} = \{t^k\}_{k=1}^K$

Projections on transforming templates $\{\langle x, g_j t^k \rangle\}, j = 1, \ldots, |G|$

Pooling (sum) using B non-linear functions (e.g., sigmoids)

$$\mu_b^k(x) = \sum_{j=1}^{|G|} \eta_b \langle x, g_j t^k \rangle, t^k \in \mathcal{T}, g_j \in G, b = 1 \ldots B$$

Signature from components

$$\Phi(x) = (\{\mu_1^1(x)\}, \{\mu_2^1(x)\}, \ldots, \{\mu_N^K(x)\}) \in \mathbb{R}^{N \times |\mathcal{T}|}$$

Number of templates (to separate $|\hat{Y}|$ classes with probability $1 - \delta^2$)

$$K \geq \frac{2}{c\epsilon^2} \log \frac{|\hat{Y}|}{\delta}$$

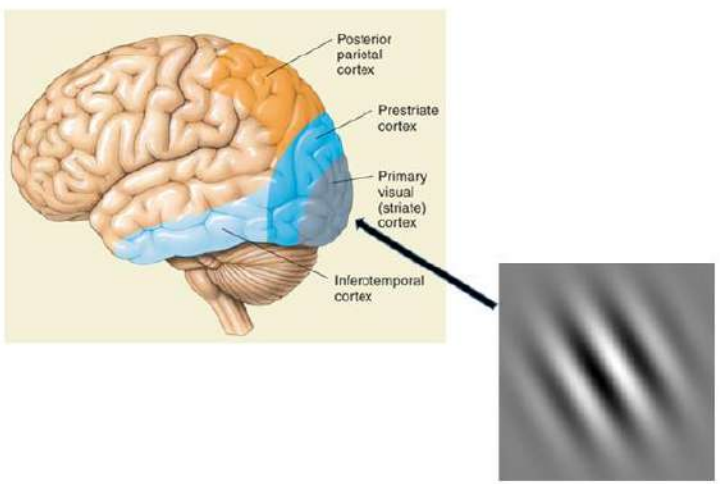
What if the group is locally compact? approximate invariance and Gabor templates

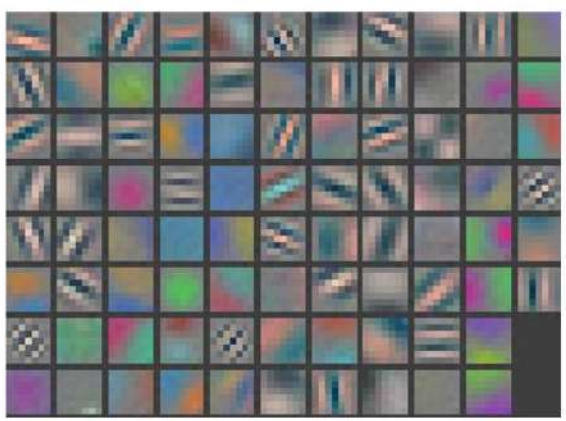
Suppose we want to implement translation and scale invariance

$$\mu_t(x) = \sum_{\lambda,y} \sigma \langle x, D_\lambda T_y t \rangle$$

Which is the best choice for $t = t^*$ to guarantee approximate invariance?

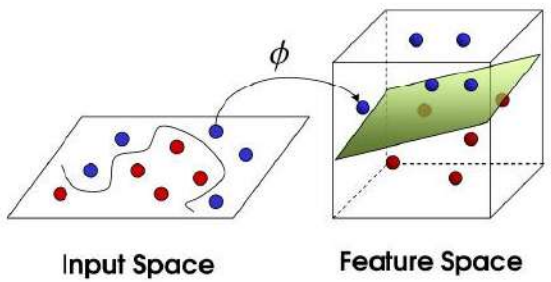
$$t^* = \arg\min_{t,\forall\ \lambda,y} |\mu_t(D_\lambda T_y x) - \mu_t(x)|$$

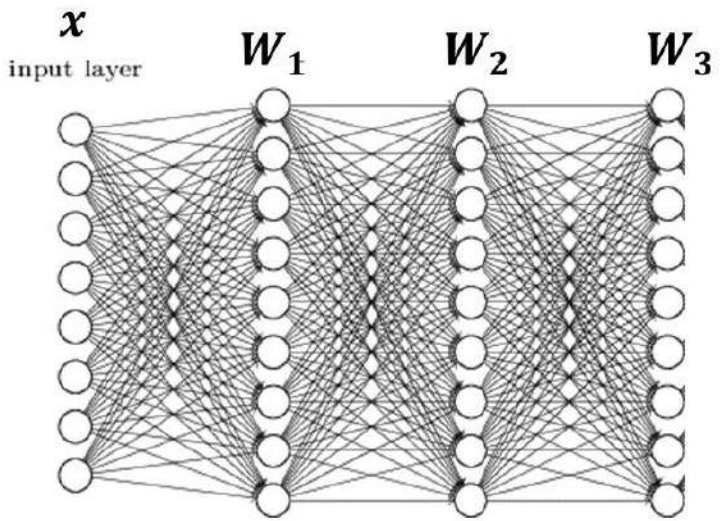




How is this translated in the context of deep networks?

Deep networks: 'smart' way to parameterize functions.





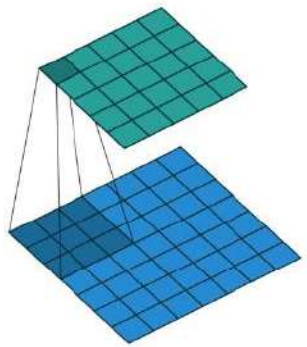
$$x \in \mathbb{R}^d, \quad \Phi : \mathbb{R}^d \to \mathbb{R}^L, \quad \Phi \equiv \Phi_L \circ \cdots \circ \Phi_1$$

$$\Phi_1(x) = \sigma \langle W_1, x \rangle$$
$$\Phi_\ell(x) = \sigma \langle W_\ell, \Phi_{\ell-1}(x) \rangle$$

$\sigma : \mathbb{R} \to \mathbb{R}$, nowadays $\sigma(\cdot) = max(0, \cdot) \equiv |\cdot|_+ \ (w \equiv t)$.

Constraints on network operations/topology

$$W = \begin{bmatrix} w_1 & w_d & w_1 & \dots & w_2 \\ w_2 & w_1 & w_2 & \dots & w_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_d & w_{d-1} & w_{d-3} & \dots & w_1 \end{bmatrix} =$$

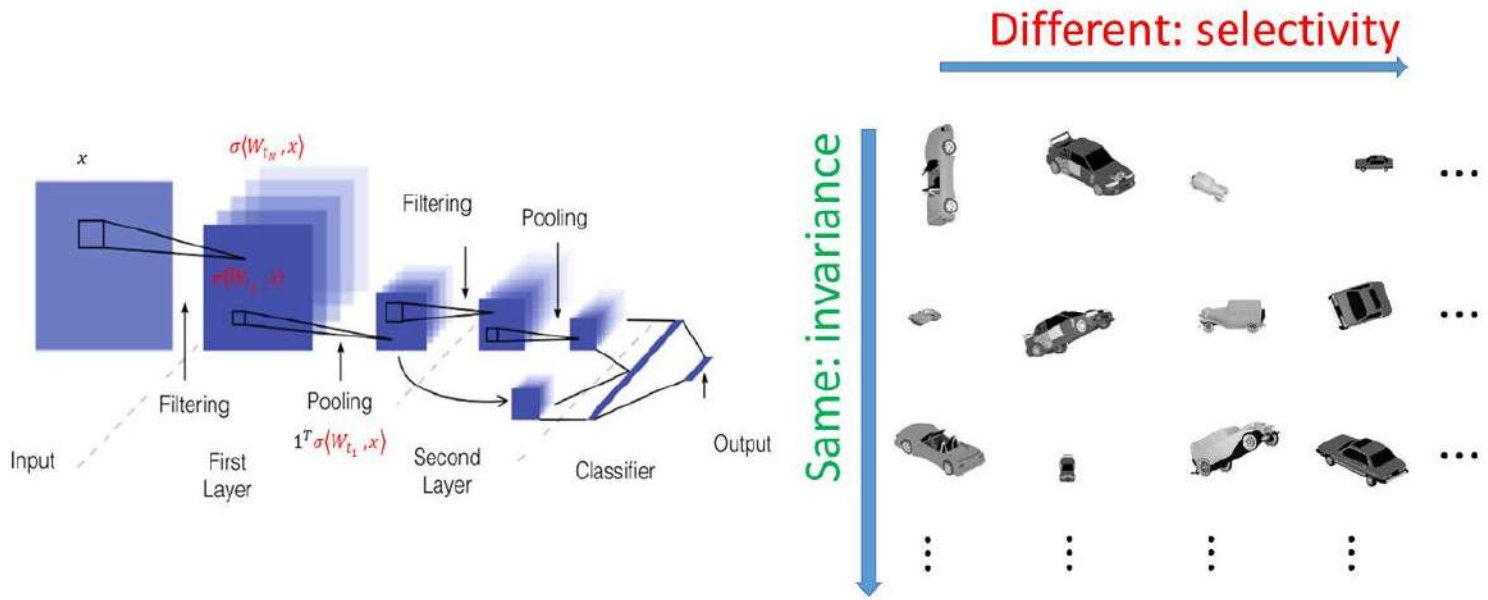


an orbit w.r.t. the cyclic group $\mathcal{G}$. This is equivalent to a convolution $x * w = W^T x$.

Remark

This construction can be extended to convolutions w.r.t. other groups.

Invariant selective Data Representation



Theorem

Suppose data are generated by $\mathcal{G}$. A CNN with convolutions w.r.t. $\mathcal{G}$ is implementing a data representation Φ that is invariant and selective i.e. $\mathbf{x} \sim \mathbf{x}' \Leftrightarrow \mathbf{\Phi}(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x}')$ *with (one layer)* $\Phi : \mathbb{R}^d \to \mathbb{R}^p$ *and:*

$$\Phi_w(x) = \sum_i |(x *_{\mathcal{G}} w)_i|_+ = \sum_i |(W^T x)_i|_+ = \sum_i |\langle x, g_i w \rangle|_+$$

Orbits and probability distribution: some definitions

Consider the map $x \to gx$ where g is the random variable. This defines a probability distribution $x \to P_x(g)$ the probability distribution supported on each orbit

$$O_x = \{x' \mid x' = gx,\ g \in \mathcal{G}\}.$$

How do we compare different distributions $P_x, P_{x'}$? We use arbitrary (continuous) test functions $f \in C_c(\mathcal{X})$ and the use the following definition

$$I(f, P_x) \equiv \int_{O_x} f(y) dP_x(y) = \int_{\mathcal{X}} f(y) dP_x(y) = \int_{\mathcal{G}} f(gx) dg,\ \ \forall\ f \in C_c(\mathcal{X})$$

with dg the Haar measure on the group. Then $P_x = P_{x'}$ iff $I(f, P_x) = I(f, P_{x'})$ for all f.

Demonstration steps

We sketch the demonstration in 3 steps:

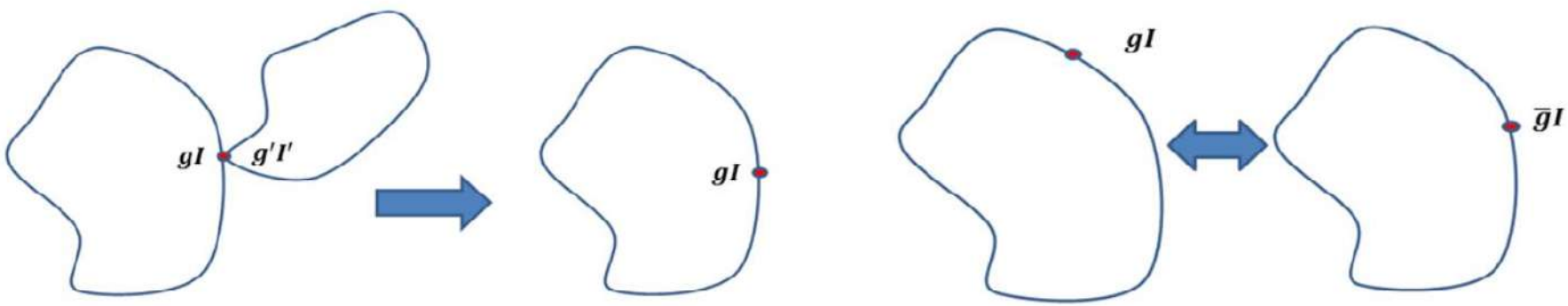
1. $x \sim x' \iff O_x = O_{x'}$
2. $O_x = O_{x'} \iff P_x = P_{x'}$
3. $P_x = P_{x'} \iff \Phi(x) = \Phi(x')$

where Φ is the CNN representation.

Equivalence classes and orbits

$$x \sim x' \iff O_x = O_{x'}$$

1. $x \sim x' \Rightarrow x = gx' \Rightarrow O_x = O_{gx'} = O_{x'}$
2. $O_x = O_{x'} \Rightarrow x = gx' \Rightarrow x \sim x'$



Equivalence Orbits-probability distribution

Theorem

$O_x = O_{x'} \Leftrightarrow P_x = P_{x'}$

$\Rightarrow$: If $O_x = O_{x'} \Rightarrow\ x = hx', \exists h \in \mathcal{G}$

Then using the definition

$$\int_{O_x} f(y)dP_x(y) = \int_{\mathcal{G}} f(gx)dg = \int_{\mathcal{G}} f(ghx')dg$$
$$= \int_{\mathcal{G}} f(gx')dg = \int_{O_{x'}} f(y)dP_{x'}(y)$$

for all f and thus $P_x = P_{x'}$.

$\Leftarrow$: If $P_x = P_{x'} \Rightarrow\ supp(P_x) = supp(P_{x'}) \Rightarrow\ O_x = O_{x'}$.

Mean probability embedding (3)

Theorem (Probability mean embedding, Smola et al. '07)

Let $x \to P_x(g)$ *induced by* $\mathcal{G}$ *and* $\Psi(x)$ *be a "rich enough" representation* [a]. *Then*

$$P_x = P_{x'} \quad \Leftrightarrow \quad \Phi(x) = \int dg\ \Psi(gx) = \int dg\ \Psi(gx') = \Phi(x')$$

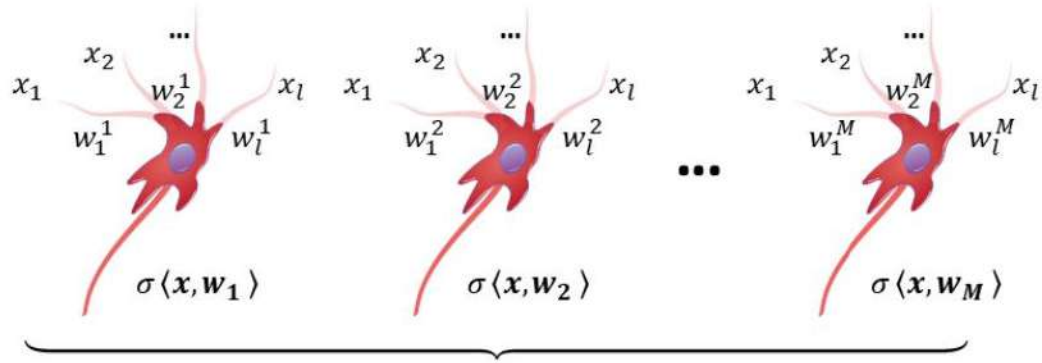
i.e. the distribution is one to one with the representation map.

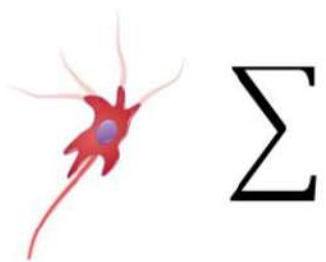
In the case of CNNs $(\Psi(g_i x))_{w,i} = |\langle x, g_i w \rangle|_+ = |(x * w)_i|_+$ *and*

$$(\Phi(x))_w = \sum_i |\langle W_t, x \rangle_i|_+ = \sum_i |\langle g_i w, x \rangle|_+$$

[a] The corresponding kernel universal, i.e. dense in the space of continuous functions

Aggregating single neurons responses: Hubel Wiesel model



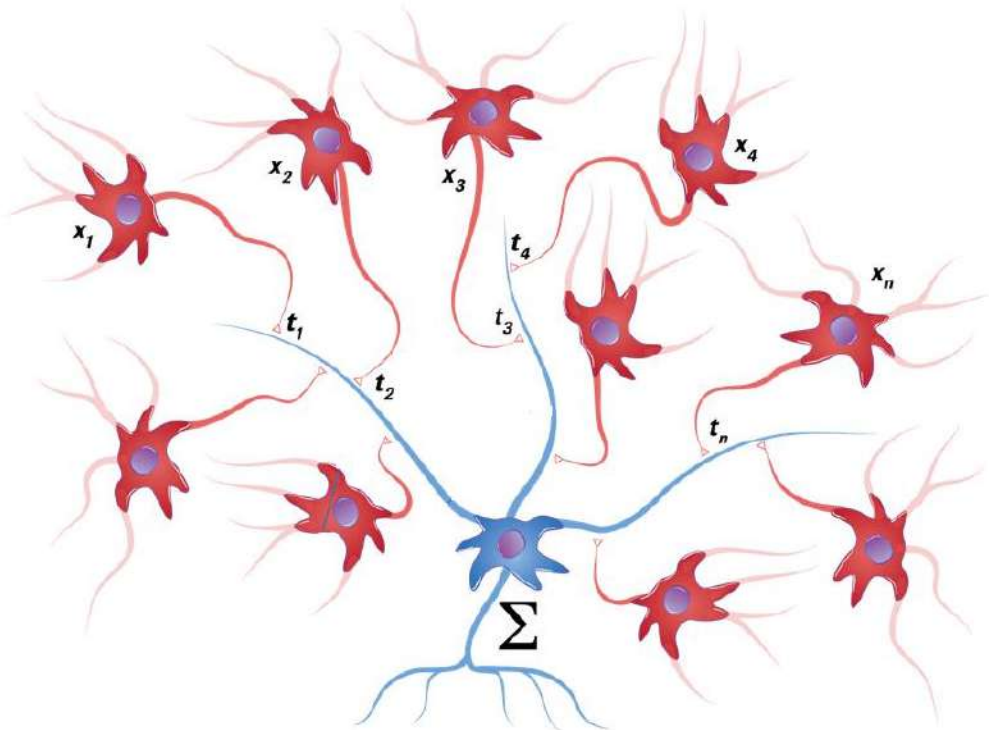


$$\sum$$

$$\mu_w(x) = \sum_{i=1}^{M} \sigma\langle x, w_i \rangle$$

This module is often called in visual cortex *Simple-Complex* and $\mu(x)$ is a neural representation of the input x.

Translating our results



Acascade of simple and complex cells is implementing an invariant and selective representation.

Robustness to more general transformations and locality

Let s a C^∞ transformation depending on $\Theta = (\theta_1, \ldots, \theta_P)$ parameters.
Expanding, e.g., around e.g. 0:

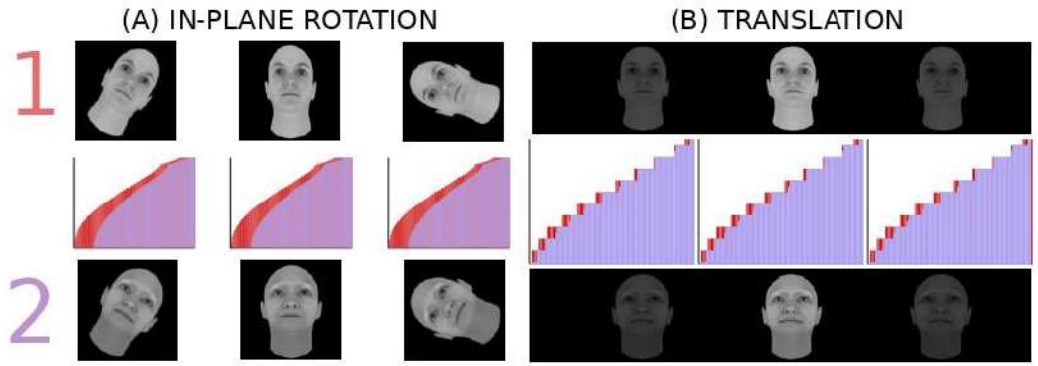
$$s(x, \Theta) = s(x, \mathbf{0}) + \sum_{i=1}^{P} \frac{\partial s(x, \Theta)}{\partial \theta_i} \theta_i + o(\|\Theta\|^2) = x + \sum_{i=1}^{P} \theta_i L_{\theta_i}(x) + o(\|\Theta\|^2)$$

where L_{θ_i} are the infinitesimal generators and Therefore locally

$$g(\Theta) = \exp(\theta_1 L_{\theta_1} + \theta_2 L_{\theta_2} + \cdots + \theta_P L_{\theta_P}).$$

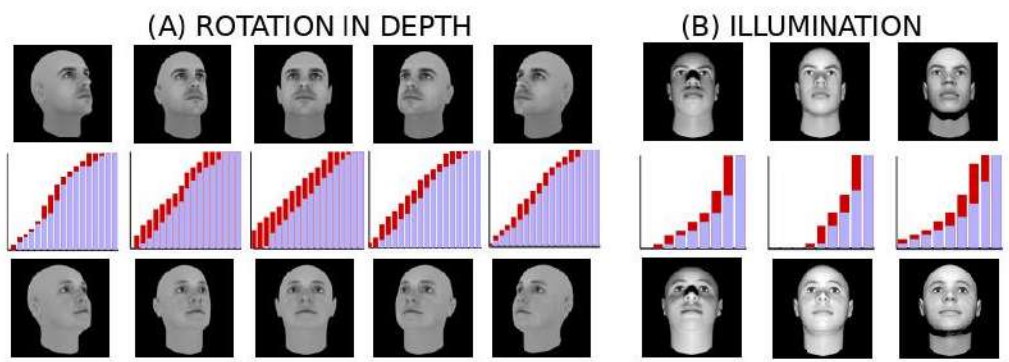
The local learned weights will be **orbits w.r.t. the local group approximating the non-group global transformation**.

Beyond groups



$$\overline{\Phi}^t(x) = \sum_i |\langle g_i t, x \rangle|_+$$

As a processing procedure the principles extend beyond compact group transformations (Leibo et al. '15)



Selectivity: probabilistic argument

(For compact groups) a probability distribution P_x on O_x is defined by measure dg via random variable $g \mapsto gx$

Theorem (Orbit-probability distribution equivalence)

$$(x \sim x' \iff) \; O_x = O'_x \iff P_x \sim P_{x'}$$

"1D" probability distributions P_x^t are defined by projections $gx \mapsto \langle gx, t \rangle$

Theorem (Cramer-Wold)

Probability distributions are uniquely determined by all 1D projections

$$P_x \equiv P_{x'} \iff P_x^t \equiv P_{x'}^t, \quad \forall t \in \mathbb{S}^d$$

1D projection can be characterized by its CDF $P_x^t \mapsto \int dg H(b - \langle x, gt \rangle)$

Learning - Transferability

Unitary groups: $\langle gx, t\rangle = \langle x, g^{-1}t\rangle, \forall x, t \in \mathcal{X}$

$$\mu(x) = \int \eta\left(\langle gx, t\rangle\right) dg = \int \eta\left(\langle x, g^{-1}t\rangle\right) = \int \eta\left(\langle x, gt\rangle\right) dg$$

Memory-based learning:

- View based/augmentation: Need $\{gx\}, \forall g \in \mathcal{G}$ $(\forall x)$
- Memory-based: Store $\{gt\}, \forall g \in \mathcal{G}$ (once)
- Trading time with space requirements
- *Biological learning, learning from sequences, CNNs*

Selective and invariant signature (recap)

Signature:

$$\mu_t(x) = \int dg\eta \langle x, gt \rangle$$

Or its computational-friendly form

$$\mu_b^k(x) = \sum_i \sigma(\langle x, g_i t_k \rangle - b)$$

The signature is invariant and selective.

From previous class: stimuli hypothesis

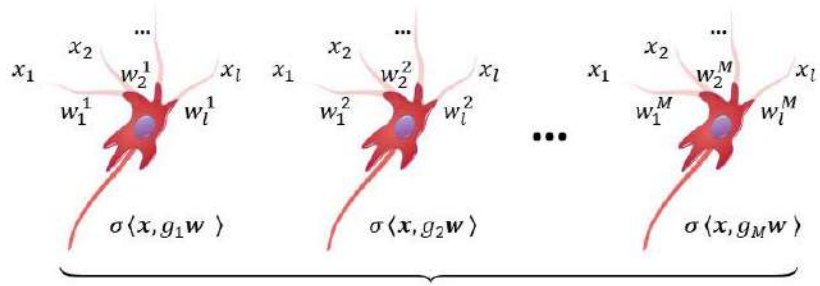
Let $\mathcal{G}$ a group. Let X set is a collection of orbits of stimuli:

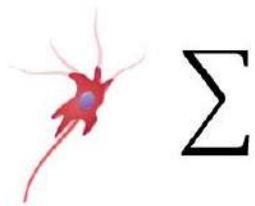
$$O_t = \{t' \mid t' = gt,\ g \in \mathcal{G},\ t \in \mathcal{X}\}, \qquad X = (O_{t_1}, \cdots, O_{t_M})$$

The group $\mathcal{G}$ induces a **partition on the data**:

$$\boxed{x' \sim x \ \Leftrightarrow\ x' = gx, \quad \exists g \in \mathcal{G}}$$

Learning question





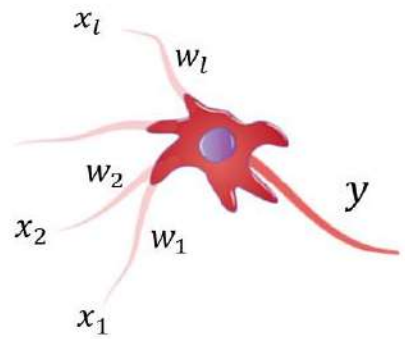
$$\sum$$

$$\mu_w(x) = \sum_{i=1}^{M} \sigma\langle x, g_i w\rangle$$

How can we learn $g_i w$? How can we learn the correct pooling over orbits?

Learning hypothesis: Hebbian learning of simple neurons synaptic weights

Neurons updates their weights during stimuli presentation. We consider a weights-normalized version of the hebbian rule $\Delta \mathbf{w} = \eta y_t x$

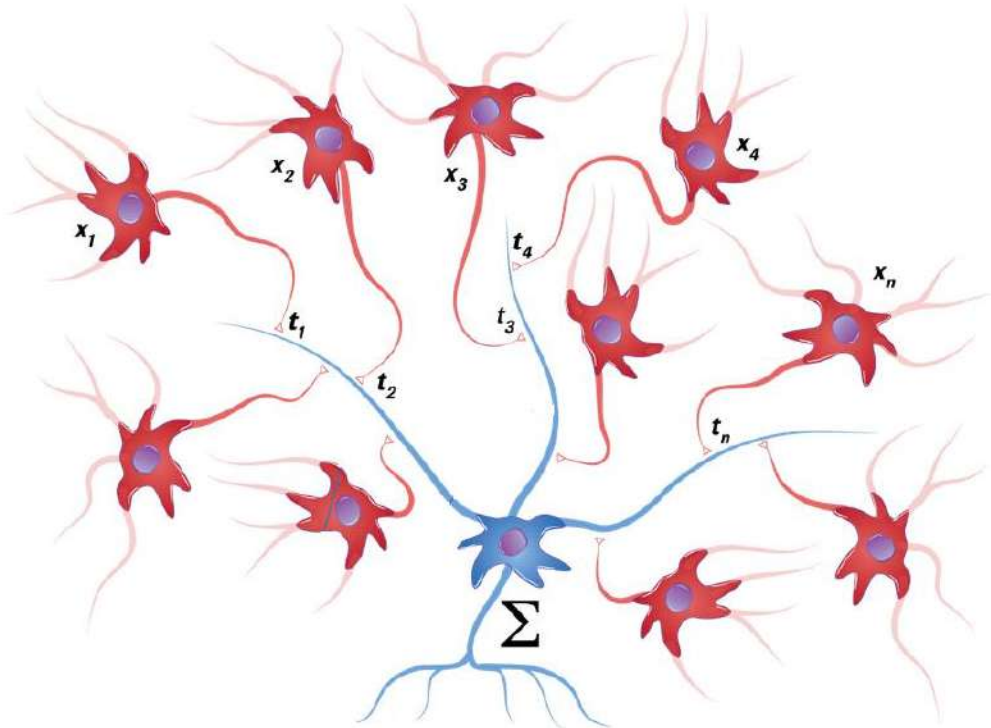


$$y_t = \langle \mathbf{x}_t, \mathbf{w}_t \rangle, \qquad \Delta \mathbf{w} = \eta y_t (\mathbf{x}_t - y_t \mathbf{x}_t)$$

$$XX^T \mathbf{w} = \lambda_{max} \mathbf{w}$$

Simple cells weights converge to the first eigenvector of the covariance of the stimuli.

Question



Can a simple-complex module, $\mu_w(x)$, under the hypotheses above, learn invariant/selective properties w.r.t. $\mathcal{G}$?

Stimuli + learning: simple neurons weights

Theorem

The weights of simple cells converge to linear combinations of elements of an orbit of $\mathcal{G}$

Simple proof:

- $X = \{g_1 t_1, \cdots, g_{|\mathcal{G}|} t_1, \cdots, g_1 t_Q, \cdots, g_{|\mathcal{G}|} t_Q\}$

$$C = XX^T = \sum_i g_i T T^T g_i^T, \quad [C, g] = Cg - gC = 0$$

and

$$Cw = \lambda w \;\; \Rightarrow \;\; Cgw = gCw = \lambda gw, \; \forall \, g \; \in \mathcal{G}, \; w \in E_\lambda$$

- In particular

$$\mathbf{E_{max}} = \mathbf{span}(\mathbf{O_w}), \;\; \forall w \in E_{max}.$$

Extension: equivariant plasticity rules

The result is true for a much broader class of plasticity rules those deriving from a Loss function of the form:

$$\mathcal{L}(X, W) = \sum_{i,j} f(\sigma \langle w_i, x_j \rangle), \qquad f \in \mathcal{C}^1.$$

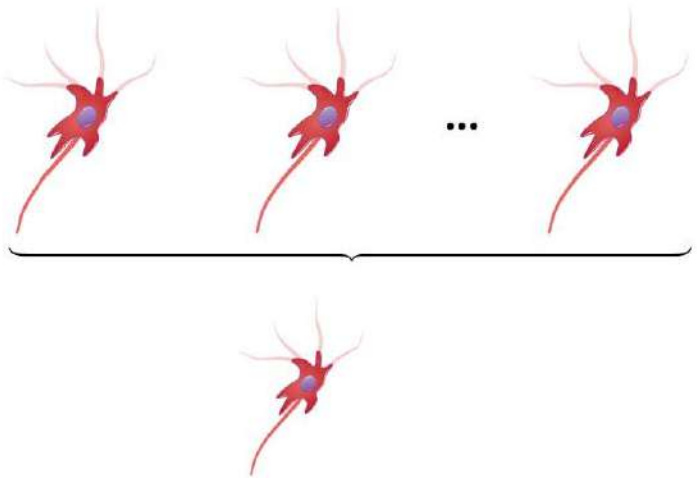
We have that the associated plasticity rule is equivariant w.r.t. $\mathcal{G}$ transformations:

$$\nabla_{w_i} \mathcal{L}(\{w_1, \cdots, g w_i, \cdots, w_N\}, X) = g \nabla_{w_i} \mathcal{L}(\{w_1, \cdots, w_i, \cdots, w_N\}, X).$$

If $\mathbf{w}$ is solution so is $g_i \mathbf{w}$.

Which simple cells is a complex cell aggregating?

Suppose simple weights are learned.



$$O_x = \{x, g_2 x, \cdots, g_M x\}$$
$$\bar{w} = \arg\max_w \langle w, x \rangle$$
$$g_2 \bar{w} = \arg\max_w \langle w, g_2 x \rangle$$
$$\langle x, \bar{w} \rangle = \langle gx, g\bar{w} \rangle$$

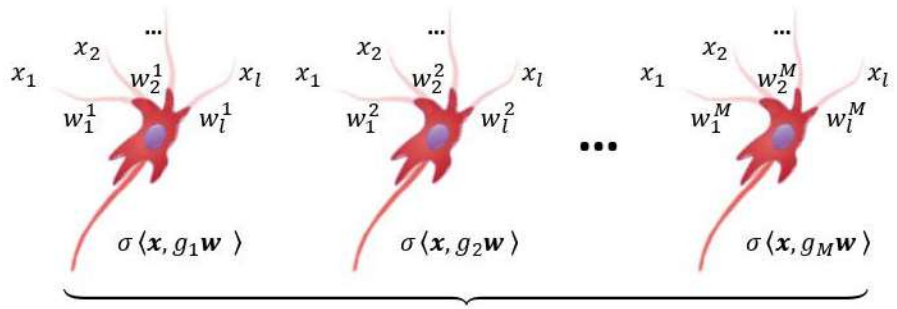
The principle is Hebbian: "fire together link together".

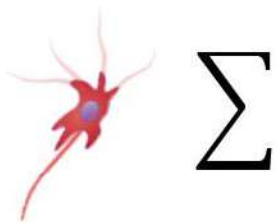
Theorem (Complex cells pooling and invariance)

A complex cell learns to aggregates over simple cells whose weights form an orbit with respect to the group $\mathcal{G}$ i.e. $w^i = g_i w$.

Summarizing:

1. The symmetries in the stimuli are "inherited" by the weights of the simple cells: $\mathbf{E_{max}} = \mathbf{span}(\mathbf{O_w}), \ \ \forall w \in E_{max}.$
2. The set of simple cells aggregated by a complex cell have weights that are orbits i.e. $\mathbf{w^i} = \mathbf{g_i w}.$





$$\sum$$

$$\mu_w(x) = \sum_{i=1}^{M} \sigma\langle x, g_i w\rangle$$

Simple-complex modulus equivariant-invariant properties

- Simple cells are permutation-equivariant to $g \in \mathcal{G}$ transformations.

$$\sigma(W^T g x) = P_g \sigma(W^T x), \quad W = (g_1 w, \cdots, g_{|\mathcal{G}|} w)$$

- Complex cells are invariant to $g \in \mathcal{G}$ transformations.

$$\mu_{\mathbf{w}}(\mathbf{x}) = \sum_i \sigma \langle x, g_i w \rangle = \mu_{\mathbf{w}}(\mathbf{gx}) \quad \forall\, g \in \mathcal{G}$$

Same properties hold for a wild class of aggregating functions (e.g. max) and pointwise non-linearities.

What about selectivity/discriminability?

- Invariance:

$$x' \sim_{\mathcal{G}} x \Rightarrow \mu(x') = \mu(x), \quad \forall x, x' \in X$$

  Elements within an orbit have the same μ.

- Selectivity:

$$x' \sim_{\mathcal{G}} x \Leftarrow \mu(x') = \mu(x), \quad \forall x, x' \in X$$

  Elements belonging to different orbits have different μ.

We want to collapse elements within orbits and tear apart different orbits, i.e. build a proper quotient space.

Discriminability: importance of the non-linearity

The information loss can be recovered considering a family of non-linearities e.g. $\{\sigma_z(\cdot) \equiv H(\cdot - z),\ z \in \mathbb{R}\}$.

$$c_z(x) \equiv (c(x))_z = \sum_{i=1}^{|\mathcal{G}|} H(\langle x, g_i w \rangle - z), \quad z \in \mathbb{R}.$$

We have:

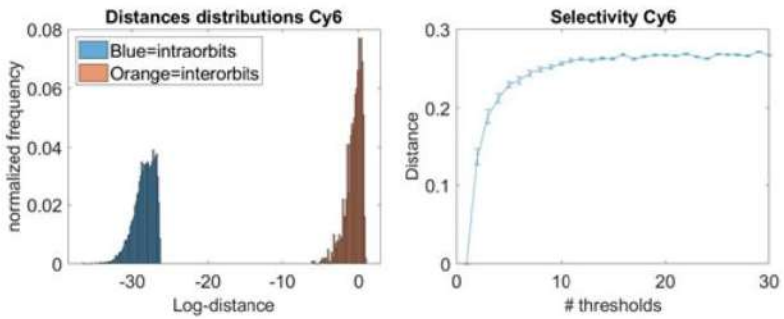
Theorem (Complex cells selectivity)

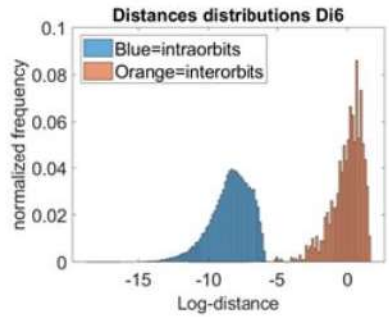
Let $x, x' \in \mathbb{R}^d$ be two input and $c(x), c(x')$ their complex cells responses. Then the distance defined as

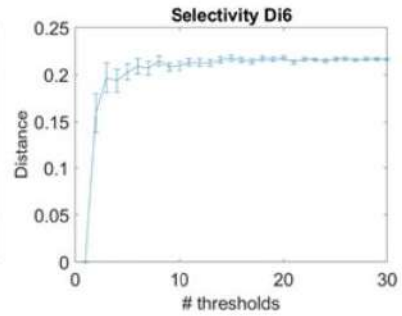
$$dist(x, x') := \|c(x) - c(x')\|_{\ell_2}.$$

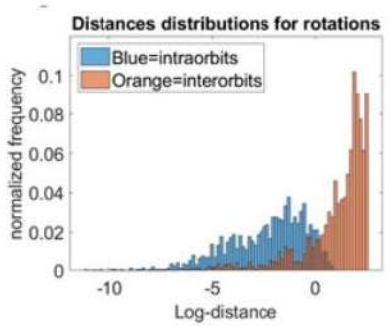
is zero iff $x \sim x'$.

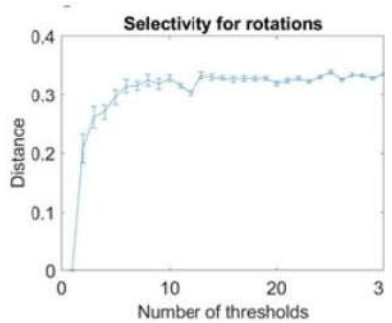
Some simulations







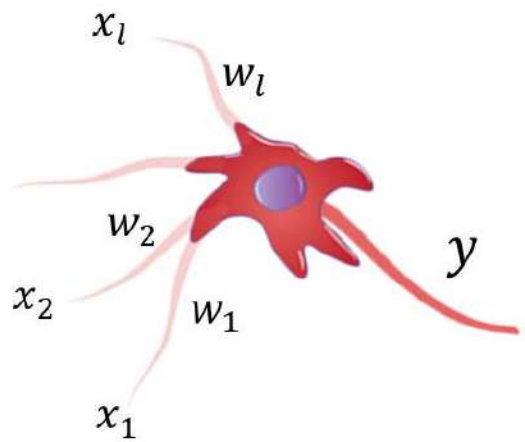




- X generated by: 1) Cyclic group, 2) Dihedral group, 3) Finite group of rotations on natural image patches.
- Plots: Intra and inter orbit distances for N orbits. Discriminability vs number of thresholds.

Oja's learning on orbits

Oja's learning:



$$w_i(t+1) = w_i(t) + \lambda y(x) x_i$$
$$||w|| = 1$$

$$w \to first\ PCA(X)$$

The covariance matrix of $D = \{(f_1, Rf_1), \cdots, (f_N, Rf_N)\}$ is

$$C = D^T D = FF^T + RFF^T R^T$$

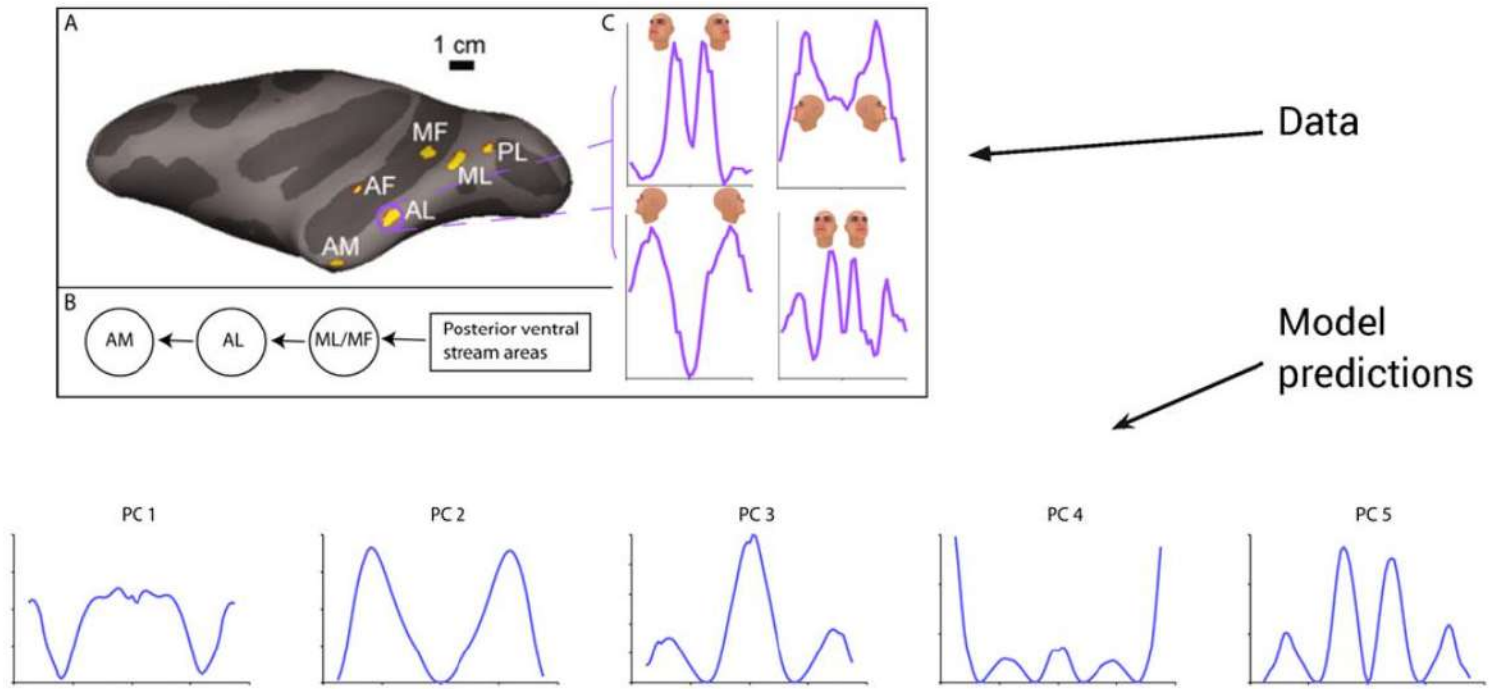
with $F = \{f, \cdots, f_N\}$ and we have

$$[C, R] = 0$$

i.e. *the eigenfunctions are such that* $Rw = w$ since Rw and w are eigenfunctions with the same eigenvalue.

Mirror symmetric neuronal receptive fields

Calculating the eigenfunctions of the covariance matrix:



View-invariance implies mirror symmetric orientation tuning curves

Geometric deep learning

- Geometric deep learning involves developing algorithms and models for analyzing **data structured in non-euclidean domains** (graphs, networks, or manifolds)

- It combines principles from graph theory, differential geometry, and topology to enable learning and processing of complex shapes and structures that are difficult to represent in traditional deep learning frameworks.

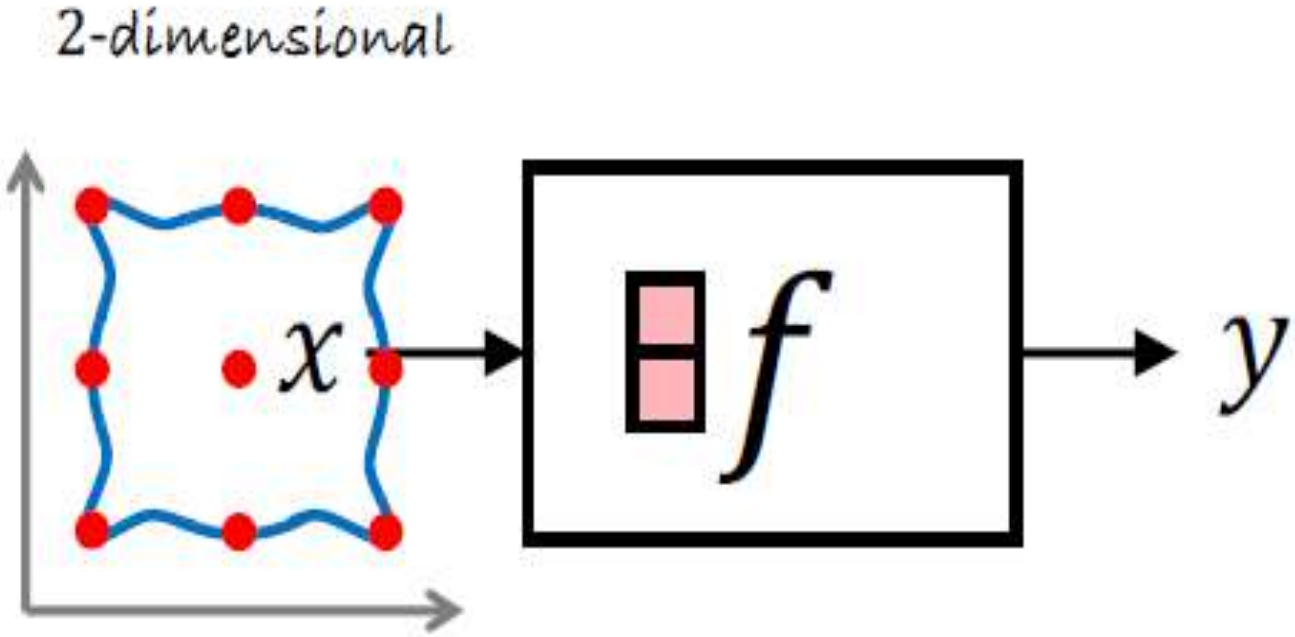
- One important direction is the understanding of the geometry of the data and the **design of meaningful priors** to decrease the sample complexity of the learning

Class 1

- Curse of dimensionality and geometric deep learning
- Fighting the curse with priors, examples
- Implicit regularization
- Symmetry

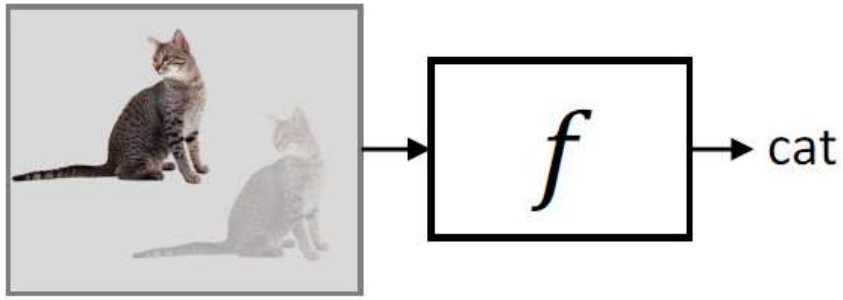
See: https://geometricdeeplearning.com/

Curse of dimensionality

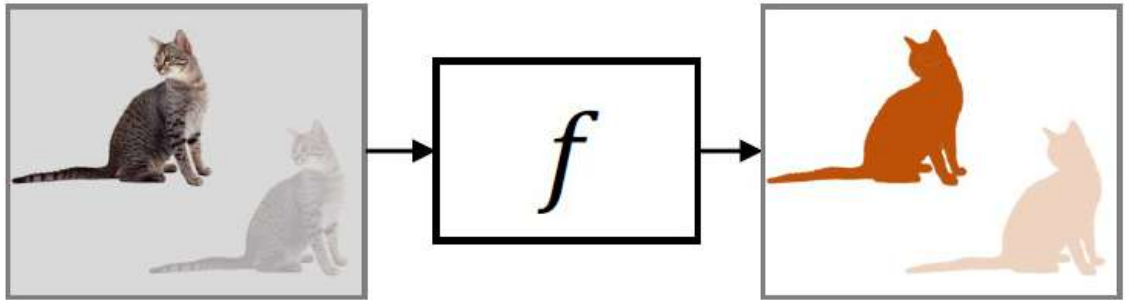


Fight against the curse: priors on the function space. Equivariance

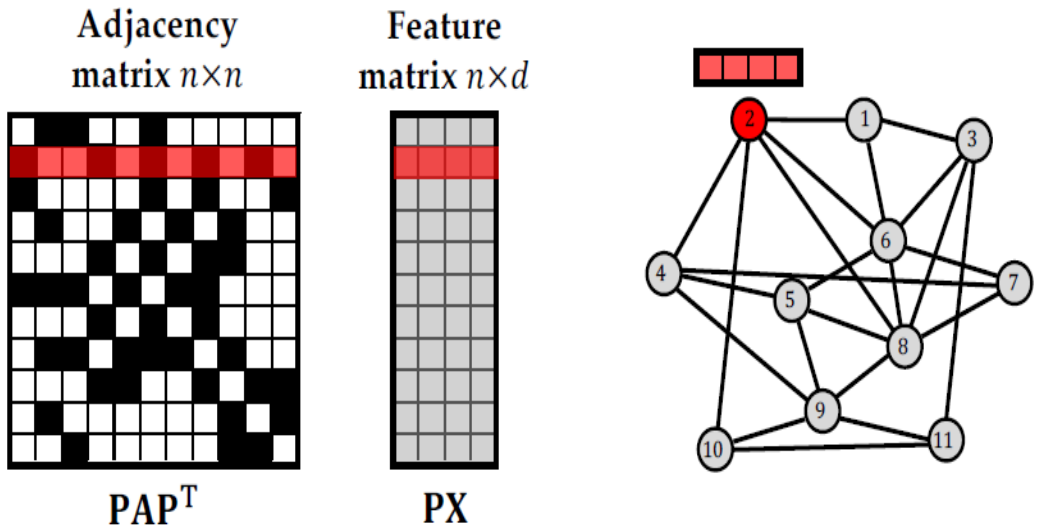
$\mathfrak{G}$-invariance $f(\rho(\mathfrak{g})x) = f(x)$

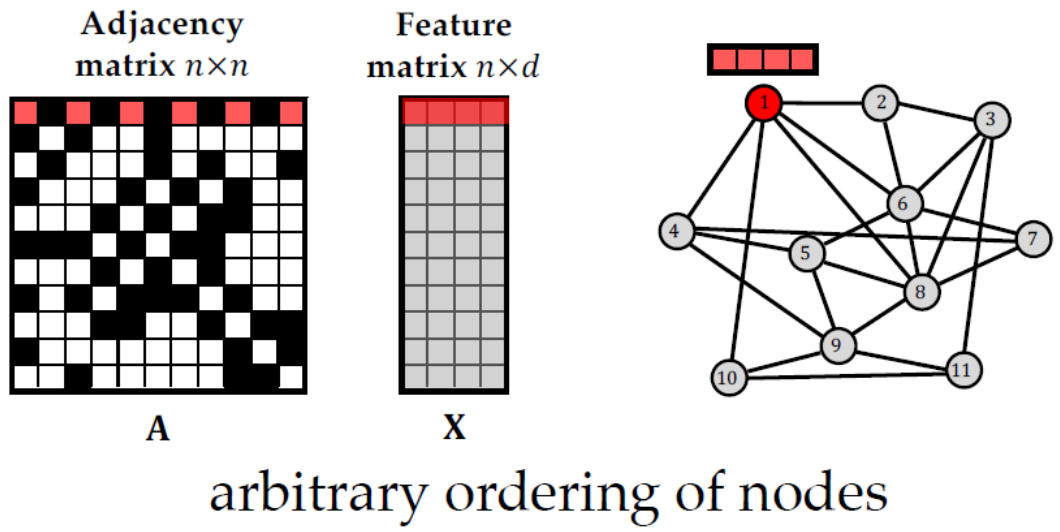


$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$



Example with graphs, graph isomorphism



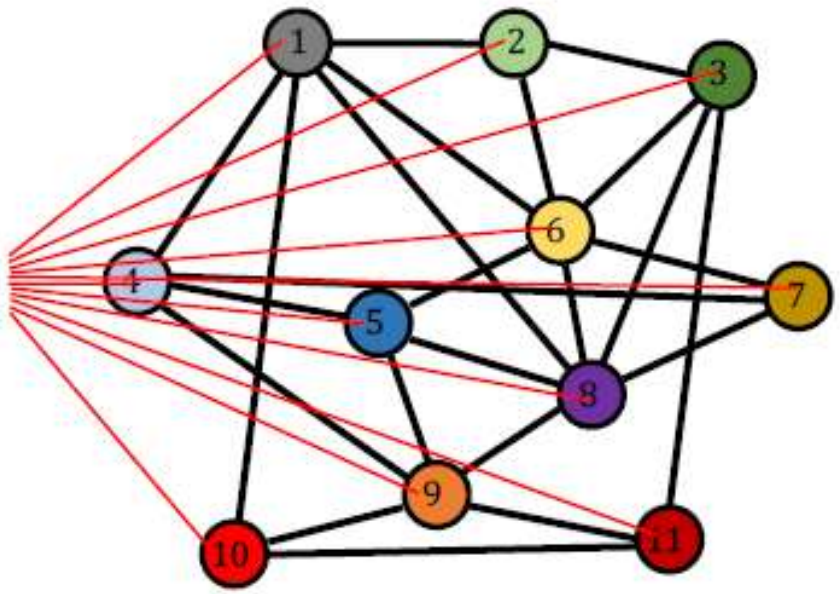


It is a very good idea to look for permutation invariant functions!

Invariant Graph Functions

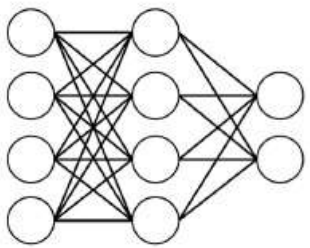
permutation-invariant

$$f(\mathbf{PX}, \mathbf{PAP}^\top) = f(\mathbf{X}, \mathbf{A})$$

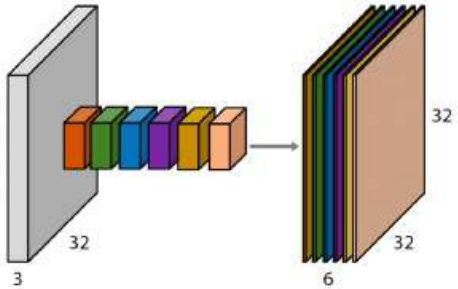


How do you contruct a permutation Invariant layer?

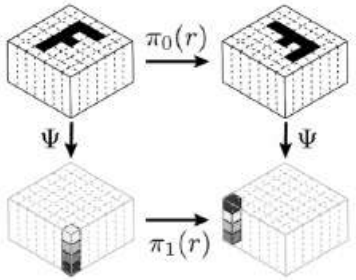
Other examples



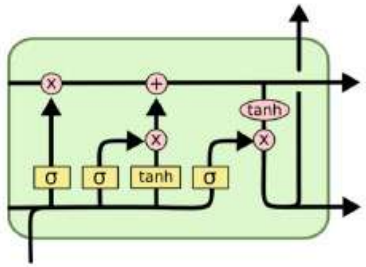
Perceptrons
Function regularity



CNNs
Translation

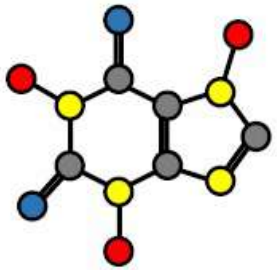


Group-CNNs
Translation+Rotation,
Global groups



LSTMs
Time warping

DeepSets / Transformers
Permutation



GNNs
Permutation

Intrinsic CNNs
Isometry / Gauge choice

Symmetries of the Label Function

- Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the label space
- Let $L : \mathcal{X} \to \mathcal{Y}$ be the ground-truth label function
- A transformation $\mathfrak{g} : \mathcal{X} \to \mathcal{X}$ is a symmetry if $L \circ \mathfrak{g} = L$

L() = L() = “dog”

Symmetries of the Parameterization

- Let $\mathcal{X}$ denote the input space, $\mathcal{Y}$ the label space, and $\mathcal{W}$ the weight space
- Let $\mathrm{f} : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$ denote a model (e.g. neural network)
- A transformation $\mathfrak{g} : \mathcal{W} \to \mathcal{W}$ is a symmetry of the parameterization if

$$f(x, \mathfrak{g}\, w) = f(x, w) \quad \text{for all } \mathrm{x} \in \mathcal{X} \text{ and } \mathrm{w} \in \mathcal{W}$$

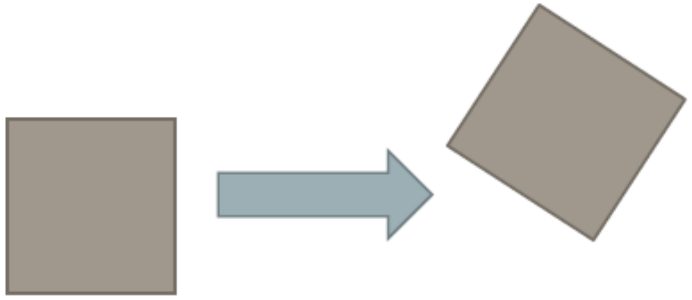
Swapping the incoming and outgoing connections of two neurons in the same layer does not change the input-output map $f(\cdot, w)$

What is the structure that allows to bridge symmetries of parameterization and data as equivalent?

Symmetries of the data: transformations, groups

Example: Euclidean planar motions acting on $\mathbb{R}^2$:

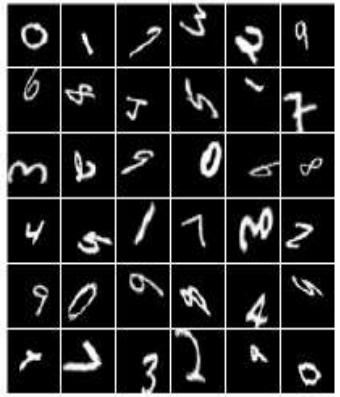
$$((\theta, t_x, t_y), (x, y)) \mapsto \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Transformations groups

- compact: rotation
- locally compact: translation, scaling (multiplication), affine
  e.g., Translation group: linear operator $T_\tau : \mathcal{X} \to \mathcal{X}$

$$T_\tau x(p) = x(p - \tau), \quad \forall p, \tau \in \mathbb{R}, x \in \mathcal{X}$$



- non-locally compact: diffeomorphisms, local/global deformations
  e.g. given a smooth map $d : \mathbb{R} \to \mathbb{R}$: linear operator $D_d : \mathcal{X} \to \mathcal{X}$

$$D_d x(p) = x(d(p)), \quad \forall p \in \mathbb{R}, x \in \mathcal{X}$$

- non-group transformations
  - e.g. 3D rotations, illumination

Group properties

A *group* is a set $\mathfrak{G}$ with a binary operation denoted $\mathfrak{gh}$ satisfying the following properties:

- *Associativity:* $(\mathfrak{gh})\mathfrak{k} = \mathfrak{g}(\mathfrak{hk})$ for all $\mathfrak{g}, \mathfrak{h}, \mathfrak{k} \in \mathfrak{G}$
- *Identity*: there exists a unique $\mathfrak{e} \in \mathfrak{G}$ satisfying

$$\mathfrak{ge} = \mathfrak{eg} = \mathfrak{g}$$

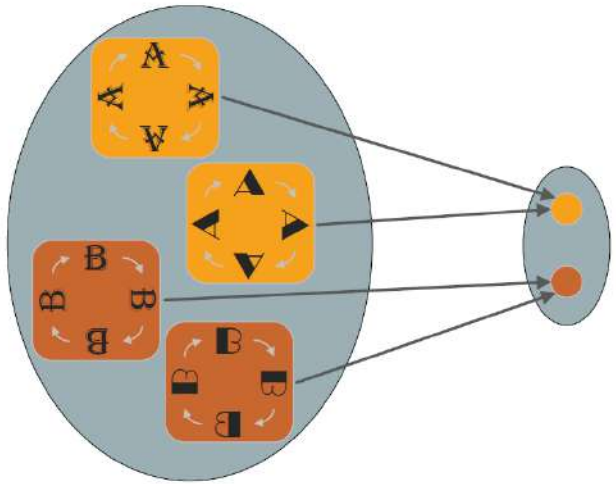
- *Inverse:* for each $\mathfrak{g} \in \mathfrak{G}$ there is a unique inverse $\mathfrak{g}^{-1} \in \mathfrak{G}$,

such that $\mathfrak{gg}^{-1} = \mathfrak{g}^{-1}\mathfrak{g} = \mathfrak{e}$

- *Closure*: for every $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$, we have $\mathfrak{gh} \in \mathfrak{G}$

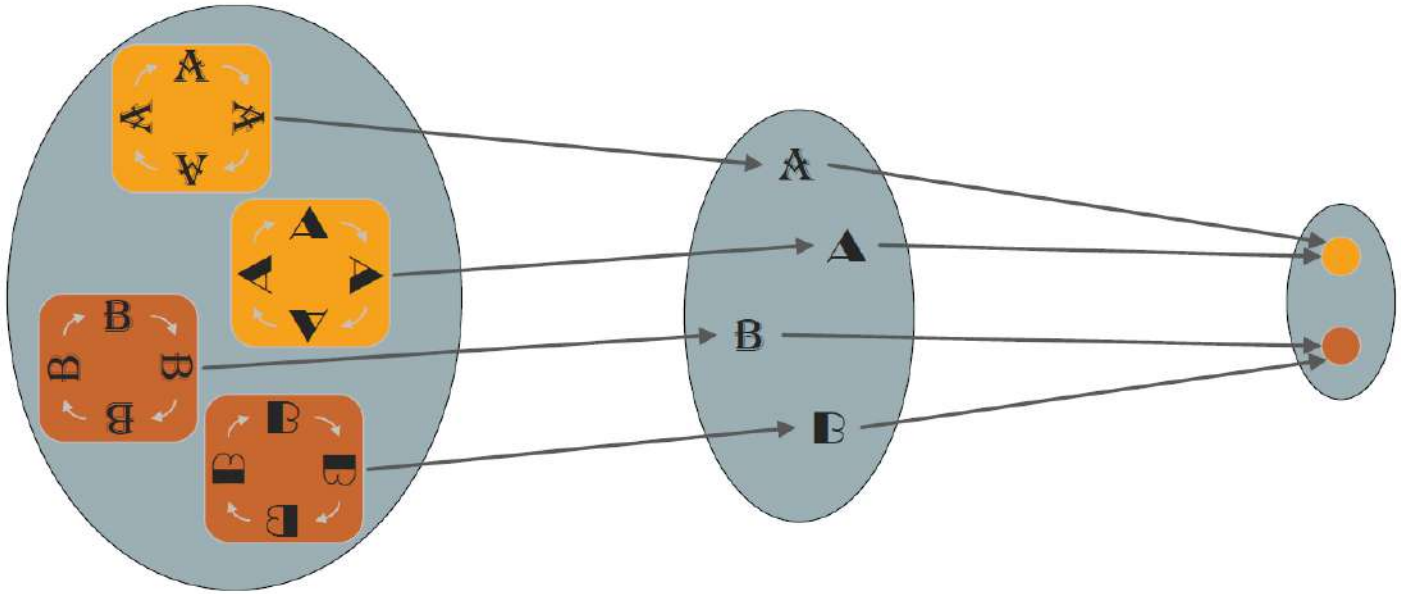
Invariant representations

Orbits & equivalence relations



$$O_x = \{gx \mid x \in X,\, g \in G\}$$

Invariant Representations



Invariant representations

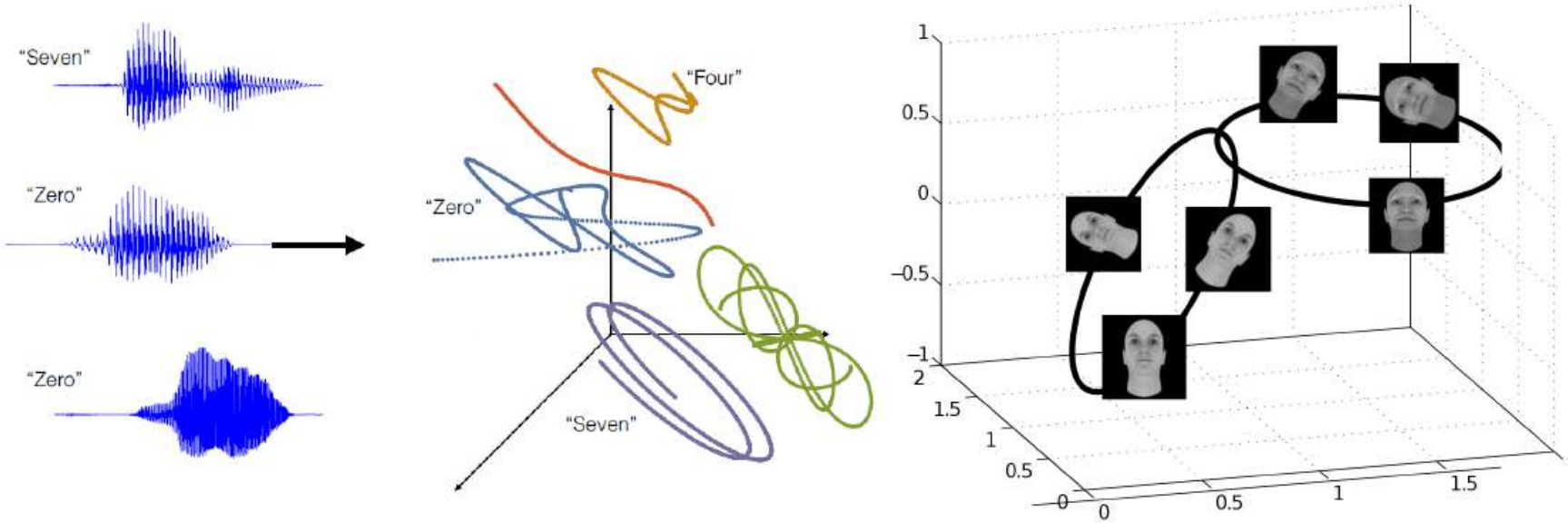
Orbit $O_s = \{gs \in \mathcal{S} | \, g \in G\} \in \mathcal{S}$

- Transformations as actions of a group element $g \in G$ on s

$$s' = gs(x) = s(g^{-1}x), x \in \mathbb{R}$$

- Equivalence relation, partition of $\mathcal{S}$

$$s \sim s' \iff \exists g \in G : s' = gs, \quad \forall s, s' \in \mathcal{S}$$



Invariant and selective representations

Invariance:

$$x' \sim x \Rightarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

Selectivity:

$$x' \sim x \Leftarrow \Phi(x') = \Phi(x), \quad \forall x, x' \in \mathcal{X}$$

How to define the equivalence classes $x' \sim x$?

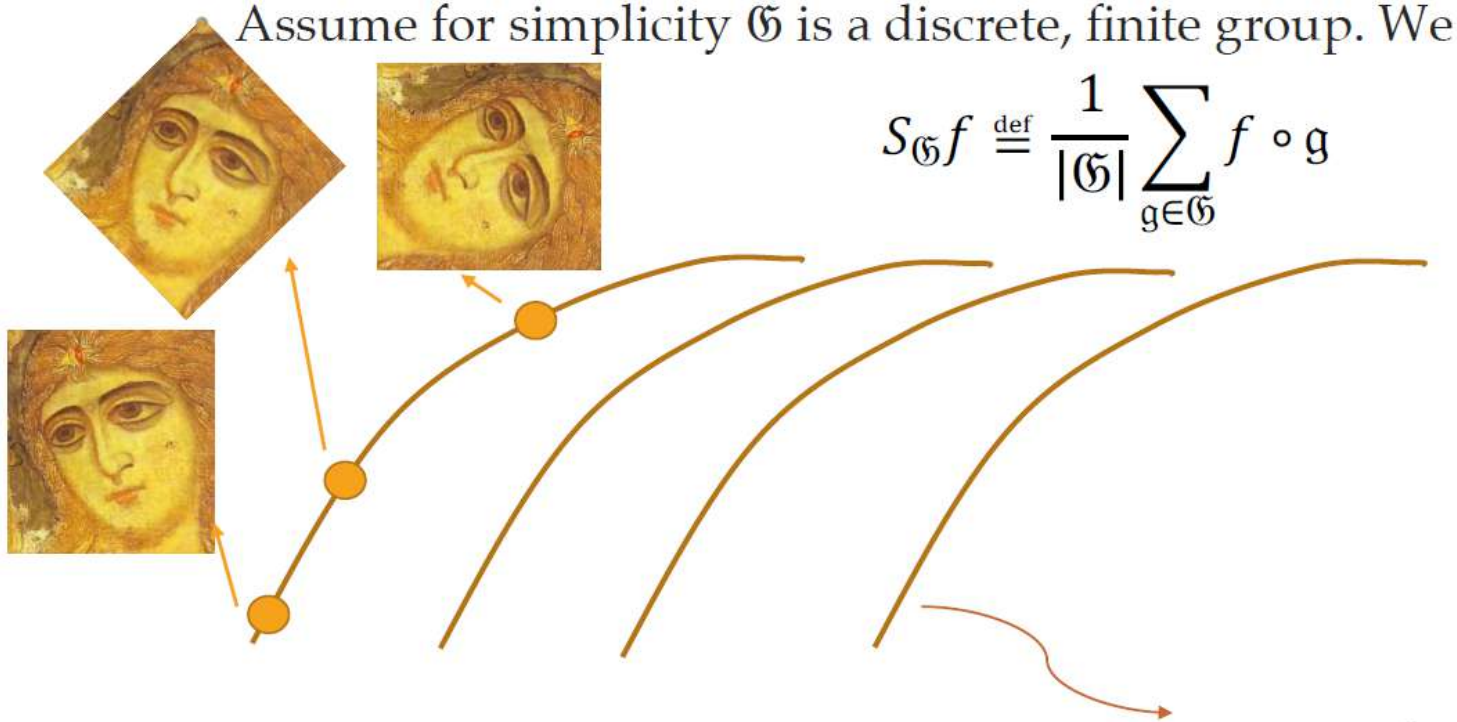
Equivalence under the action of a transformation $x' = gx, \quad g \in G$

Invariant Function Classes

- We may first consider an abstract (non-algorithmic) option, using $\mathfrak{G}$-*smoothing operators:*

Assume for simplicity $\mathfrak{G}$ is a discrete, finite group. We define

$$S_{\mathfrak{G}} f \stackrel{\text{def}}{=} \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f \circ \mathfrak{g}$$



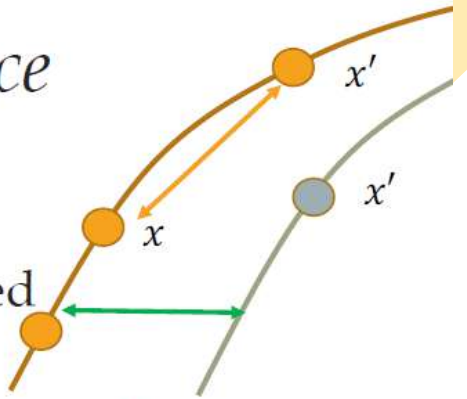
$$S_{\mathfrak{G}} f(x) = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f(\mathfrak{g}.x)$$

$S_{\mathfrak{G}}$ thus averages a function over group orbits.

Observe that $S_{\mathfrak{G}} f^* = f^*$.

Group Orbits $\mathfrak{G}.x = \{\mathfrak{g}.x; \mathfrak{g} \in \mathfrak{G}\}$

On the Sample Complexity of Learning under Invariance



- We can consider, as before, the Lipschitz class $\mathcal{F}$, and its $\mathfrak{G}$-smoothed version.

$$f \in \mathcal{F}: |f(x) - f(x')| \leq \beta \|x - x'\|; \;\; f \in S_{\mathfrak{G}}\mathcal{F}: |f(x) - f(x')| \leq \beta \, inf_{\mathfrak{g}} \|x - \mathfrak{g}.x'\|$$

- **Theorem [BVB'21]:** Using a $\mathfrak{G}$-*invariant kernel ridge regression, the generalization error of learning a Lipschitz,* $\mathfrak{G}$-*invariant function* f^* *satisfies*

$$\mathbb{E}\mathcal{R}(\tilde{f}) \lesssim (|\mathfrak{G}| \, n)^{\frac{-1}{d}}$$

  - Group size $|\mathfrak{G}|$ can be exponential in dimension (local translations, or $\mathcal{S}_d$).

[Mei, Misiakiewicz, Montanari, 21][Bietti, Venturi, **B**.'21][Bousquet & Von Luxburg'04]

Invariant kernels and Convolutional representations

Convolution layer representation (with average pooling),

$$\Phi(x) = \left(\sum_g s(\langle x, gt_1 \rangle), \ldots, \sum_g s(\langle x, gt_T \rangle) \right), \quad x \in \mathcal{X}.$$

The **associated kernel** can be written as

$$K(x, x) = \langle \Phi(x), \Phi(x') \rangle = \sum_t \left(\sum_g s(\langle x, gt_1 \rangle) \cdot \sum_g s(\langle x', gt_1 \rangle) \right)$$

Using linearity

$$K(x,x') = \sum_g \sum_{g'} \underbrace{\left(\sum_t s(\langle x, gt_1 \rangle) \cdot s(\langle x', g't_1 \rangle) \right)}_{K'(x,x')}.$$

Convolution kernels

$$K(x,x') = \sum_g \sum_{g'} K'(gx, g'x')$$

- Kernel of the above form are a special case of **convolution kernels**.
- If $g_1, \ldots, g_G$ form a **group** then the kernel is invariant $K(x,x') = K(x,gx')$, i.e. $\Phi(x) = \Phi(gx)$.

Compositionality prior

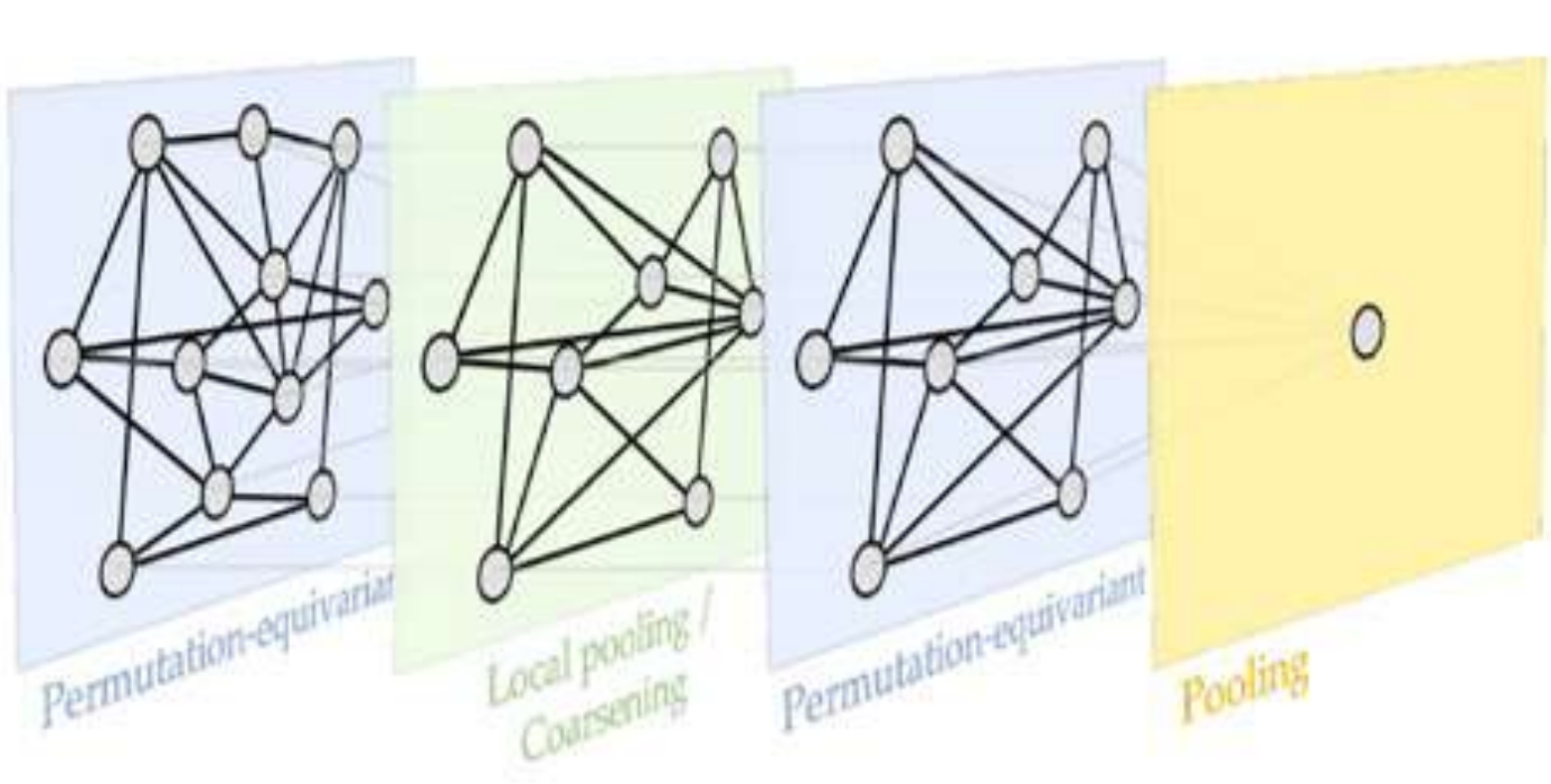
- What is the computational advantage?

How?

- This suggests a constructive approach to build rich invariants with multiscale structure, with building blocks:
  - *Linear $\mathfrak{G}$-equivariant layer* $B : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$, satisfying $B(\mathfrak{g}.x) = \mathfrak{g}.B(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$.
  - *Nonlinearity* $\sigma : \mathcal{C} \to \mathcal{C}'$ applied element-wise as $\big(\boldsymbol{\sigma}(x)\big)(u) = \sigma(x(u))$.
  - *Local pooling (coarsening)* $P : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C})$, such that $\Omega' \subseteq \Omega$.
  - *$\mathfrak{G}$-invariant layer (global pooling)* $A : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$, satisfying $A(\mathfrak{g}.x) = A(x)$for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in \mathcal{X}(\Omega, \mathcal{C})$.

- These blocks can be defined under very mild conditions on the geometric domain Ω.

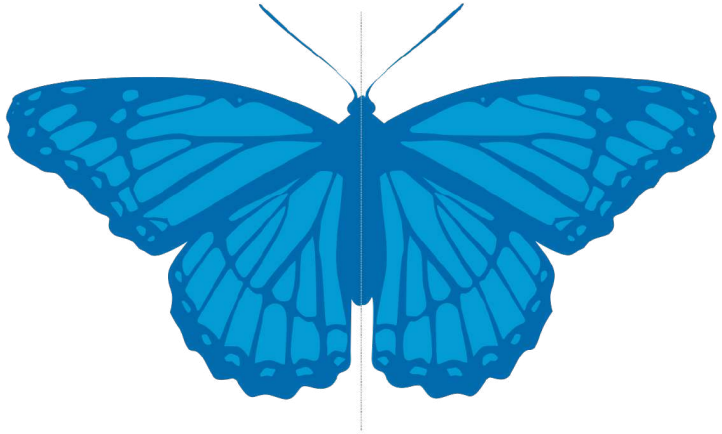
An example with graphs



Takeaway

- Geometric Priors enable a novel class of hypothesis spaces that together may tame the curse of dimensionality.
- Symmetries must be combined with scale separation.
- The language of group invariance and equivariance provides a principled architecture compatible with these priors, while preserving large approximation power.
- Theory side: what precise assumptions on non-linear coarsening operators?

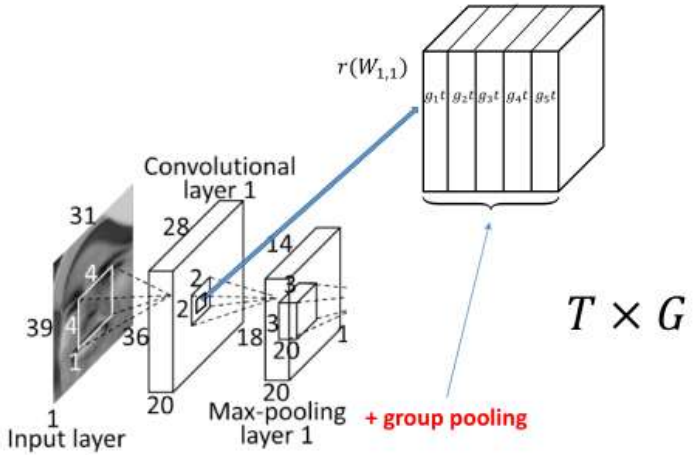
Class 1/2: Learning symmetries



What if the symmetry is not evident?

Rationale

Standard CNNs use translational convolution. A reason why they need such big training sets might be that the translational invariance is not the optimal choice to produce the best invariant representation. We want to learn the symmetry from the data.



Message: **the statistics of the data might be invariant to a broader set of symmetries then translations and to learn them will reduce the sample complexity.**

Let $\mathcal{G}$ a group. Let X set is a collection of orbits of stimuli:

$$O_t = \{t' \mid t' = gt,\ g \in \mathcal{G},\ t \in \mathcal{X}\}, \qquad X = (O_{t_1}, \cdots, O_{t_M})$$

How do we learn the group of symmetry?

Main observation to built the regularization term

If $W = \{g_1 t, \cdots, g_{|\mathcal{G}|} t\},\ t \in \mathbb{R}^d,\ g \in \mathcal{G},\ \mathbb{R}^{d \times d}$

$$(W^T W)_{i,j} = \langle w_i, w_j \rangle = \langle g_i t, g_j t \rangle = \langle t, g_i^{-1} g_j t \rangle$$

Let $\mathcal{G} = \{e, g\}$, $O_t = \{t, gt\}$:

$$W^T W = \begin{bmatrix} \langle t, e^{-1} e t \rangle & \langle t, e^{-1} g t \rangle \\ \langle t, g^{-1} e t \rangle & \langle t, g^{-1} g t \rangle \end{bmatrix} = \begin{bmatrix} \langle t, et \rangle & \langle t, gt \rangle \\ \langle t, gt \rangle & \langle t, et \rangle \end{bmatrix}$$

The Gramian's **columns are permutations of a single vector**.

1. The problem can be chosen to be low dimensional.
2. We map the problem from $\mathcal{G}$ to the permutation group.

Explanation: group multiplication tables and Cayley theorem

Let $\mathcal{G} = \{e, a, b, c, d, e, f\}$.

*	e	a	b	c	d	f	permutation
e	e	a	b	c	d	f	e
a	a	e	d	f	b	c	(12)(35)(46)
b	b	f	e	d	c	a	(13)(26)(45)
c	c	d	f	e	a	b	(14)(25)(36)
d	d	c	a	b	f	e	(156)(243)
f	f	b	c	a	e	d	(165)(234)

Theorem (Cayley)

Every group $\mathcal{G}$ is isomorphic to a subgroup of the symmetric group acting on $\mathcal{G}$.

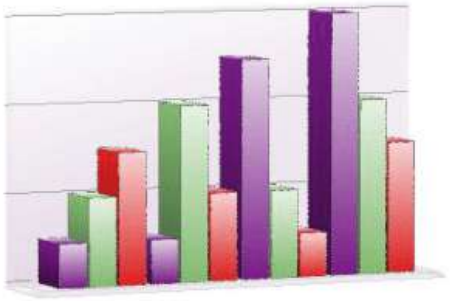
Separating permutation orbits

We need to test if the Gramian columns $G_{:,i}$ are permuted version of the same vector. A choice is to consider the distribution function of the vector values:

$$\mu_\lambda(G_{:,i}) = \sum_{k=1}^{|\mathcal{G}|} \delta(G_{k,i} - \lambda)$$

and impose equality of the distribution:

$$\|\mu(G_{:,i}) - \mu(G_{:,j})\|_2^2 = \sum_{k,l=1}^{|\mathcal{G}|} \int d\lambda \, (\mu_\lambda(G_{k,i}) - \mu_\lambda(G_{:,j}))^2 = 0$$



Separating permutation orbits

For all pairs of Gramian columns we apply:

$$\|\mu(G_{:,i}) - \mu(G_{:,j})\|_2^2 \ \forall\, i,j \ \Leftrightarrow \ \sum_{ikjl=1}^{|\mathcal{G}|} (1 - |\mathcal{G}|\delta_{ij})\delta(w_i^T w_k - w_j^T w_l) = 0$$

or if we approximate δ by its smooth form (letting $\sigma \to 0$)

$$\sum_{ikjl=1}^{|\mathcal{G}|} (1 - |\mathcal{G}|\delta_{ij}) e^{-\frac{(w_i^T w_k - w_j^T w_l)^2}{\sigma^2}} = 0.$$

More compact:

$$r(W) = \tau^T g_\sigma(C\text{vec}(W^T W)) = 0$$

Enforcing the Gramian constraint

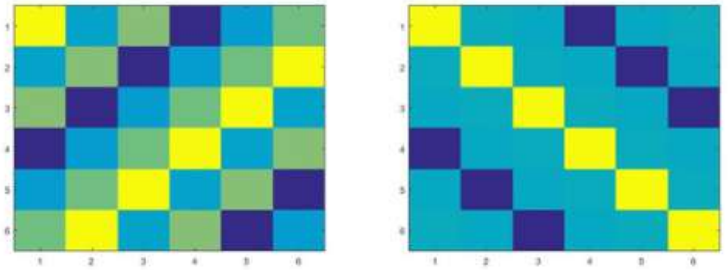
Theorem (Symmetry regularization)

Let matrix $W \in \mathbb{R}^{d \times |\mathcal{G}|}$ *and* $r : \mathbb{R}^{|\mathcal{G}| \times |\mathcal{G}|} \to \mathbb{R}_+$ *with*

$$r(W) = \tau^T g_\sigma(C\,\mathrm{vec}(W^T W))$$

C, τ *is a constant matrix and vector and function* g_σ *acts as the elementwise gaussian function. If* $r(W) = 0$*, then* $W^T W$ *is a permuted matrix and viceversa.*

W $\qquad$ $W^T W$



Alternative strategy. Google softmax

```
import sys
!git clone https://github.com/google-research/fast-soft-sort/
softsort_path = './fast-soft-sort'
sys.path.append(softsort_path)

from fast_soft_sort.pytorch_ops import soft_sort
```

Testing the algorithm

- Test set: randomly permuted orbits of vectors w.r.t. Cyclic group ($|\mathcal{G}| = 6$), Dihedral group ($|\mathcal{G}| = 12$), Pyritohedral group ($|\mathcal{G}| = 12$) in dimension 6.

X: Randomly permuted orbits dataset

- We calculate the intra and infra orbit elements distance with a ground truth orbit vs learned orbit. If the learned weights are an orbit of $\mathcal{G}$ then:
  1. $\|\Phi(x) - \Phi(x')\| = 0 \quad \Rightarrow \quad O_x = O_{x'}$
  2. $\|\Phi(x) - \Phi(x')\| \neq 0 \quad \Rightarrow \quad O_x \neq O_{x'}$

To test invariance and selectivity we plot the distances in a confusion matrix and distance distributions.

Recap

The following approximates the distribution of the values $\langle g_i I, t^k \rangle$ for one template t^k:

$$\mu_n^k(I) = \sum_i \left(\langle g_i I, t^k \rangle \right)^n$$

$\forall I, I'$ images we have:

Theorem 2 *The signature* $\mu(I) = (\mu_1^1(I), \cdots, \mu_N^K(I))$

- *is* **invariant** *i.e.* $\mu(g_i I) = \mu(I), \ \forall g_i \in G$
- *is* **selective (among classes)** *i.e.* $\mu(I) = \mu(I') \quad iff \quad I \sim I'$

If G is unitary, we have $\langle g_i I, t^k \rangle = \langle I, g_i^{-1} t^k \rangle \Rightarrow$ we have only one orbit of an arbitrary template to implement invariance of an image seen only once.

Framework for illumination transformations

∗ **Contrast Functions**: continuous, monotonic, and positive functions acting on an image.

We assume the contrast function $e : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^+$ has the following properties:

1. $e(x, 0) = 1$
2. $e(x, \zeta)e(x, \zeta') = e(x, \zeta \circ \zeta'), \quad \zeta, \zeta' \in \mathbb{R}_+$

where $\circ$ is a group composition. The transformation of an image I is:

$$I_\zeta(x) \equiv I(x, \zeta) = e(x, \zeta)I(x, 0).$$

and similarly for template t. We have the following signature for an image under change in illumination:

$$\mu_n^k(I) = \int_0^\infty \left(\langle I_0, t_\zeta^k \rangle \right)^n d\zeta$$

Protocol

* **Invariance**: We calculated the euclidean distance between images and their transformations. For invariance we tested:

$$||\mu(I_0) - \mu(I_\zeta)||^2 \sim 0$$

where we fixed 1 template, with $\mu(I) \in \mathbb{R}^{n \times k}$, and averaged the euclidean distance over images and their transformations.

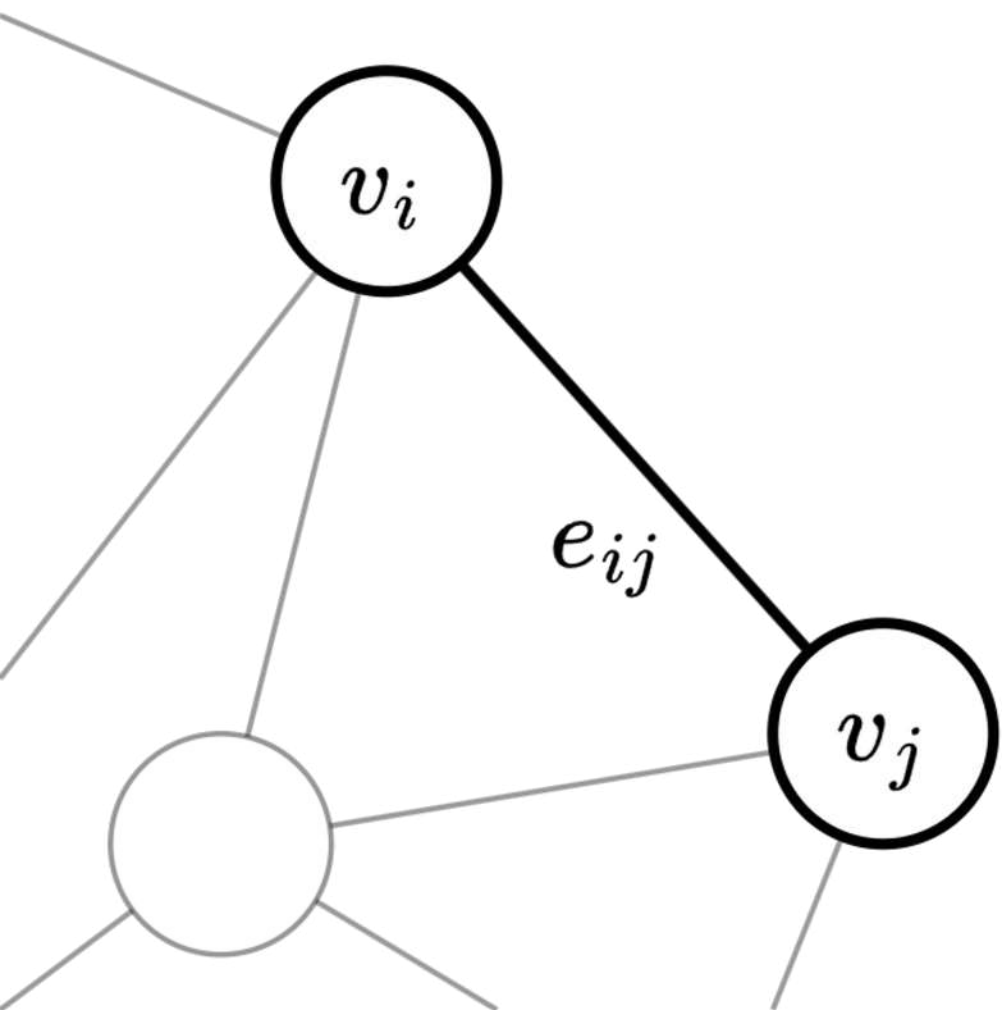
* **Selectivity**: We calculated the euclidean difference between different images. For selectivity we tested:

$$||\mu(I_0^i) - \mu(I_0^j)||^2 \nsim 0, \ i \neq j$$

* **Control**: We defined a "fake orbit" as a set of randomly selected images from the datasets.

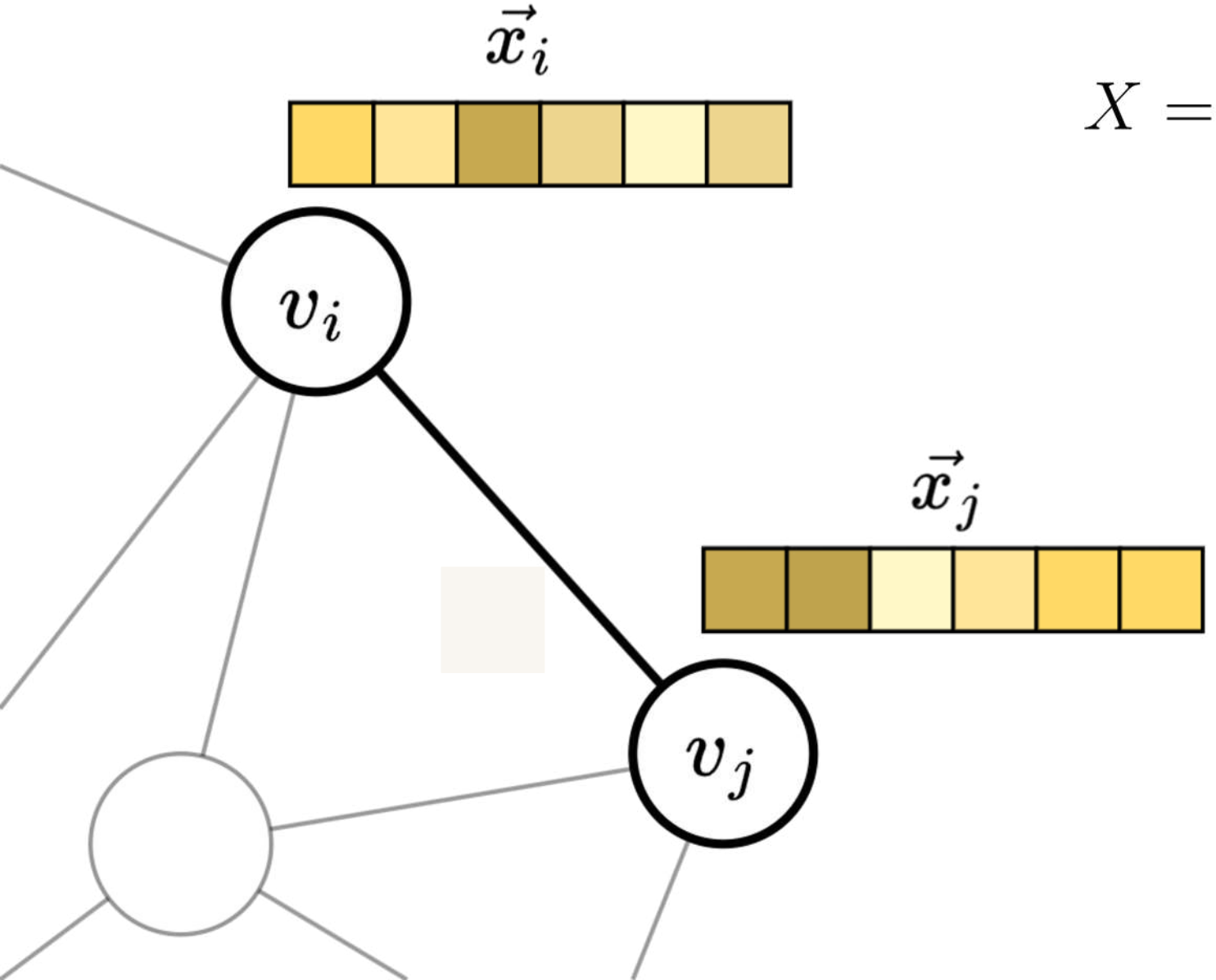
For each of the datasets, we let I_0 be a face under one illumination condition and I_ζ be all other illumination conditions of the face. Our set of templates $\{t_\zeta^k\}$, contained faces under all illumination conditions with $t^k \neq I$.

$$\mathcal{G} = (V, E)$$



Adjacency matrix of $\mathcal{G}$

$$A_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin E \\ 1 & \text{if } (i,j) \in E \end{cases}$$



$$X = \begin{pmatrix} \vec{x_1} \\ \vdots \\ \vec{x_i} \\ \vdots \\ \vec{x_n} \end{pmatrix}$$

node features of $\mathcal{G}$

for example:

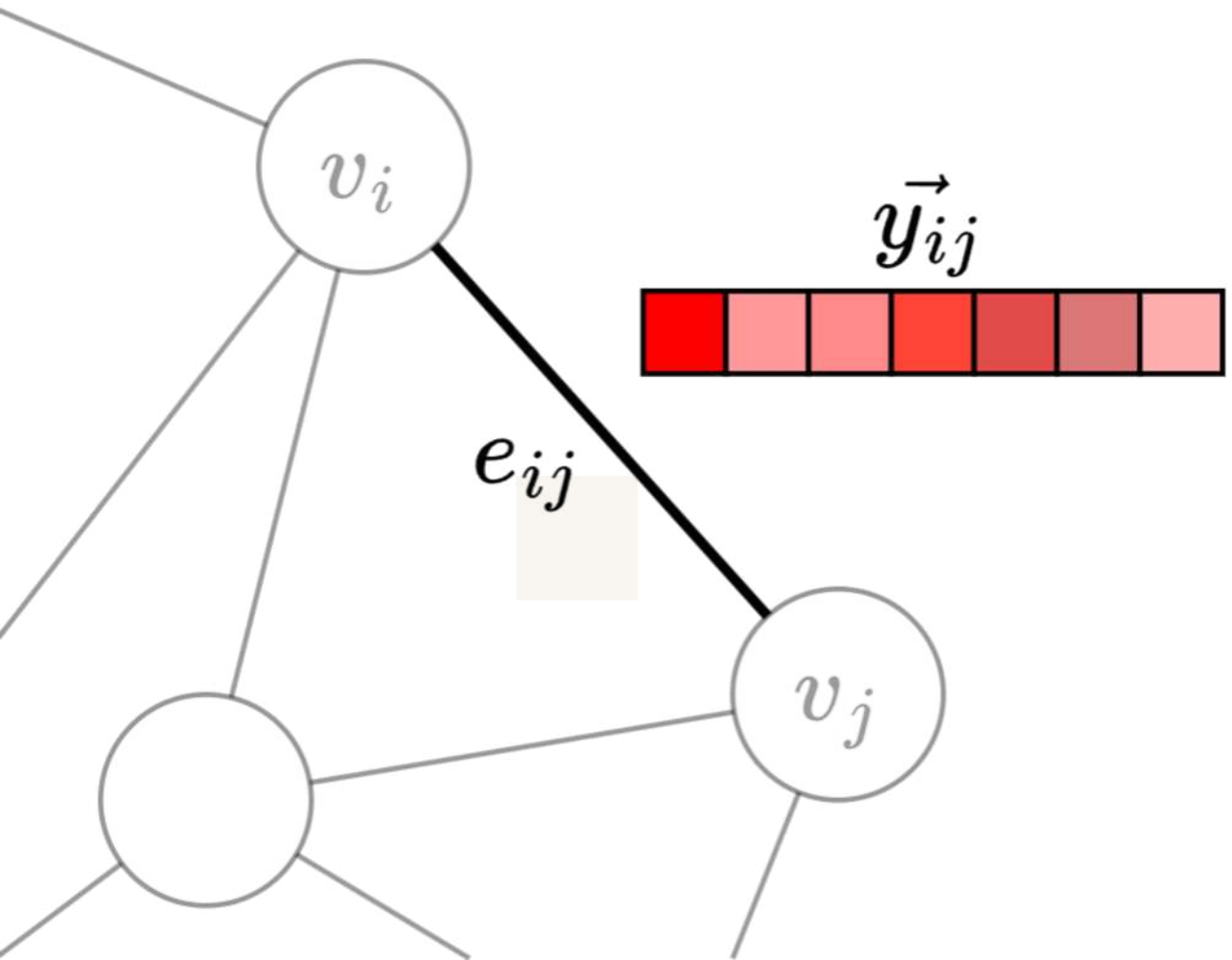
- properties of the atom
- data of the account
- number/type of crossroads

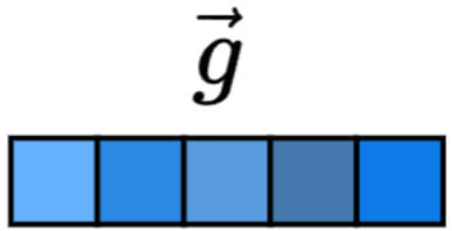
$$Y_{ij} = \vec{y_{ij}}$$

edge features of $\mathcal{G}$

for example:

- type of bond
- date of friend request, relationship
- average traffic, number of car parks





global features of $\mathcal{G}$

for example:

- known properties
- year
- day of the week

Tasks on graphs

Global tasks

tasks on the **whole graph**.
Both classification and regression.

Examples:

- predicting properties of a molecule
- predicting the hobby shared by all the people
- predicting the likelihood of a car accident

Tasks on graphs

Node tasks

tasks at the level of the **single node**.
Both classification and regression.

Examples:

- predicting if an atom is part of an aromatic ring
- predicting political stance of each users on a topic
- classifying intersections based on predicted traffic

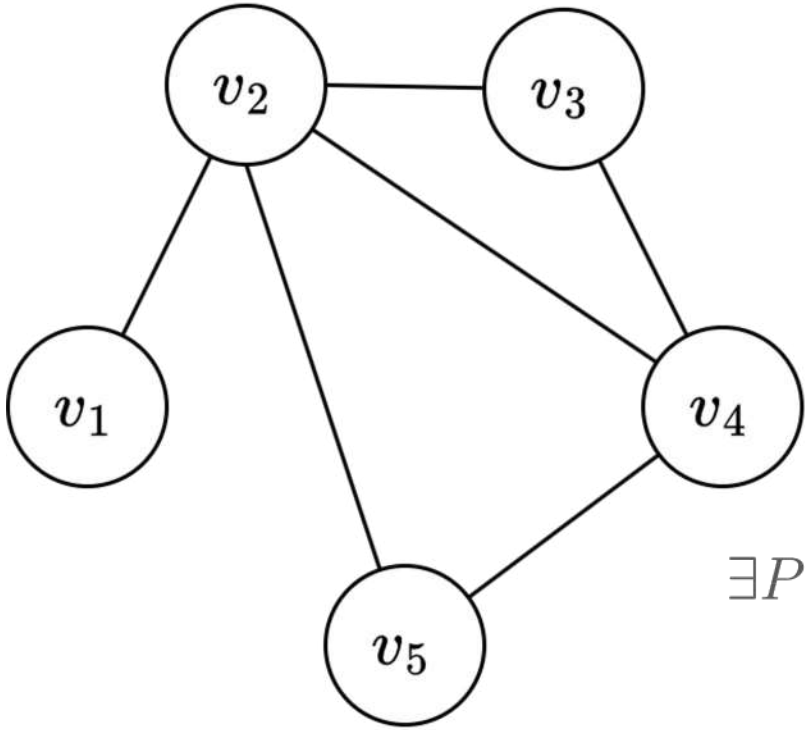
Tasks on graphs

tasks at the level of the **single edge.**
Both classification and regression.

Examples:

- predicting which bond is easier to break
- predicting how likely two users will become friends
- predicting how busy a road will be in an hour

Edge tasks

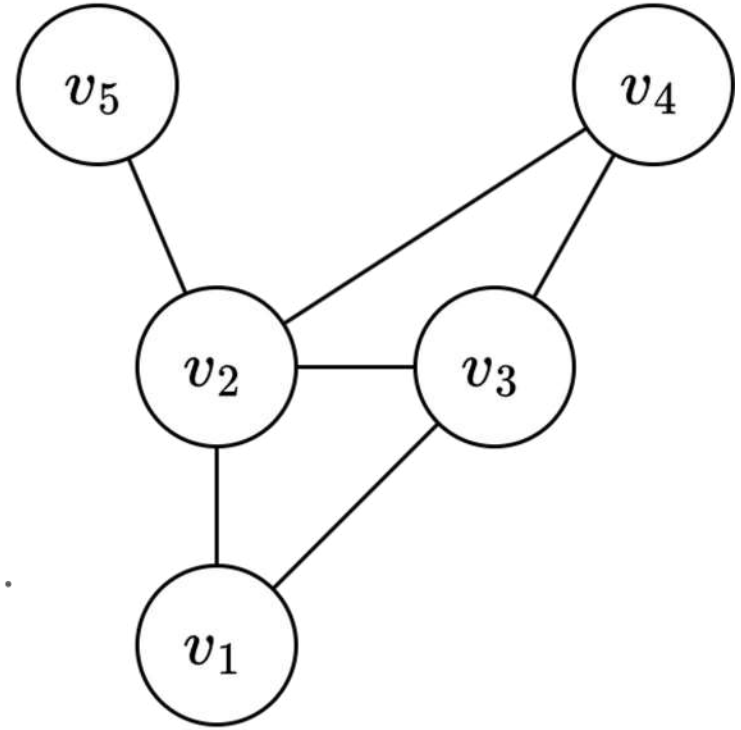


$\cong$

$\iff$

$\exists P$ permutation matrix s.t.

$$PA_1P^{-1} = A_2$$



$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$\forall P$ permutation matrix

invariance in the context of global tasks

$$F_G(A, X, Y, \vec{g}) = F_G(PAP^{-1}, PX, PYP^{-1}, \vec{g}) \quad \in \mathbb{R} \text{ or } \in \mathbb{R}^d$$

equivariance in the context of node or edge tasks

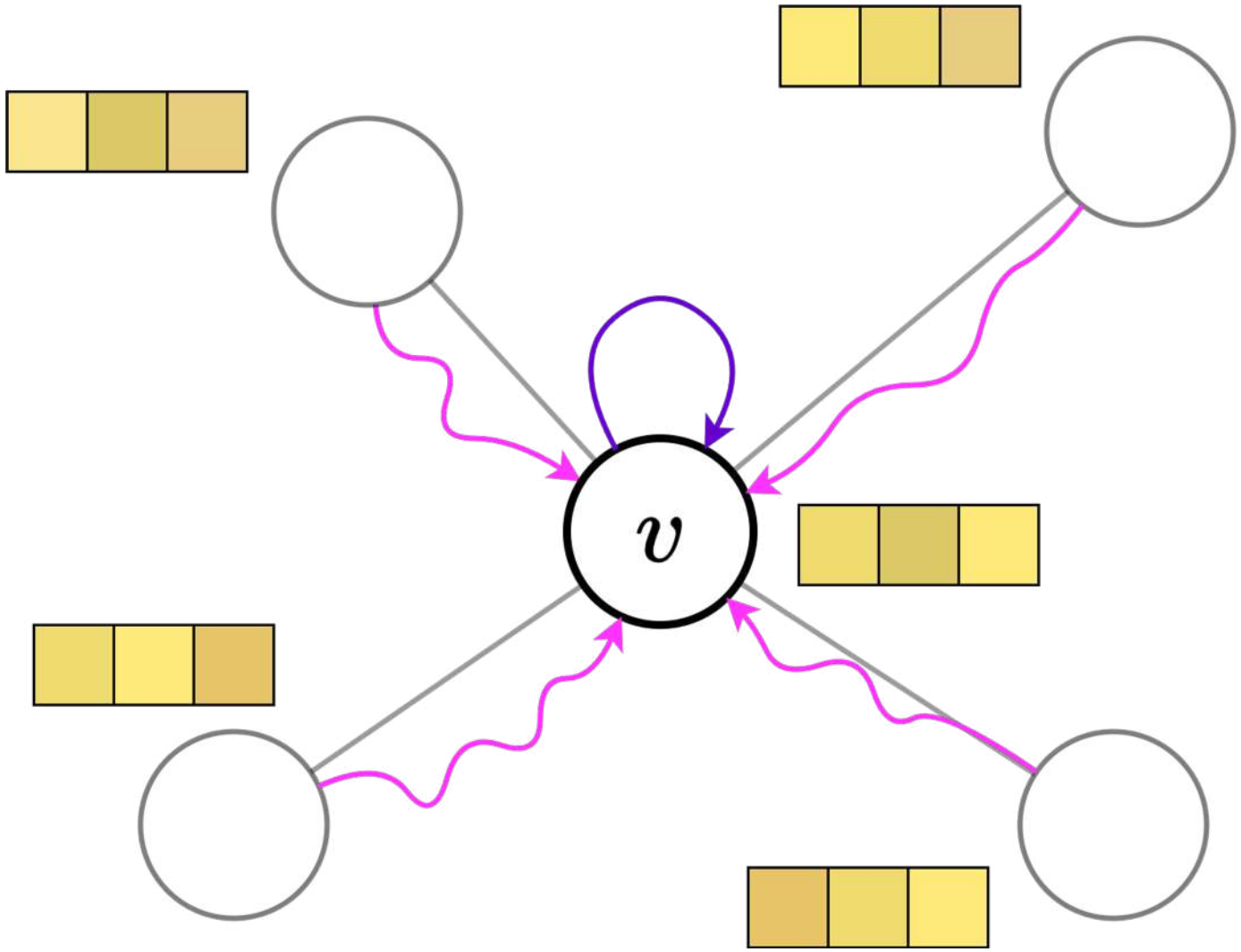
$$F_V(A, X, Y, \vec{g}) = P^{-1} F_V(PAP^{-1}, PX, PYP^{-1}, \vec{g})$$
$$\in \mathbb{R}^n \text{ or } \in \mathbb{R}^{n \cdot d}$$

$$F_E(A, X, Y, \vec{g}) = P^{-1} F_E(PAP^{-1}, PX, PYP^{-1}, \vec{g}) P$$
$$\in \mathcal{M}_{\mathbb{R}}(n, n) \text{ or } \in \mathcal{M}_{\mathbb{R}^d}(n, n)$$

in the context of node/global tasks

input: A , X

- **Aggregating** over $\mathcal{N}_v$
- **Updating** $\vec{x_v}$



Convolutional approach

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} c_{vu}\psi(\vec{x_u})\right)$$

Aggregation

must be a **permutation invariant** operator

- mean
- max
- sum

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} c_{vu}\psi(\vec{x_u})\right)$$

Update

$$\phi(\vec{x}, \vec{z}) = \sigma(W\vec{x} + U\vec{z} + \vec{b})$$

learnable

coefficient

in the case of summation can be

$$c_{uv} = \frac{1}{\sqrt{deg(u) \cdot deg(v)}}$$

transformation of the features

$$\psi(\vec{x}) = W\vec{x} + \vec{b}$$

learnable

Attentional approach

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} a(\vec{x_v}, \vec{x_u})\psi(\vec{x_u})\right)$$

coefficients are **learned**
via NN and softmax
normalized

$$a(\vec{x_v}, \vec{x_u}) \in \mathbb{R}$$

Message-passing approach

$$\vec{x_v} = \phi\left(\vec{x_v}, \bigoplus_{u \in \mathcal{N}_v} \psi(\vec{x_v}, \vec{x_u})\right)$$

ψ here is a **learnable message function**

$\psi(\vec{x_v}, \vec{x_u}) \in \mathbb{R}^m$

1-neuron ReLu implicit bias

Notations. We use bold-faced letters to denote vectors, e.g., $\mathbf{x} = (x_1, \ldots, x_d)$. For $\mathbf{x} \in \mathbb{R}^d$ we denote by $\|\mathbf{x}\|$ the Euclidean norm.

Single-neuron networks. Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be a training dataset, where for every i we have $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We consider empirical-loss minimization of a single-neuron network, with respect to the square loss. Thus, the objective is given by

$$\mathcal{L}(\mathbf{w}) := \frac{1}{2} \sum_{i=1}^{n} (\sigma(\langle \mathbf{x}_i, \mathbf{w} \rangle) - y_i)^2 \ , \tag{1}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function. Let $X \in \mathbb{R}^{n \times d}$ denote the data matrix, i.e., the rows of X are $\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top$, and let $\mathbf{y} = (y_1, \ldots, y_n)$. We have

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2} \|\sigma(X\mathbf{w}) - \mathbf{y}\|^2 \ ,$$

where σ is applied component-wise. We assume that the data is realizable, that is, $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = 0$. Moreover, we focus on settings where the network is *overparameterized*, in the sense that $\mathcal{L}$ has multiple (or even infinitely many) global minima.

We analyze the implicit regularization of gradient flow on the objective given by Eq. 1. This setting captures the behaviour of gradient descent with infinitesimally small step size. Let $\mathbf{w}(t)$ be the trajectory of gradient flow, where $\mathbf{w}(0)$ is the initial point. The dynamics of $\mathbf{w}(t)$ is given by the differential equation

$$\dot{\mathbf{w}}(t) := \frac{d\mathbf{w}(t)}{dt} = -\nabla \mathcal{L}(\mathbf{w}(t)) \ .$$

If $\lim_{t \to \infty} \mathbf{w}(t)$ exists then we denote it by $\mathbf{w}(\infty)$. We note that gradient flow is not guaranteed to converge to a global minimum (cf. Yehudai and Shamir [2020]). In the overparameterized setting there can be infinitely many global minima, and we study to which one gradient flow converges, assuming that it converges to a global minimum.

The implicit regularization of a ReLU neuron is approximately the ℓ_2 norm

While the implicit regularization of ReLU neuron cannot be expressed as a function $\mathcal{R}(\mathbf{w})$, in this section we show that it can be expressed approximately, within a factor of 2, by the ℓ_2 norm. This implies that even without early stopping, if the data can be labeled by a ReLU neuron with small ℓ_2 norm, then gradient flow will converge to a ReLU neuron whose ℓ_2 norm is not much larger. Since a ReLU neuron is just a linear function composed with a fixed nonlinearity, this can be used to derive good statistical generalization guarantees, via standard techniques (cf. Shalev-Shwartz and Ben-David [2014]).

Theorem 5.1. *Consider gradient flow on the objective given by Eq. 1, where $\sigma : \mathbb{R} \to \mathbb{R}$ is a monotonically non-decreasing activation function (e.g., ReLU). Assume that $\mathbf{w}(\infty)$ exists and $\mathcal{L}(\mathbf{w}(\infty)) = 0$. Let $\mathbf{w}^* \in \operatorname{argmin}_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}(0)\|$ s.t. $\mathcal{L}(\mathbf{w}) = 0$. Then, $\|\mathbf{w}(\infty) - \mathbf{w}(0)\| \leq 2 \cdot \|\mathbf{w}^* - \mathbf{w}(0)\|$.*

Proof. First, note that

$$\begin{aligned}\langle \nabla \mathcal{L}(\mathbf{w}(t)), \mathbf{w}(t) - \mathbf{w}^* \rangle &= \left\langle \sum_{i=1}^{n} \left(\sigma(\mathbf{x}_i^\top \mathbf{w}(t)) - \sigma(\mathbf{x}_i^\top \mathbf{w}^*)\right) \sigma'(\mathbf{x}_i^\top \mathbf{w}(t)) \mathbf{x}_i, \mathbf{w}(t) - \mathbf{w}^* \right\rangle \\ &= \sum_{i=1}^{n} \left(\sigma(\mathbf{x}_i^\top \mathbf{w}(t)) - \sigma(\mathbf{x}_i^\top \mathbf{w}^*)\right) \sigma'(\mathbf{x}_i^\top \mathbf{w}(t)) (\mathbf{x}_i^\top \mathbf{w}(t) - \mathbf{x}_i^\top \mathbf{w}^*) \,.\end{aligned}$$

Since σ is monotonically non-decreasing then

$$\left(\sigma(\mathbf{x}_i^\top \mathbf{w}(t)) - \sigma(\mathbf{x}_i^\top \mathbf{w}^*)\right) (\mathbf{x}_i^\top \mathbf{w}(t) - \mathbf{x}_i^\top \mathbf{w}^*) \geq 0 \,,$$

and $\sigma'(\mathbf{x}_i^\top \mathbf{w}(t)) \geq 0$. Hence,

$$\langle \nabla \mathcal{L}(\mathbf{w}(t)), \mathbf{w}(t) - \mathbf{w}^* \rangle \geq 0 \,.$$

Therefore, we have

$$\frac{d}{dt}\left(\frac{1}{2}\|\mathbf{w}(t) - \mathbf{w}^*\|^2\right) = \langle \mathbf{w}(t) - \mathbf{w}^*, \dot{\mathbf{w}}(t) \rangle = \langle \mathbf{w}(t) - \mathbf{w}^*, -\nabla \mathcal{L}(\mathbf{w}(t)) \rangle \leq 0 \,.$$

Thus, $\|\mathbf{w}(\infty) - \mathbf{w}^*\| \leq \|\mathbf{w}(0) - \mathbf{w}^*\|$.

Hence,

$$\begin{aligned} \|\mathbf{w}(\infty) - \mathbf{w}(0)\| &= \|\mathbf{w}(\infty) - \mathbf{w}^* + \mathbf{w}^* - \mathbf{w}(0)\| \\ &\leq \|\mathbf{w}(\infty) - \mathbf{w}^*\| + \|\mathbf{w}^* - \mathbf{w}(0)\| \\ &= 2 \cdot \|\mathbf{w}^* - \mathbf{w}(0)\| \,. \end{aligned}$$

□

Remark 5.1. *Note that Theorem 5.1 implies that if* $\mathbf{w}(0) = \mathbf{0}$ *then* $\|\mathbf{w}(\infty)\| \leq 2 \cdot \|\mathbf{w}^*\|$. *By Theorem 4.2, the implicit regularization cannot be expressed as a function of* $\mathbf{w}$. *Hence, while the implicit regularization of a ReLU neuron cannot be expressed exactly as a function of* $\mathbf{w}$, *it can be expressed approximately, within a factor of* 2, *by the* ℓ_2 *norm.*

Non-linearity choice and implicit bias

For the ReLu expansion of a function $f : \mathbb{R} \to \mathbb{R}$ we have:

$$f(x) = \int d\beta \; c(\beta) ReLu(x - \beta) \tag{6}$$

with

$$f'(x) = \int d\beta \; c(\beta) Step(x - \beta) \tag{7}$$

Taking into account of saturation we can use instead of a ReLu the following:

$$SReLu(x) = \frac{1}{w}\Big(ReLu(x - \beta) - ReLu(x - (\beta + w))\Big) \tag{8}$$

where the factor $1/w$ is a normalization factor. Indeed

$$f'(x) = \int d\beta \; d(\beta) \frac{1}{w} Box_w(x - \beta) = \int d\beta \; d(\beta) \bar{Box}_w(x - \beta). \tag{9}$$

where $(1/w)Box = \bar{Box}$ is the Box function at $\beta + w$ with integral normalized to unit.
How can we pass from one representation to the other? Note that:

$$Box_{\beta,w}(x) = Step_\beta(x) - Step_{\beta+w}(x) = (Id - T_w)Step_\beta(x) \equiv \Delta_w Step_\beta(x) \tag{10}$$

where $T : \mathbb{R} \to \mathbb{R}$, $T_w f(x) = f(x + w)$ is a translation operator.
Thus we have

$$d(\beta) = \Delta_w^{-1} c(\beta) \tag{11}$$

where the operator can be explicitly written as a sum of translation operators since:

$$\Delta_w^{-1} = (Id - T_w)^{-1} = \sum_{n \in \mathbb{N}} T_w^n. \tag{12}$$

Indeed:

$$Step_\beta(x) = \Delta_w^{-1} Box_\beta(x) = \sum_{n \in \mathbb{N}} T_w^n Box_{\beta,w}(x). \tag{13}$$

Explicitly:

$$d(\beta) = \sum_{n \in \mathbb{N}} c(\beta - nw) = \sum_{n \in \mathbb{N}} \frac{1}{w} f''(\beta - nw). \tag{14}$$

Check for constants.

Non-linearity choice and implicit bias

Supposing an $N = ceil(x_{max}/w)$ maximum (finite range of neural activations):

$$\left\|f''\right\|_2^2 = \int d\beta\, c^2(\beta) = \int d\beta \left|\sum_{n=0}^{N-1} f''(\beta - nw)\right|^2 = \int dk \left|\mathcal{F}(\sum_{n=0}^{N-1} \frac{1}{w} T_{nw} \partial_\beta^2 f(\beta))\right|^2$$

$$= \int dk\, k^4 \left|\sum_{n=0}^{N-1} \frac{e^{inwk}}{w}\right|^2 \tilde{f}^2(k).$$

Using

$$\left|\sum_{n=0}^{N-1} \frac{e^{inwk}}{w}\right|^2 = \frac{\sin^2(\frac{1}{2}Nwk)}{w^2 \sin^2(\frac{1}{2}wk)}$$

we finally have:

$$\boxed{\left\|f''\right\|_2^2 = \int dk\, \frac{k^4}{w^2} \frac{\sin^2(\frac{1}{2}Nwk)}{\sin^2(\frac{1}{2}wk)} \tilde{f}^2(k)}$$

To compare in the ReLu case a similar calculation would lead to:

$$\left\|f''\right\|_2^2 = \int dk\, k^4\, \tilde{f}^2(k).$$

Suppose $f(t) = \cos(w_1 t) + \cos(w_2 t)$ is a sound wave and let $\sigma(t) = t^2$. Then

$$(\sigma \circ f)(t) = \frac{1}{2}[2 + \cos(2w_1 t) + \cos(2w_2 t) + 2\cos((w_1 + w_2)t) + 2\cos((w_1 - w_2)t)].$$

$$FT\phi(x) = F \sum_n a_n \underbrace{x \odot \cdots \odot x}_{\text{n-times}} = \sum_n a_n \underbrace{\hat{x} * \cdots * \hat{x}}_{\text{n-times}},$$

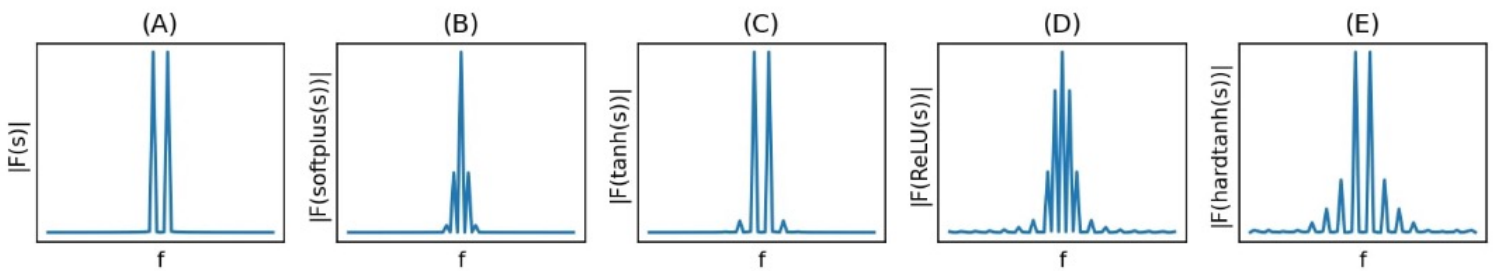
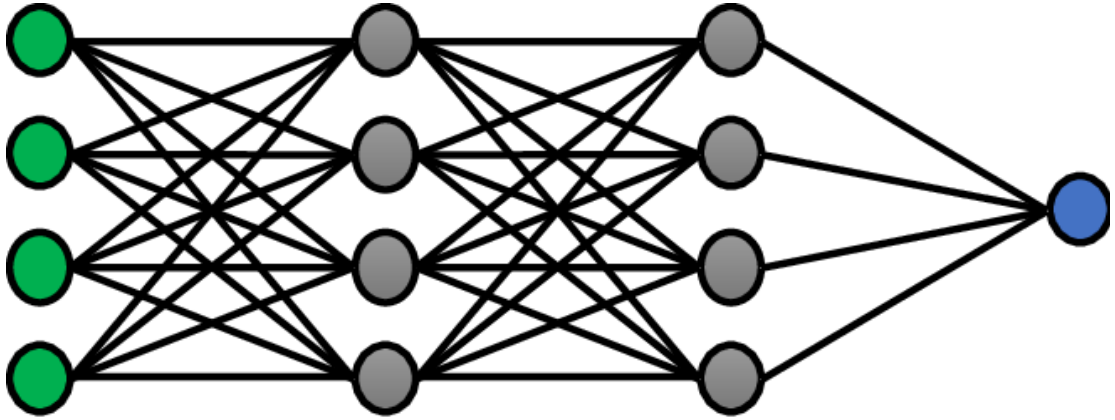


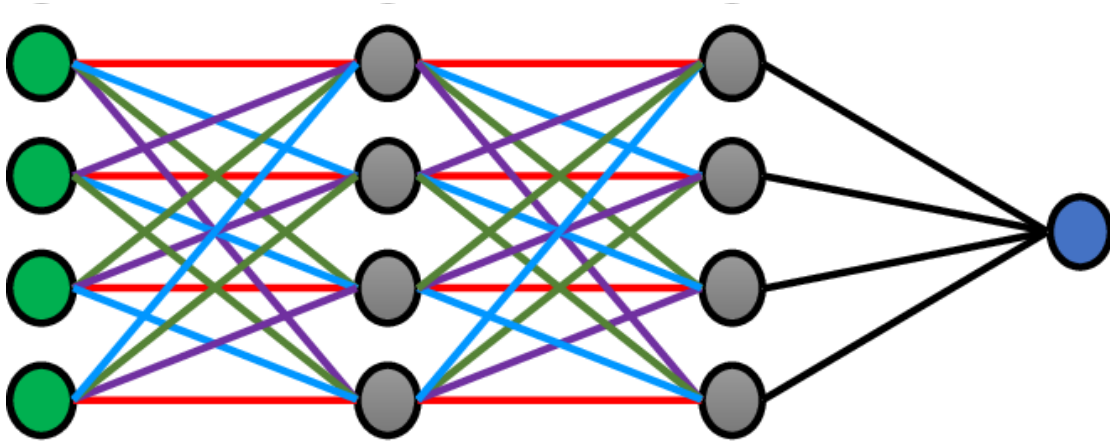
Figure 1. Non-linear distortions in the frequency domain due to the application of (B) softplus, (C) tanh, (D) ReLU and (E) hardtanh non-linear activations on $s(\cdot) = \sin(\cdot)$ (A).

Implicit bias of convolutional and fully connected networks



(a) Fully connected network of depth L

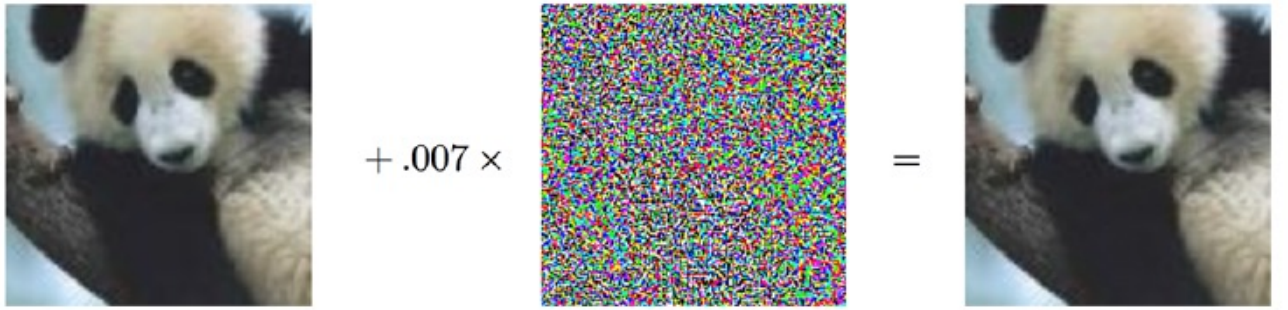
$\overline{\boldsymbol{\beta}}^{\infty} \propto \underset{\forall n,\, y_n \langle \mathbf{x}_n, \boldsymbol{\beta} \rangle \geq 1}{\operatorname{argmin}} \|\boldsymbol{\beta}\|_2$ (independent of L)



(b) Convolutional network of depth L

$\overline{\boldsymbol{\beta}}^{\infty} \propto$ first order stationary point of $\underset{\forall n,\, y_n \langle \mathbf{x}_n, \boldsymbol{\beta} \rangle \geq 1}{\operatorname{argmin}} \|\widehat{\boldsymbol{\beta}}\|_{2/L}$

Adversarial attacks



Definition 1 (Adversarial Attack)**.** *Let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be a data point belong to class $\mathcal{C}_i$. Define a target class $\mathcal{C}_t$. An **adversarial attack** is a mapping $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^d$ such that the perturbed data*

$$\boldsymbol{x} = \mathcal{A}(\boldsymbol{x}_0)$$

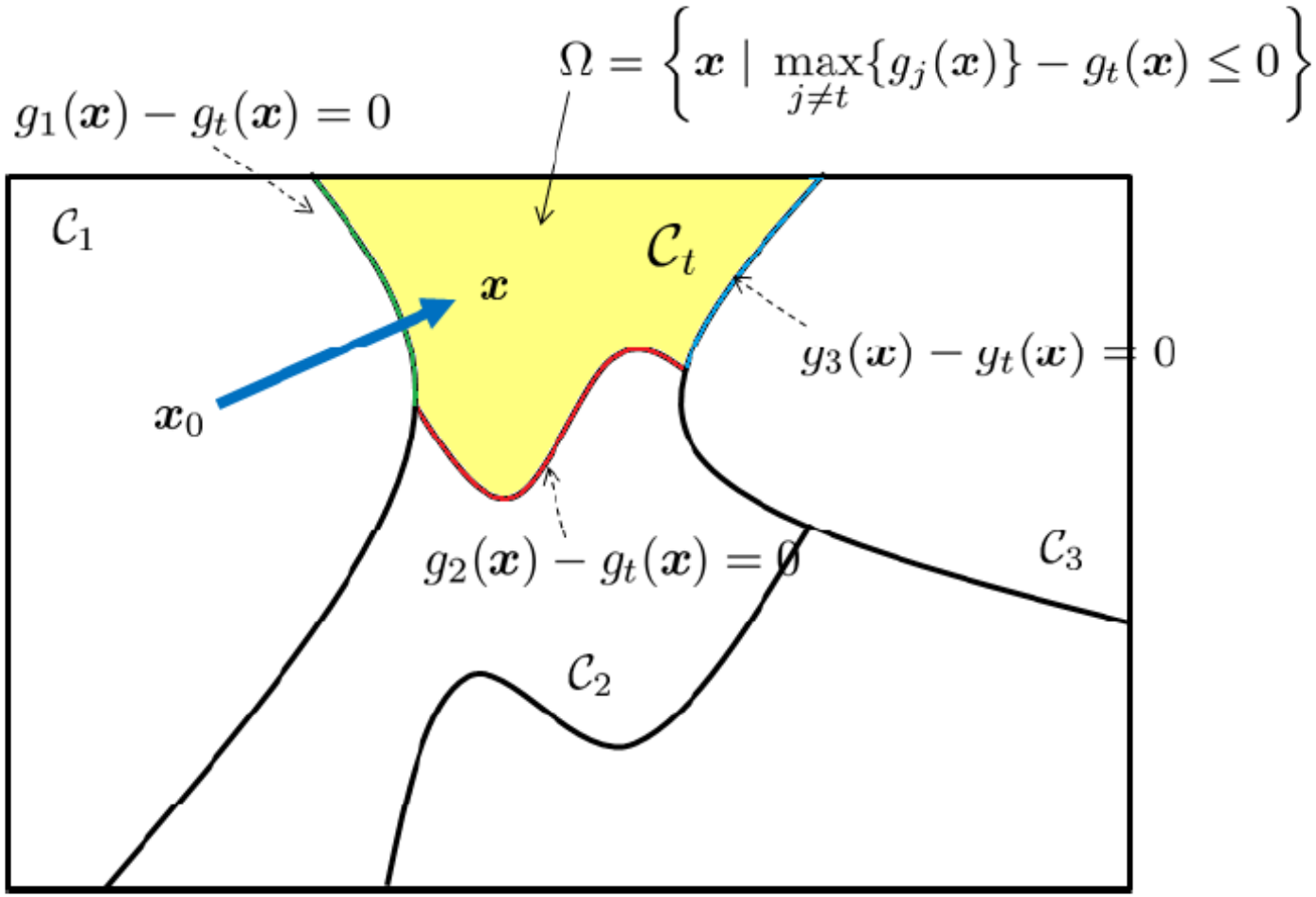
is misclassified as $\mathcal{C}_t$.

Definition 2 (Additive Adversarial Attack)**.** *Let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be a data point belong to class $\mathcal{C}_i$. Define a target class $\mathcal{C}_t$. An **additive** adversarial attack is an addition of a perturbation $\boldsymbol{r} \in \mathbb{R}^d$ such that the perturbed data*

$$\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{r}$$

is misclassified as $\mathcal{C}_t$.

Geometry



$$\Omega = \left\{ \boldsymbol{x} \;\middle|\; \begin{array}{ll} g_1(\boldsymbol{x}) - g_t(\boldsymbol{x}) & \leq 0 \\ g_2(\boldsymbol{x}) - g_t(\boldsymbol{x}) & \leq 0 \\ \vdots & \\ g_k(\boldsymbol{x}) - g_t(\boldsymbol{x}) & \leq 0 \end{array} \right\}$$

Attacks

Definition 3 (Minimum Norm Attack). *The* ***minimum norm attack*** *finds a perturbed data* $\boldsymbol{x}$ *by solving the optimization*

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{x} - \boldsymbol{x}_0\| \\ \text{subject to} \quad & \max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0, \end{aligned} \tag{3.3}$$

where $\|\cdot\|$ *can be any norm specified by the user.*

Definition 4 (Maximum Allowable Attack). *The* ***maximum allowable attack*** *finds a perturbed data* $\boldsymbol{x}$ *by solving the optimization*

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \\ \text{subject to} \quad & \|\boldsymbol{x} - \boldsymbol{x}_0\| \leq \eta, \end{aligned} \tag{3.4}$$

where $\|\cdot\|$ *can be any norm specified by the user, and* $\eta > 0$ *denotes the magnitude of the attack.*

Definition 5 (Regularization-based Attack). *The* ***regularization-based attack*** *finds a perturbed data* $\boldsymbol{x}$ *by solving the optimization*

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\| + \lambda \left(\max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right) \tag{3.5}$$

where $\|\cdot\|$ *can be any norm specified by the user, and* $\lambda > 0$ *is a regularization parameter.*

Linear attacks

To gain insights about these constraints, we consider a k-class linear classifier. In this case, each discriminant function takes the form

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i,0}. \tag{3.9}$$

The decision boundary between the i-th class and the t-th class is therefore

$$g(\boldsymbol{x}) = (\boldsymbol{w}_i - \boldsymbol{w}_t)^T \boldsymbol{x} + w_{i,0} - w_{t,0} = 0. \tag{3.10}$$

Since we want the perturbed data $\boldsymbol{x}$ to live in $\mathcal{C}_t$, the constraint set becomes

$$\begin{bmatrix} \boldsymbol{w}_1^T - \boldsymbol{w}_t^T \\ \vdots \\ \boldsymbol{w}_{t-1}^T - \boldsymbol{w}_t^T \\ \boldsymbol{w}_{t+1}^T - \boldsymbol{w}_t^T \\ \vdots \\ \boldsymbol{w}_k^T - \boldsymbol{w}_t^T \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} w_{1,0} - w_{t,0} \\ \vdots \\ w_{t-1,0} - w_{t,0} \\ w_{t+1,0} - w_{t,0} \\ \vdots \\ w_{k,0} - w_{t,0} \end{bmatrix} \leq \boldsymbol{0} \quad \Leftrightarrow \quad \boldsymbol{A}^T \boldsymbol{x} \leq \boldsymbol{b} \tag{3.11}$$

where $\boldsymbol{A} = [\boldsymbol{w}_1 - \boldsymbol{w}_t, \ldots, \boldsymbol{w}_k - \boldsymbol{w}_t] \in \mathbb{R}^{d \times (k-1)}$, and $\boldsymbol{b} = [w_{t,0} - w_{1,0}, \ldots, w_{t,0} - w_{k,0}]^T$. This is summarized in the following Lemma.

Lemma 1 (Constraint Set of Linear Classifier). *Consider a k-class linear classifier with discriminant function $g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$ for $i = 1, \ldots, k$. Define*

$$\boldsymbol{A} = [\boldsymbol{w}_1 - \boldsymbol{w}_t, \ldots, \boldsymbol{w}_k - \boldsymbol{w}_t] \in \mathbb{R}^{d \times (k-1)}, \quad \text{and} \quad \boldsymbol{b} = [w_{t,0} - w_{1,0}, \ldots, w_{t,0} - w_{k,0}]^T \in \mathbb{R}^{k-1},$$

Then, the constraint set is

$$\Omega = \{\boldsymbol{x} \mid \boldsymbol{A}^T \boldsymbol{x} \leq \boldsymbol{b}\}. \tag{3.12}$$

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \quad \text{subject to} \quad \boldsymbol{A}^T\boldsymbol{x} \leq \boldsymbol{b}$$

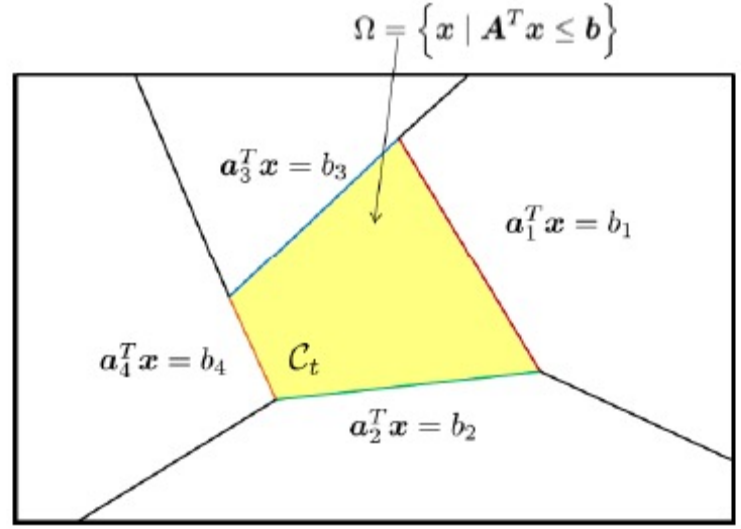


Figure 3.4: Geometry of the constraint set Ω for a linear classifier. The constraint set Ω is now a polygon with decision boundaries defined by $\boldsymbol{a}_i^T\boldsymbol{x} = b_i$, where $\boldsymbol{a}_i = \boldsymbol{w}_i - \boldsymbol{w}_t$ and $b_i = w_{i0} - w_{t0}$.

Linear attacks

Minimum norm attack

A. Minimum-Norm Attack

In order to gain insight of the adversarial attack, it is useful to consider a simple linear classifier with only two classes. A linear classifier has a discriminant function

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}, \tag{3.22}$$

and therefore by defining $\boldsymbol{w} \stackrel{\text{def}}{=} \boldsymbol{w}_i - \boldsymbol{w}_t$ and $w_0 \stackrel{\text{def}}{=} w_{i0} - w_{t0}$, we can show that

$$g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0. \tag{3.23}$$

Recalling Equation (3.3), the adversarial attack can be formulated as

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \\ \text{subject to} & g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) = 0. \end{array} \tag{3.24}$$

In this optimization, we simplify $\max_{j \neq t} \{g_j(\boldsymbol{x})\}$ to $g_i(\boldsymbol{x})$ because there are only two classes in our problem. If the index is not t, then it has to be i. We also replaced the inequality $g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0$ as an equality, because the projection theorem in Equation (3.17) suggests that the solution must be on the decision boundary.

Theorem 2 (Minimum ℓ_2 Norm Attack for Two-Class Linear Classifier). *The adversarial attack to a two-class linear classifier is the solution of*

$$\underset{\boldsymbol{x}}{\textit{minimize}} \ \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \ \textit{subject to} \ \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0, \tag{3.25}$$

which is given by

$$\boldsymbol{x}^* = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T \boldsymbol{x}_0 + w_0}{\|\boldsymbol{w}\|_2} \right) \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}. \tag{3.26}$$

Proof. The Lagrange multiplier of the constrained optimization is given by

$$\mathcal{L}(\boldsymbol{x}, \lambda) = \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + \lambda(\boldsymbol{w}^T \boldsymbol{x} + w_0).$$

The solution of the optimization is the saddle point $(\boldsymbol{x}^*, \lambda^*)$ such that $\nabla_{\boldsymbol{x}} \mathcal{L} = 0$ and $\nabla_\lambda \mathcal{L} = 0$. Taking derivative with respect to $\boldsymbol{x}$ and λ yields

$$\begin{aligned} \nabla_{\boldsymbol{x}} \mathcal{L} &= \boldsymbol{x} - \boldsymbol{x}_0 + \lambda \boldsymbol{w} = 0, \\ \nabla_\lambda \mathcal{L} &= \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0. \end{aligned}$$

Regularization norm attack

C. Regularization-based Attack

In both minimum norm attack and maximum allowable attack, the optimizations are constrained. For certain simple cases, e.g., binary linear classifiers, closed-form solutions can be derived. However, for advanced classifiers such as deep neural networks, solving an optimization involving constraints are typically very difficult. The regularization-based attack alleviates the problem by considering

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2+\lambda\left(\max_{j\neq t}\{g_j(\boldsymbol{x})\}-g_t(\boldsymbol{x})\right).$$

For two-class linear classifier, this is simplified to

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2+\lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0).$$

Since the regularization-based attack is an unconstrained version of the minimum norm attack and the maximum allowable attack, one should expect to have a very similar solution.

Theorem 6 (Regularization-based Attack for Two-Class Linear Classifier). *The regularization-based attack for a two-class linear classifier generates the attack by solving*

$$\underset{\boldsymbol{x}}{\textit{minimize}} \quad \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2+\lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0), \tag{3.46}$$

of which the solution is given by

$$\boldsymbol{x}=\boldsymbol{x}_0-\lambda\boldsymbol{w}. \tag{3.47}$$

Proof. Let $\varphi(\boldsymbol{x})=\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2+\lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0)$. Taking the first order derivative and sending to zero yields

$$0=\nabla\varphi(\boldsymbol{x})=(\boldsymbol{x}-\boldsymbol{x}_0)+\lambda\boldsymbol{w}. \tag{3.48}$$

Therefore, we have $\boldsymbol{x}=\boldsymbol{x}_0-\lambda\boldsymbol{w}$. □

Non-linear attacks

Definition 6 (DeepFool Attack by Moosavi-Dezfooli et al. 2016). *The DeepFool attack for a two-class classification generates the attack by solving the optimization*

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2 \quad \text{subject to} \quad g(\boldsymbol{x})=0, \tag{3.29}$$

where $g(\boldsymbol{x})=0$ is the nonlinear decision boundary separating the two classes.

Due to the nonlinearity of $g(\boldsymbol{x})$, it is generally very difficult to derive a closed-form solution. The numerical procedure to compute the solution can be derived by iteratively updating $\boldsymbol{x}$, where each iteration minimizes $\boldsymbol{x}$ over the first order approximation of $g(\boldsymbol{x})$:

$$g(\boldsymbol{x}) \approx g(\boldsymbol{x}^{(k)}) + \nabla_x g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}-\boldsymbol{x}^{(k)}),$$

where $\boldsymbol{x}^{(k)}$ is the k-th iterate of the solution. In other words, we are defining a procedure

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\text{argmin}} \quad \|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|^2 \quad \text{subject to} \quad g(\boldsymbol{x}^{(k)}) + \nabla_x g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}-\boldsymbol{x}^{(k)}) = 0. \tag{3.30}$$

By identifying $\boldsymbol{w}^{(k)} = \nabla_x g(\boldsymbol{x}^{(k)})$ and $w_0^{(k)} = g(\boldsymbol{x}^{(k)}) - \nabla_x g(\boldsymbol{x}^{(k)})^T\boldsymbol{x}^{(k)}$, the linearized problem Equation (3.30) becomes

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\text{argmin}} \quad \|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|^2 \quad \text{subject to} \quad (\boldsymbol{w}^{(k)})^T\boldsymbol{x} + w_0^{(k)} = 0$$

The solution can thus be found by using Equation (3.26), yielding

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \left(\frac{(\boldsymbol{w}^{(k)})^T\boldsymbol{x}^{(k)} + w_0^{(k)}}{\|\boldsymbol{w}^{(k)}\|^2}\right)\boldsymbol{w}^{(k)} \\ &= \boldsymbol{x}^{(k)} - \left(\frac{g(\boldsymbol{x}^{(k)})}{\|\nabla_x g(\boldsymbol{x}^{(k)})\|^2}\right)\nabla_x g(\boldsymbol{x}^{(k)}).\end{aligned}$$

This result is summarized in the following corollary.

Corollary 1 (DeepFool Algorithm for Two-Class Problem). *An iterative procedure to obtain the DeepFool attack solution is*

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \underset{\boldsymbol{x}}{\textit{argmin}} \quad \|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|^2 \quad \textit{subject to} \quad g(\boldsymbol{x}^{(k)}) + \nabla_x g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}-\boldsymbol{x}^{(k)}) = 0 \\ &= \boldsymbol{x}^{(k)} - \left(\frac{g(\boldsymbol{x}^{(k)})}{\|\nabla_x g(\boldsymbol{x}^{(k)})\|^2}\right)\nabla_x g(\boldsymbol{x}^{(k)}).\end{aligned} \tag{3.31}$$

Can we attack with random noise?

3.4 Can Random Noise Attack?

Adversarial attack works because the perturbation is carefully designed. What if we perturb the data by pure i.i.d. Gaussian noise?

Recall that when launching attacks for a linear classifier, the perturbation always takes the form of $\boldsymbol{x} = \boldsymbol{x}_0 + \lambda \boldsymbol{w}$. That is, we want to move $\boldsymbol{x}_0$ along $\boldsymbol{w}$ by an amount λ so that $\boldsymbol{x}$ is misclassified. Now, if we do not move along $\boldsymbol{w}$ but along a random vector $\boldsymbol{r}$ such that

$$\boldsymbol{x} = \boldsymbol{x}_0 + \sigma_r \boldsymbol{r}, \tag{3.57}$$

where $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, then we can ask if we can still misclassify the data point $\boldsymbol{x}$. Clearly, this requires us to check whether $\boldsymbol{w}^T \boldsymbol{r} > 0$. If $\boldsymbol{w}^T \boldsymbol{r} > 0$, then $\boldsymbol{r}$ and $\boldsymbol{w}$ will form an acute angle and so for sufficient step size we will be able to move $\boldsymbol{x}_0$ to another class. If $\boldsymbol{w}^T \boldsymbol{r} < 0$, then $\boldsymbol{w}$ and $\boldsymbol{r}$ are form an obtuse angle and so $\boldsymbol{r}$ will move $\boldsymbol{x}_0$ to an opposite direction.

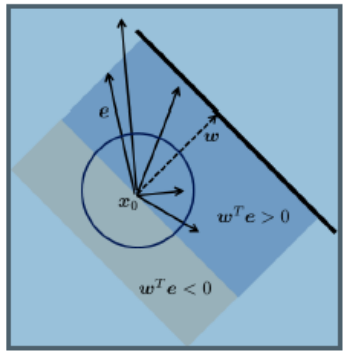


Figure 3.10: Attacking the linear classifier with i.i.d. noise is equivalent to putting an uncertainty circle around $\boldsymbol{x}_0$ with radius σ_r. The possible attack directions are those that form acute angle with the normal vector $\boldsymbol{w}$. Therefore, among all the possible $\boldsymbol{r}$'s, we are only interested in those such that $\boldsymbol{w}^T \boldsymbol{r} > 0$, which occupy half of the space in 2D.

Figure 3.10 provides a pictorial illustration of the noise attack. From this figure, we are tempted to think that i.i.d. noise can achieve 50% attack rate because $\boldsymbol{w}^T \boldsymbol{r} > 0$ occupies half of the space. The problem of such argument is that in high dimension, the probability of $\boldsymbol{w}^T \boldsymbol{r} > 0$ is diminishing very quickly as the dimensionality of $\boldsymbol{r}$ grows. This is a well-known

Can we attack with random noise?

phenomenon called **curse of dimensionality**. To illustrate the idea, let us evaluate the probability of $\boldsymbol{w}^T\boldsymbol{r} \geq \epsilon$ for some $\epsilon > 0$. To this end, let us consider

$$\mathbb{P}\left[\frac{1}{d}\boldsymbol{w}^T\boldsymbol{r} \geq \epsilon\right] = \mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \epsilon\right],$$

where d is the dimensionality of $\boldsymbol{w}$, i.e., $\boldsymbol{w} \in \mathbb{R}^d$. The tolerance level ϵ is a small positive constant that stays away from 0.

Example. Before we discuss the theoretical results, it will be useful to do a quick numerical experiment. Consider a special case where $\boldsymbol{w} = \mathbf{1}_{d\times 1}$, i.e., a d-dimensional all-one vector, and $\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. In this case, we define the average as

$$Y \overset{\text{def}}{=} \frac{1}{d}\sum_{j=1}^{d} r_j. \tag{3.58}$$

It follows that Y is a Gaussian random variable because linear combination of Gaussian remains a Gaussian. The mean and variance are

$$\mathbb{E}[Y] = 0, \quad \text{and} \quad \text{Var}[Y] = \frac{1}{d}. \tag{3.59}$$

Therefore, the probability of the event $\{Y > \epsilon\}$ is

$$\begin{aligned}\mathbb{P}[Y > \epsilon] &= \int_{\epsilon}^{\infty} \frac{1}{\sqrt{2\pi/d}} \exp\left\{-\frac{t^2}{2/d}\right\} dt \\ &= \int_{\epsilon\sqrt{\frac{d}{2}}}^{\infty} \frac{1}{\sqrt{\pi}} \exp\left\{-t^2\right\} dt \\ &= \frac{1}{2}\text{erfc}\left(\epsilon\sqrt{d/2}\right),\end{aligned} \tag{3.60}$$

where erfc is the complementary error function defined as $\text{erfc}(z) \overset{\text{def}}{=} \frac{2}{\sqrt{\pi}}\int_z^{\infty} e^{-t^2} dt$. As we can see in Figure 3.11, the probability drops rapidly as d increases.

Let us discuss the general case. One classical result to derive the probability is the tail bound on Gaussian by Feller 1968. [1] If Y is a Gaussian random variable with $Y \sim \mathcal{N}(\mu, \sigma)$, then

$$\mathbb{P}[Y \geq \mu + \sigma\epsilon] \leq \frac{1}{\epsilon}\frac{e^{-\epsilon^2/2}}{\sqrt{2\pi}}. \tag{3.61}$$

Can we attack with random noise?

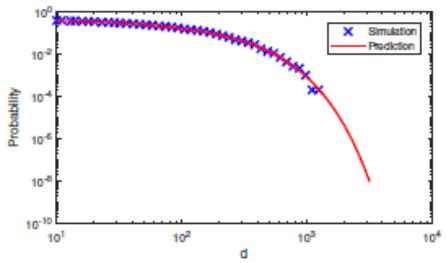


Figure 3.11: Empirical probability and the theoretically predicted value. In this experiment, we let $Y = \frac{1}{d}\sum_{j=1}^{d} r_j$ where $r_j \sim \mathcal{N}(0,1)$ is an iid Gaussian random variable. We plot the probability $\mathbb{P}[Y > \epsilon]$ as a function of d. Note the tightness of the prediction, and the rapid decaying behavior of the probability. The result shows that when d grows, the value Y is concentrated at 0.

Now, let $Y = \frac{1}{d}\sum_{j=1}^{d} w_j r_j$. Since a linear combination of Gaussian remains a Gaussian, it holds that Y is Gaussian and

$$\mu = \mathbb{E}[Y] = 0, \quad \text{and} \quad \sigma^2 = \mathrm{Var}[Y] = \frac{1}{d^2}\sum_{j=1}^{d} w_j^2 = \frac{\|\boldsymbol{w}\|^2}{d^2}. \tag{3.62}$$

Therefore, by substituting $\varepsilon = \sigma\epsilon$ we can show that

$$\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right] = \mathbb{P}[Y \geq \varepsilon] \leq \frac{\sigma}{\varepsilon}\frac{e^{-\frac{\varepsilon^2}{2\sigma^2}}}{\sqrt{2\pi}} = \frac{\|\boldsymbol{w}\|}{\varepsilon d\sqrt{2\pi}} \exp\left\{-d^2\frac{\varepsilon^2}{2\|\boldsymbol{w}\|^2}\right\}. \tag{3.63}$$

As $d \to \infty$, it holds that $\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right] \to 0$. That means, the probability of getting a "good attack direction" is diminishing to zero exponentially. Putting everything together we have the following theorem.

Theorem 7. *Let $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$ be the decision boundary of a linear classifier, and let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be an input data point. Suppose we attack the classifier by adding i.i.d. Gaussian noise $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to $\boldsymbol{x}_0$. The probability of a successful attack at a tolerance level ε is $\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right]$, and such probability is upper bounded by*

$$\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right] \leq \frac{\|\boldsymbol{w}\|}{\varepsilon d\sqrt{2\pi}} \exp\left\{-d^2\frac{\varepsilon^2}{2\|\boldsymbol{w}\|^2}\right\}. \tag{3.64}$$

Therefore, as $d \to \infty$ it becomes increasingly more difficult for i.i.d. Gaussian noise to succeed in attacking.

Equivalence robust classifier/maximum margin

Definition 2.2 (Dual norm)**.** Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The associated *dual norm*, denoted $\|\cdot\|_*$, is defined as $\|\delta\|_* = \sup_{\boldsymbol{x}}\{|\langle \delta, x\rangle| \mid \|x\| \le 1\}$.

Definition 2.3 (Linear Separability)**.** We say a dataset is linearly separable if there exists w, b such that $y_i(w^\top x_i + b) > 0$ for all i.

Lemma 2.1 (Maximally Robust Linear Classifier (Ben-Tal et al. [14], §12))**.** *For linear models and linearly separable data, the following problems are equivalent; i.e., from a solution of one, a solution of the other is readily found.*

$$\textit{Maximally robust classifier:} \quad \arg\max_{w,b}\{\varepsilon \mid y_i(w^\top(x_i+\delta)+b) > 0,\ \forall i, \|\delta\| \le \varepsilon\}\,, \tag{4}$$

$$\textit{Maximum margin classifier:} \quad \arg\max_{w,b:\|w\|_*\le 1} \{\varepsilon \mid y_i(w^\top x_i + b) \ge \varepsilon,\ \forall i\}\,, \tag{5}$$

$$\textit{Minimum norm classifier:} \quad \arg\min_{w,b}\{\|w\|_* \mid y_i(w^\top x_i + b) \ge 1,\ \forall i\}\,. \tag{6}$$

The expression $\min_i y_i(w^\top x_i + b)/\|w\|$ *is the margin of a classifier* w *that is the distance of the nearest training point to the classification boundary, i.e. the line* $\{v : w^\top v = -b\}$.

Proof. We first show that the maximally robust classifier is equivalent to a robust counterpart by removing δ from the problem,

$$\arg\max_{w,b}\{\varepsilon \mid y_i(w^\top(x_i+\delta)+b) > 0,\ \forall i,\ \|\delta\| \le \varepsilon\}$$

(homogeneity of p-norm)

$$= \arg\max_{w,b}\{\varepsilon \mid y_i(w^\top(x_i+\varepsilon\delta)+b) > 0,\ \forall i,\ \|\delta\| \le 1\}$$

(if it is true for all δ it is true for the worst of them)

$$= \arg\max_{w,b}\{\varepsilon \mid \inf_{\|\delta\|\le 1} y_i(w^\top(x_i+\varepsilon\delta)+b) > 0,\ \forall i\}$$

$$= \arg\max_{w,b}\{\varepsilon \mid y_i(w^\top x_i+b) + \varepsilon \inf_{\|\delta\|\le 1} w^\top\delta > 0,\ \forall i\}$$

(definition of dual norm)

$$= \arg\max_{w,b}\{\varepsilon \mid y_i(w^\top x_i+b) > \varepsilon\|w\|_*,\ \forall i\}$$

Assuming $w \neq 0$, which is a result of linear separability assumption, we can divide both sides by $\|w\|_*$ and change variables,

$$= \arg\max_{w,b}\{\varepsilon \mid y_i(w^\top x_i+b) \ge \varepsilon,\ \forall i, \|w\|_* \le 1\}\,,$$

where we are also allowed to change $>$ to $\ge$ because any solution to one problem gives an equivalent solution to the other given $w \neq 0$.

Proof continues

Now we show that the robust counterpart is equivalent to the minimum norm classification problem by removing ε. When the data is linearly separable there exists a solution with $\varepsilon > 0$,

$$\begin{aligned}
&\arg\max_{\boldsymbol{w},b} \{\varepsilon \mid y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) > \varepsilon \|w\|_*,\ \forall i\} \\
&= \arg\max_{\boldsymbol{w},b} \left\{\varepsilon \mid y_i \left(\frac{\boldsymbol{w}^\top}{\varepsilon\|\boldsymbol{w}\|_*}\boldsymbol{x}_i + \frac{b}{\varepsilon\|\boldsymbol{w}\|_*}\right) \geq 1,\ \forall i\right\}
\end{aligned}$$

This problem is invariant to any non-zero scaling of (w, b), so with no loss of generality we set $\|w\|_* = 1$.

$$= \arg\max_{\boldsymbol{w},b} \left\{\varepsilon \mid y_i \left(\frac{\boldsymbol{w}^\top}{\varepsilon}\boldsymbol{x}_i + b\right) \geq 1,\ \forall i, \|\boldsymbol{w}\|_* = 1\right\}$$

Let $w' = w/\epsilon$, then the solution to the following problem gives a solution for w,

$$\begin{aligned}
&\arg\max_{\boldsymbol{w}',b} \left\{\frac{1}{\|\boldsymbol{w}'\|_*} \mid y_i(\boldsymbol{w}'^\top \boldsymbol{x}_i + b) \geq 1,\ \forall i\right\} \\
&= \arg\min_{\boldsymbol{w}',b} \left\{\|\boldsymbol{w}'\|_* \mid y_i(\boldsymbol{w}'^\top \boldsymbol{x}_i + b) \geq 1,\ \forall i\right\}.
\end{aligned}$$

Robustness and implicit Bias

Theorem 3.1 (Implicit Bias of Steepest Descent (Gunasekar et al. [12] (Theorem 5))). *For any separable dataset* $\{x_i, y_i\}$ *and any norm* $\|\cdot\|$, *consider the steepest descent updates from* (7) *for minimizing the empirical risk* $\mathcal{L}(w)$ *(defined in Section 2) with the exponential loss,* $\zeta(z) = \exp(-z)$. *For all initializations* w_0, *and all bounded step-sizes satisfying a known upper bound, the iterates* w_t *satisfy*

$$\lim_{t\to\infty} \min_i \frac{y_i w_t^\top x_i}{\|w_t\|} = \max_{w:\|w\|\le 1} \min_i y_i w^\top x_i \,. \tag{8}$$

In particular, if a unique maximum margin classifier $w^*_{\|\cdot\|} = \arg\max_{w:\|w\|\le 1} \min_i y_i w^\top x_i$ *exists, the limit direction converges to it, i.e.* $\lim_{t\to\infty} \frac{w_t}{\|w_t\|} = w^*_{\|\cdot\|}$.

Robustness and implicit Bias

Corollary 1 (Implicit Robustness of Steepest Descent). *For any linearly separable dataset and any norm* $\|\cdot\|$*, steepest descent iterates minimizing the empirical risk,* $\mathcal{L}(\boldsymbol{w})$*, satisfying the conditions of Theorem 3.1, converge in direction to a maximally robust classifier,*

$$\arg\max_{\boldsymbol{w}}\{\varepsilon \,|\, y_i \boldsymbol{w}^\top (\boldsymbol{x}_i + \boldsymbol{\delta}) > 0,\ \ \forall i,\ \|\boldsymbol{\delta}\|_* \leq \varepsilon\}\,.$$

In particular, a maximally robust classifier against ℓ_1*,* ℓ_2*, and* ℓ_∞ *is reached, respectively, by sign gradient descent, gradient descent, and coordinate descent.*

Proof. By Theorem 3.1, the margin of the steepest descent iterates, $\min_i \frac{y_i \boldsymbol{w}_t^\top \boldsymbol{x}_i}{\|\boldsymbol{w}_t\|}$, converges as $t \to \infty$ to the maximum margin, $\max_{\boldsymbol{w}:\|\boldsymbol{w}\|\leq 1} \min_i y_i \boldsymbol{w}^\top \boldsymbol{x}_i$. By Lemma 2.1, any maximum margin classifier w.r.t. $\|\cdot\|$ gives a maximally robust classifier w.r.t. $\|\cdot\|_*$. □

IB Fourier and maximal robustness

Definition 3.3 (Discrete Fourier Transform). $\mathcal{F}(\boldsymbol{w}) \in \mathbb{C}^d$ denotes the Fourier coefficients of $\boldsymbol{w}$ where $[\mathcal{F}(\boldsymbol{w})]_d = \frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} [\boldsymbol{w}]_k \exp(-\frac{2\pi j}{d} kd)$ and $j^2 = -1$.

Theorem 3.2 (Implicit Bias towards Fourier Sparsity (Gunasekar et al. [22], Theorem 2, 2.a)). *Consider the family of L-layer linear convolutional networks and the sequence of gradient descent iterates, $\boldsymbol{w}_t$, minimizing the empirical risk, $\mathcal{L}(\boldsymbol{w})$, with the exponential loss, $\exp(-z)$. For almost all linearly separable datasets under known conditions on the step size and convergence of iterates, $\boldsymbol{w}_t$ converges in direction to the classifier minimizing the norm of the Fourier coefficients given by*

$$\arg\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_L} \{\|\mathcal{F}(\boldsymbol{w})\|_{2/L} \mid y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \geq 1, \forall i\}. \tag{9}$$

In particular, for two-layer linear convolutional networks the implicit bias is towards the solution with minimum ℓ_1 norm of the Fourier coefficients, $\|\mathcal{F}(\boldsymbol{w})\|_1$. For $L > 2$, the convergence is to a first-order stationary point.

IB Fourier and maximal robustness

Corollary 2 (Maximally Robust to Perturbations with Bounded Fourier Coefficients). *Consider the family of two-layer linear convolutional networks and the gradient descent iterates, $\boldsymbol{w}_t$, minimizing the empirical risk. For almost all linearly separable datasets under conditions of Theorem 3.2, $\boldsymbol{w}_t$ converges in direction to a maximally robust classifier,*

$$\arg\max_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_L} \{\varepsilon \,|\, y_i \varphi_{conv}(\boldsymbol{x}_i + \boldsymbol{\delta}; \{\boldsymbol{w}_l\}_{l=1}^{L}) > 0, \ \forall i, \ \|\mathcal{F}(\boldsymbol{\delta})\|_\infty \leq \varepsilon\} .$$

Fourier, proof

B.2 Proof of Maximally Robust to Perturbations Bounded in Fourier Domain (Corollary 2)

The proof mostly follows from the equivalence for linear models in Appendix B.1 by substituting the dual norm of Fourier-ℓ_1. Here, $\boldsymbol{A}^*$ denotes the complex conjugate transpose, $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^\top \boldsymbol{v}^*$ is the complex inner product, $[\boldsymbol{F}]_{ik} = \frac{1}{\sqrt{D}} \omega_D^{ik}$ the DFT matrix where $\omega_D = e^{-j2\pi/D}$, $j = \sqrt{-1}$.

Let $\| \cdot \|$ be a norm on $\mathbb{C}^n$ and $\langle \cdot, \cdot \rangle$ be the complex inner product. Similar to $\mathbb{R}^n$, the associated dual norm is defined as $\|\boldsymbol{\delta}\|_* = \sup_{\boldsymbol{x}} \{ |\langle \boldsymbol{\delta}, \boldsymbol{x} \rangle| \mid \|\boldsymbol{x}\| \leq 1 \}$.

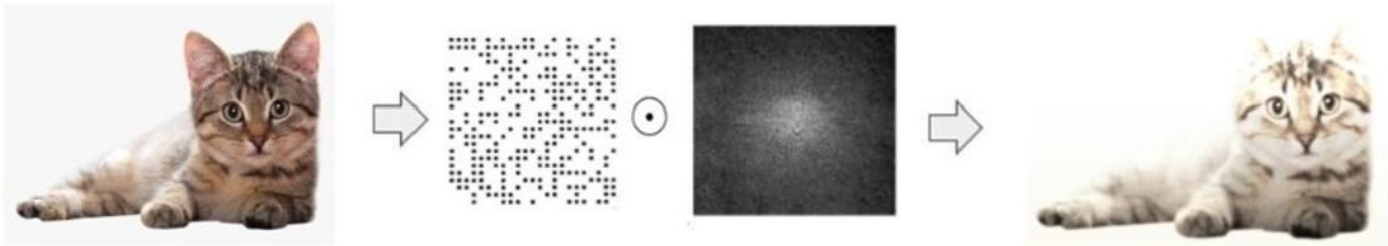
$$
\begin{aligned}
&\|\mathcal{F}(\boldsymbol{w})\|_1 \\
&= \sup_{\|\boldsymbol{\delta}\|_\infty \leq 1} |\langle \mathcal{F}(\boldsymbol{w}), \boldsymbol{\delta} \rangle| \\
&\qquad \text{(Expressing DFT as a linear transformation.)} \\
&= \sup_{\|\boldsymbol{\delta}\|_\infty \leq 1} |\langle \boldsymbol{F}\boldsymbol{w}, \boldsymbol{\delta} \rangle| \\
&= \sup_{\|\boldsymbol{\delta}\|_\infty \leq 1} |\langle \boldsymbol{w}, \boldsymbol{F}^*\boldsymbol{\delta} \rangle| \\
&\qquad \text{(Change of variables and } \boldsymbol{F}^{-1} = \boldsymbol{F}^*\text{.)} \\
&= \sup_{\|\boldsymbol{F}\boldsymbol{\delta}\|_\infty \leq 1} |\langle \boldsymbol{w}, \boldsymbol{\delta} \rangle| \\
&= \sup_{\|\mathcal{F}(\boldsymbol{\delta})\|_\infty \leq 1} |\langle \boldsymbol{w}, \boldsymbol{\delta} \rangle| \,.
\end{aligned}
$$

An aspect of the implicit bias: important Fourier features. A step towards testing for interpretability

Question

Which Fourier features are essential to the network to solve the task?

x $\quad M_\Phi \odot \mathcal{F}x$ $\quad \mathcal{F}^{-1}(M_\Phi \odot \mathcal{F}x)$



$$M_\Phi(\lambda, p) = \text{argmin}_{M_\Phi} \sum_{x \in \mathcal{X}_V} e^{[\mathcal{L}(\Phi(\bar{x}),y) - \mathcal{L}(\Phi(x),y)]^2} + \lambda \| M_\Phi \|_p, \quad \lambda \in \mathbb{R}_+,$$

Result: M_Φ highlights the important Fourier features.

The study hasn't been done in the context of neuroscience but the technique can be applied.

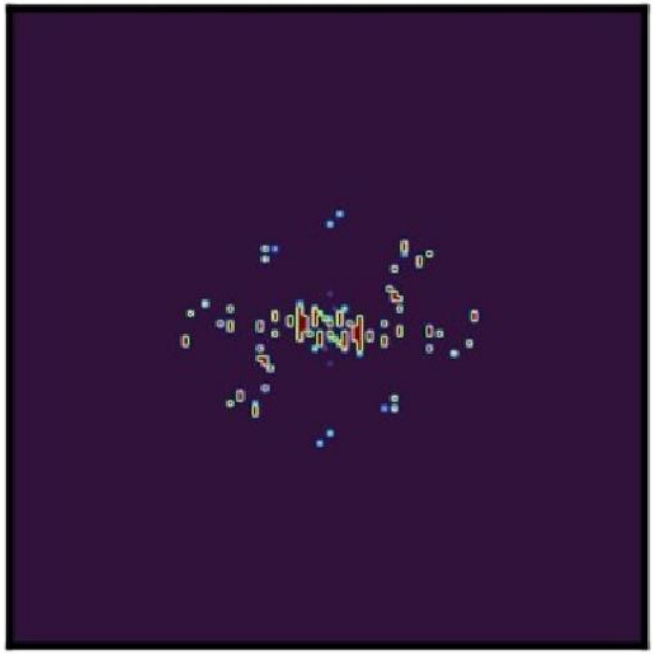
Algorithm

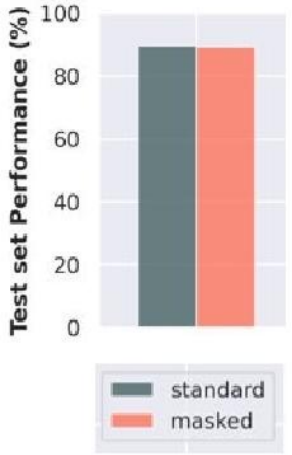
- Train the model
- Freeze the weights and add a mask layer
- Train the mask to
  1. preserve the score
  2. maximize the frequency sparsity

Why Fourier? Many successful stories

- Abello et al. (2021). Dissecting the high-frequency bias in convolutional neural In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.
- Caro et al. (2020). Local convolutions cause an implicit bias towards high frequency adversarial examples.
- Christian et al. (2021). Ringing relus: Harmonic distortion analysis of nonlinear feedforward networks. ICLR
- Geirhos, R.et al. Imagenet trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. ICLR. Li, Z. et al. (2022). Robust deep learning object recognition models rely on low frequency information in natural images.
- Ortiz-Jimenez, G. at al. (2020). Neural anisotropy directions.
- Sharma, Y., Ding, G. W., and Brubaker, M. A. (2019). On the Effectiveness of Low Frequency Perturbations. IJCAI

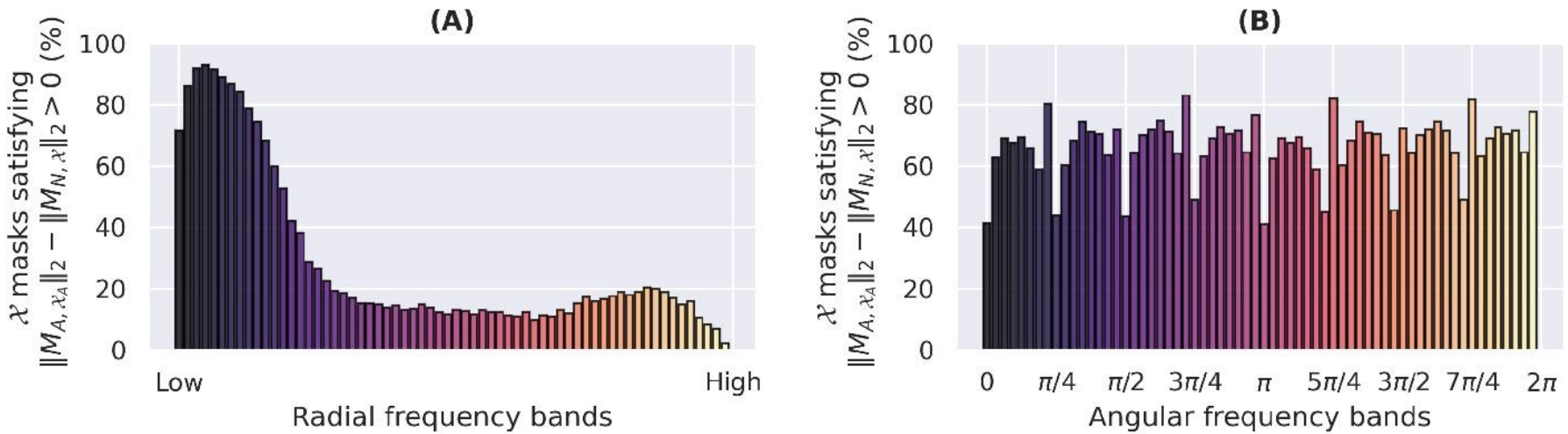
Single image Fourier masks





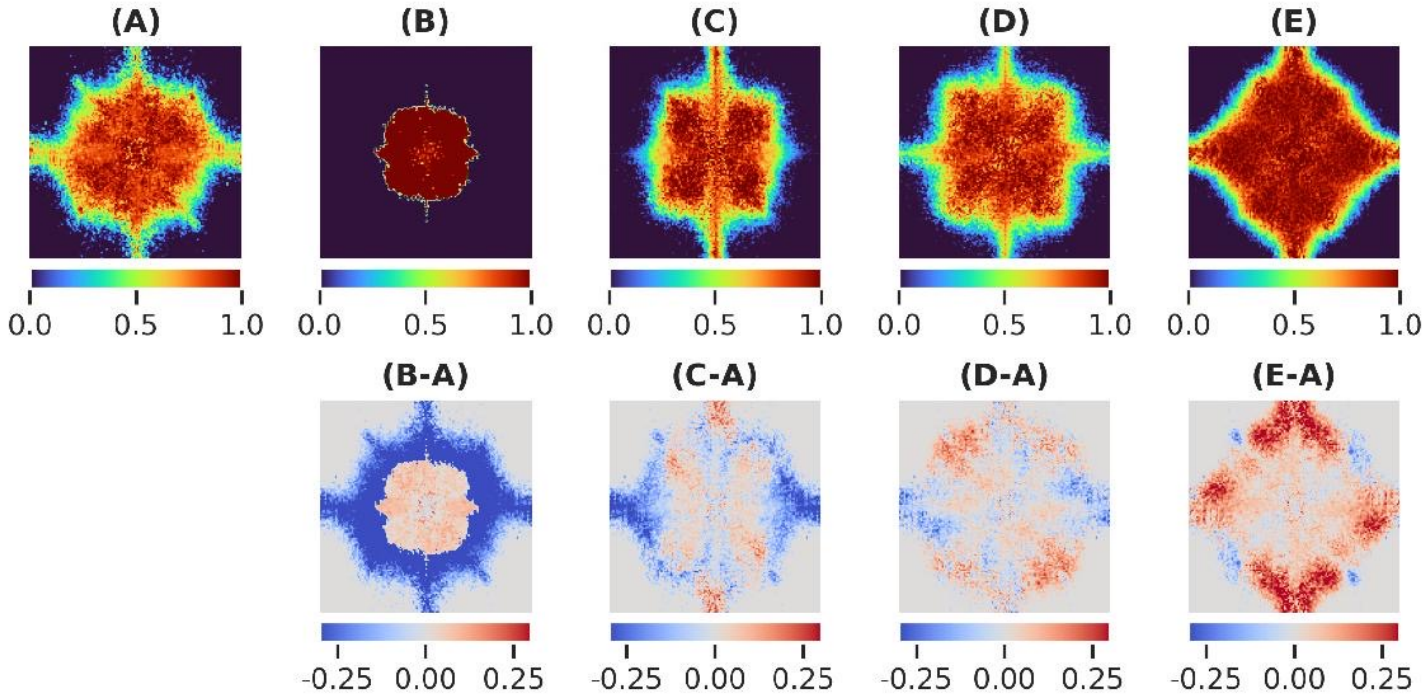
- Important frequencies are very sparse.
- The performance is preserved with the mask
- The performance drops 50% with complementary mask

Energy distribution



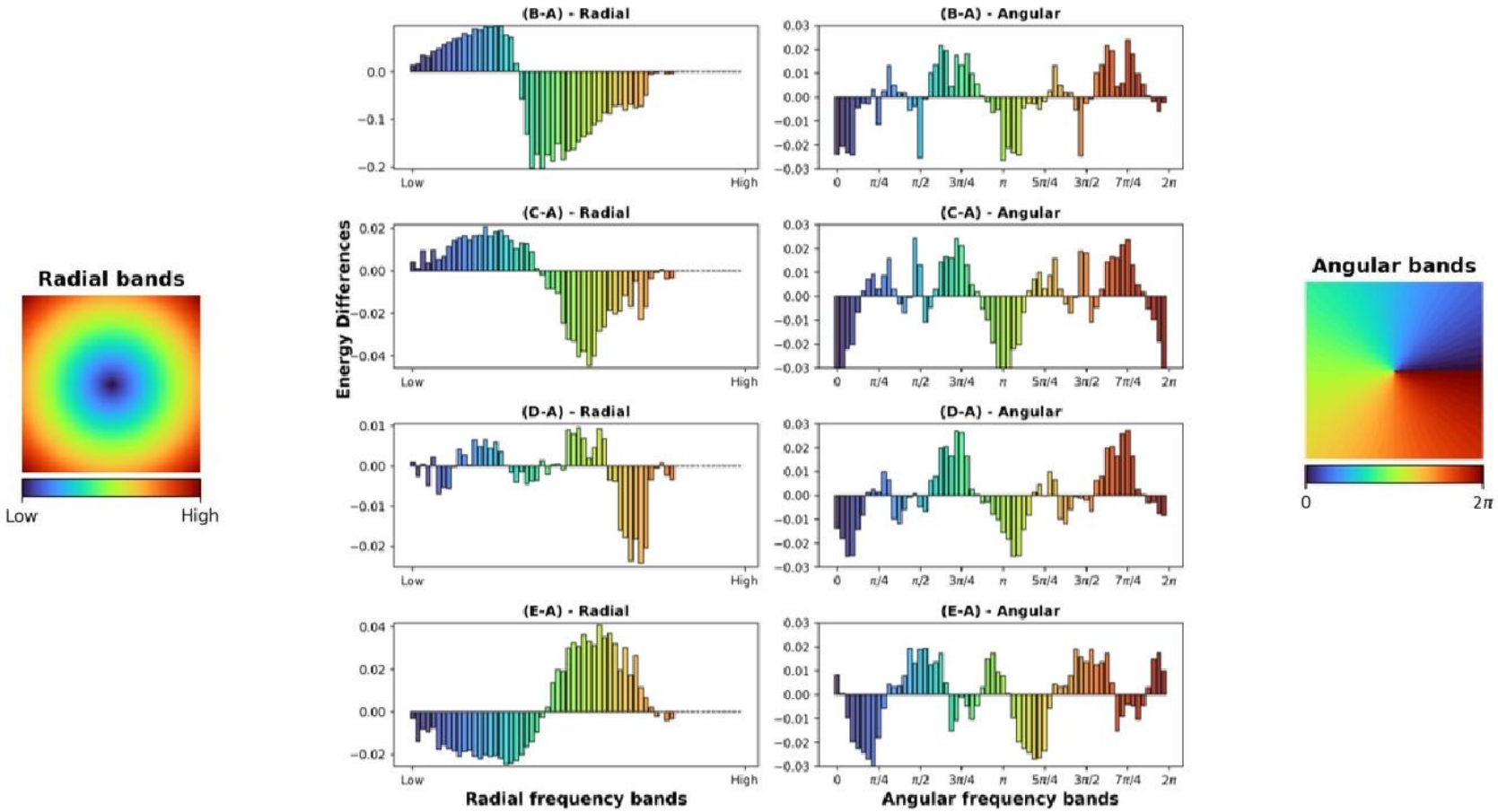
Lower frequencies are more important in the adversarially trained net w.r.t. the naive net.

Data augmentation (full dataset): testing the low-frequency bias



A: Naive
B: Adversarially trained
C: Scale augmentation
D: Translation augmentation
E: Rotation augmentation

Data augmentation (full dataset): testing the low-frequency bias



*Adversarial training and scale are biasing towards lower frequencies.
*The bias is highly direction dependent.

Where does this leave us?

- Although reaching good performance ANNs are relying on features that do not seem to be those the brain is relying on.
- Test "brain-like" Anns:
  1. Train an ANN to a recognition task that an animal also performs well.
  2. Find the essential frequencies and produce filtered images.
  3. Show back to the animal the filtered images and test for the animal performance.
- Generate "brain-like" Anns:
  1. Online learning of a frequency filter that maximize real neurons activity (how?)
  2. Learn a network to do e.g. recognition on mask-filtered images.