Imperial College London

# Theories of Deep Learning

Deep Learning Course

Kevin Webster, Pierre Harvey Richemond

# Course outline

1. Introduction to deep learning
2. Neural networks optimisation
3. Convolutional neural networks
4. Introduction to reinforcement learning - part 1
5. Introduction to reinforcement learning - part 2
6. Sequence models
7. Generative adversarial networks (GANs)
8. Variational autoencoders (VAEs)
9. Normalising flows
10. Theories of deep learning

# Why theories ?

- As of writing date, no single mathematical theory manages to capture the surprising phenomena we have seen occur in deep learning.
- ’Deep Learning is X’ (X=renormalization group, etc...) usually is a very incomplete characterization.
- Some sub-fields of mathematics do shed light and quantify particular aspects; the purpose of today’s course is to illustrate a few of those.

# Information bottleneck theory

# Mutual information refresher

Morally, *mutual information* measures the information that r.v.s $X$ and $Y$ share: it measures how much knowing one of these variables reduces uncertainty about the other[1]. Extreme cases = independence, deterministic dependence $Y = f(X)$. It is symmetric and positive.

$$I(X;Y) := \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy = KL\Big[p(x,y) || p(x) \odot p(y)\Big]$$

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X) \\ &= H(X,Y) - H(X|Y) - H(Y|X) \end{aligned}$$

$$I(\phi(X);\psi(Y)) = I(X,Y) \text{ for invertible } \phi, \psi$$

[1] If unfamiliar with the notion, first think of it as a non-linear covariance measure, in the information theory setting. For instance two $\rho-$correlated normal standard rv's have MI $= -\frac{1}{2}\log(1-\rho^2)$.

# Preliminaries

- Information bottleneck theory = information-theoretic tradeoff between compression and prediction.
- Argues that the goal of any supervised learning is to capture and efficiently represent the relevant information in the input variable about the output (label) variable, as measured by mutual information.
- Hence, lossy compression = extract an approximate minimal sufficient statistics of the input with respect to the output.
- Setting = deep feedforward classification networks with data $X$, labeled ground truth $Y$, predictions $\hat{Y}$, and $m$ sigmoidal layers

$$h_k = \sigma(W_k h_{k-1} + b_k)$$

# Information bottleneck - Motivation

The information bottleneck theory applied to neural networks consists in computing

1. the mutual information between the data and the learned hidden representations on the one hand, and
2. between labels and again hidden learned representations on the other hand.
3. Training should maximize the information with respect to the labels and simultaneously minimize the information with respect to the input data, preventing overfitting and leading to a good generalization.

# Information Bottleneck - 1

- Pre-activation neurons implement hyperplane separation, and hyperplanes can optimally classify data when the inputs are conditionally independent. The goal of the successive layers is therefore to learn representations that can approximately decouple inputs.
- The ratio $I(Y;\hat{Y})/I(X;Y)$ quantifies how much of the relevant information $I(X;Y)$ is captured by the network ($I(Y;\hat{Y})$ is a natural quantifier of the quality of the DNN).
- To this end, at each layer we want to find *minimal sufficient statistics* of the previous ones. So the problem is finding a short code for $X$ that preserves the maximum information about $Y$.
- This is constrained optimization.

# Information Bottleneck - 2

- If we want to find the relevant part of $X$ with respect to $Y$ - an optimal representation denoted by $\hat{X}$ - we need to minimize the Lagrangian $\mathcal{L}[p(\hat{x}|x)]$:

$$\min_{\hat{X}} \quad \underbrace{I(\hat{X};X)}_{\text{minimal}} - \beta \cdot \underbrace{I(\hat{X};Y)}_{\text{sufficient}}$$

- This is a variational problem with closed-form solution (Gibbs measure) - see Shannon-Kolmogorov rate distortion theorem.

$$p(\hat{x}|x) = \frac{p(\hat{x})}{Z(x,\beta)} \exp(-\beta \cdot KL\Big[p(y|x)||p(y|\hat{x})\Big])$$

- This tells us how to train a one-layer network... but we can iterate, and divide and conquer our coding problem.
- Each layer $h_i$ should hence attempt to maximize $I(Y;h_i)$ while minimizing $I(h_{i-1};h_i)$ as much as possible.

# Information Bottleneck - 3

- Conveniently defining $h_0 = X$ and $h_{m+1} = \hat{Y}$, the information bottleneck principle is that network training enforces

$$\forall i \in [1; m+1], \quad h_i \leftarrow \min_{h_i} \quad I(h_{i-1}; h_i) - \beta \cdot I(h_i; Y)$$

- Since $\frac{\delta I(\hat{X}; Y)}{\delta I(\hat{X}; X)} = \beta^{-1}$, the joint distribution $p(X, Y)$ of the dataset imposes an optimal tradeoff curve $(\beta_t)$ in the $\beta$-information plane. Network layers are single points on that plane.
- Shifting from low to higher level representations is analogous to successively decreasing $\beta$ (deterministic annealing).
- The network and all its hidden layers can be directly compared to the optimal IB limit, by estimating the mutual information between each layer and the input and the output variables, on the $\beta$-information plane.

# The $\beta$-information plane

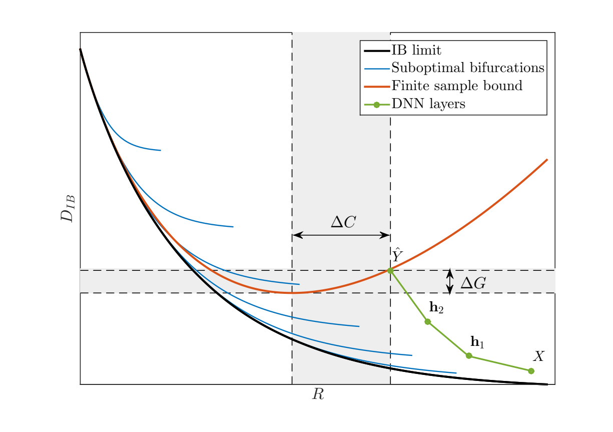

**Figure 1:** A qualitative $\beta$-information plane. The black line is the optimal achievable IB limit ($\beta$) from dataset $p(X, Y)$. The red line is the upper bound on the *out-of-sample* IB distortion when training from a finite sample. Green is the hypothesized path of the layers in a typical DNN on training data. Blue lines are suboptimal IB bifurcations.

# Looking at SGD in the information plane : results

- To understand the dynamics of SGD, we visualize network layers $T_i$ in another information plane, against $I_X = I(X; T_i)$ and $I_Y = I(T_i; Y)$. We observe its evolution during training to gain the following insights.
- Most of the training epochs in standard DL are spent on compression of the input to efficient representation and not on fitting the training labels.
- The representation compression phase begins when the training errors becomes small and SGD epochs change from a fast drift to smaller training error into a stochastic relaxation, or random diffusion, constrained by the training error.

# Looking at SGD in the information plane : results (2)

- The converged layers lie on or very close to the IB theoretical bound, and the maps from the input to any hidden layer and from this hidden layer to the output satisfy the IB self-consistent equations. This generalization through noise mechanism is unique to Deep Neural Networks and absent in one layer networks.
- The training time is dramatically reduced when adding more hidden layers. Thus the main advantage of the hidden layers is computational. This can be explained by the reduced relaxation time, as this it scales super-linearly (exponentially for simple diffusion) with the information compression from the previous layer.
- As we expect critical slowing down of the stochastic relaxation near phase transitions on the IB curve, we expect the hidden layers to converge to such critical points.

# SGD dynamics in the information plane

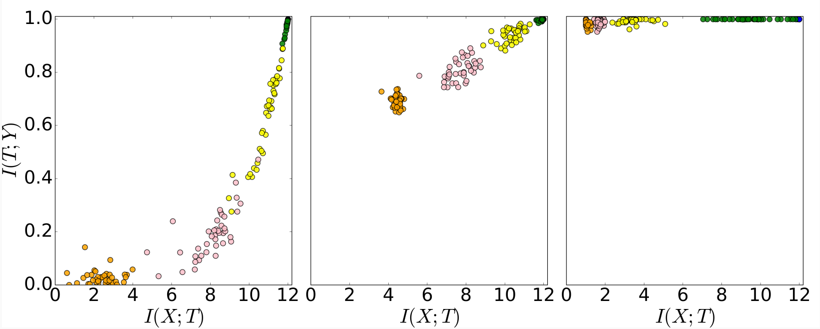

**Figure 2:** Snapshots of layers (different colors) of 50 randomized networks during the SGD optimization process in the *information plane*: **left** - with initial weights; **center** - at 400 epochs; **right** - after 9000 epochs. See online video of the process, outlining first learning and then compression. (Orange layer is deep.)

# SGD dynamics in the information plane

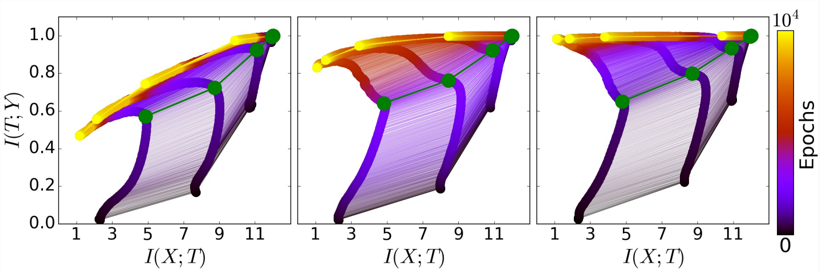

**Figure 3: Evolution of the layers with the training epochs in the information plane, for different training samples**. On the left - 5% of the data, middle - 45% of the data, and right - 85% of the data. Colors indicate the number of SGD training epochs from 0 to 10000. Green paths correspond to the SGD drift-diffusion phase transition (grey line on Figure 4).

# SGD dynamics - Gradients mean & variance

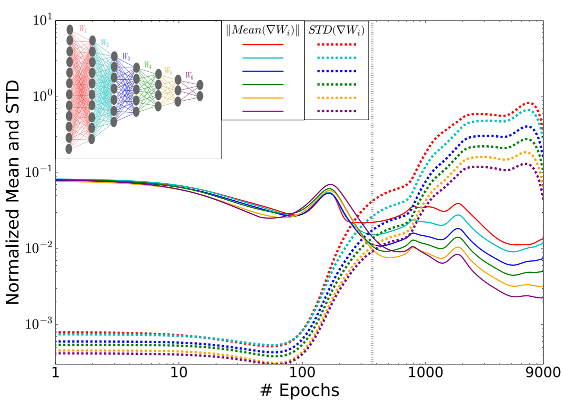

**Figure 4: The layers' stochastic gradients distributions during optimization.** The norm of the means and standard deviations of the weights gradients for each layer, as function of the number of training epochs (in log-log scale). Values are normalized by the L2 norms of the weights for each layer. The grey line ($\sim$ 350 epochs) marks the phase transition.

# In summary

- Information bottleneck theory claims to discover two distinct phases of stochastic gradient descent (SGD) : error minimization and representation compression.
- During the error minimization phase, training (and test) error goes down substantially (*drift*), and the mean of the gradients dominates, whereas...
- During the representation compression phase, SGD ’pumps’ noise into the model (*diffusion*), and washes out the irrelevant bits of information about the input (irrelevant for the task at hand); the variance of the gradients dominates.
- Both transitions in information and gradients magnitude seem to precisely coincide.

# Practical issues and controversy

- Experiments on toy models only for now and only for *tanh* vs *ReLU* (but rebuttal by Tishby)
- Extremely hard to accurately estimate mutual information in practice, many competing estimators
- Objection : ’Networks right before they enter the compression phase already achieve good generalization performance (in fact, the transition to the compression phase seems to take place at a point where the test error starts to asymptote). This makes it questionable whether compression can really explain the bulk of the generalization performance. Relatedly, the compression phase seems to take place over an unrealistically long period. In practical problems, networks are rarely, if ever, trained for that long’. See Saxe et al.

# Spin-glass physics

# Introductory example - Ising model

For a configuration (family) of binary *spins* $\sigma = (\sigma_i)_{i \leq N} \in \{-1; +1\}$, sitting on a $d$-dimensional lattice, the Ising model is given by

$$\mathbb{P}(\sigma) = \frac{e^{-H(\sigma)}}{Z}$$

$$H(\sigma) = -\sum_i h_i \sigma_i - \sum_{i,j} J_{ij} \sigma_i \sigma_j$$

for $i$ and $j$ neighbors, meaning we have an energy model taking into account pairwise interactions.

This model arises from finding a random max-entropy distribution with first and second moments matching a given data (write down the Lagrangian to convince yourself).

So we are matching one- and two-point correlation functions from data, and *the form of H looks very close to that of a two-layer neural network* (also see Hopfield networks !).

# General notions : Hamiltonians

A *configuration* $\boldsymbol{\sigma} \in \Sigma_N \subset \mathbb{R}^N$ is a set of spins, or neural networks weights, for instance. For a given Hamiltonian function $H(\boldsymbol{\sigma})$, which is characteristic of the problem at hand, and a parameter $\beta$ that physically represents the inverse of an outside-world imposed temperature, we weigh every configuration proportionally to its ”Boltzmann factor” $\exp(-\beta H_N(\boldsymbol{\sigma}))$. This defines the Gibbs measure as a probability measure on $\Sigma_N$ defined by

$$G_N(\sigma) = \frac{\exp(-\beta H_N(\boldsymbol{\sigma}))}{Z_N} \tag{1}$$

where the normalizing factor, or *partition function*, is given by

$$Z_N = Z_N(\beta) = \sum_{\boldsymbol{\sigma}} \exp(-\beta H_N(\boldsymbol{\sigma})) \tag{2}$$

The partition function is a number. The minus sign in the Boltzmann factor comes from the fact that the system favours *low* (and not high) energy configurations. The normalizing factor $Z_N$ is crucial; it’s the sum of many terms of widely different orders of magnitude.

# General notions : Hamiltonians - 2

Given a function $f$ on $\Sigma_N$, we denote by $\langle f \rangle$ its average for the Gibbs measure $G_N$:

$$\langle f \rangle = \int f(\boldsymbol{\sigma}) dG_N(\boldsymbol{\sigma}) = \frac{1}{Z_N} \sum_{\boldsymbol{\sigma}} f(\boldsymbol{\sigma}) \exp(-H_N(\boldsymbol{\sigma})) \tag{3}$$

Depending on the temperature, the Gibbs measure is more or less dominated by its maximum (think how states are averaging in statistical physics). As such the objects of interest will be either its mode (average) or its large deviations.

# General notions : Hamiltonians - 3

The average is the partition function $Z_N$; it's exponentially large, and better studied on a log scale via $N^{-1}\log Z_N$. As this is a random quantity, we denote by $p_N$ its expectation

$$p_N = \frac{1}{N}\mathbb{E}\log Z_N = \frac{1}{N}\mathbb{E}\Big[\log\sum_{\boldsymbol{\sigma}}\exp(-H_N(\boldsymbol{\sigma}))\Big]. \tag{4}$$

As $N \to \infty$, the number $p_N$ captures much important information about the random variable $N^{-1}\log Z_N$. This is because in all cases of interest, this number stays bounded above and below independently of $N$, while under rather general conditions, the fluctuations of the random variable become small as $N \to \infty$; in physics' terminology, the random quantity $N^{-1}\log Z_N$ is ”self-averaging”. $p_N$ will be called the *free energy* of the system. For the many models we will consider, the computation of $p_N$ will be a central objective. We will be able to perform this computation in many cases at “high temperature”, but the computation at “low temperature” remains a formidable challenge.

# Spherical pure p-spin Hamiltonians

Moving away from binary valued spins, the Hamiltonian of the *spherical pure p-spin* spin-glass model is given by

$$H_N(\boldsymbol{\sigma}) = \frac{1}{N^{(p-1)/2}} \sum_{i_1,\ldots,i_p=1}^{N} J_{i_1,\ldots,i_p} \sigma_{i_1} \cdots \sigma_{i_p}, \quad \boldsymbol{\sigma} \in \mathbb{S}^{N-1} \tag{5}$$

This can also be seen as a polynomial function of degree $p$ on the $\sigma$-sphere of radius $\sqrt{N}$ where $\sigma$ is an $N$-dimensional vector, with random coefficients $J$. Besides the physical interpretation, this can also be seen as the ”energy” of a neural network with depth $p$ and at most $N$ parameters (weights) described by $\sigma$. $N$ will typically be large; these methods can be seen as ”high-dimensional probability”.

# Neural network analogy : setting

We follow the presentation of Choromanska et al. Simple model of the fully-connected feed-forward deep network $\mathcal{N}$ with a single output and ReLUs. We focus on a binary classification task.

- Let $X$ be the random input vector of dimensionality $d$.
- Let $(H-1)$ denote the number of hidden layers in the network and we will refer to the input layer as the $0^{\text{th}}$ layer and to the output layer as the $H^{\text{th}}$ layer.
- Let $n_i$ denote the number of units in the $i^{\text{th}}$ layer (note that $n_0 = d$ and $n_H = 1$).
- Let $W_i$ be the matrix of weights between $(i-1)^{\text{th}}$ and $i^{th}$ layers of the network.
- Also, let $\sigma$ denote the activation function that converts a unit's weighted input to its output activation. We consider linear rectifiers, $\sigma(x) = \max(0, x)$.

# Neural network analogy : setting - 2

We can therefore write the (random) network output $Y$ as

$$Y = q \cdot \sigma(W_H^\top \sigma(W_{H-1}^\top \dots \sigma(W_1^\top X))) \dots),$$

with $q = \sqrt{(n_0 n_1 ... n_H)^{(H-1)/2H}}$ a normalization factor. The same expression for the output of the network can be re-expressed in the following way:

$$Y = q \cdot \sum_{i=1}^{n_0} \sum_{j=1}^{\gamma} X_{i,j} A_{i,j} \prod_{k=1}^{H} w_{i,j}^{(k)}, \tag{6}$$

where the first summation is over the network inputs and the second one is over all paths from a given network input to its output.

# Equivalence between FF-NN and spin-glass Hamiltonian

The mass of the network $\Psi$ is the total number of all paths between all network inputs and outputs: $\Psi = \prod_{i=0}^{H} n_i$. Also let $\Lambda$ as $\Lambda = \sqrt[H]{\Psi}$.
The size of the network $N$ is the total number of network parameters: $N = \sum_{i=0}^{H-1} n_i n_{i+1}$.

Under a few more assumptions, we have the result that the loss function of the neural network has the form of a centered Gaussian process on the sphere $\mathcal{S} = S^{\Lambda-1}(\sqrt{\Lambda})$, which is equivalent to the Hamiltonian of the $H$-spin spherical spin-glass model, given as

$$\mathcal{L}_{\Lambda,H}(\tilde{\boldsymbol{w}}) = \frac{1}{\Lambda^{(H-1)/2}} \sum_{i_1,i_2,\ldots,i_H=1}^{\Lambda} X_{i_1,i_2,\ldots,i_H} \tilde{w}_{i_1} \tilde{w}_{i_2} \ldots \tilde{w}_{i_H}, \tag{7}$$

with spherical constraint

$$\frac{1}{\Lambda} \sum_{i=1}^{\Lambda} \tilde{w}_i^2 = 1. \tag{8}$$

# Motivation : Nonconvex optimization

This analogy unlocks a wealth of results from the rich literature of statistical physics, including for our purposes, at critical points (zero gradient):

- results on the *large deviations of the network's loss function* (energy level);
- results on the local curvature of the loss landscape, via the *number of negative eigenvalues of the Hessian* (index).

# Results - Large deviations - 1

Additional assumption on the loss : we assume either MAE $\mathcal{L}^{a}_{\Lambda,H}(\mathbf{w}) = \mathbb{E}_A[\,|Y_t - Y|]$ or hinge loss $\mathcal{L}^{h}_{\Lambda,H}(\mathbf{w}) = \mathbb{E}_A[\max(0, 1 - Y_t Y)]$. Let $u \in \mathbb{R}$ and $k$ be an integer such that $0 \leq k < \Lambda$. Define

- $\mathcal{C}_{\Lambda}(B)$ := total number of critical values of $\mathcal{L}_{\Lambda,H}(w)$.
- $\mathcal{C}_{\Lambda,k}(u)$ := random number of critical values of $\mathcal{L}_{\Lambda,H}(\mathbf{w})$ in the set $\Lambda B = \{\Lambda X : x \in (-\infty, u)\}$ with index[2] equal to $k$.

Let us also introduce the number that we will refer to as $E_\infty$. We will refer to this important threshold as the *energy barrier* and define it as

$$E_\infty := E_\infty(H) = 2\sqrt{\frac{H-1}{H}}.$$

[2] The number of negative eigenvalues of the Hessian $\nabla^2 \mathcal{L}_{\Lambda,H}$ at $\mathbf{w}$ is also called index of $\nabla^2 \mathcal{L}_{\Lambda,H}$ at $\mathbf{w}$.

# Results - Large deviations - 2

We will now define two non-decreasing, continuous functions on $\mathbb{R}$, $\Theta_H$ and $\Theta_{k,H}$.

$$\Theta_H(u) = \begin{cases} \frac{1}{2}\log(H-1) - \frac{(H-2)u^2}{4(H-1)} - I(u) & \text{if } u \leq -E_\infty \\ \frac{1}{2}\log(H-1) - \frac{(H-2)u^2}{4(H-1)} & \text{if } -E_\infty \leq u \leq 0 \\ \frac{1}{2}\log(H-1) & \text{if } 0 \leq u \end{cases},$$

and for any integer $k \geq 0$:

$$\Theta_{k,H}(u) = \begin{cases} \frac{1}{2}\log(H-1) - \frac{(H-2)u^2}{4(H-1)} - (k+1)I(u) & \text{if } u \leq -E_\infty \\ \frac{1}{2}\log(H-1) - \frac{H-2}{4(H-1)} & \text{if } u \geq -E_\infty \end{cases}$$

where

$$I(u) = -\frac{u}{E_\infty^2}\sqrt{u^2 - E_\infty^2} - \log(-u + \sqrt{u^2 - E_\infty^2}) + \log E_\infty.$$

# Results - Large deviations - 3

**Theorem (Auffinger et al.)**

*For all $H \geq 2$ and $k \geq 0$ fixed*

$$\lim_{\Lambda \to \infty} \frac{1}{\Lambda} \log \mathbb{E}[\mathcal{C}_{\Lambda,k}(u)] = \Theta_{k,H}(u) \quad \text{and} \quad \lim_{\Lambda \to \infty} \frac{1}{\Lambda} \log \mathbb{E}[\mathcal{C}_{\Lambda}(u)] = \Theta_{H}(u).$$

- Gradient descent gets trapped roughly at the barrier $-\Lambda E_\infty$.
- The number of critical values of the loss below the level $\Lambda u$ is roughly $e^{\Lambda \Theta_H(u)}$.
- The number of critical points in the band $(-\Lambda E_0(H), -\Lambda E_\infty(H))$ increases exponentially as $\Lambda$ grows, and local minima dominate over saddle points. For large-size networks the probability of recovering a saddle point in the band $(-\Lambda E_0(H), -\Lambda E_\infty(H))$, rather than a local minima, goes to 0.
- So past a certain point, essentially no barriers in the landscape of critical points' energies

# Experimental validity of the analogy

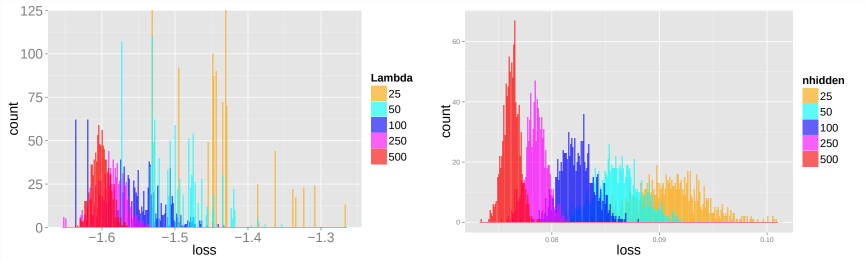

**Figure 5:** Distributions of the scaled test losses for the spin-glass (left) and the neural network (right) experiments.

# Results - Index (nonconvexity)

Expression for the index as a function of the loss/error :

$$\alpha \approx \frac{4\sqrt{2}}{3\pi} \left[ \frac{\epsilon - \epsilon_c}{\epsilon_c} \right]^{\frac{3}{2}}$$

where

- $\alpha$, the **index**, is the number of negative eigenvalues of the Hessian
- $\epsilon$ is the energy or loss value at the critical point
- $\epsilon_c$ is the energy of the minimizer

Statistically, the more we optimize, the more convex we get !

# Curvature and mean-field theory

# Mean-field equations - 1

Consider a deep feedforward network with $D$ layers of weights $\mathbf{W}^1, \dots, \mathbf{W}^D$ and $D+1$ layers of neural activity vectors $x^0, \dots, x^D$, with $N_l$ neurons in each layer $l$, so that $x^l \in \mathbb{R}^{N_l}$ and $\mathbf{W}^l$ is an $N_l \times N_{l-1}$ weight matrix. The feedforward dynamics elicited by an input $x^0$ is given by

$$x^l = \phi(\mathbf{h}^l) \qquad \mathbf{h}^l = \mathbf{W}^l\,{}^{l-1} + \mathbf{b}^l \quad \text{for } \; l = 1, \dots, D, \tag{9}$$

where $\mathbf{b}^l$ is a vector of biases, $\mathbf{h}^l$ is the pattern of inputs to neurons at layer $l$, and $\phi$ is a single neuron scalar nonlinearity that acts component-wise to transform inputs $\mathbf{h}^l$ to activities ${}^l$.

We therefore study ensembles of random networks in which each of the synaptic weights $\mathbf{W}^l_{ij}$ are drawn i.i.d. from a zero mean Gaussian with variance $\sigma_w^2/N_{l-1}$, while the biases are drawn i.i.d. from a zero mean Gaussian with variance $\sigma_b^2$.

# Mean-field equations - 2

$$q^l = \frac{1}{N_l} \sum_{i=1}^{N_l} (\mathbf{h}_i^l)^2. \tag{10}$$

For large $N_l$, this empirical distribution converges to a zero mean Gaussian by central limit, since each $\mathbf{h}_i^l = \sum_j \mathbf{W}_{ij}^l \phi(\mathbf{h}_j^{l-1}) + \mathbf{b}_i^l$ .

By propagating this Gaussian distribution across one layer, we obtain an iterative map for $q^l$:

$$q^l \;=\; \mathcal{V}(q^{l-1} \,|\, \sigma_w, \sigma_b) \;\equiv\; \sigma_w^2 \int \mathcal{D}z \, \phi\left(\sqrt{q^{l-1}}z\right)^2 + \sigma_b^2, \quad \text{for} \quad l = 2, \ldots, D, \tag{11}$$

where $\mathcal{D}z = \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$ is the standard Gaussian measure, and the initial condition is $q^1 = \sigma_w^2 q^0 + \sigma_b^2$, where $q^0 = \frac{1}{N_0} x^0 \cdot x^0$ is the length in the initial activity layer.

# Polarization - 1

Now consider the layer-wise propagation of two inputs $x^{0,1}$ and $x^{0,2}$. The geometry of these two inputs as they propagate through the network is captured by the 2 by 2 matrix of inner products:

$$q^l_{ab} = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathsf{h}^l_i(x^{0,a})\, \mathsf{h}^l_i(x^{0,b}) \qquad a, b \in \{1, 2\}. \tag{12}$$

$$q^l_{12} = \mathcal{C}(c^{l-1}_{12},\, q^{l-1}_{11},\, q^{l-1}_{22} \,|\, \sigma_w, \sigma_b) \equiv \sigma_w^2 \int \mathcal{D}z_1\, \mathcal{D}z_2\, \phi(u_1)\, \phi(u_2) + \sigma_b^2, \tag{13}$$

$$u_1 = \sqrt{q^{l-1}_{11}} z_1, \qquad u_2 = \sqrt{q^{l-1}_{22}} \left[ c^{l-1}_{12} z_1 + \sqrt{1 - (c^{l-1}_{12})^2} z_2 \right],$$

# Polarization - 2

We obtain an iterative correlation coefficient map, or $\mathcal{C}$-map, for $c_{12}^l$:

$$c_{12}^l = \frac{1}{q^*}\mathcal{C}(c_{12}^{l-1},\, q^*,\, q^* \,|\, \sigma_w, \sigma_b). \tag{14}$$

This map always has a fixed point at $c^* = 1$ as can be checked by direct calculation. However, the stability of this fixed point depends on the slope of the map at 1, which is

$$\chi_1 \equiv \left.\frac{\partial c_{12}^l}{\partial c_{12}^{l-1}}\right|_{c=1} = \sigma_w^2 \int \mathcal{D}z \left[\phi'\left(\sqrt{q^*}z\right)\right]^2. \tag{15}$$

If the slope $\chi_1 < 1$, then the C-map is above the unity line, the fixed point at 1 is stable, and nearby points become more similar over time. $\chi_1$ directly reflects the typical multiplicative growth or shrinkage of a random perturbation across one layer. Correlations typically take about 20 layers to approach the fixed point, while lengths need only 4.

# Propagation on Riemannian manifolds

- We also do this analysis for pairs of points on a manifold. The theory for the propagation of pairs of points applies to all pairs of points on the manifold, so intuitively, we expect that in the chaotic phase of a sigmoidal network, the manifold should in some sense de-correlate, and become more complex, while in the ordered phase the manifold should contract around a central point.
- The object of interest becomes the *extrinsic curvature*, which is the inverse of the radius of the ’osculating circle’ to the curve.
- For very broad classes of nonlinearities, even random deep neural networks can construct hidden internal representations whose global extrinsic curvature grows exponentially with depth but not width.

# Propagation of manifold geometry : Grassmannian

Poole et al. derive an iterative formula for the extrinsic curvature and Euclidean metric of this manifold as it propagates through the layers of a deep network:

$$\bar{g}^{E,l} = \chi_1 \, \bar{g}^{E,l-1} \qquad (\bar{\kappa}^l)^2 = 3\frac{\chi_2}{\chi_1^2} + \frac{1}{\chi_1}(\bar{\kappa}^{l-1})^2, \qquad \bar{g}^{E,1} = q^*, \qquad (\bar{\kappa}^1)^2 = 1/q^*. \tag{16}$$

where $\chi_1$ is the stretch factor defined in (15) and $\chi_2$ is defined analogously as

$$\chi_2 = \sigma_w^2 \int \mathcal{D}z \left[\phi''\left(\sqrt{q^*}z\right)\right]^2. \tag{17}$$

$\chi_2$ is closely related to the second derivative of the $\mathcal{C}$-map in (14) at $c_{12}^{l-1} = 1$; this second derivative is $\chi_2 q^*$.

# Geometric interpretation

- This implies that in the chaotic phase, the global curvature measure grows exponentially with depth.
- The curve explores many different tangent directions in hidden representation space. This further implies that the coordinate functions of the embedding become highly complex curved basis functions on the input manifold coordinate, allowing a deep network to compute exponentially complex functions over simple low dimensional manifolds.

# Geometric interpretation

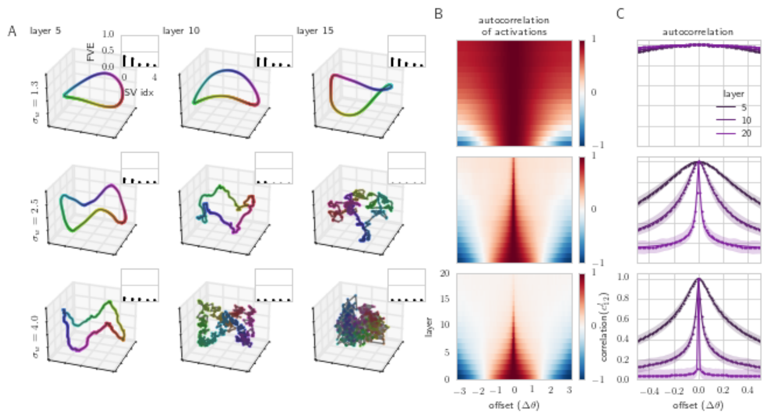

**Figure 6:** Propagating a circle through three random sigmoidal networks with varying $\sigma_w$. Notice growing entanglement as a function of weights variance. From Poole 2016.

# Random matrices & Free probability

# Random Matrix Theory preliminaries

In general, given a random matrix X (a *weight initialization scheme*), its limiting spectral density is defined as

$$\rho_X(\lambda) \equiv \left\langle \frac{1}{N} \sum_{i=1}^{N} \delta(\lambda - \lambda_i) \right\rangle_X , \tag{18}$$

where $\langle \cdot \rangle_X$ denotes an average w.r.t to the distribution over the random matrix X.
We begin with recalling two seminal results of random matrix theory to illustrate that in the limit of an infinite number of bins, two Gaussian ensembles can be characterized by their limiting spectral distributions known in closed form.

# RMT example limit 1 - The Wigner semicircle

For a matrix $M$ of the real Wigner matrix ensemble whose entries $M_{ij} \sim \mathcal{N}(0, \sigma^2)$, the limiting spectral density is the semi-circle law

$$\rho_{SC}(\lambda; \sigma; \phi) = \frac{1}{2\pi\sigma^2}\sqrt{4\sigma^2 - \lambda^2} \quad \text{if} |\lambda| \leq 2\sigma$$

# RMT example limit 2 - The Marcenko-Pastur distribution

Similarly, if $M = XX^T$ is a real Wishart matrix with $X \in \mathbb{R}^{n \times p}$ and $X_{i,j} \sim \mathcal{N}(0, \sigma^2/p)$, then the limiting spectral density can be shown to be the Marcenko-Pastur distribution

$$\rho_{MP}(\lambda; \sigma, \phi) = \rho(\lambda) \quad \text{if} \phi < 1, \quad (1 - \frac{1}{\phi})\delta(\lambda) + \rho(\lambda) \quad \text{otherwise}$$

where $\phi = n/p$ and

$$\rho(\lambda) = \frac{1}{2\pi\lambda\sigma\phi}\sqrt{(\lambda - \lambda_-)(\lambda_+ - \lambda)}$$

# Transforming the spectral density: equivalent forms

The complex *Stieltjes transform* of $\rho_X$ is defined as,

$$G_X(z) \equiv \int_{\mathbb{R}} \frac{\rho_X(t)}{z-t} dt \,, \qquad z \in \mathbb{C} \setminus \mathbb{R} \,, \tag{19}$$

which can be inverted using,

$$\rho_X(\lambda) = -\frac{1}{\pi} \lim_{\epsilon \to 0^+} \operatorname{Im} G_X(\lambda + i\epsilon) \,. \tag{20}$$

$G_X$ is related to the moment generating function $M_X$,

$$M_X(z) \equiv z G_X(z) - 1 = \sum_{k=1}^{\infty} \frac{m_k}{z^k} \,, \tag{21}$$

where $m_k$ is the $k$th moment of the distribution $\rho_X$.

$$m_k = \int d\lambda \; \rho_X(\lambda) \lambda^k = \frac{1}{N} \langle \operatorname{tr} \mathsf{X}^k \rangle_X \,. \tag{22}$$

# Free probability

- The notion of *free probability* is a concept that generalizes the notion of independence between two random variables when they don't commute.
- The assumption needs to be empirically validated in the case of neural networks, but is generally approximately true in our very high dimensional case of interest.
- This additional assumption is crucial to make statements about the eigenvalue spectrum of a sum or a product of freely independent matrices.

# Free probability transforms - S-transform

We denote the functional inverse of $M_X$ by $M_X^{-1}$, which satisfies $M_X(M_X^{-1}(z)) = M_X^{-1}(M_X(z)) = z$. The *S-transform* is defined as

$$S_X(z) = \frac{1+z}{zM_X^{-1}(z)}\,. \tag{23}$$

The utility of the S-transform arises from its behavior under multiplication. Specifically, if **A** and **B** are two freely independent random matrices, then the S-transform of the product random matrix ensemble **AB** is simply the product of their S-transforms,

$$S_{AB}(z) = S_A(z)S_B(z)\,. \tag{24}$$

# Free probability transforms - R-transform

Given the Stieltjes transform $G$ of a probabiliity distribution $\rho$, the $\mathcal{R}$ transform is defined as the solution of the functional equation

$$\mathcal{R}(G(z)) + \frac{1}{G(z)} = z$$

The benefit of the $\mathcal{R}$ transform is that it is an additive morphism for free convolution, in the sense that,

$$\mathcal{R}_{A+B}(z) = \mathcal{R}_A(z) + \mathcal{R}_B(z)$$

# Computing eigenspectra with these transforms

We now have a strategy for computing the limiting eigenspectra[3] of weight matrices which are products and/or sums of matrices originally belonging to an RMT ensemble (think of it as your weight initialization):

1. Write down the limiting distribution of your input ensembles
2. Compute their Stieltjes transforms, then their S or R transforms
3. Combine those using the appropriate morphism
4. Invert these (step 2 backwards).
5. If we don't get closed forms, we can at least identify single moments term by term.

[3] That is, for an infinitely large layer.

# Results - 1

Consider an $L$-layer feed-forward neural network of width $N$ with synaptic weight matrices $\mathbf{W}^l \in \mathbb{R}^{N\times N}$, bias vectors $\mathbf{b}^l$, pre-activations $\mathbf{h}^l$, and post-activations $\mathbf{x}^l$, with $l = 1, \ldots, L$. The forward-propagation dynamics are given by,

$$\mathbf{x}^l = \phi(\mathbf{h}^l)\,, \quad \mathbf{h}^l = \mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l\,, \tag{25}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a pointwise nonlinearity and the input is $\mathbf{x}^0 \in \mathbb{R}^N$. Now consider the input-output Jacobian $\mathbf{J} \in \mathbb{R}^{N\times N}$ given by

$$\mathbf{J} = \frac{\partial \mathbf{x}^L}{\partial \mathbf{x}^0} = \prod_{l=1}^{L} \mathbf{D}^l\mathbf{W}^l. \tag{26}$$

Here $\mathbf{D}^l$ is a diagonal matrix with entries $D^l_{ij} = \phi'(h^l_i)\,\delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta function.

# Results - 2

We are going to use our S-transform strategy to compute the large-width limit of the eigenvalue distribution of the input-output Jacobian.

The S-transform depends only on traces of moments through (21), and that these traces are invariant under cyclic permutations, to derive a simple expression for the S-transform of $JJ^T$,

$$S_{JJ^T} = \prod_{l=1}^{L} S_{(D^l)^2} S_{(W^l)^T W^l} = S^L_{D^2} S^L_{W^T W}. \tag{27}$$

# Results - 3

In addition to numerically extracting the spectrum of $\mathbf{J}\mathbf{J}^T$, we can also calculate its moments $m_k$ encoded in the function

$$M_{JJ^T}(z) \equiv \sum_{k=1}^{\infty} \frac{m_k}{z^k} \,. \tag{28}$$

These moments in turn can be computed in terms of the series expansions of $S_{W^TW}$ and $M_{D^2}$, which we define as

$$S_{W^TW}(z) \equiv \quad \sigma_w^{-2} \left(1 + \textstyle\sum_{k=1}^{\infty} s_k z^k\right) \tag{29}$$

$$M_{D^2}(z) \equiv \quad \textstyle\sum_{k=1}^{\infty} \frac{\mu_k}{z^k} \,, \tag{30}$$

where the moments $\mu_k$ of $\mathbf{D}^2$ are given by,

$$\mu_k = \int \mathcal{D}h \; \phi'(\sqrt{q^*}h)^{2k} \,. \tag{31}$$

# Results - 4

We obtain equations for the unknown moments $m_k$ in terms of the known moments $\mu_k$ and $s_k$. We can solve for the low-order moments by expanding in powers of $z^{-1}$. By equating the coefficients of $z^{-1}$ and $z^{-2}$, we find equations for $m_1$ and $m_2$ yielding,

$$\begin{aligned} m_1 &= (\sigma_w^2 \mu_1)^L \\ m_2 &= (\sigma_w^2 \mu_1)^{2L} \, L \left( \frac{\mu_2}{\mu_1^2} + \frac{1}{L} - 1 - s_1 \right) . \end{aligned} \tag{32}$$

On this critical boundary where the mean $m_1$ of the spectrum of $\mathbf{J}\mathbf{J}^T$ is one for any depth $L$, the variance

$$\sigma_{JJ^T}^2 = m_2 - m_1^2 = L \left( \frac{\mu_2}{\mu_1^2} - 1 - s_1 \right) \tag{33}$$

grows linearly with depth $L$ for generic values of $\mu_1$, $\mu_2$ and $s_1$. Thus $\mathbf{J}$ can be highly ill-conditioned at large depths $L$ for generic choices of nonlinearities and weights, even when $\sigma_w$ and $\sigma_b$ are tuned to criticality.

# Interpretation

- For deep learning we want

$$\frac{\partial \sigma^2_{JJ^T}}{\partial L} = 0 = \frac{\mu_2}{\mu_1^2} - 1 - s_1$$

- So there is interplay between activation functions and initialization ensemble in Jacobian conditioning.
- In particular, the combination of sigmoid activations and orthogonal initializations works well in MLPs.
- While that result does fully not hold in the convolutional setting yet (only fast initialization schemes), it suggests we might be able to do away completely with some of the gradient flow stabilizers like batch normalization and skip connections (?) in the future...

# Conclusion : What else is there ?

Harmonic analysis of convolutional networks (see here and here)
Approximate message passing and variational Bayes
Spectral and PAC-bounds
Tensor decomposition methods
Compressed sensing
Approximation theory
*and so much more...*

# Conclusion

- An exact theory of deep learning is likely to be intractable or uninformative…
- Theory abounds and keeps on coming
- Empirical best practices and auto-differentiation will likely continue to inform deep learning theory for the time being
- Thank you for taking this course !

# References - Information Bottleneck

The Information Bottleneck method
Opening the black box of deep networks through information
Deep Learning and the Information Bottleneck principle
On the Information Bottleneck theory of Deep Learning
SGD performs variational inference
Mutual Information Neural Estimator

# References - Statistical physics and neural theory

The loss surfaces of multilayer networks
Random Matrices and complexity of Spin Glasses
The statistics of critical points of Gaussian fields on large-dimensional spaces
On the energy landscape of deep networks
High-Dimensional Random Fields and Random Matrix Theory

# References - Gradient flow, RMT/free probability

Exponential expressivity in deep neural networks
Geometry of neural network loss surfaces via RMT
The emergence of spectral universality in deep networks
Nonlinear random matrix theory for deep learning
Resurrecting the sigmoid in deep learning
Dynamical isometry and mean field theory of CNNs