

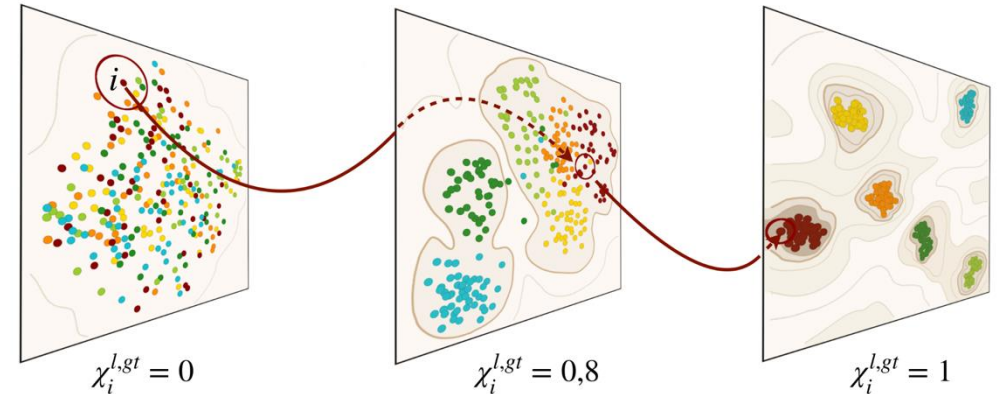
# Neural collapse and pruning

ATML Track1

Rik Sarkar

# Recap : Embedding sequence in NN

- With each layer the NN computes a different embedding in a high dimensional space
  - Alternatively, a different probability density in High dim space
- Large neural networks (with many parameters) can isolate every point into a cell
  - And map each point to the class cluster
  - Achieve any classification



- Example: ResNet152 (2D projections)
  - Input, Conv4, output
  - Initial input scattered and mixed
  - Slowly becomes more clustered
  - $\chi_i$ : fraction of 5 nearest neighbors of  $i$  that are in the same class

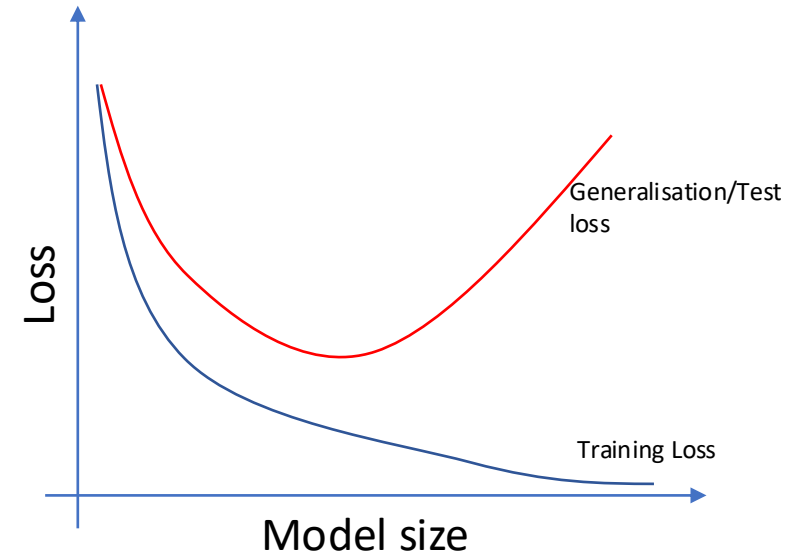
[Doimo 2020]: Hierarchical nucleation in deep neural networks

# Today

- Overparameterisation
- Double descent
- Neural collapse
- Pruning

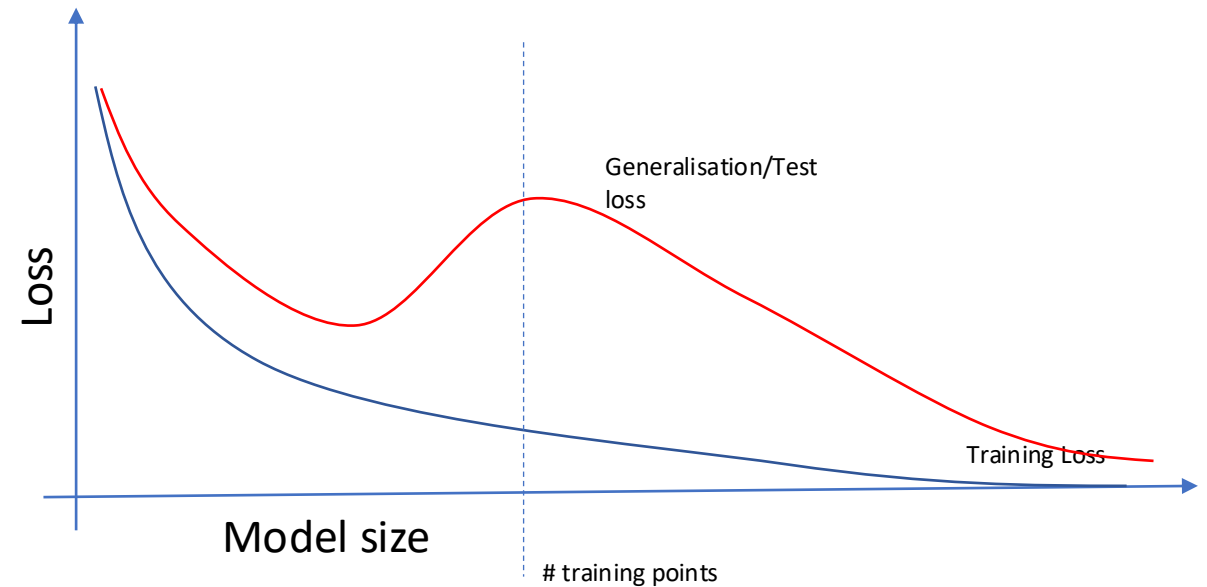
# Overparameterised neural networks

- Large networks with more parameters than training points
- As we increase model size, we expect more overfitting
  - Larger generalization loss
  - Predicted by traditional statistical ML
    - More complex models need more data
  - Observed in traditional models



# Overparameterised neural networks

- Show double descent
- When number of parameters are large
  - Test loss drops again
- Neural networks with large number of parameters
  - Training loss goes to (almost) zero
  - Show surprisingly good generalization
  - Instead of overfitting

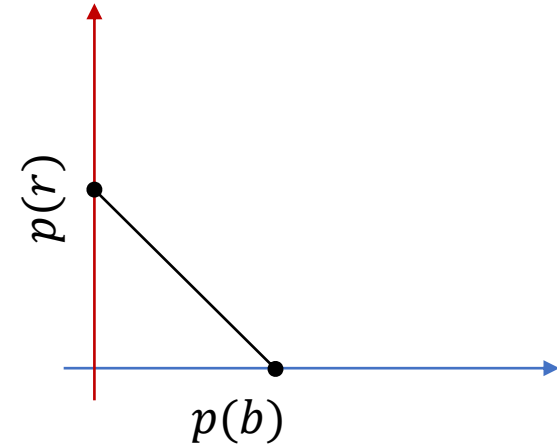


# Neural collapse: Final layers of the network

- Take an overparametrized neural network
- Train for a long time
- In the late layers of the network, the training data forms a few tight clusters
  - One for each class

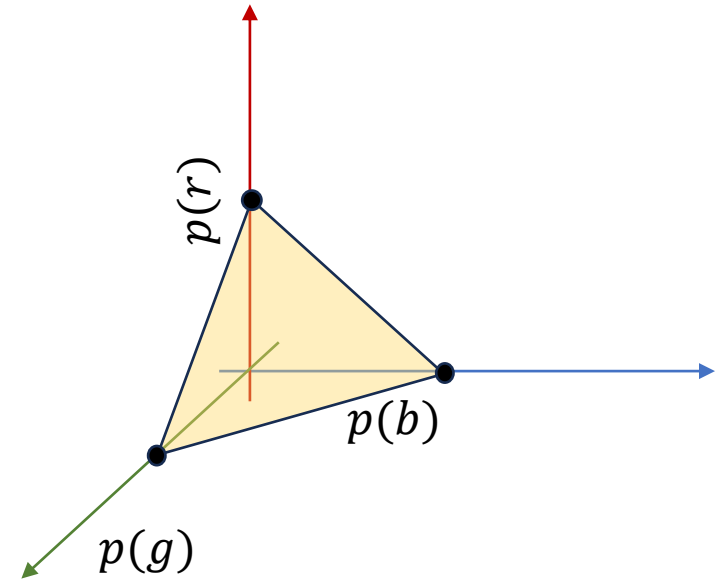
# Softmax output

- Suppose 2 classes are  $r$  and  $b$
- For any data point
  - $p(r) + p(b) = 1$
- Output lies on a line segment
  - Connecting  $(1,0)$  and  $(0,1)$



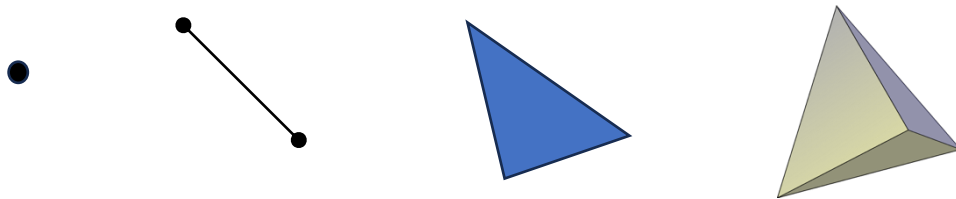
# Higher dimensions

- If there are three classes
- $p(r) + p(b) + p(g) = 1$
- The output lies on an equilateral triangle



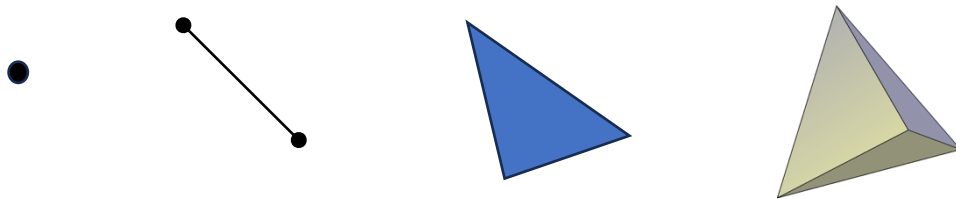
# Definition: $k$ -Simplex

- Take  $k + 1$  points  $u_0, u_1 \dots u_k$  that are affinely independent
  - No point can be expressed as a linear combination of the others
  - Vectors  $u_1 - u_0, \dots u_k - u_0$  are linearly independent
- The  $k$ -dim simplex defined by these points is the set of points:
  - $C = \{a_0u_0 + a_1u_1 + \dots + a_ku_k \mid \sum_{i=0}^k a_i = 1, \text{ and } a_i \geq 0, \text{ for } i = 0, \dots k\}$
- Simplices of dim 0, 1, 2, 3



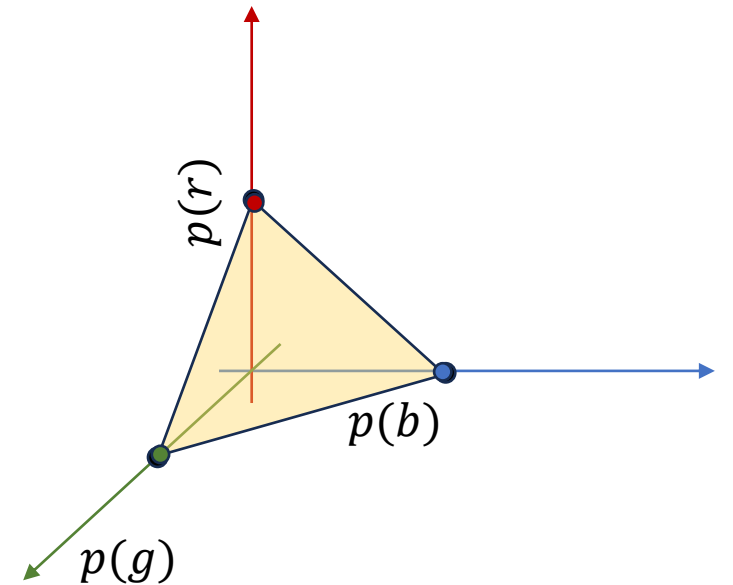
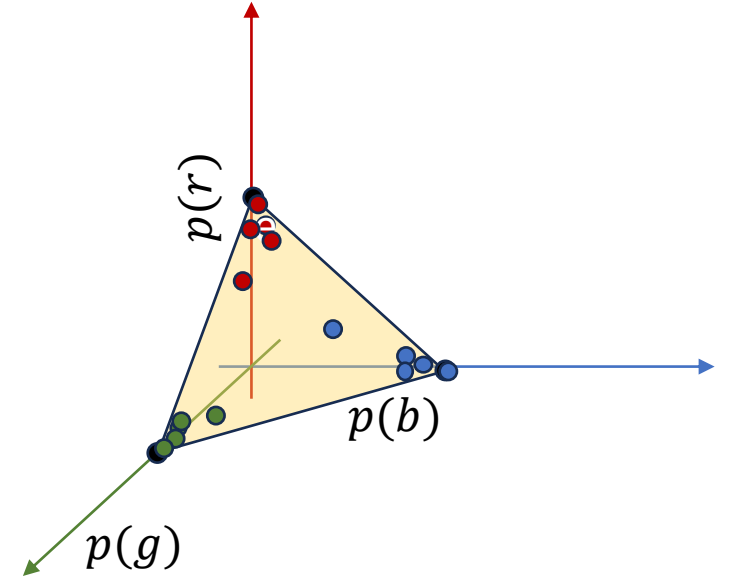
# Simplex

- Observe: a simplex includes the interior
  - Not just the boundary
  - The boundary of a  $k$ -simplex consists of  $k - 1$  simplices
- A regular simplex is a regular polytope
  - All edge lengths are same
- A standard simplex or probability simplex
  - The softmax output
  - Vertices are the standard unit vectors in  $\mathbb{R}^{k+1}$



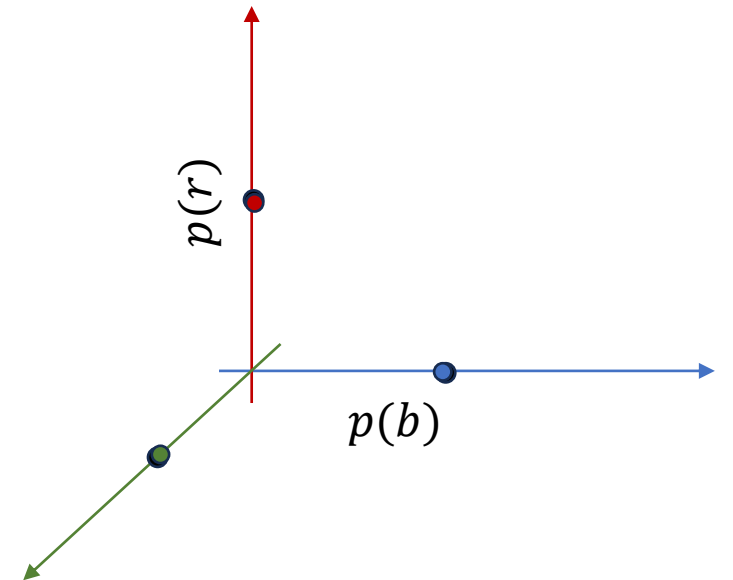
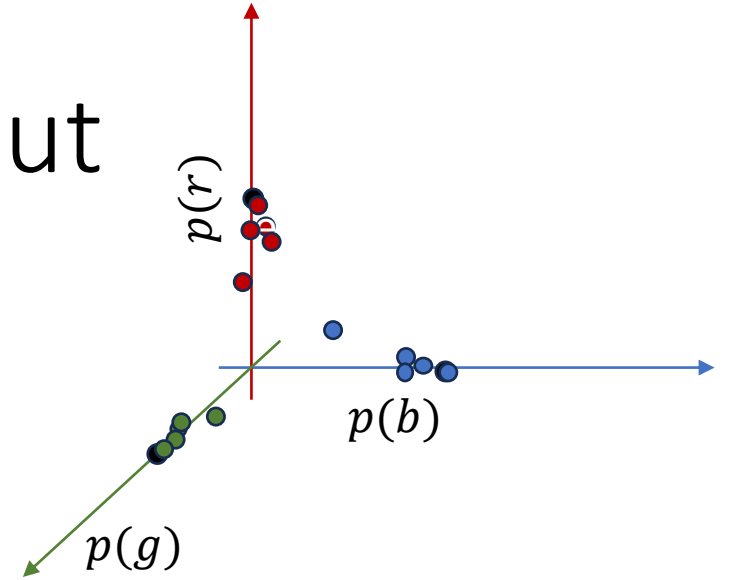
# The softmax output

- For a well trained NN
  - Mostly clustered near vertices
  - Some points a bit far away
- For an overparameterized NN
- Trained to terminal phase of training
  - When training error is almost zero
  - All the training points converge to simplex vertex



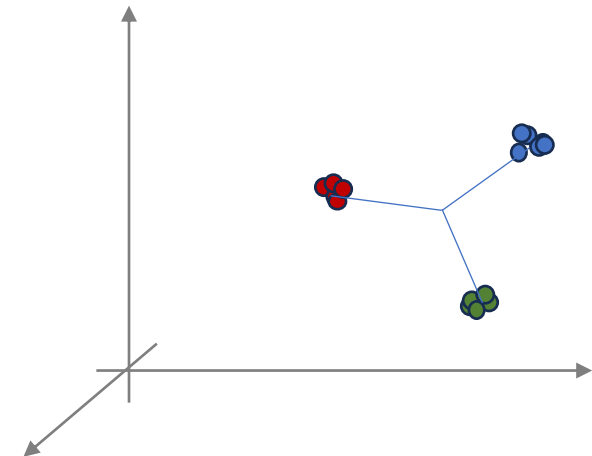
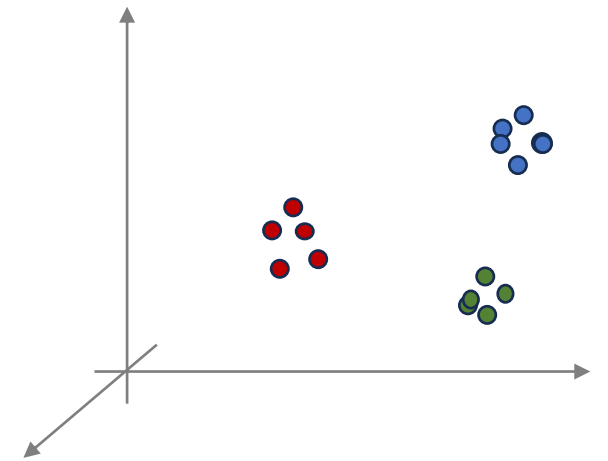
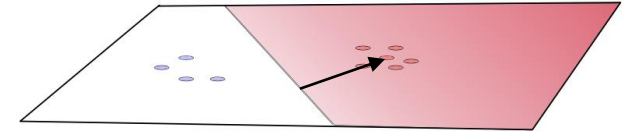
# What happens to the logit output

- Similar clustering
  - But not exactly at the unit vector points
  - They can be larger values
- 
- All red points have to have similar activation on the red neuron
    - Similarly green and blue...



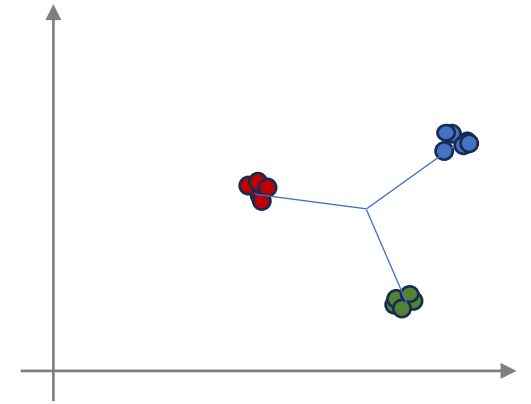
# Penultimate Layer : Layer before the logit layer

- Has many more dimensions
- Each logit neuron is a linear (ReLU) classifier
  - For the corresponding class
- All the red points have to be similar distance from the red boundary
  - To achieve similar red ReLU activation
- Similarly blue and green also form tight clusters for their class
- Converges to the vertices of a regular simplex



# Neural collapse

- Training an overparameterized network
- Beyond terminal phase of training gives
- NC1: Variability collapse
  - Within class variation of activations disappears
- NC2: Converges to Equiangular tight frame (ETF)
  - Simplex, with equal angles from the center between any two classes
- NC3: Converge to self duality
  - The ReLU vector points in direction from center to class mean
- NC4: Simplification to nearest class center
  - Classification comes down to picking the nearest center.



# Neural collapse is a good thing!

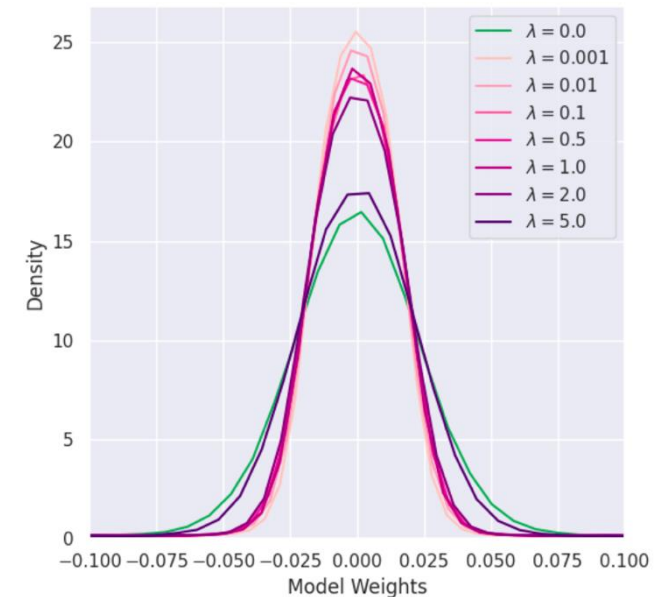
- Complex data reduced to simple clusters through layers of NN
- Simpler representation => simpler classifier models in final layers
  - Better generalization and robustness
- Finds important features
- Useful for interpretability

# Lottery ticket hypothesis

- Hypothesis: A randomly initialised dense NN already contains a sparse subnetwork (a winning ticket) that can give good performance.
  - Sparse means few non-zero edges compared to total no. of edges
- Algorithm to find the winning ticket
  - Initialise a network to random weights
  - Train for some iterations
  - Prune  $p\%$  of edges with small weights
  - Reset the remaining edge to their original random weights
- Works surprisingly well on MNIST, CIFAR with test performance comparable to a well trained network [Frankle and Carbin, 2019]

# Distribution of weights on trained NNs

- A large fraction of weights are close to zero
- Small fraction is far from zero
- Observation:
  - Zero weight edges have no effect – do not conduct information
  - Almost zero weights: Little effect
- Conclusion: While NNs have large number of parameters, after training, many of them have little to no effect!



Histogram of weights

# Pruning

- Idea: take all the edges that are tiny weights, and remove them!
  - These edges are not propagating much information forward
- Observations
  - Can sometimes remove 80% - 90% of edges (e.g. in image classifiers)
  - Retains comparable performance and sometimes better generalization
  - In some other setups (e.g. language models) fewer edges are removed, but still a non-trivial fraction

# Standard pruning methods

- One shot:
  - Train
  - Remove edges with small weights
  - Return to initialization weights of remaining edges
  - Retrain
  - Stop
- Iterative
  - Set random weights
  - Train
  - Remove edges with small weights
  - Start over with initial weights

# Other results

- Theoretical proofs (special cases, few layers etc)
  - [Malach et al. 2020, Bartoldson et al. 2020]
- Pruning and finding winning tickets without data
  - [Wang et al. 2020, Tanaka et al. 2020]

# Question

- If a small network is good enough, why are we using a large one?

# Question

- If a small network is good enough, why are we using a large one?
- Having a large network available makes it easier to find a one of the many sparser networks (and weight assignments) that work well
  - Effectively, exploring many architectures
  - In a smaller network, we are stuck with a smaller range of architectures and have to find exactly the right weights to get good results
- $\mathcal{H}$  is high dimensional, but contains low dimensional subspaces (smaller sets of weights) that can be searched to find good solutions.

# Many variants of pruning

- Magnitude pruning: remove edges with small weights
- Gradient based: Remove parameters that do not contribute to the gradient during training
- Activation based: Remove neurons that do not often have high activation
- Data-free Analysis: Estimate edge importance using network structure (without data)
- Layer pruning: prune only specific layers

# Some comments on overparameterized networks

- Contain sparse subnetworks that work well
- Fits entire training data including noisy elements
  - But usually generalizes well!
- Relevant theory topic:
  - Neural tangent kernels
    - Consider very wide shallow networks and argue that there are reasons for good generalization
- Implicit regularization and stability