
UKRI
Science and Technology Facilities Council
Hartree Centre
Welcome

Science and Technology Facilities Council

Hartree Centre

INTELLECTUAL PROPERTY RIGHTS NOTICE:

The User may only download, make and retain a copy of the materials for their use for non-commercial and research purposes. If you intend to use the materials for secondary teaching purposes it is necessary first to obtain permission.

The User may not commercially use the material, unless a prior written consent by the Licensor has been granted to do so. In any case, the user cannot remove, obscure or modify copyright notices, text acknowledging or other means of identification or disclaimers as they appear.

Science and Technology Facilities Council

Hartree Centre

For further details, please email us: hartreetraining@stfc.a.c.uk

Science and Technology Facilities Council

Hartree Centre

Week 12: Neural Architecture Search

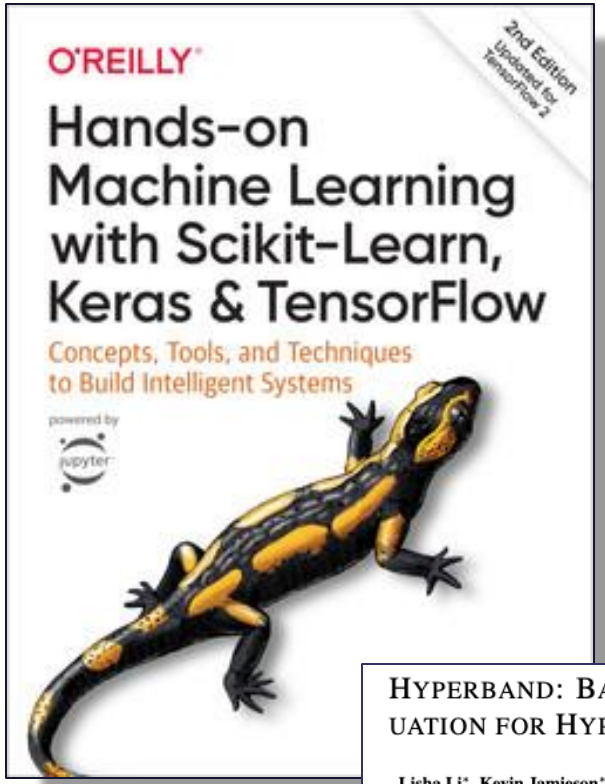
Dom Richards
Artificial Intelligence Group Leader, Hartree Centre

Lecture Outline

- The aim for this lecture is to present some more advanced material, to give you a feel for what's happening at the cutting edge
- We'll be looking at neural networks and neural architecture search
  - It's ***not examinable.***

Recommended Reading

- This lecture is self-contained, but the following might also be useful
  - Hands-on Machine Learning, 2nd Edition 2019
    - Available online from the university library:
    - https://libguides.liverpool.ac.uk/online
    - Chapter 10 discusses neural architecture search
  - Academic paper on Hyperband:
    - Li, Lisha, et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization." The Journal of Machine Learning Research 18.1 (2017): 6765-6816.



HYPERBAND: BANDIT-BASED CONFIGURATION EVALUATION FOR HYPERPARAMETER OPTIMIZATION

Lisha Li*, Kevin Jamieson, Giulia DeSalvo†, Afshin Rostamizadeh‡, and Ameet Talwalkar***
*UCLA, **UC Berkeley, †NYU, and ‡Google
{lishal,ameet}@cs.ucla.edu, kjamieson@berkeley.edu
desalvo@cims.nyu.edu, rostami@google.com

ABSTRACT

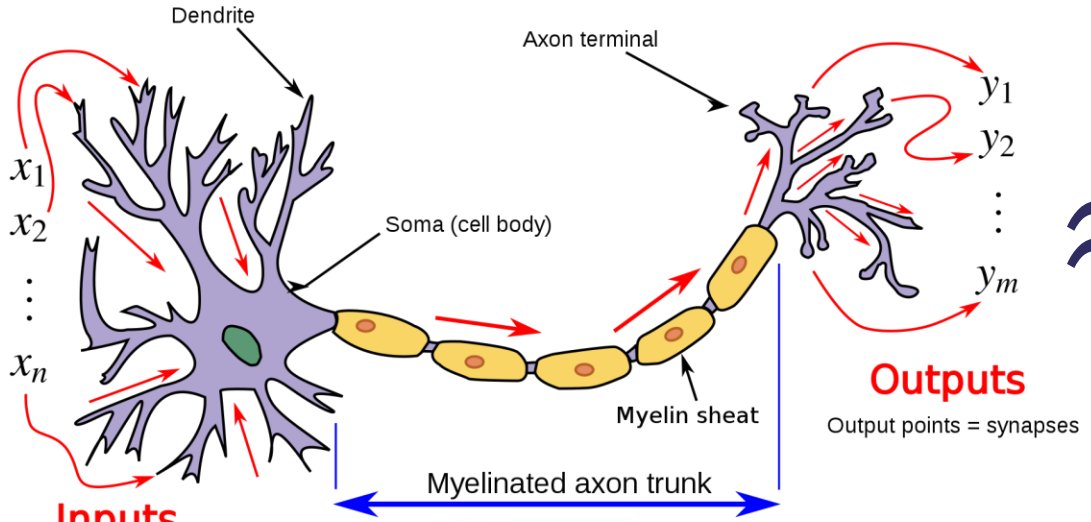
Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While recent approaches use Bayesian Optimization to adaptively select configurations, we focus on speeding up random search through adaptive resource allocation. We present HYPERBAND, a novel algorithm for hyperparameter optimization that is simple, flexible, and theoretically sound. HYPERBAND is a principled early-stoppping method that adaptively allocates a predefined resource, e.g., iterations, data samples or number of features, to randomly sampled configurations. We compare HYPERBAND with popular Bayesian Optimization methods on several hyperparameter optimization problems. We observe that HYPERBAND can provide more than an order of magnitude speedups over competitors on a variety of neural network and kernel-based learning problems.

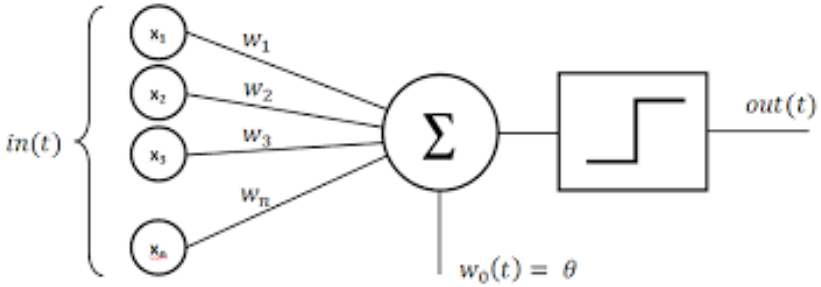
1 INTRODUCTION

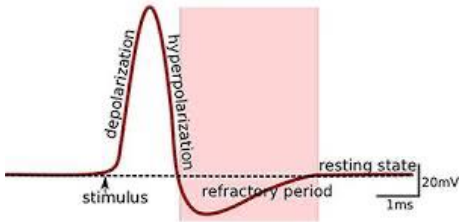
The task of hyperparameter optimization is becoming increasingly important as modern data analysis pipelines grow in complexity. The quality of a predictive model critically depends on its hyperparameter configuration, but it is poorly understood how these hyperparameters interact with each other to affect the quality of the resulting model. Consequently, practitioners often default to either hand-tuning or automated brute-force methods like random search and grid search.

Artificial Neural Networks

- Artificial neural networks are a class of machine learning models that were originally inspired by the structure of the brain
- Neurons receive and emit electrical pulses, and have a nonlinear activation function
- We can create a simple linear mathematical function that mimics this in a very basic way, receiving floating point inputs, weighting them and passing them through a non-linear activation function (such as a step function) to produce a floating point output:



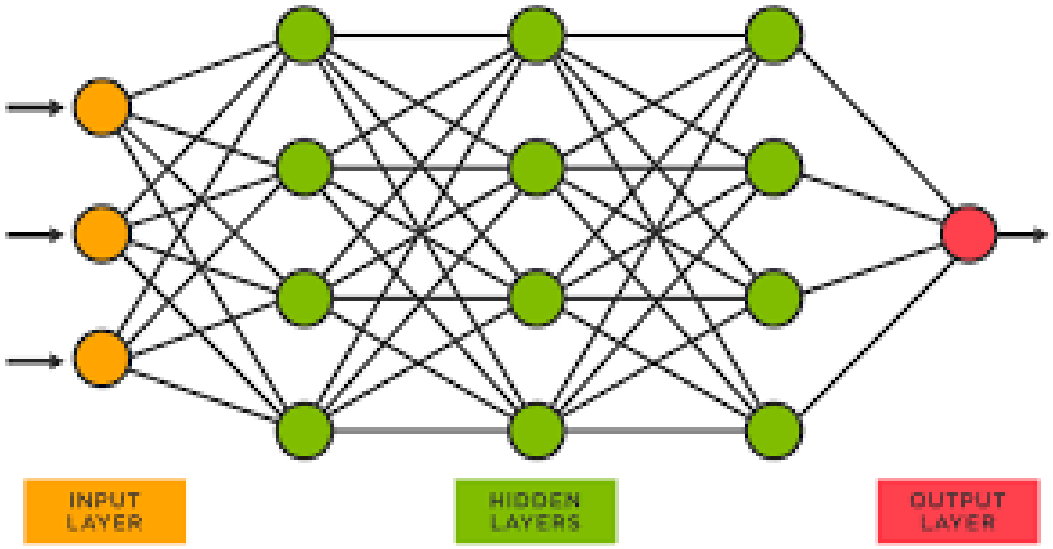




Science and Technology Facilities Council
Hartree Centre

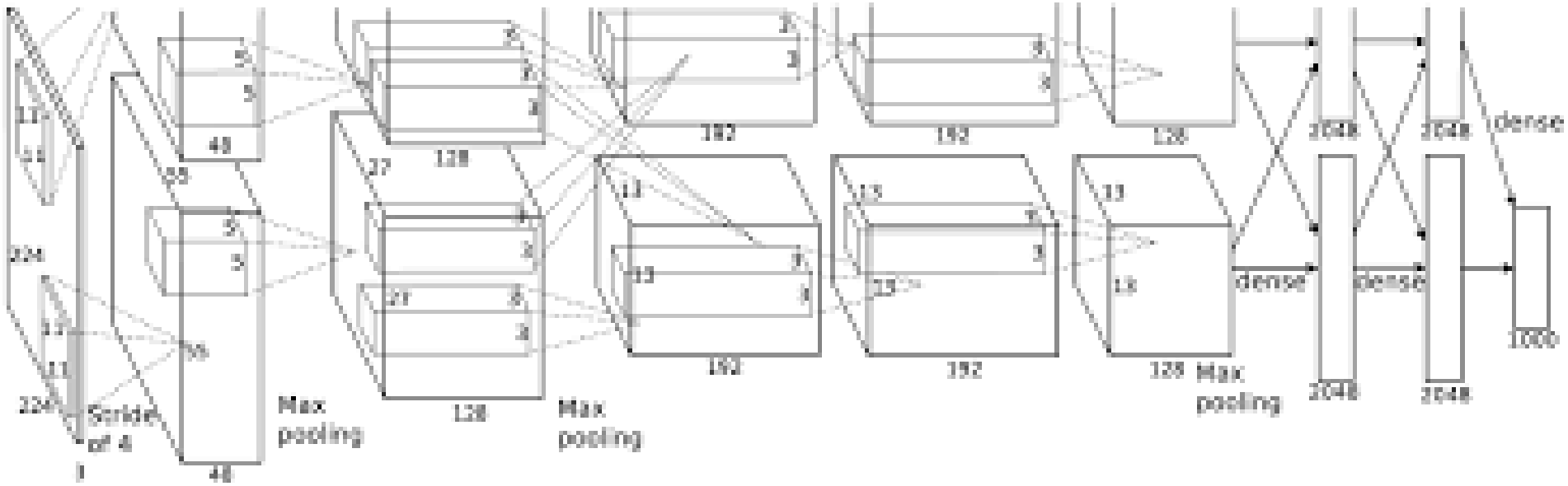
Artificial Neural Networks II

- These artificial neurons can be connected into artificial neural networks
- Artificial neural networks can be used for supervised learning (week 8)
- As with any supervised learning model (e.g. SVMs from week 8) they need to be trained
  - i.e. their parameters need to be adjusted so that the network's output does a good prediction for a given input
- The parameters of a neural network are its weights (see previous slide)



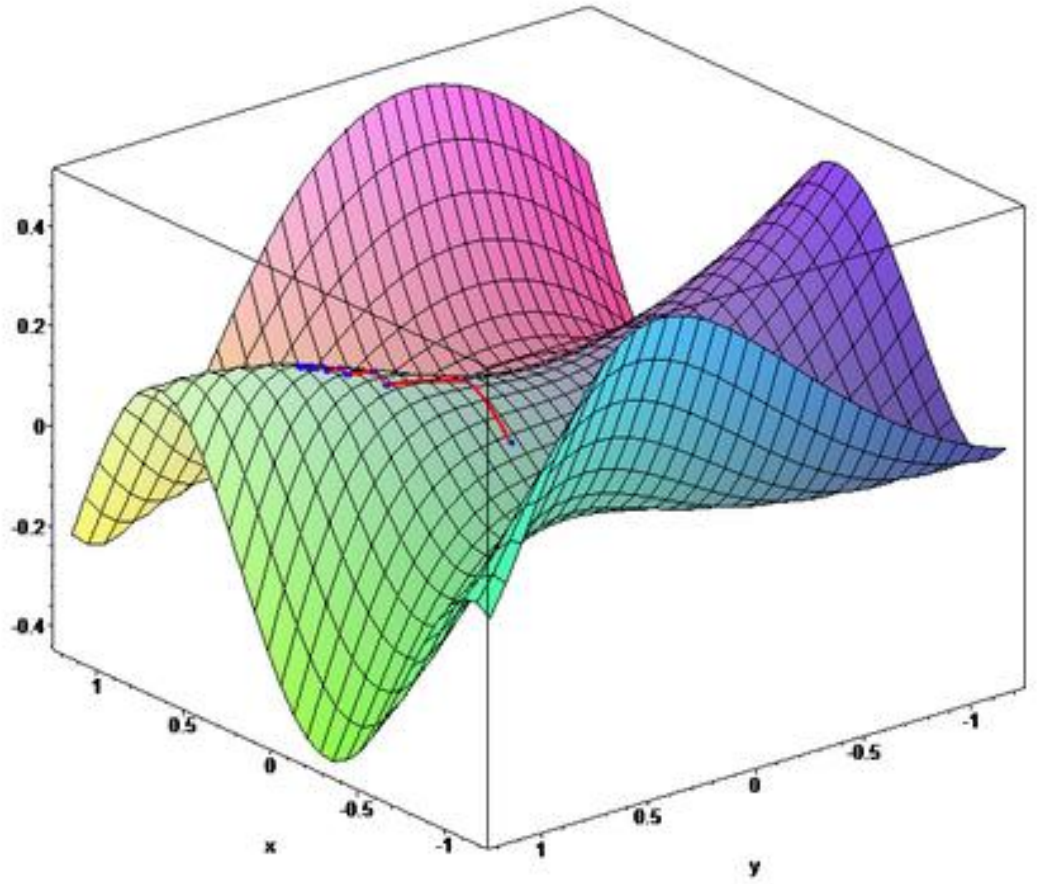
Deep Networks

- Artificial neural network have become very complex over the years.
- E.g. AlexNet (below), which was one of the first well-known ***deep*** artificial neural networks (AKA ***deep networks***), and was published in 2012
- These deep networks are incredibly powerful, and are receiving a ***lot*** of attention in academia and industry at the moment



Parameter Optimisation

- Deep networks often have hundreds of thousands or millions of parameters (i.e. weights)
- Predictive performance is measured with a loss function, which in one way or another measures the difference between actual y values and values predicted by the network given the corresponding observations, X, in a dataset
- The network's parameters are typically trained with *gradient descent*, to minimize the loss
  - E.g. in the plot, choose any starting point and then follow the gradient downhill until you get to a minimum
- Back in 2012, AlexNet took 6 days to train (on 2x Nvidia GTX 580s)
- Nowadays, GPUs are hundreds of times faster
  - RTX 3080 = 285 TFlops FP16 (on tensor cores)
    GTX 580 = 1.6 TFlops FP32
- Now it typically takes minutes to hours to train a neural network

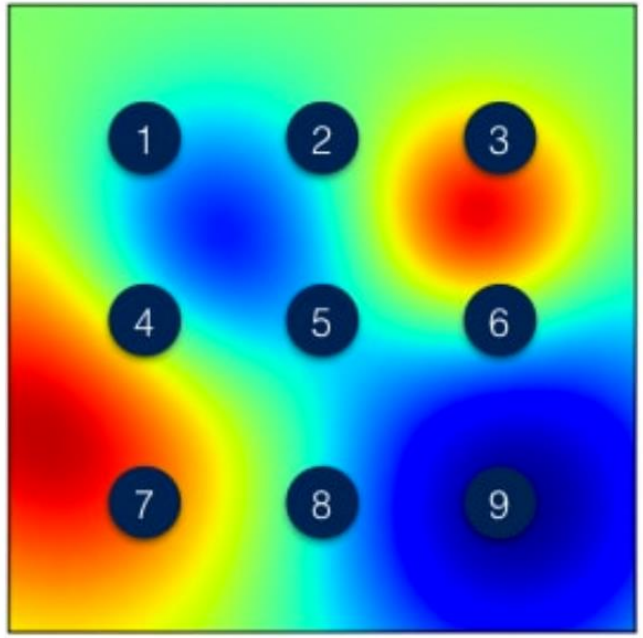


Neural Architecture Search

- So far, we've seen how to optimize a given network for a given prediction task
- One very difficult problem to solve for deep networks (and neural networks in general) is which specific architecture to select for a given task
- In the days of AlexNet, deep nets took days to train. Network architectures were tuned by hand, and this was seen as something of an art form
- Now that we have faster hardware, deep nets often take minutes or hours to train, and we can automate the task of neural architecture search:
  - The sizes and number of different functional blocks in the deep network are represented by ***hyperparameters***
  - Neural architecture search algorithms efficiently search the ***hyperparameter space*** for the best deep network
- We'll now see a few neural architecture search algorithms, starting off simple and culminating with a state-of-the-art algorithm called Hyperband

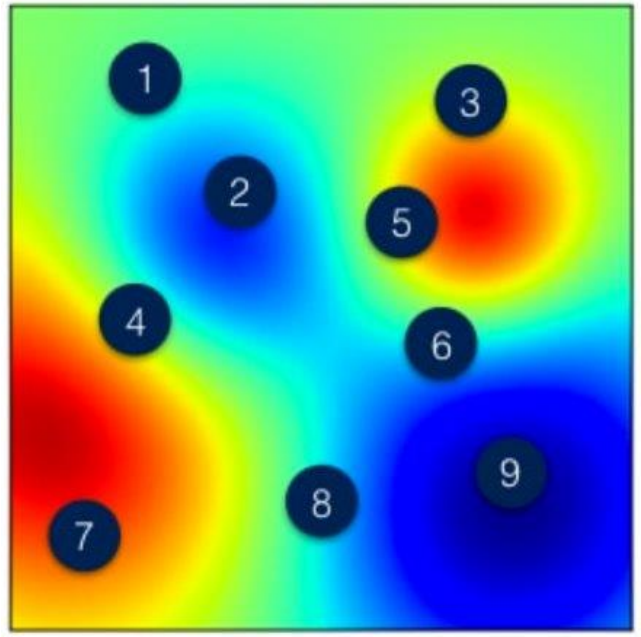
Grid Search

- Choose a hypercube of evenly-spaced points in the hyperparameter search space
- Each of these points corresponds to a different deep network architecture
- Different regions of the hyperparameter space correspond to different losses, as shown in the figure below
- We train each of the networks and find evaluate its loss function, then pick the network with the lowest loss
- Grid search works well for low-dimensional hyperparameter spaces, but struggles with high dimensional spaces because too many points are needed in the grid
- In the plot (from the Hyperband paper):
  - We have 2 hyperparameters => 2 dimensions in the hyperparameter space
  - The heatmap corresponds to losses for the deep network which is trained with a given set of hyperparameters
  - The red areas are more preferable (i.e. have the lowest losses)
  - The black circles represent the samples
  - The numbers represent the order in which they were chosen
  - So, the best choice of hyperparameters will be either 3 or 7



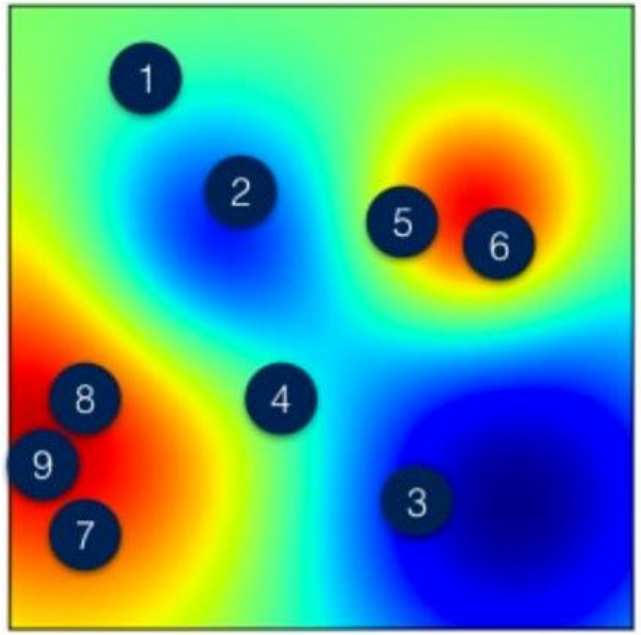
Random Search

- Random search randomly picks points in the hyperparameter space, then trains the corresponding deep networks and picks the one with the lowest loss
- It explores the space less evenly than grid search, but is more tractable for high-dimensional hyperparameter spaces
- In the plot, the best choice of hyperparameters will be 7



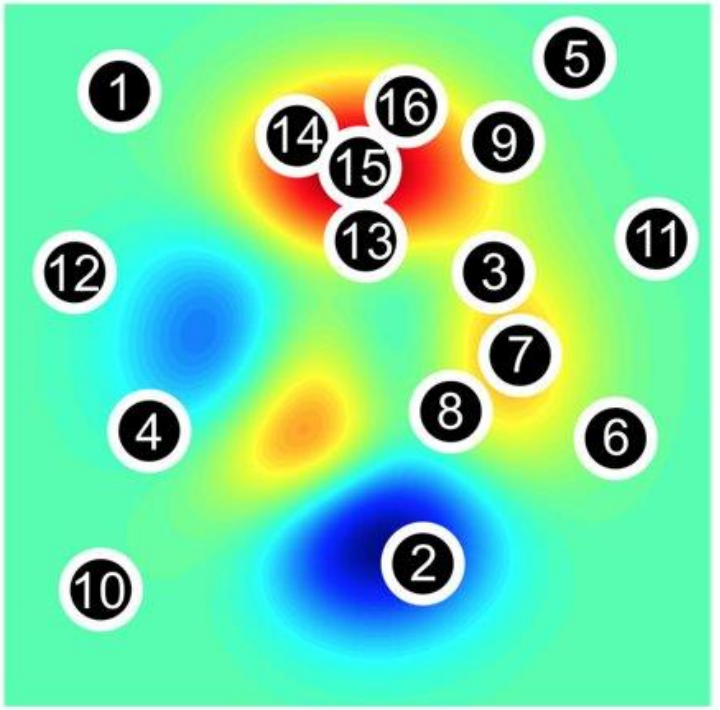
Adaptive Selection

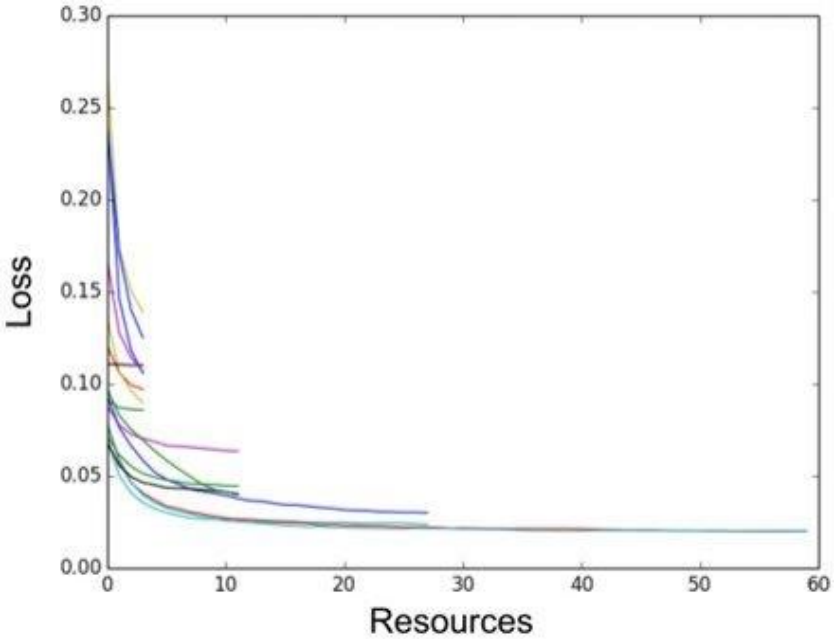
- Adaptive selection is an extension of random search
- It first performs a coarse random search, and then chooses more points around the best points in the original coarse search
- This works well if the space is small enough that the initial coarse search can provide reasonable coverage
- In the plot:
  - The red areas are more preferable (i.e. have the lowest losses)
  - The black circles represent the samples
  - The numbers represent the order in which they were chosen
  - Note that the later samples are focused on the red regions
  - The best choice of hyperparameters will be 9



Successive Halving

- Successive halving is another extension of random search
- It first makes large number of random hyperparameter choices, and *partially trains* them
- Then throw away most of them, based on the losses after this partial training
- It then trains the remainder some more, then throws most of those away based on their losses
- Continue until you're left with one model
- In the original paper (below), half of the models were discarded after each iteration, and hence the name "successive halving"
  - However, later work has suggested that throwing away 2/3 is a better idea
- In the plot:
  - Again we're trying to find the red areas of the heatmap
  - The numbers show the order in which points were discarded
  - Note that 13 – 16 are all in the most promising region of the hyperparameter space





Jamieson, Kevin, and Ameet Talwalkar. "Non-stochastic best arm identification and hyperparameter optimization." Artificial Intelligence and Statistics. PMLR, 2016.

Successive Halving Algorithm

- The successive halving algorithm is listed below
- There are 3 subroutines:
  - GetHyperparameterConfiguration(n) randomly picks n sets of hyperparameters
  - RunThenReturnValLoss(t, r) trains deep net t for r epochs and returns the (validation) loss
  - TopK(T, L, n) keeps the best n models from T given their losses L
- R (for resource) is the number of epochs for which the best model should be trained

Algorithm 1: Successive Halving

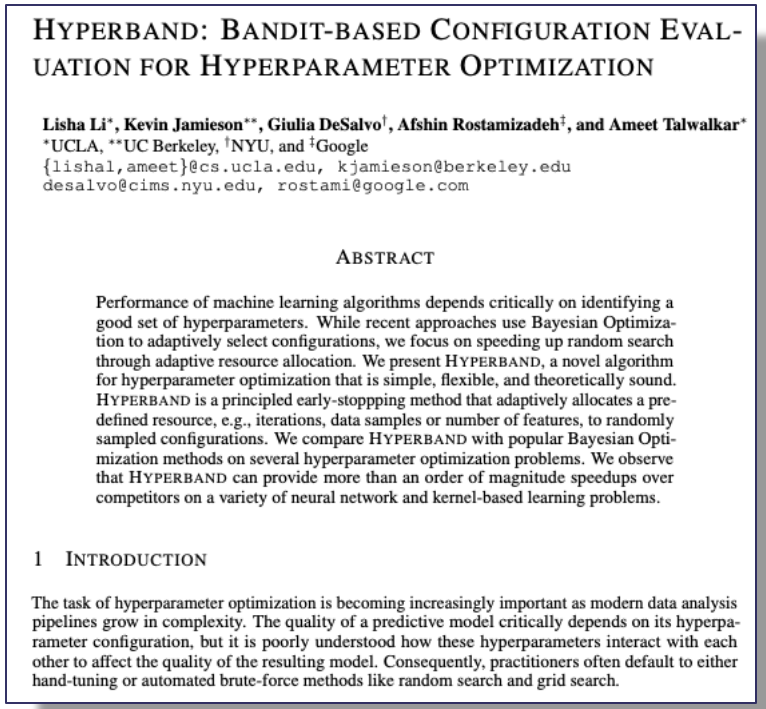
input : s, η (default $\eta = 3$)
initialise: $R = \eta^s$
begin
  $T = \text{GetHyperparameterConfiguration}(R)$
  for $i \in \{0, \ldots, s\}$ **do**
    $n_i = \eta^{s-i}$
    $r_i = \eta^i$
    $L = \{\text{RunThenReturnValLoss}(t, r_i) : t \in T\}$
    $T = \text{TopK}(T, L, \frac{n_i}{\eta})$
  end
  return configuration with the smallest intermediate loss seen so far.
end

Successive Halving Improvements

- The successive halving algorithm listed in the previous slide:
  - Initially trains $\eta^s = R$ models for 1 epoch each
  - Then trains η^{s-1}models for η epochs each
  - …
  - And finally trains a single model for $\eta^s = R$ epochs
- So, we throw away most of the initial models after training for just one epoch
- Would it be better to train all the initial models for more epochs before starting to throw models away?
- If so, what would be the best number of initial epochs?
- It's hard to know the best number so the next algorithm, Hyperband, tries all feasible numbers instead…

Hyperband

- Hyperband is an extension of successive halving
- It adds an outer loop around successive halving, and calls into it with different numbers of initial training epochs before we start throwing networks away
- We have removed the requirement that $R = \eta^s$ which complicates the algorithm a bit



HYPERBAND: BANDIT-BASED CONFIGURATION EVALUATION FOR HYPERPARAMETER OPTIMIZATION

Lisha Li*, **Kevin Jamieson****, **Giulia DeSalvo**†**, **Afshin Rostamizadeh**‡**, and **Ameet Talwalkar***
*UCLA, **UC Berkeley, †NYU, and ‡Google
{lishal,ameet}@cs.ucla.edu, kjamieson@berkeley.edu
desalvo@cims.nyu.edu, rostami@google.com

ABSTRACT

Performance of machine learning algorithms depends critically on identifying a good set of hyperparameters. While recent approaches use Bayesian Optimization to adaptively select configurations, we focus on speeding up random search through adaptive resource allocation. We present HYPERBAND, a novel algorithm for hyperparameter optimization that is simple, flexible, and theoretically sound. HYPERBAND is a principled early-stoppping method that adaptively allocates a predefined resource, e.g., iterations, data samples or number of features, to randomly sampled configurations. We compare HYPERBAND with popular Bayesian Optimization methods on several hyperparameter optimization problems. We observe that HYPERBAND can provide more than an order of magnitude speedups over competitors on a variety of neural network and kernel-based learning problems.

1 INTRODUCTION

The task of hyperparameter optimization is becoming increasingly important as modern data analysis pipelines grow in complexity. The quality of a predictive model critically depends on its hyperparameter configuration, but it is poorly understood how these hyperparameters interact with each other to affect the quality of the resulting model. Consequently, practitioners often default to either hand-tuning or automated brute-force methods like random search and grid search.

Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.

input : R, η (default $\eta = 3$)
initialization : $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$

for $s \in \{s_{\max}, s_{\max} - 1, \ldots, 0\}$ **do**
  $n = \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil$, $r = R\eta^{-s}$
  // begin SUCCESSIVEHALVING with (n, r) inner loop
  $T =$ `get_hyperparameter_configuration`(n)
  for $i \in \{0, \ldots, s\}$ **do**
    $n_i = \lfloor n\eta^{-i} \rfloor$
    $r_i = r\eta^i$
    $L = \{$`run_then_return_val_loss`$(t, r_i) : t \in T\}$
    $T =$ `top_k`$(T, L, \lfloor n_i/\eta \rfloor)$
  end
end
return *Configuration with the smallest intermediate loss seen so far.*

Summary of Previous Lectures

- Clustering (week 6):
  - Hierarchical Agglomerative Clustering
  - K-Means
  - DBSCAN
- Classification (week 8):
  - K-Nearest Neighbors
  - Linear Support Vector Classifiers
  - SVM
- Text analysis:
  - Used PCA, clustering and classification for text-based machine learning problems
  - Vector space representations of text documents:
    - Bag of words
    - Term log-frequency
    - Term frequency, inverse document frequency (TF-IDF)
  - Applications:
    - Latent semantic analysis
    - Clustering for search
    - Document classification

UKRI Science and Technology Facilities Council

Hartree Centre

Questions?

Science and Technology Facilities Council

Hartree Centre

Thank you