Lecture 6 - AutoML

Resources:

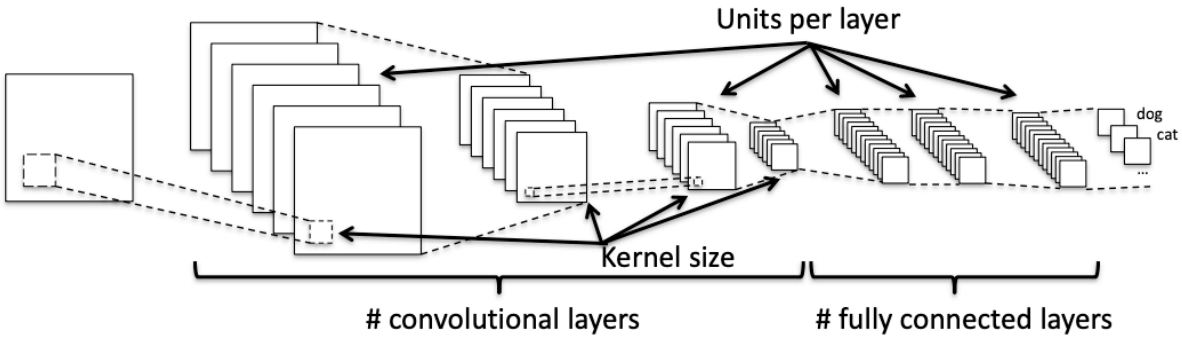
- Automl book: https://www.automl.org/book/

Section 1. Motivation

Success of deep learning; we have deployments everywhere and successes in different domains/applications

However, there is a big problem: **The performance of deep learning models is very sensitive to many hyper-parameters!**

- Architectural hyper parameters



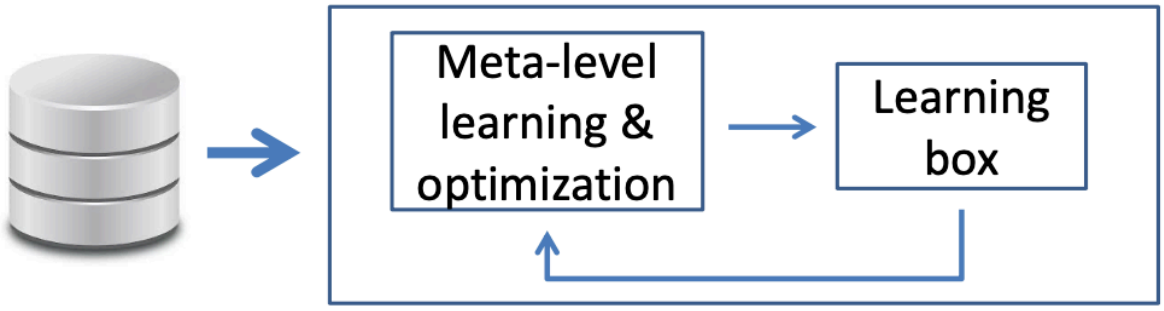
- Optimization algorithm, learning rates, momentum, batch normalization, batch sizes, dropout rates, weight decay, data augmentation….

Easily 20-50 design decisions

Section 2. Goal of AutoML

The current deep learning practice requires the expert to choose the architecture and the hyper parameters, train the deep learning model end-to-end and then iterate

The promise of AutoML is **true end-to-end learning**



The learning box can contain multiple ML-related operations:

- Clean and preprocess the data
- Select/engineer better features
- Select the model family
- Set the hyper parameters
- Construct ensembles of models
- ....

Section 3. Modern Hyperparameter Optimization

Section 3.1 AutoML as Hyperparameter Optimization

What is hyper parameter optimization?

Definition: Hyperparameter Optimization (HPO)

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of a ML algorithm A with domain $\boldsymbol{\Lambda}$,
- $\mathcal{L}(A_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of A, using hyperparameters $\boldsymbol{\lambda}$ trained on D_{train} and evaluated on D_{valid}.

The hyperparameter optimization (HPO) problem is to find a hyperparameter configuration $\boldsymbol{\lambda}^*$ that minimizes this loss:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \mathcal{L}(A_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

We have different types of hyper parameters: continuous (learning rate), discrete (number of unites), categorical (finite domain, e.g., algorithm, activation functions etc)

We also have conditional hyperparameters they are activated only given another hyperpameter value.

Example 1:
A = choice of optimizer (Adam or SGD)
B = Adam's second momentum hyperparameter (only active if A=Adam)

Example 2:
A = choice of classifier (RF or SVM)
B = SVM's kernel parameter (only active if A = SVM)

Definition: Combined Algorithm Selection and Hyperparameter Optimization (CASH)

Let

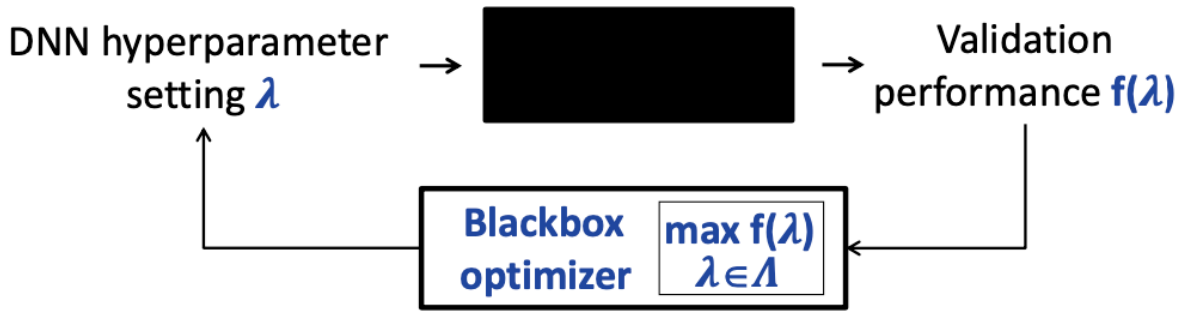
- $\mathcal{A} = \{A^{(1)}, \ldots, A^{(n)}\}$ be a set of algorithms
- $\mathbf{\Lambda}^{(i)}$ denote the hyperparameter space of $A^{(i)}$, for $i = 1, \ldots, n$
- $\mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of $A^{(i)}$, using $\lambda \in \mathbf{\Lambda}^{(i)}$ trained on D_{train} and evaluated on D_{valid}.

The Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem is to find a combination of algorithm $A^* = A^{(i)}$ and hyperparameter configuration $\boldsymbol{\lambda}^* \in \mathbf{\Lambda}^{(i)}$ that minimizes this loss:

$$A^*_{\boldsymbol{\lambda}^*} \in \underset{A^{(i)} \in \mathcal{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}^{(i)}}{\arg\min} \mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

Top-level hyperparameter and then all other hyperparamters are conditional hyperparamters.

Section 3.2 Blackbox optimization



Evaluating the function is expensive (remember our previous discussions on model evaluation in the last class).

Methods for blackbox optimization:

1. ***Grid Search***

2. ***Random Search***

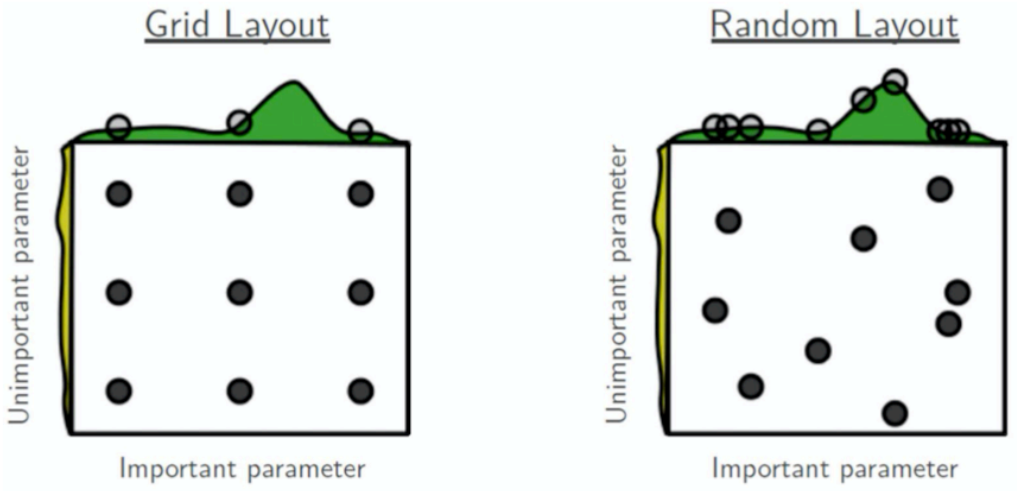


Image source: Bergstra & Bengio, JMLR 2012

3. ***Bayesian optimization***

- **Approach**
  - Fit a proabilistic model to the function evaluations $\langle \lambda, f(\lambda) \rangle$
  - Use that model to trade off exploration vs. exploitation

- **Popular since** Mockus [1974]
  - Sample-efficient
  - Works when objective is nonconvex, noisy, has unknown derivatives, etc
  - Recent convergence results [Srinivas et al, 2010; Bull 2011; de Freitas et al, 2012; Kawaguchi et al, 2016]

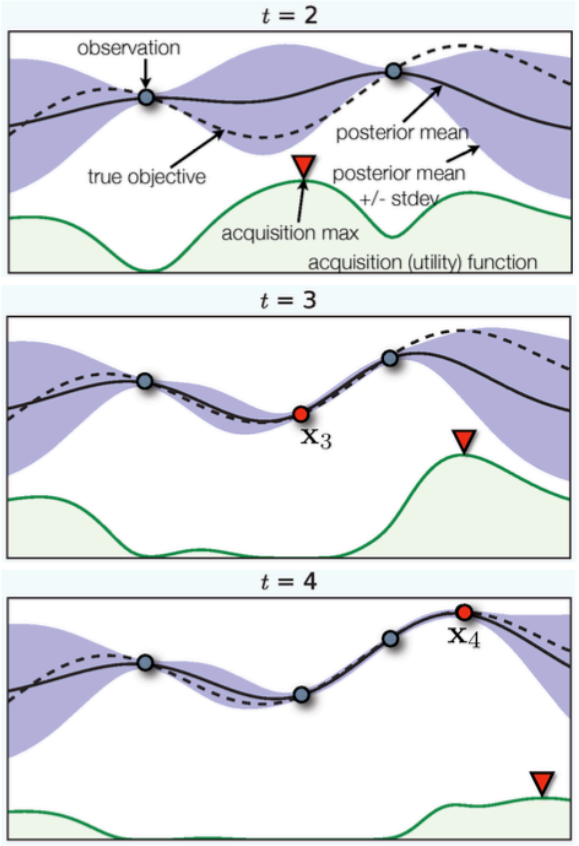


Image source: Brochu et al, 2010

See lecture notes for details

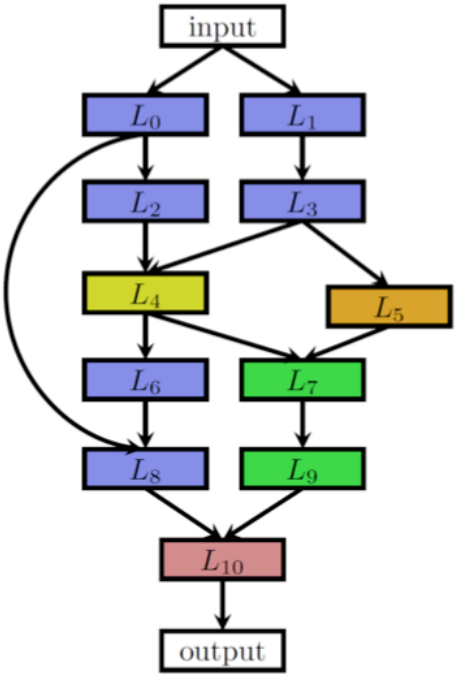
Bayesian optimization can be very slow. Different methods to speed things up!

Section 3.3. Beyond Hyperparameter Optimization

See:
https://media.neurips.cc/Conferences/NIPS2018/Slides/hutter-vanschoren-part1-2.pdf

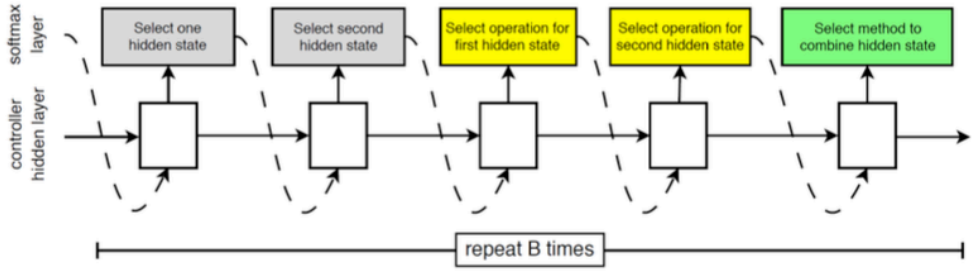
Section 4. Neural Architecture Search Spaces

Chain-structured space
(different colours:
different layer types)



More complex space
with multiple branches
and skip connections

- Cell search space by Zoph et al [CVPR 2018]

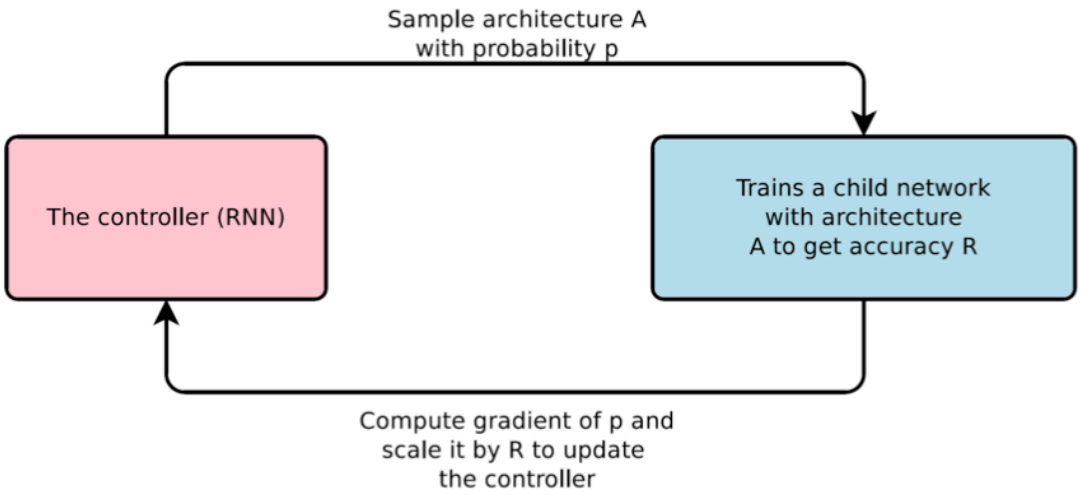


  - 5 categorical choices for Nth block:
    - 2 categorical choices of hidden states, each with domain {0, ..., N-1}
    - 2 categorical choices of operations
    - 1 categorical choice of combination method

    → Total number of hyperparameters for the cell: 5B (with B=5 by default)

- Unrestricted search space
  - Possible with conditional hyperparameters (but only up to a prespecified maximum number of layers)
  - Example: chain-structured search space
    - Top-level hyperparameter: number of layers L
    - Hyperparameters of layer k conditional on L >= k

Use of reinforcement learning combined with recurrent neural networks to learn a better model



Read work: https://arxiv.org/pdf/1611.01578.pdf
http://rll.berkeley.edu/deeprlcoursesp17/docs/quoc_barret.pdf

In Neural Architecture Search, we use a controller to generate architectural hyperparameters of neural networks. To be flexible, the controller is implemented as a recurrent neural network. Let's suppose we would like to predict feedforward neural networks with only convolutional layers, we can use the controller to generate their hyperparameters as a sequence of tokens:

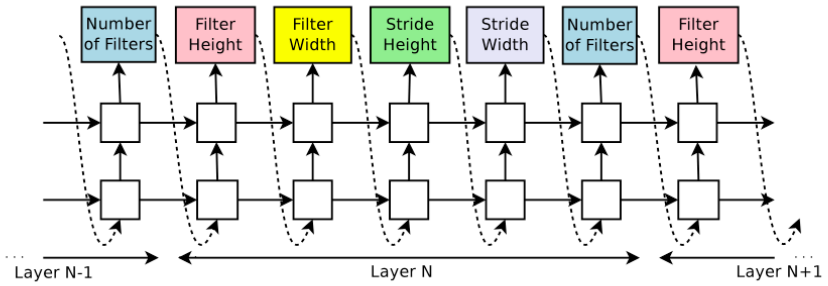


Figure 2: How our controller recurrent neural network samples a simple convolutional network. It predicts filter height, filter width, stride height, stride width, and number of filters for one layer and repeats. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.