

Bayesian multimodeling

Spring 2025

Topics in this term

- Hyperparameter optimization
- Structure selection
- Meta-optimization
- Knowledge transfer
- Multi-task learning
- SSM models
- Model ensembles and hierarchical models
- Latent space projection
- NTK
- Bayesian RL agents

Projects

Model selection: coherent inference

First level: select optimal parameters:

$$w = \arg \max \frac{p(\mathcal{D}|w)p(w|h)}{p(\mathcal{D}|h)},$$

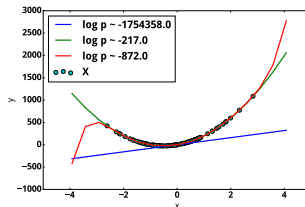
Second level: select optimal model (hyperparameters).

Evidence:

$$p(\mathcal{D}|h) = \int_w p(\mathcal{D}|w)p(w|h)dw.$$



Model selection scheme



Example: polynomials

Hyperparameters

Definition

Prior for parameters w and structure Γ of the model f is a distribution $p(W, \Gamma | h) : \mathbb{W} \times \Gamma \times \mathbb{H} \rightarrow \mathbb{R}^+$, where \mathbb{W} is a parameter space, Γ is a structure space.

Definition

Hyperparameters $h \in \mathbb{H}$ of the models are the parameters of $p(w, \Gamma | h)$ (parameters of prior f).

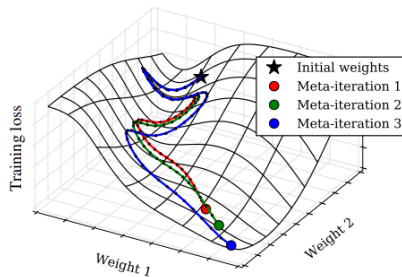
Gradient methods

Idea: Optimize hyperparameters using the full parameter optimization trajectory.

Pros:

- Hyperparameter optimization will consider the features of the parameter optimization.
- The complexity is linear on the number of hyperparameters.

Cons: the computational cost is very expensive. Also can be problems with numerical precision of reverse method.



Maclaurin et. al, 2015. Example.

Project: Gradient-based hyperparameter optimization

Hyperparameter optimization is a problem of finding suitable hyperparameters given a validation (or sometimes test) dataset. In contrast to classical hyperparameter optimization methods, gradient-based methods allow the researchers to perform hyperparameter optimization over a billion-dimension search space.

Recommended stack: JAX or pytorch. As with Evidence-based operators, the main problem here is to make this usable for a broad-class of tasks (provide good API).

The code of 2 algorithms is already implemented on JAX by Konstantin Yakovlev.

Knowledge distillation [Hinton et al., 2015]

Given a teacher model. Knowledge distillation is a transfer of the knowledge from the teacher model to a small model that is more suitable for deployment.



Teacher model

Information



Student model


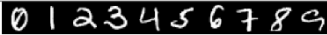










Hinton distillation

$$\lambda \log p(y | \mathbf{SM}(f_{\text{student}}(X) / T)) + (1 - \lambda) \log p(\mathbf{SM}(f_{\text{teacher}} / T) | \mathbf{SM}(f_{\text{student}}(X) / T)),$$

where f_{student} and f_{teacher} are student and teacher outputs, T is a temperature:

- When $T \rightarrow \infty$, we get uniform distribution;
- With T is high enough and outputs are zero-mean we get l_2 minimization of distance between student and teacher logits.
- When $T \rightarrow 0$, we get one-hot labels from teacher.

Distillation without dataset (Lopes et al., 2017)

Activation Record	Means	Randomly sampled example
MNIST		
Top Layer Statistics		
All Layers Statistics		
All Layers + Dropout		
Spectral All Layers		
Spectral Layer Pairs		

Project: Data-free distillation

Knowledge distillation is an approach to train/improve target model performance (student) using information from already trained large model (teacher). Classical approaches to this problem perform distillation using logits, responses or hidden state from teacher obtained from data. However in some cases we cannot use the original data, and thus this method becomes unapplicable. The idea of data-free distillation is to obtain some elements of data from the teacher and then perform distillation.

Some mechanism of analyzing hidden states must be implemented (in pytorch this can be implemented using hooks or via computational graph inspection, for example).

State-space models

An n -th order linear State space model (SSM) has the form:

$$\dot{x}(t) = Az(t) + Bu(t),$$

$$y(t) = Cz(t) + Du(t),$$

here $z \in \mathbb{R}^n$ is a state, u is a control variable.

Why this model is important?

- Can describe the dynamics of n -th order linear ODE (!)
- Allows us to use methods from DE in usual machine learning way.

Kalman filter

$$x_t = Az_t + \epsilon_x,$$

$$z(t) = Cz_{t-1} + \epsilon_z,$$

where ϵ is Gaussian noise.

To compute posterior use Bayes rule:

$$p(z_t|x_t) \propto p(x_t|z_t, x_{t-1})p(z_t|x_{t-1}).$$

Model parameter optimization can be done in different ways (the most simple: using EM algorithm).

Kalman filter: similar models and extensions

$$x_t = Az_t + \epsilon_x,$$

$$z(t) = Cz_{t-1} + \epsilon_z,$$

- Linear RNN (or just SSM): Kalman filter without noise;
- HMM: discrete state.
- Extended Kalman filter: nonlinear KF using Taylor series.
- Also: we can approximate posterior with NN to make the model non-linear.
- **KF is a special case of Gaussian process**

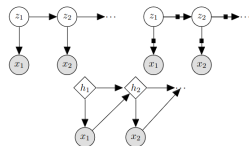


Figure 1: **Generative Models of Sequential Data:** (Top Left) Hidden Markov Model (HMM), (Top Right) Deep Markov Model (DMM) ■ denotes the neural networks used in DMMs for the emission and transition functions, (Bottom) Recurrent Neural Network (RNN), ◇ denotes a deterministic intermediate representation. Code for learning DMMs and reproducing our results may be found at: github.com/clinicalml/structuredinference

(Krishnan et al., 2016)

Project: Kalman filter and extensions

Surprisingly, this class of models is underrepresented in DL community. This project is focused on the very simple and clear implementation of this class of models.

It would be good to make the code somehow compatible with S4 or other current SSM SOTAs.

Neural ensemble search

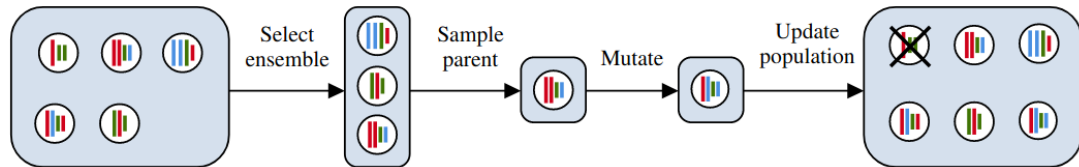


Figure 3: Illustration of one iteration of NES-RE. Network architectures are represented as colored bars of different lengths illustrating different layers and widths. Starting with the current population, ensemble selection is applied to select parent candidates, among which one is sampled as the parent. A mutated copy of the parent is added to the population, and the oldest member is removed.

Project: Neural ensemble search

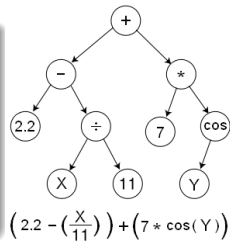
Neural ensemble search is a special case neural architecture search allowing to select not a single model but an ensembles of models. The project goal is to implement baseline algorithms for this problem as well as methods of uncertainty estimation.

The aleatoric/epistemic uncertainty estimation must be implemented. It would be also good to make the code compatible with nni. The code itself will obviously require usage of some GPU resources (collab is ok).

Symbolic models: recap

Definition, Koza

The problem of **symbolic function identification** is developing a composition of terminals and functions that can return the correct value of the function after seeing a finite sampling of combinations of the independent variable associated with the correct value of the dependent variable

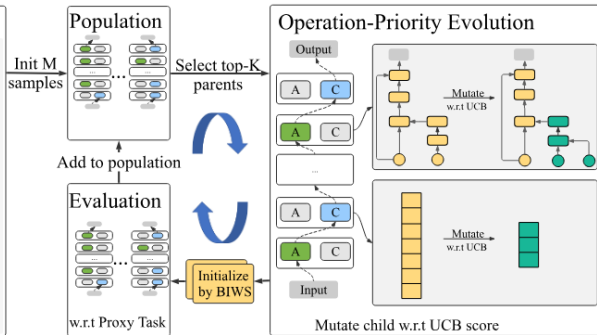
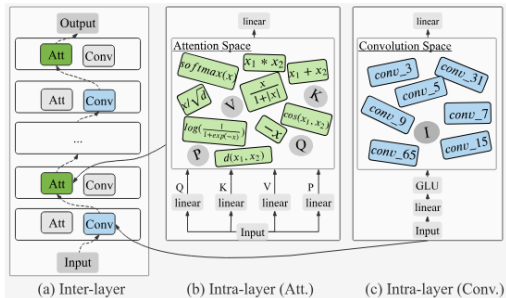


Definition, wikipedia

Symbolic regression is a type of regression analysis that searches the space of mathematical expressions to find the model that best fits a given dataset, both in terms of accuracy and simplicity.

Expression tree as it can be used in symbolic regression to represent a function

Search Space



Project: AutoML-Zero for deep learning

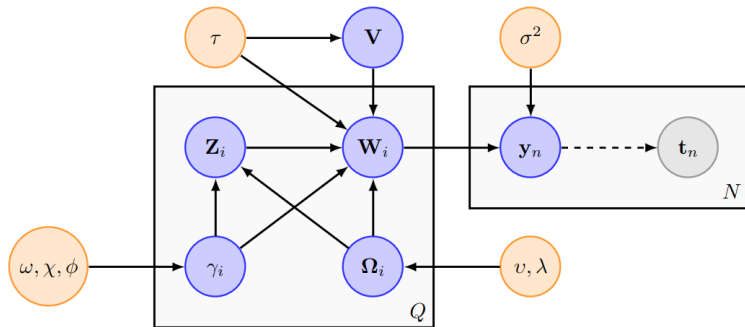
Original paper on automl-zero involves symbolic programming for machine learning algorithms. Current extension of this idea led to multiple heterogenous papers devoted to optimization of neural architecture search, loss function optimization and optimizer optimization. The goal of the project is to consolidate all the results.

The project requires a lot of computation, so some GPU could be good (working with collab might be challenging). Note that AutoBERT uses proxy functions to speedup optimization (this might be implemented in the project). For our purposes we don't need to implement some intensive architecture search, obtaining some small model on MNIST/FashionMNIST/CIFAR will be ok (but this will require some thinkin on search space).

Sparse Bayesian Multi-Task Learning, 2011

$$y_n = Wx_n + \mu + \varepsilon_n;$$

$$\varepsilon_n \sim \mathcal{N}(0, \Sigma).$$



Project: Multi-task learning with task relation

Multi-task learning is a machine learning paradigm which involves optimization model parameters for multiple diverse tasks. Vanilla multitask optimization presumes that the tasks are optimized without any hierarchy, but with some possible weights over tasks. The project goal is to implement different methods to assign weights or hierarchy to tasks to make the optimization more effective.

Attention: some methods like "sparse Bayesian MT learning" are poorly compatible with DL models. Some extension must be proposed.

Hometask

Fill the google form with project preferences.

Deadline: next Monday (!).

Talks for the next classes are available.