... in 30 sec

- Deep networks critically depend on hyperparameters, but training is expensive
- To automate a heuristic that experts use, we built a probabilistic model to forecast the asymptotic accuracy of a given parameter setting and stop all but the most promising runs
- Simulation resulted in a 2.7-fold reduction of overall runtime

Motivation

- It takes very few SGD iterations for a human expert to tell good from bad parameter settings
- Yet in hyperparameter optimization every setting is run to the very end
- Automating the prediction of performance could save a lot of time and speed up preliminary evaluations during development

Model Search

- Search over structure and hyperparameters of deep networks:
  - 81 parameters in total, namely 9 network parameters and 12 parameters for each of up to 6 layers
- Neural network software: Caffe [Jia 2013]
- Bayesian optimization methods:
  - SMAC (Sequential Model-based Algorithm Configuration) is based on Gaussian Mixture Models. Supports conditional, continuous and discrete parameters and also priors over them. [Bergstra, Bardenet, Bengio, and Kégl, 2011]
  - TPE (Tree Parzen Estimator) is based on Gaussian Mixture Models. Supports continuous, discrete and conditional hyperparameters. [Hoos, Hutter, and Leyton-Brown, 2011]
- 5 runs of both SMAC and TPE
- Evaluated a total of 800 networks
- Dataset: k-means features extracted from CIFAR10 [Krizhevsky 2009; Coates 2011]

Extrapolation

- Problem definition
  - Given data points \( y_{m} \) we would like to forecast the future performance \( y_{\text{last}} \) probabilistically

Approach

- Selected \( k = 10 \) parametric model families that roughly match learning curves' shape (typically increasing, saturating functions)
  - Representative power increased by convex combination of individual models:
    - \( f(x) = \sum_{i=1}^{k} w_i f_i(x; \theta_i) + \epsilon \) with \( \epsilon \sim N(0, \sigma^2) \) and \( \sum_{i=1}^{k} w_i = 1 \)
  - Model uncertainty captured by MCMC
    - The prior encoded monotonicity assumption of each of the models
  - Model configuration 
    - We obtained \( S = 10\,000\) samples from 100 parallel chains of length 1500 with a burn-in of 500
    - Let \( \xi \) be the model's parameters \( (w_1, \ldots, w_k; \theta_1, \ldots, \theta_k; \sigma^2) \)
    - Probability of improving over current best parameter setting:
      - \( P(O_{\text{last}} \geq y_{\text{best}} | y_{m}) = 1 \frac{\xi^T \sum_{i=1}^{k} P(O_{\text{last}} \geq y_{\text{best}} | \xi^T y_{m})}{\sum_{i=1}^{k} P(O_{\text{last}} \geq y_{\text{best}} | \xi^T y_{m})} \)

Learning curves

- Random subset of learning curves:

Experiments

- Example extrapolation:
- Example of model being misled by unusual shape of the learning curve:
- Quality of predictions:
  - RMSE of residual \( E[y_{\text{rm}}] - y_{\text{last}} \):
    - \% train 10% 30% 50% 70% 90%
    - RMSE 0.082 0.046 0.026 0.010 0.011
- \( y_{\text{last}} \) in/over/under 90% interval:
  - \% train 10% 30% 50% 70% 90%
  - \( y_{\text{last}} \) in 42.54% 48.51% 61.94% 80.45% 91.04%
  - \( y_{\text{last}} \) over 12.69% 9.70% 8.96% 6.77% 6.72%
  - \( y_{\text{last}} \) under 44.77% 41.79% 29.10% 12.78% 2.24%
- Model tends to be overconfident based on little data, but rarely underpredict
- Simulated early stopping in optimization
  - Replayed all 800 runs
  - Stopped a run when probability of improving over current best got too small: \( P(O_{\text{last}} \geq y_{\text{best}} | y_{m}) < 1\%
- Reached the same accuracy
- 2.7-fold speedup

Ongoing/Future Work

- Use early stopping in model search
- Control early stopping via Bayesian optimization