Random Search for Hyper-Parameter Optimization

1 Citation

Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of Machine Learning Research 13.Feb (2012): 281-305.

`http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a`

2 Abstract

Random search is better than grid search for hyperparameter optimization.

3 Introduction

Algorithm $\mathcal{A}$, parametrized by hyperparameters λ (call this $\mathcal{A}_\lambda$) and given training set $\mathcal{X}^{(train)}$ produces function f to minimize $\mathcal{L}(x; f)$ over points sampled i.i.d from the training set: $f = \mathcal{A}_\lambda(\mathcal{X}^{(train)})$. Hyperparameter optimizations aims to optimize generalization error (we simplify this by approximating generalization error with a validation set and by considering a set of trial points instead of the full hyperparameter space):

$$\begin{aligned}
\lambda^{(*)} &= \text{argmin}_{\lambda\in\Lambda}\mathbb{E}_{\mathcal{X}\sim\mathcal{G}_\mathcal{X}}[\mathcal{L}(x; \mathcal{A}_\lambda(\mathcal{X}^{(train)}))] && (1)\\
&\approx \text{argmin}_{\lambda\in\Lambda}\text{mean}_{x\in\mathcal{X}^{(valid)}}\mathcal{L}(x; \mathcal{A}_\lambda(\mathcal{X}^{(train)})) && (2)\\
&= \text{argmin}_{\lambda\in\Lambda}\Psi^{(valid)}(\lambda) && (3)\\
&\approx \text{argmin}_{\lambda\in\{\lambda^{(1)},...,\lambda^{(S)}\}}\Psi^{(valid)}(\lambda) = \hat{\lambda} && (4)
\end{aligned}$$

Random search and grid search aim to pick the $\lambda^{(1)}, ..., \lambda^{(S)}$.

Suppose there are D hyperparameters, each with L levels. Suppose the number of hyperparameters that have a meaningful impact on accuracy is E. Then grid search only really needs to consider E^L cells, but actually considers D^L cells. Random search does not suffer from this issue. Note that if $D \approx E$ (this usually happens when D is small), then grid search might be better. Another advantage to random search is that it doesn't matter if you have a few extra trials or failed trials because trials are i.i.d.

4 Random vs. Grid for Optimizing Neural Networks

$$\mathbb{V}^{(valid)}(\lambda) = \frac{\Psi^{(valid)}(\lambda)(1 - \Psi^{(valid)}(\lambda))}{|\mathcal{X}^{(valid)}| - 1} \quad (5)$$

Treating $\Psi^{(valid)}$ as the mean of a Bernoulli random variable gives us the variance above. We can define $\Psi^{(test)}$ and $\mathbb{V}^{(test)}(\lambda)$ analogously. It is typical to report test error as $\Psi^{(test)}(\lambda^{(s)})$ where $s = \text{argmax}_{s' \in 1 \ldots S} \Psi^{(valid)}(\lambda^{(s')})$. We propose a way to combine all the trials to estimate test error, Basically, we create a Gaussian Mixture Model with S components, where component s has mean $\Psi^{(valid)}(\lambda^{(s)})$ and variance $\mathbb{V}^{(valid)}(\lambda^{(s)})$. The weights are defined as follows (we can estimate them by simulating 10-20 draws from the distribution):

$$w_s = P(Z^{(s)<Z^{(s')}}, \forall s' \neq s) \tag{6}$$

$$Z^{(i)} \sim \mathcal{N}(\Psi^{(valid)}(\lambda^{(i)}), \mathbb{V}^{(valid)}(\lambda^{(i)})) \tag{7}$$

Our test error can be reported as $\mu_z = \sum_{s=1}^{S} w_s \mu_s$ with variance $\sigma_z^2 = \sum_{s=1}^{S} w_s(\mu_s^2 + \sigma_s^2) - \mu_z^2$. Here's how we can use these numbers to make a useful visualization. First, we run S trials of random search. Now, we consider some "experiment size" s such that $sN = S$ for some integer N. Then, we break the trials into N experiments with s trials each. We can then plot a box-plot for this experiment size (using μ_z and σ_z^2 for this experiment). If we make the y-axis into test error and the x-axis into experiment size, then we can plot a sequence of vertical box-plots that get shorter as experiment size grows larger. This Random Experiment Efficiency Curve shows us how quickly random search converages.

We consider the follwing datasets: (1) MNIST, (2) MNIST but where the digit foreground pixels are replaced with pixels from a natural image, (3) MNIST but where the digit foreground pixels are random, (4) MNIST but where each digit is randomly rotated, (5) MNIST but with random rotation and natural foreground mask, (6) randomly generated rectangles (the model must guess if they are tall or wide), (7) rectangles where the forground and background are pixels from different natural images, (8) an image with pixels with value 1 or 0 (the model must predict whether the 1 pixels form a convex set).

We train a neural network with the following hyperparameters (1) data preprocessing (none, normalize, PCA) (2) PCA fraction of variance to keep (two possible distributions), (3) random seeds, (4) number of hidden units, (5) minibatch size, (6) learning rate, (7) L2 regularization parameter, (8) number of steps before cutting learning rate. Doing grid search is not tractable for this hyperparameter space. However, if we consider S grid cells, we can compare against random search. For a given S, random search tends to do better than grid search. For our datasets, random search tends to converge pretty quickly (sometimes in like 6 trials and other times in like 32).

5 The Low Effective Dimension of Ψ

We use a Gaussian Process with squared exponential kernels to estimate $\Psi(\lambda)$ from λ. We can use this to estimate the effective number of hyperparameters. We found that there are usually just a few hyperparameters that really matter, but these effective hyperparameters vary depending on the dataset that we are training on. Thus, grid search is a poor choice.

6 Grid Search and Sets with Low Effective Dimensionality

We generate a random hyperspace and embed a random target hyperspace inside it. We then see how many tries it takes random search vs. grid search (we considered several possible grids over the parent hyperspace) to find a point in the target hyperspace. If the target hyperspace has volume v, the parent hyperspace has volumn V, and the number of trials is T, then random search will find the target hyperspace with probability $1 - (1 - \frac{v}{V})^T$. We find that grid search tends to perform worse than random search.

7 Random Search vs. Sequential Manual Optimization

In practice, people tend to do some manual search along with grid search. We compare this approach with random search on training a Deep Belief Network (i.e. stacked Restricted Boltzmann Machines trained with contrastive divergence). We have like 13 hyperparameters in this problem. There's a well known paper by Larochelle et al. that does a manual search + grid search to optimize the hyperparameters, so we compare against their technique. We find that random search beats the manual + grid approach on 1 dataset, ties on 4, and loses on 3. So random search isn't quite as good as manual + grid search, but it certainly competitive and much simpler to implement.

8 Future Work

You could combine random search with grid search (i.e. grid search on a few hyperparameters and randomly pick the rest). There's a bunch of work on hyperparameter optimization, so maybe we could combine random search with some of them. These other algorithms are not as popular as grid search because they are much harder to implement and require more complex systems (e.g. a master server that picks the next set of hyperparameters to try).

9 Conclusion

When you have more than a couple hyperparameters, use random search instead of grid search. Random search is easy to implement, doesn't care if experiments fail, allows launching more experiments, and provides a nice visualization for test set error convergence.