Thesis proposal

Modeling Diversity in the Machine Learning Pipeline

Jesse Dodge
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
jessed@cs.cmu.edu

February 16, 2018

Abstract

Randomness is a foundation on which many aspects of the machine learning pipeline are built. From training models with stochastic gradient descent to tuning hyperparameters with random search, independent random sampling is ubiquitous. While independent sampling can be fast, it can also lead to undesirable properties, such as when samples are very similar. Abstaining from the independence assumption, we propose to examine the role of diversity in these samples, especially in cases where we have limited computation, limited space, or limited data. We address three applications: tuning hyperparameters, subsampling data, and ensemble generation.

Hyperparameter optimization requires training and evaluating numerous models, which can often be time consuming. Random search allows for training and evaluating these models fully in parallel; we model diversity explicitly in this regime, and find that sets of hyperparameter assignments which are more diverse lead to better optima than conventional random search, a low discrepancy sequence, and even a sequential Bayesian optimization approach.

Drawing a subset of a dataset is useful in many applications, such as when the full training set is too large to fit into memory. How these subsets are drawn is important, as their distribution may differ significantly from that of the full dataset, especially in the case when some labels are much more common than others. We address this by proposing to sample smaller datasets that are diverse, both in terms of labels and features.

Ensembling models is a popular technique when minimizing task-specific errors is more important than computational efficiency or interpretability. Diversity among the models in an ensemble is directly tied to the the ensemble's error, with more diversity often leading to lower error. We propose to apply our diversity promoting techniques to ensemble generation, first when selecting data to train the base models (as in bagging), and second when choosing which already-trained models will comprise an ensemble.

Contents

1 Introduction

Randomness is ubiquitous in machine learning. From random search for hyperparameter optimization to training with stochastic gradient descent, randomness plays a part in many stages of the machine learning pipeline. We propose to examine the role of *diversity* in this randomness, especially in cases where we have limited computation, limited space, or limited data.

Many machine learning algorithms use some form of independent random sampling. A set of independent samples may have undesirable properties, however, such as when the samples happen to be very similar. We propose to forego this mutual independence assumption and model the interaction between samples in specific applications. Our approaches can elegantly model both the quality of the individual samples and the similarity between all pairs of samples. Even with this strong statistical dependence, we introduce an efficient algorithm for drawing diverse samples from any space from which we can draw uniform samples.

We apply our approach to a number of common problems in machine learning, including hyperparameter optimization and ensemble construction. In a few cases, current methods are special cases of our approach; here our proposed techniques will likely do no worse than conventional approaches, while explicitly modeling diversity may lead to better performance, consumption of fewer resources (like compute time), or both. We also address applications where there are new opportunities for modeling diversity. Below we list three application areas.

Hyperparameter Search: Hyperparameters such as the learning rate of an artificial neural network or regularization strength of regularized logistic regression are vital to the empirical success of machine learning applications. Currently, choosing good hyperparameter values requires knowledge of both the training algorithms and application domain, which limits the adoption of machine learning to new areas. This is exemplified by the rise in popularity of neural networks, which notoriously have many hyperparameters to tune [Zhang and Wallace, 2015]; the development of effective hyperparameter optimization algorithms will help alleviate these difficulties.

Random search for hyperparameter optimization has the advantage that all the hyperparameter values can be evaluated in parallel [Bergstra and Bengio, 2012]. We model diversity explicitly in random hyperparameter search and find better settings than conventional random search, a low discrepancy sequence, and even a sequential Bayesian optimization approach designed for tree-structured spaces. Additionally, we find that Spearmint, a popular Bayesian optimization package , takes more than ten times as long to find an optimum that leads to less than 0.15 percent gain in accuracy over our approach (on average) on a particular hyperparameter optimization task.

Subsampling Data: Many datasets are so large that it is not possible to use them in entirety. Sampling data uniformly can lead to poor coverage, especially when some labels are much more frequent than others; we propose to model the diversity of the labels in sampled data so we can control the expected label proportions. When training requires a significant amount of resources, each example is important, and redundant examples are wasteful. In addition to label diversity, we will also promote feature diversity in sets of examples, so we can sample datasets which are unlikely to contain very similar examples.

Adaptive sample size methods start by fully training a model on a small set of data, then use that trained model as an initialization point for training a new model on a larger set [Mokhtari and

Ribeiro, 2017]. These small sets of data dictate the initial direction for the parameters in these models, though because of their size they are likely to deviate significantly from the full dataset. Similar to our proposed work on subsampling data to create balanced datasets, we propose to sample datasets when using adaptive sample size methods for training so that they form a diverse, high-quality set, and will evaluate our approach on training speed and model quality.

Ensembles Generation: An ensemble, created by combining predictions from a set of models, can have lower error than the individual models themselves. Perhaps the most important consideration when building an ensemble is selecting the models which comprise it; understanding the diversity of these models is considered the "holy grail" of ensemble methods [Zhou, 2012]. As we see in Section 5, the diversity of an ensemble is directly related to its error.

Diversity can play a role in ensemble creation in a number of ways. Bagging, for example, generates new datasets which are used to train models using the same algorithm. The success of bagging comes from the diversity in the data among the generated datasets. We will model this diversity directly, which will allow us to control in what way and to what degree the datasets are diverse.

Given a set of learners, selecting a subset to include in an ensemble is a common and challenging step in ensemble creation. Many measures exist which can be used to compute the diversity of an ensemble [Zhou, 2012]. Each diversity measurement has advantages and disadvantages, and while diversity is valuable, other considerations (such as the error of the individual models) are also important. We propose to develop an approach which selects models to be included in an ensemble by trading off between multiple objectives, and we hypothesize doing so effectively will lead to better-quality ensembles.

2 Notation and Background

In this thesis we propose to use determinantal point processes (DPPs) [Macchi, 1975, Kulesza et al., 2012] for modeling diversity and quality of items in a set. DPPs are statistical models of global, negative correlations. Given a base set of items $\mathcal{B}$, a DPP defines a distribution over all subsets of $\mathcal{B}$. A DPP is defined by a matrix which is constructed using a kernel $\mathcal{K}$ that measures the similarity between items, and a quality measure q_i for each individual item. One draw from a DPP is then a subset $\mathcal{J}$ of the items in $\mathcal{B}$, where diverse sets $\mathcal{J}$ (according to $\mathcal{K}$) with high-quality items are more likely. While much of our discussion will relate to a $\mathcal{B}$ which is a finite set of items, we also will address the case where some or all of the dimensions of $\mathcal{B}$ are continuous.

2.1 DPP Notation

The following notation will be used:

- $\mathcal{B}$ is a base set, which the DPP will sample from. It might be discrete (e.g. examples in a training set), or $\mathcal{B}$ might be continuous, or some dimensions of $\mathcal{B}$ might be continuous and others discrete (e.g. hyperparameter space including learning rate and type of nonlinearity in a neural network).

- $\mathcal{Y} = \{1, \ldots, N\}$, a set of indices which correspond to items in $\mathcal{B}$. Only defined when $\mathcal{B}$ is discrete.
- $\boldsymbol{\phi}_i \in \mathbb{R}^d$ is a feature vector representing item i in $\mathcal{B}$, used to compute similarity between items.
- $q_i \in \mathbb{R}^+$ is a scalar representing the quality of item i .
- $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$ is a similarity kernel (e.g. cosine distance). $\mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_i) = 1$, and $\mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) = 0$ when $\boldsymbol{\phi}_i, \boldsymbol{\phi}_j$ are as dissimilar as possible.
- $\mathbf{L} \in \mathbb{R}^{N \times N}$ is a symmetric, positive-semidefinite Gram matrix which defines a DPP (specifically, an L-ensemble), where $\mathbf{L}_{i,j} = q_i q_j \mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j)$.
- $\mathbf{Y} \sim \text{DPP}(\mathcal{K})$, $\mathbf{Y} \subseteq \mathcal{Y}$, and when drawn from k-DPP, $|\mathbf{Y}| = k$.
- k is the size of one draw from a k-DPP. $k \leq N$ always holds true.

3 Diversity in Hyperparameters

Hyperparameter values—regularization strength, model family choices like depth of a neural network or which nonlinear functions to use, procedural elements like dropout rates, stochastic gradient descent step sizes, and data preprocessing choices—can make the difference between a successful application of machine learning and a wasted effort. To search among many hyperparameter values requires repeated execution of often-expensive learning algorithms, creating a major obstacle for practitioners and researchers alike.

In general, on iteration (evaluation) k, a hyperparameter searcher suggests a hyperparameter configuration x_k, a worker trains a model using x_k, and returns a validation loss of y_k computed on a hold out set. In this work we say a hyperparameter searcher is **open loop** if x_k depends only on $\{x_i\}_{i=1}^{k-1}$; examples include choosing x_k uniformly at random [Bergstra and Bengio, 2012], or x_k coming from a low-discrepancy sequence (c.f., Iacò [2015]). We say a searcher is **closed loop** if x_k depends on both the past configurations and validation losses $\{(x_i, y_i)\}_{i=1}^{k-1}$; examples include Bayesian optimization [Snoek et al., 2012] and recent reinforcement learning methods [Zoph and Le, 2016]. Note that open loop methods can draw an infinite sequence of configurations before training a single model, whereas closed loop methods rely on validation loss feedback in order to make suggestions.

While sophisticated closed loop selection methods have been shown to empirically identify good hyperparameter configurations faster (i.e., with fewer iterations) than open loop methods like random search, **two trends have rekindled interest in embarrassingly parallel open loop methods**: 1) modern deep learning models can take days or weeks to train with no signs of efficiency breakthroughs, and 2) the rise of cloud resources available to anyone that charge not by the number of machines, but by the number of CPU-hours used so that 10 machines for 100 hours costs the same as 1000 machines for 1 hour.

In this section we explore the landscape of open loop methods, identifying tradeoffs that are rarely considered, if at all acknowledged. While random search is arguably the most popular open loop method and chooses each x_k independently of $\{x_i\}_{i=1}^{k-1}$, it is by no means the only choice. In many ways uniform random search is the least interesting of the methods we will discuss because we will advocate for methods where x_k depends on $\{x_i\}_{i=1}^{k-1}$ to promote **diversity**. In particular, we

will focus on drawing $\{x_i\}_{i=1}^k$ from a k**-determinantal point process (DPP)** [Kulesza et al., 2012]. We introduce a sampling algorithm which allows DPPs to support real, integer, and categorical dimensions, any of which may have a tree structure.

In synthetic experiments, we find our diversity-promoting open-loop method is competitive or outperforms other open loop methods on a number of measures. In practical hyperparameter optimization experiments, we find that it significantly outperforms other approaches in cases where the hyperparameter values have a large effect on performance. Finally, we compare against a closed loop Bayesian optimization method, and find that sequential Bayesian optimization takes, on average, more than ten times as long to find a good result, for a gain of only 0.15 percent accuracy on a particular hyperparameter optimization task.

3.1 Related Work

While this work focuses on open loop methods, the vast majority of recent work on hyperparameter tuning has been on closed loop methods, which we briefly review.

3.1.1 Closed Loop Methods

Much attention has been paid to sequential model-based optimization techniques such as Bayesian optimization [Bergstra et al., 2011, Snoek et al., 2012], which sample hyperparameter spaces adaptively. These techniques first choose a point in the space of hyperparameters, then train and evaluate a model with the hyperparameter values represented by that point, then sample another point based on how well previous point(s) performed. When evaluations are fast, inexpensive, and it's possible to evaluate a large number of points (e.g. $k >> 2^d$) these approaches can be advantageous, but in the more common scenario where we have limited time or a limited evaluation budget, the sequential nature of closed loop methods can be cumbersome. In addition, it has recently been observed that many Bayesian optimization methods, when run for k iterations, can be outperformed by sampling $2k$ points uniformly at random [Li et al., 2017], indicating that even simple open loop methods can be competitive.

Parallelizing Bayesian optimization methods has proven to be nontrivial, though many agree that it's vitally important. Many algorithms exist which can sample more than one point at each iteration [Contal et al., 2013, Desautels et al., 2014, González et al., 2016, Kandasamy et al., 2018], but the sequential nature of Bayesian optimization methods prevent the full parallelization open loop methods can employ. Even running two iterations (with batches of size $k/2$) will take on average twice as long as fully parallelizing the evaluations, as you can do with open loop methods like grid search, sampling uniformly, or sampling according to a DPP.

One recent line of research has examined the use of DPPs for optimizing hyperparameters in the context of parallelizing Bayesian optimization [Kathuria et al., 2016, Wang et al., 2017]. At each iteration within one trial of Bayesian optimization, instead of drawing a single new point to evaluate from the posterior, they define a DPP over a relevance region from which they sample a diverse set of points. They found their approach to beat state-of-the-art performance on a number of hyperparameter optimization tasks, and they proved that generating batches by sampling from a DPP has better regret bounds than a number of other approaches (which include drawing batch samples which maximize the DPP, and non-batch approaches). They show that a previous batch sampling approach which selects a batch by sequentially choosing a point which has the highest

posterior variance [Contal et al., 2013] is equivalent to maximizing the probability of a sample drawn from a DPP. As another connection, we show in Section 3.3.2 that the DPP sampling approach is actually equivalent to sampling points proportional to the posterior variance. We use the work of Kathuria et al. [2016] as a foundation for our exploration of fully-parallel optimization methods.

So-called configuration evaluation methods have been shown to perform well by adaptively allocating resources to different hyperparameter settings [Swersky et al., 2014, Li et al., 2017]. They initially choose a set of hyperparameters to evaluate (often uniformly), then partially train a set of models for these hyperparameters. After some fixed training budget (e.g., time, or number of training examples observed), they compare the partially trained models against one another and allocate more resources to those which perform best. Eventually, these algorithms produce one (or a small number) of fully trained, high-quality models. In some sense, these approaches are orthogonal to open vs. closed loop methods since both can be applied with these methods.

3.1.2 Open Loop Methods

As discussed above, recent trends have renewed interest in open loop methods. While there exist many different batch BO algorithms, analyzing these in the open loop regime (when there are no results from function evaluations) is often rather simple. As there is no information with which to update the posterior mean, either function evaluations are approximated using the prior or points are drawn only using information about the posterior variance. For example, in the open loop regime, Kandasamy et al. [2018]'s approach without "hallucinated" observations first draws a sample from the posterior of the GP, then selects the max of this sample. Without function evaluations with which to update the posterior, the probability of being the max of the posterior sample is the same for every point, so this is equivalent to uniform sampling. Similarly, open loop optimization in SMAC [Hutter et al., 2012] is equivalent to first Latin hypercube sampling to make a large set of diverse candidate points, then sampling k uniformly among these points.

Recently, uniform sampling was shown to be competitive with sophisticated closed loop methods for modern hyperparameter optimization tasks like optimizing the hyperparameters of deep neural networks [Li et al., 2017], inspiring other works to explain the phenomenon [Ahmed et al., 2016]. Bergstra and Bengio [2012] offer one of the most comprehensive studies of open loop methods to date, and focus attention on comparing random search and grid search. A main takeaway of the paper is that uniform random sampling is generally preferred to grid search[1] due to the frequent observation that some hyperparameters have little impact on performance, and random search promotes more diversity in the dimensions that matter. Essentially, if points are drawn uniformly at random in d dimensions but only $d' < d$ dimensions are relevant, those same points are uniformly distributed (and just as diverse) in d' dimensions. Grid search, on the other hand, distributes configurations aligned with the axes so if only $d' < d$ dimensions are relevant, many configurations are essentially duplicates.

However, grid search does have one favorable property that is clear in just one dimension. If k points are distributed on $[0, 1]$ on a grid, the maximum spacing between points is equal to $\frac{1}{k-1}$. But if points are uniformly at random drawn on $[0, 1]$, the expected largest gap between points scales

[1] Grid search uniformly grids $[0, 1]^d$ such that $x_k = (\frac{i_1}{m}, \frac{i_2}{m}, \ldots, \frac{i_d}{m})$ is a point on the grid for $i_j = 0, 1, \ldots, m$ for all j, with a total number of grid points equal to $(m + 1)^d$.

as $\frac{1}{\sqrt{k}}$. If you are unlucky enough to have your minimum located in this largest gap, this difference could be considerable.

This is an important concept in numerical integration and one way to quantify this property of a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ is star discrepancy:

$$D_k(\mathbf{x}) = \sup_{u_1,\ldots,u_d \in [0,1]} \left| \mathcal{A}_k(\mathbf{x}, u_j) - \prod_{j=1}^{d} u_j \right|, \text{where}$$

$$\mathcal{A}_k(\mathbf{x}, u_j) = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1} \left\{ x_i \in \textstyle\prod_{j=1}^{d} [0, u_j) \right\}.$$

Star discrepancy has been well-studied. One can interpret the star discrepancy as a multidimensional version of the Kolmogorov-Smirnov statistic between the sequence $\mathbf{x}$ and the uniform measure. It is well-known that a sequence chosen uniformly at random from $[0, 1]^d$ has an expected star discrepancy of at least $\sqrt{\frac{1}{k}}$ (and is no greater than $\sqrt{\frac{d \log(d)}{k}}$) [Shalev-Shwartz and Ben-David, 2014] whereas sequences are known to exist with star discrepancy less than $\frac{\log(k)^d}{k}$ [Sobol', 1967], where both bounds depend on absolute constants. These low-discrepancy sequences, as they are known, include the Sobol sequence, which was also given brief mention in Bergstra and Bengio [2012] and shown to outperform random search and grid search.[2] We also note that the Sobol sequence is used as an initialization procedure for some Bayesian Optimization schemes [Snoek et al., 2012]. However, the Sobol sequence is defined only for the $[0, 1]^d$ hypecrube, so for hyperparameter search which involves conditional hyperparameters (i.e. those with tree structure) it is not appropriate.

The final open loop method we study is the DPP, which has been given considerably less attention in the hyperparameter optimization literature. Comparing the star discrepancy of sampling uniformly and Sobol, the bounds suggest that as d grows large relative to k, Sobol starts to suffer. Indeed, Bardenet and Hardy [2016] notes that the Sobol rate is not even valid until $k = \Omega(2^d)$ which motivates them to study a formulation of a DPP that has a star discrepancy between Sobol and random and holds for all k, small and large. They primarily approached this problem from a theoretical perspective, and didn't include experimental results. Their work, in part, motivates us to look at DPPs as a solution for hyperparameter optimization.

3.2 Comparison of Open Loop Methods

Optimization performance—how close a point in our sequence is to the true, fixed minimum—is our goal, not a sequence with low discrepancy. However, as Bergstra and Bengio [2012] observed, the rare "large gap" that can occur in random sequences without the low discrepancy property can affect optimization performance, on average. One natural surrogate of average optimization performance is to define a hyperparameter space on $[0, 1]^d$ and measure the distance from a fixed point, say $\frac{1}{2}\mathbf{1} = (\frac{1}{2}, \ldots, \frac{1}{2})$, to the nearest point in the length k sequence in the Euclidean norm squared: $\min_{i=1,\ldots,k} ||x_i - \frac{1}{2}\mathbf{1}||_2^2$. The Euclidean norm (squared) is motivated by a quadratic Taylor

[2] Bergstra and Bengio [2012] found that the Niederreiter and Halton sequences performed similarly to the Sobol sequence, and that the Sobol sequence outperformed Latin hypercube sampling. Thus, our experiments include the Sobol sequence as a representative low-discrepancy sequence.

series approximation around the minimum of the hypothetical function we wish to minimize. The first question we wish to answer is: is low discrepancy a surrogate for optimization performance? In the first and second columns of Figure 1 we plot the star discrepancy and smallest distance from the center $\frac{1}{2}\mathbf{1}$, respectively, as a function of the length of the sequence (in one dimension[3]) for the Sobol sequence, uniform at random, and a DPP (see the next section for details). We observe that even though the Sobol sequence was designed specifically to minimize discrepancy, samples drawn from a DPP perform comparably, and both strongly outperform uniform sampling. However, all methods appear comparable when it comes to distance to the center.

Acknowledging the fact that practitioners define the search space themselves more often than not, we realize that if the search space bounds are too small, the optimal solution often is found on the edge, or in a corner of the hypercube (as the true global optima are outside the space). Thus, in some situations it makes sense to *bias* the sequence towards the edges and the corners, the very opposite of what low discrepancy sequences attempt to do. While Sobol and uniformly random sequences will not bias themselves towards the corners, a DPP does. This happens because points from a DPP are sampled according to how distant they are from the existing points; this tends to favor points in the corners. This same behavior of sampling in the corners is also very common for Bayesian optimization schemes, which is not surprise due to the known connections between sampling from a DPP and Gaussian processes (see Section 3.3.2). In the third column of Figure 1 we plot the distance to the origin which is just an arbitrarily chosen corner of hypercube. As expected, we observe that the DPP tends to outperform uniform at random and Sobol in this metric.

Figure 1 includes k up to 500, in one dimension. To generate the k-DPP samples, we sequentially drew samples proportional to the (updated) posterior variance (using an RBF kernel, with $\sigma = \sqrt{2}/k$), as described in Section 3.3.2. Scaling Bayesian optimization to large k is considered one of the most important problems in the field, and this shows our approach is able to scale well: drawing a sample of $k = 500$ took less than a minute on a machine with 16 cores.

In what follows, we study the DPP in more depth and how it performs on real-world hyperparameter tuning problems.

3.3 Method

We begin by reviewing DPPs and k-DPPs.

Let $\mathcal{B}$ be a domain from which we would like to sample a finite subset. (In our use of DPPs, this is the set of hyperparameter assignments.) In general, $\mathcal{B}$ could be discrete or continuous; here we assume it is discrete with N values, and we define $\mathcal{Y} = \{1, \ldots, N\}$ to be a a set which indexes $\mathcal{B}$ (this index set will be particularly useful in Algorithm 1). In Section 3.3.3 we address when $\mathcal{B}$ has continuous dimensions. A DPP defines a probability distribution over $2^{\mathcal{Y}}$ (all subsets of $\mathcal{Y}$) with the property that two elements of $\mathcal{Y}$ are more (less) likely to both be chosen the more dissimilar (similar) they are. Let random variable $\boldsymbol{Y}$ range over finite subsets of $\mathcal{Y}$.

There are several ways to define the parameters of a DPP. We focus on $\mathbf{L}$-ensembles, which define the probability that a specific subset is drawn (i.e., $P(\boldsymbol{Y} = \mathcal{A})$ for some $\mathcal{A} \subset \mathcal{Y}$) as:

$$P(\boldsymbol{Y} = \mathcal{A}) = \frac{\det(\mathbf{L}_{\mathcal{A}})}{\det(\mathbf{L} + I)}. \tag{1}$$

[3]Computation of star discrepancy is NP-Hard in general [Doerr et al., 2014], but feasible in one dimension.

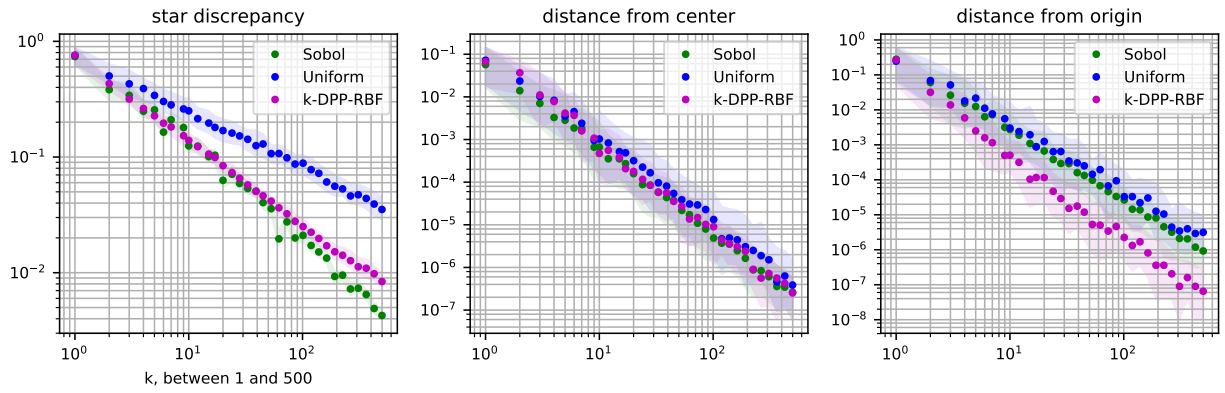

Figure 1: Comparison of the Sobol sequence (with uniform noise), samples a from k-DPP, and uniform random for three metrics of interest. These log-log plots show k-DPP-RBF performs comparably to the Sobol sequence in terms of star discrepancy. On one natural surrogate of optimization performance (distance to the center) all methods perform similarly, but on another (distance to the origin) k-DPP-RBF samples outperform the Sobol sequence and uniform sampling.

As shown in Kulesza et al. [2012], this definition of $\mathbf{L}$ admits a decomposition to terms representing the *quality* and *diversity* of the elements of $\mathcal{Y}$. For any $y_i, y_j \in \mathcal{Y}$, let:

$$\mathbf{L}_{i,j} = q_i q_j \mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j), \tag{2}$$

where $q_i > 0$ is the quality of y_i, $\boldsymbol{\phi}_i \in \mathbb{R}^d$ is a featurized representation of y_i, and $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ is a similarity kernel (e.g. cosine distance). (We will discuss how to featurize hyperparameter settings in Section 3.3.4.)

Here, we fix all $q_i = 1$; in future work, closed loop methods might make use of q_i to encode evidence about the quality of particular hyperparameter settings to adapt the DPP's distribution over time.

3.3.1 Sampling from a k-DPP

DPPs have support over all subsets of $\mathcal{Y}$, including $\emptyset$ and $\mathcal{Y}$ itself. In many practical settings, one may have a fixed budget that allows running the training algorithm k times, so we require precisely k elements of $\mathcal{Y}$ for evaluation. k-DPPs are distributions over subsets of $\mathcal{Y}$ of size k. Thus,

$$P(\boldsymbol{Y} = \mathcal{A} \mid |\boldsymbol{Y}| = k) = \frac{\det(\mathbf{L}_{\mathcal{A}})}{\sum_{\mathcal{A}' \subset \mathcal{Y}, |\mathcal{A}'| = k} \det(\mathbf{L}_{\mathcal{A}'})}. \tag{3}$$

Sampling from k-DPPs has been well-studied. When the base set $\mathcal{B}$ is a set of discrete items, exact sampling algorithms are known which run in $\mathcal{O}(Nk^3)$ Kulesza et al. [2012]. When the base set is a continuous hyperrectangle, a recent exact sampling algorithm was introduced, based on a connection with Gaussian processes (GPs), which runs in $\mathcal{O}(dk^2 + k^3)$ [Hennig and Garnett, 2016]. We are unaware of previous work which allows for sampling from DPPs defined over any other base sets.

3.3.2 Sampling proportional to the posterior variance of a Gaussian process

GPs have long been lauded for their expressive power, and have been used extensively in the hyperparameter optimization literature. Hennig and Garnett [2016] show that drawing a sample from a k-DPP with kernel $\mathcal{K}$ is equivalent to sequentially sampling k times proportional to the (updated) posterior variance of a GP defined with covariance kernel $\mathcal{K}$. This lets us easily draw connections between our approach and other Bayesian optimization approaches. Additionally, this has a nice information theoretic justification: since the entropy of a Gaussian is proportional to the log determinant of the covariance matrix, points drawn from a DPP have probability proportional to $\exp(\text{information gain})$, and the most probable set from the DPP is the set which maximizes the information gain. With our MCMC algorithm presented in Algorithm 2, we can draw samples with these appealing properties from any space for which we can draw uniform samples. The ability to draw DPP samples by sequentially sampling points proportional to the posterior variance grants us another boon: if one has a sample of size k and wants a sample of size $k+1$, only a single additional point needs to be drawn, unlike with the sampling algorithms presented in Kulesza et al. [2012].

3.3.3 Sampling k-DPPs defined over arbitrary base sets

Anari et al. [2016] present a Metropolis-Hastings algorithm (included here as Algorithm 1) which is a simple and fast alternative to the exact sampling procedures described above. However, it is restricted to discrete domains. We propose a generalization of the MCMC algorithm which preserves relevant computations while allowing sampling from any base set from which we can draw uniform samples, including those with discrete dimensions, continuous dimensions, some continuous and some discrete dimensions, or even (conditional) tree structures (Algorithm 2).

Algorithm 1 Drawing a sample from a discrete k-DPP [Anari et al., 2016]

Input: $\mathbf{L}$, a symmetric, $N \times N$ matrix where $\mathbf{L}_{i,j} = q_i q_j \mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j)$ which defines a DPP over a finite base set of items $\mathcal{B}$, and $\mathcal{Y} = \{1, \ldots, N\}$, where $\mathcal{Y}_i$ indexes a row or column of $\mathbf{L}$

Output: $\mathcal{B}_{\mathbf{Y}}$ (the points in $\mathcal{B}$ indexed by $\mathbf{Y}$)

Initialize $\mathbf{Y}$ to k elements sampled from $\mathcal{Y}$ uniformly
while not mixed **do**
    uniformly sample $u \in \mathbf{Y}, v \in \mathcal{Y} \setminus \mathbf{Y}$
    set $\mathbf{Y}' = \mathbf{Y} \cup \{v\} \setminus \{u\}$
    $p \leftarrow \frac{1}{2} min(1, \frac{\det(\mathbf{L}_{\mathbf{Y}'})}{\det(\mathbf{L}_{\mathbf{Y}})})$
    with probability p: $\mathbf{Y} = \mathbf{Y}'$
Return $\mathcal{B}_{\mathbf{Y}}$

Algorithm 1 proceeds as follows: First, initialize a set $\mathbf{Y}$ with k indices of $\mathbf{L}$, drawn uniformly. Then, at each iteration, sample two indices of $\mathbf{L}$ (one within and one outside of the set $\mathbf{Y}$), and with some probability replace the item in $\mathbf{Y}$ with the other.

When we have continuous dimensions in the base set, however, we can't define the matrix $\mathbf{L}$, so sampling indices from it is not possible. We propose Algorithm 2, which samples points directly from the base set $\mathcal{B}$ instead (assuming continuous dimensions are bounded), and computes only the principal minors of $\mathbf{L}$ needed for the relevant computations on the fly.

Algorithm 2 Drawing a sample from a k-DPP defined over a space with continuous and discrete dimensions

Input: A base set $\mathcal{B}$ with some continuous and some discrete dimensions, a quality function $\mathbf{\Psi} : \mathbf{Y}_i \rightarrow q_i$, a feature function $\mathbf{\Phi} : \mathbf{Y}_i \rightarrow \boldsymbol{\phi}_i$

Output: $\boldsymbol{\beta}$, a set of k points in $\mathcal{B}$

Initialize $\boldsymbol{\beta}$ to k points sampled from $\mathcal{B}$ uniformly
while not mixed **do**
    uniformly sample $u \in \boldsymbol{\beta}, v \in \mathcal{B} \setminus \boldsymbol{\beta}$
    set $\boldsymbol{\beta}' = \boldsymbol{\beta} \cup \{v\} \setminus \{u\}$
    compute the quality score for each item, $q_i = \mathbf{\Psi}(\boldsymbol{\beta}_i), \forall i$, and $q'_i = \mathbf{\Psi}(\boldsymbol{\beta}'_i), \forall i$
    construct $\mathbf{L}_{\boldsymbol{\beta}} = [q_i q_j \mathcal{K}(\mathbf{\Phi}(\boldsymbol{\beta}_i), \mathbf{\Phi}(\boldsymbol{\beta}_j))], \forall i, j$
    construct $\mathbf{L}_{\boldsymbol{\beta}'} = [q'_i q'_j \mathcal{K}(\mathbf{\Phi}(\boldsymbol{\beta}'_i), \mathbf{\Phi}(\boldsymbol{\beta}'_j))], \forall i, j$
    $p \leftarrow \frac{1}{2} min(1, \frac{\det(\mathbf{L}_{\boldsymbol{\beta}'})}{\det(\mathbf{L}_{\boldsymbol{\beta}})})$
    with probability p: $\boldsymbol{\beta} = \boldsymbol{\beta}'$
Return $\boldsymbol{\beta}$

Even in the case where the dimensions of $\mathcal{B}$ are discrete, Algorithm 2 requires less computation and space than Algorithm 1 (assuming the quality and similarity scores are stored once computed, and retrieved when needed). Previous analyses claimed that Algorithm 1 should mix after $\mathcal{O}(N \log(N))$ steps. There are $\mathcal{O}(N^2)$ computations required to compute the full matrix L, and at each iteration we will compute at most $O(k)$ new elements of L, so even in the worst case we will save space and computation whenever $k \log(N) < N$. In expectation, we will save significantly more.

3.3.4 Constructing L for hyperparameter optimization

Let $\boldsymbol{\phi}_i$ be a feature vector for $y_i \in \mathcal{Y}$, a modular encoding of the attribute-value mapping assigning values to different hyperparameters, in which fixed segments of the vector are assigned to each hyperparameter attribute (e.g., the dropout rate, the choice of nonlinearity, etc.). For a hyperparameter that takes a numerical value in range $[h_{\text{min}}, h_{\text{max}}]$, we encode value h using one dimension (j) of $\boldsymbol{\phi}$ and project into the range $[0, 1]$:

$$\boldsymbol{\phi}[j] = \frac{h - h_{\text{min}}}{h_{\text{max}} - h_{\text{min}}} \tag{4}$$

This rescaling prevents hyperparameters with greater dynamic range from dominating the similarity calculations. A categorical-valued hyperparameter variable that takes m values is given m elements of $\boldsymbol{\phi}$ and a one-hot encoding. Ordinal-valued hyperparameters can be encoded using a unary encoding. (For example, an ordinal variable which can take three values would be encoded with [1,0,0], [1,1,0], and [1,1,1].) Additional information about the distance between the values can be incorporated, if it's available. In this work, we then compute similarity using an RBF kernel, $\mathcal{K} = \exp\left(-\frac{||\boldsymbol{\phi}_i - \boldsymbol{\phi}_j||^2}{2\sigma^2}\right)$, and hence label our approach k-DPP-RBF. Values for σ^2 lead to models with different properties; when σ^2 is small, points that are spread out interact little with one another, and when σ^2 is large, the increased repulsion between the points encourages them to be as far apart as possible.

3.3.5 Tree-structured hyperparameters

Many real-world hyperparameter search spaces are tree-structured. For example, the number of layers in a neural network is a hyperparameter, and each additional layer adds at least one new hyperparameter which ought to be tuned (the number of nodes in that layer). For a binary hyperparameter like whether or not to use regularization, we use a one-hot encoding. When this hyperparameter is "on," we set the associated regularization strength as above, and when it is "off" we set it to zero. Intuitively, with all other hyperparameter settings equal, this causes the off-setting to be closest to the least strong regularization. One can also treat higher-level design decisions as hyperparameters [Komer et al., 2014], such as whether to train a logistic regression classifier, a convolutional neural network, or a recurrent neural network. In this construction, the type of model would be a categorical variable (and thus get a one-hot encoding), and all child hyperparameters for an "off" model setting (such as the convergence tolerance for logistic regression, when training a recurrent neural network) would be set to zero.

3.4 Hyperparameter Optimization Experiments

In this section we present our hyperparameter optimization experiments. Our experiments consider a setting where hyperparameters have a large effect on performance: a convolutional neural network for text classification [Kim, 2014]. The task is binary sentiment analysis on the Stanford sentiment treebank [Socher et al., 2013]. On this balanced dataset, random guessing leads to 50% accuracy. We use the CNN-non-static model from Kim [2014], with skip-gram [Mikolov et al., 2013] vectors. The model architecture consists of a convolutional layer, a max-over-time pooling layer, then a fully connected layer leading to a softmax. All DPP samples are drawn using Algorithm 2.

3.4.1 Simple tree-structured space

We begin with a search over three continuous hyperparameters and one binary hyperparameter, with a simple tree structure: the binary hyperparameter indicates whether or not the model will use L_2 regularization, and one of the continuous hyperparameters is the regularization strength. We assume a budget of $k = 20$ evaluations by training the convolutional neural net. L_2 regularization strengths in the range $[e^{-5}, e^{-1}]$ (or no regularization) and dropout rates in $[0.0, 0.7]$ are considered. We consider three increasingly "easy" ranges for the learning rate:

- Hard: $[e^{-5}, e^{5}]$, where the majority of the range leads to accuracy no better than chance.
- Medium: $[e^{-5}, e^{-1}]$, where half of the range leads to accuracy no better than chance.
- Easy: $[e^{-10}, e^{-3}]$, where the entire range leads to models that beat chance.

Figure 2 shows the accuracy (averaged over 50 runs) of the best model found after exploring $1, 2, \ldots, k$ hyperparameter settings. We see that k-DPP-RBF finds better models with fewer iterations necessary than the other approaches, especially in the most difficult case. Figure 2 compares the sampling methods against a Bayesian optimization technique using a tree-structured Parzen estimator [BO-TPE; Bergstra et al., 2011]. This technique evaluates points sequentially, allowing the model to choose the next point based on how well previous points performed (a closed loop

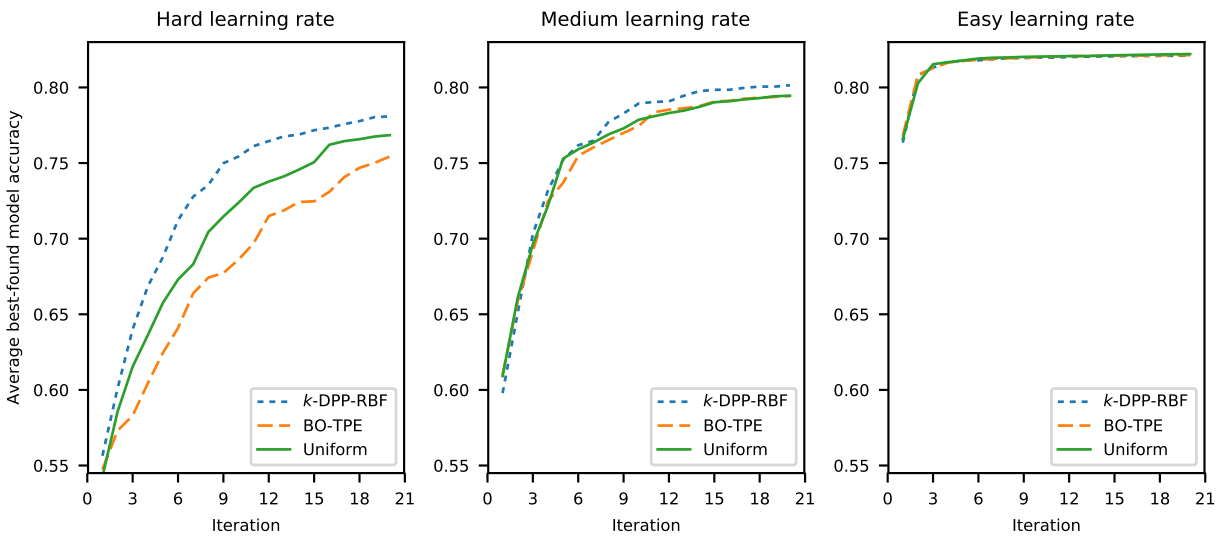

Figure 2: Average best-found model accuracy by iteration when training a convolutional neural network on three hyperparameter search spaces (defined in Section 3.4.1), averaged across 50 trials of hyperparameter optimization, with $k = 20$.

approach). It is state-of-the-art on tree-structured search spaces (though its sequential nature limits parallelization). Surprisingly, we find it performs the worst, even though it takes advantage of additional information. We hypothesize that the exploration/exploitation tradeoff in BO-TPE causes it to commit to more local search before exploring the space fully, thus not finding hard-to-reach global optima. Even though the tree structure for this search space is simple, it's not clear how to search it using a low-discrepancy sequence like the Sobol sequence.

Note that when considering points sampled uniformly or from a DPP, the order of the k hyperparameter settings in one trial is arbitrary (though this is not the case with BO-TPE as it is an iterative algorithm). In all cases the variance of the best of the k points is lower than when sampled uniformly, and the differences in the plots are all significant with $p < 0.01$.

3.4.2 Optimizing within ranges known to be good

Zhang and Wallace [2015] analyzed the stability of convolutional neural networks for sentence classification with respect to a large set of hyperparameters, and found a set of six which they claimed had the largest impact: the number of kernels, the difference in size between the kernels, the size of each kernel, dropout, regularization strength, and the number of filters. We optimized over their prescribed "Stable" ranges for three open loop methods and one closed loop method; average accuracies with 95 percent confidence intervals from 50 trials of hyperparameter optimization are shown in Figure 3, across $k = 5, 10, 15, 20$ iterations. We find that even when optimizing over a space for which all values lead to good models, k-DPP-RBF outperforms the other methods.

Our experiments reveal that, while the hyperparameters proposed by Zhang and Wallace [2015], can have an effect, the learning rate, which they do not analyze, is at least as impactful.

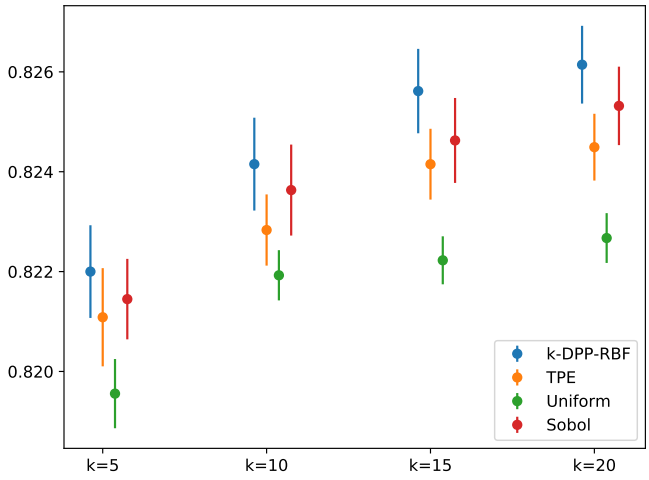

Figure 3: Average best-found model accuracy by iteration when training a convolutional neural network on the "Stable" search space (defined in Section 3.4.2), averaged across 50 trials of hyperparameter optimization, with $k = 5, 10, 15, 20$, with 95 percent confidence intervals. The k-DPP-RBF outperforms uniform sampling, TPE, and the Sobol sequence.

3.4.3 Wall clock time comparison with Spearmint

Here we compare our approach against Spearmint [Snoek et al., 2012], perhaps the most popular Bayesian optimization package. Figure 4 shows wall clock time and accuracy for 25 runs on the "Stable" search space of four hyperparameter optimization approaches: k-DPP-RBF (with $k = 20$), batch Spearmint with 2 iterations of batch size 10, batch Spearmint with 10 iterations of batch size 2, and sequential Spearmint[4]. Each point in the plot is one hyperparameter assignment evaluation. The vertical lines represent how long, on average, it takes to find the best result in one run. We see that all evaluations for k-DPP-RBF finish quickly, while even the fastest batch method (2 batches of size 10) takes nearly twice as long on average to find a good result. The final average best-found accuracies are 82.61 for k-DPP-RBF, 82.65 for Spearmint with 2 batches of size 10, 82.7 for Spearmint with 10 batches of size 2, and 82.76 for sequential Spearmint. Thus, we find it takes on average more than ten times as long for sequential Spearmint to find its best solution, for a gain of only 0.15 percent accuracy.

4 Subsampling data

Much of modern machine learning is built on datasets which are too large to fit in memory. For example, ImageNet [Russakovsky et al., 2015] has been a major driver of progress in computer vision, and it contains more than fourteen million examples. Clearly this wealth of data has proven invaluable, but it is of such a scale that even using specialized techniques training models can take days or even weeks.

One technique commonly employed in such situations is subsampling of the data, but doing so naively can lead to poor results. For example, a desirable property of a subsampled dataset

[4]When in the fully parallel, open loop setting, Spearmint simply returns the Sobol sequence.

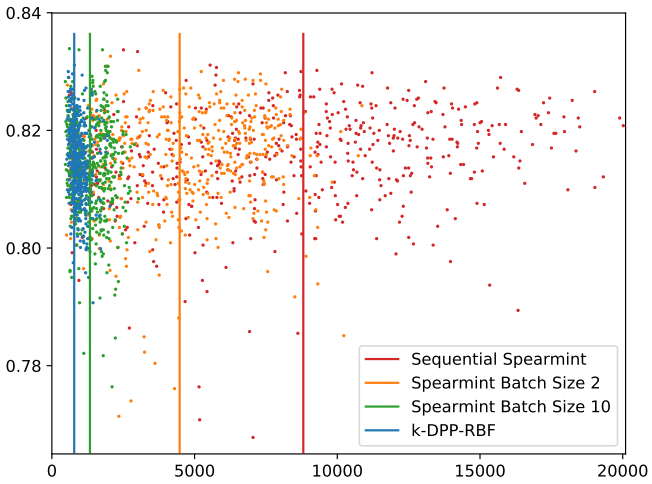

Figure 4: Wall clock time (in seconds) for 25 hyperparameter trials of hyperparameter optimization (each with $k = 20$) on the "Stable" search space define in Section 3.4.2. The vertical lines represent the average time it takes too find the best hyperparameter assignment in a trial.

is that it contains at least one example for every label in the full dataset. If we're interested in how many examples must be drawn from the full dataset so each label is represented once, this is a reframing of the well-known coupon collector's problem [Feller, 1957]. To get at least one example for each label (an admittedly weak requirement), we would need to draw $\Theta(M \log_e(M))$ samples (uniformly, in expectation), where M is the number of labels. For a dataset like ImageNet with $M = 21,841$ labels, we would expect to draw 218,225 examples. Drawing the same number of samples using an approach designed to keep label proportions even (like stratified sampling) would lead to a set with nearly ten examples for each label.

As shown by Zhang et al. [2017], DPPs generalize stratified sampling. That is, stratified sampling can be be recovered as sampling from a DPP with a particular form. Similarly, Dodge et al. [2017] showed that uniform sampling can also be recovered as sampling from a DPP. The construction of the DPP allows for a smooth transition between sampling uniformly and stratified sampling, making it a natural choice for this problem.

4.1 Diversity in data

Zhang et al. [2017] presented an approach for diversifying minibatch samples using a DPP, and showed that the variance of the gradients of minibatches that were diverse is lower than the variance of uniformly sampled minibatches, thus speeding up training. They advocated measuring diversity in two ways: label diversity and feature diversity. Promoting diversity among the labels of the items in pushes the distribution of labels in a sample towards uniform; this is of particular benefit when the full dataset is unbalanced and has some rare labels. To promote diversity among input features for image datasets, we will use distances between images using features from the first fully-connected layer of a pre-trained convolutional neural network, as was done in Zhang et al. [2017]. For text data, we will use edit distance between sentences or documents and cosine distance between average word vectors.

4.2 Proposed work and evaluation

We will evaluate the quality of our data sampling approach in two ways. First, from a full training dataset D which contains N examples, we will select L subsets $D'_1, D'_2, \ldots, D'_L$ of increasing size, so that D'_i contains $\frac{i}{L}$ of the examples in the full dataset. Then, we will train supervised models on these subsets, and compare the learning curves of models trained on data that was sampled using a DPP and using uniform sampling. As Zhang et al. [2017] found diverse minibitaches to lead to faster training, we expect training on subsets of the data which are diverse will lead to better performance than training on subsets of the data which are sampled uniformly at random.

In a similar vein, we will evaluate the effect of training set diversity on adaptive sample size methods. These approaches start by fully training a model on a small dataset, then use that trained model as an initialization for training on a larger dataset (e.g. twice as large) [Mokhtari and Ribeiro, 2017]. Both theoretically and empirically, these approaches show an advantage in convergence rates over traditional methods, but little work has addressed how sensitive they are to the initial, small training set's properties. We hypothesize the variance reduction (and faster training) found by Zhang et al. [2017] will extend to adaptive sample size methods, leading to even better performance of the trained models.

5 Diversity in ensembles

When minimizing task-specific errors is more important than computational efficiency or interpretability, practitioners often turn to ensembles, as they have been shown to achieve better performance than individual learners. Choosing the set of models in an ensemble is important but non-trivial; individual learners that achieve high accuracy and are uncorrelated often lead to the best performance, as we will see below. However, deciding how many learners to include, and how to balance the quality and diversity of the learners, are open questions.

Intuitively, if we have a set of learners which have uncorrelated errors, we should be able to combine their output to get a prediction with lower error. We can see this is true for ensembles in the following analysis.

Let H be an ensemble of T learners $h_1 \ldots h_T$ (combined with equal weight), let $\boldsymbol{x}$ be some training example, and define the *ambiguity* of a learner on $\boldsymbol{x}$ as the difference between its prediction and the prediction of the ensemble:

$$ambi(h_i|\boldsymbol{x}) = (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2.$$

We can then integrate out $\boldsymbol{x}$ to get a data-agnostic measure of disagreement, $ambi(h_i)$. Let $\overline{ambi}(H) = \frac{1}{T}\sum ambi(h_i)$ be the average of the ambiguity of the individual learners, and $\overline{err}(H) = \frac{1}{T}\sum err(h_i)$ be the average of the errors of the learners (using squared error loss). Zhou [2012] showed that for an ensemble H,

$$err(H) = \overline{err}(H) - \overline{ambi}(H).$$

This indicates that ensembled learners can have lower error than the individual learners, as our intuition suggested. Additionally, since $\overline{ambi}(H) \geq 0$, H will never have error worse than the average of the base learners. However, selecting a set of models which minimizes the above quantity is difficult.

To take another look at how diversity impacts ensembles, we can examine the well-known bias-variance decomposition. Complex models are able to capture complex dependencies within data, but they are sensitive to even small sources of noise, while simple models are more robust to small changes in the data but might not be able to capture some of the phenomena present. We can formalize this as:

$$err(h_i) = bias(h_i)^2 + variance(h_i).$$

For an ensemble, Zhou [2012] showed that this can be further expanded to the following bias-variance-covariance decomposition:

$$err(H) = \overline{bias}(H)^2 + \frac{1}{T}\overline{variance}(H) + (1 - \frac{1}{T})\overline{covariance}(H),$$

where $\overline{bias}$, $\overline{variance}$, and $\overline{covariance}$ are the bias, variance, and covariance averaged across models in H. Here we see that as the average covariance between the models increases, so does the error, which gives us a second look at how large of a role the diversity of the models in an ensemble plays. Understanding this diversity, according to Zhou [2012], is the "holy grail" of ensemble methods.

5.1 Measures of diversity

Zhou [2012] discussed fourteen measures of diversity in ensembles, from disagreement to information-theoretic measures such as multi-information, but claimed that generating high-quality ensembles is not as simple as maximizing diversity; as we have seen, the error of an ensemble depends on the average bias and variance as well as the average covariance. In this work, we propose to promote diversity in ensemble creation in two regimes: hyperparameter optimization and bagging.

5.2 Hyperparameter optimization is wasteful

When tuning hyperparameters, usually all models except the single best are discarded, but this wastes all the computation spent training the discarded models. In Section 3, we found that drawing samples from a hyperparameter space using a DPP provided better coverage of that space, leading to better optima. We hypothesize that the models trained on diverse sets of hyperparameters also would lead to better ensembles. Instead of throwing out the $N - 1$ worst models, we can select a subset of the N models to form an ensemble.

5.2.1 Ensembles from hyperparameter search review

Feurer et al. [2015] used models trained during hyperparameter optimization to build ensembles. Given a set of trained models, they greedily added a model to their ensemble (with replacement) based on which model increased development accuracy the most, until their ensemble contained fifty models. They found that ensembles built from models trained during hyperparameter search outperformed the individual learners, but their sequential, greedy process didn't leverage any information about the diversity or quality of the individual learners. Since we have seen the ensemble error is related to the quality and diversity of the learners, their approach almost surely didn't optimize the quality of the entire ensemble.

In different work, Lévesque et al. [2016] adjusted the objective function in Bayesian optimization to optimize the learners within the ensemble; they computed a posterior over the hyperparameter space for the ith model in their ensemble, then used standard Bayesian optimization techniques for choosing the next hyperparameter assignment to evaluate. They found better ensemble performance than Feurer et al. [2015], but again used a greedy approach to select each learner in their ensemble without leveraging information about the quality of the learners or the similarity between the learners in their ensemble.

5.2.2 Proposed work and evaluation

We propose to select an ensemble of size k from a set of N models trained during hyperparameter search. We can use a set of trained models, $\mathcal{B}$, as a base set over which we define a DPP as follows: set q_i (the quality term for item i) to be the development accuracy of model i, and set $\mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j)$ (the similarity between model i and model j) to be one (or a combination) of the similarity measures described by Zhou [2012], like mutual information. The DPP can capture high-order interactions between the models in the set (unlike the greedy approaches used in previous work), where sets with more diversity have higher probability. Here we find an additional property of DPPs appealing, namely that DPPs are distributions over all subsets, including subsets of different sizes. If we happen to train a set of N models which are all extremely similar or in which only a small number achieve high accuracy, a draw will likely contain a small number of models. On the other hand, if a large fraction of our models are high-quality and naturally diverse, a draw will likely contain more models.

We will evaluate the effect of encouraging diversity in two locations: when selecting hyperparameters used to train the models (as in Section 3), and when selecting among already-trained models. We will compare against the two greedy approaches from previous work, and against sampling uniformly at random, using a linear model (logistic regression) and neural model (a convolutional neural network, like in Section 3) as the base learners. We will run experiments on a text classification task and an image classification task, to show our approach is broadly applicable.

5.2.3 Connections between proposed and related work

We can recover the approach of Feurer et al. [2015] as a special case of the proposed work in Section 5.2.2 as follows: set the similarity of all model pairs to 1 (i.e. $\mathcal{K}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) = 1, \forall i, j$), and the quality q_i to be the accuracy of the ensemble which includes that model (with replacement). Then, choose the max probability sample of size 1 from the DPP, and add it to the ensemble. Our proposed approach includes a scalable tradeoff between diversity and ensemble accuracy, and the sets of models drawn using our method will have global diversity encouraged, as opposed to their greedy, sequential maximization.

5.3 Bagging

Bagging (Bootstrap Aggregating) is one of the most successful ensemble methods [Breiman, 1996]. Given a dataset D of size N, bagging builds m datasets of size N' by drawing examples from D uniformly, with replacement. The success of bagging comes from the diversity inherent in the sampled datasets; when $N' = N$ and samples are drawn with replacement, in expectation

each sampled dataset contains 63.2% of the training items in D. The m datasets sampled likely contain somewhat different subsets of the data, so models trained on these sampled datasets will likely have different decision boundaries. We hypothesize that encouraging diversity within each individual dataset and between datasets sampled by bagging will improve the quality of the final ensembles. To do this, we propose extend the Markov DPP [Affandi et al., 2012].

5.3.1 Proposed work and evaluation

Affandi et al. [2012] developed a Markov DPP, where each draw i is DPP distributed, and the union of points from i and $i + 1$ is also DPP distributed. They describe an efficient algorithm for constructing the distribution, and show standard DPP sampling algorithms can be used. We propose to extend the Markov assumption in their approach beyond adjacent draws. With an infinite horizon, we can draw a set of datasets which are each DPP distributed, and for which the union of all the datasets is also DPP distributed. This will be the first model which draws samples with this property.

To evaluate this model, we will use it to draw datasets for bagging. Buja and Stuetzle [2006] showed an equivalence for bagging when sampling with replacement and sampling without replacement. We will follow Affandi et al. [2012] in assuming the different draws will contain no overlapping points; DPPs assign zero probability to samples which contain duplicates, so in this work we will focus on bagging when sampling without replacement.

We hypothesize that promoting diversity between the datasets will lead to diversity between the trained models' predictions, and the diversity within each dataset will give the individual models high quality subsets of the data on which to train. Within each new dataset sampled from D, our model will encourage diversity among the labels (which will help with datasets which have skewed label distributions) and among the features (to encourage diversity in the input). As is described in Section 4, we will promote diversity between images by measuring distance using using features from the first fully-connected layer of a pre-trained convolutional neural network, and will promote diversity in text data using edit distance between sentences and cosine distance between average word vectors. Similarly to Section 5.2.2, we will logistic regression models and convolutional neural networks.

6 Conclusion

In this proposal we have explored the role of diversity in the machine learning pipeline. In hyperparameter optimization we've seen that evaluating a diverse set of hyperparameter assignments (drawn from a DPP) can lead to better optima than a number of other approaches. In synthetic experiments, we showed k-DPP samples perform well on a number of important metrics, even for large values of k. We propose two lines of work: subsampling data for learning with limited computational resources and ensemble construction. In both cases, we build on previous work which showed benefit from promoting diversity in data.

References

Raja Hafiz Affandi, Alex Kulesza, and Emily B. Fox. Markov determinantal point processes. In *Uncertainty in Artificial Intelligence (UAI)*, 2012.

Mohamed Osama Ahmed, Bobak Shahriari, and Mark Schmidt. Do we need harmless Bayesian optimization and first-order Bayesian optimization? In *Proc. of Advances in Neural Information Processing Systems (NIPS) Bayesian Optimization Workshop*, 2016.

Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte Carlo Markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes. In *Proc. of Conference on Learning Theory (COLT)*, 2016.

Rmi Bardenet and Adrien Hardy. Monte Carlo with determinantal point processes. In *arXiv:1605.00361*, 2016.

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

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2011.

Leo Breiman. Bagging predictors. In *Machine Learning*, 1996.

Andreas Buja and Werner Stuetzle. Observations on bagging. *Statistica Sinica*, 2006.

Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013.

Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. In *Proc. of Journal of Machine Learning Research (JMLR)*, 2014.

Jesse Dodge, Catrìona Anderson, and Noah A Smith. Random search for hyperparameters using determinantal point processes. *arXiv preprint arXiv:1706.01566*, 2017.

Carola Doerr, Michael Gnewuch, and Magnus Wahlstrm. Calculation of discrepancy measures and applications. In *Proc. of A Panorama of Discrepancy Theory*, 2014.

William Feller. An introduction to probability theory and its applications. In *John Wiley and Sons*, 1957.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, 2015.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial Intelligence and Statistics*, 2016.

Philipp Hennig and Roman Garnett. Exact sampling from determinantal point processes. In *arxiv:1609.06840*, 2016.

Frank Hutter, H. Holger Hoos, and Kevin Leyton-Brown. Exact sampling from determinantal point processes. In *Proc. of the Learning and Intelligent OptimizatioN Conference (LION) 6*, 2012.

Maria Rita Iacò. Low discrepancy sequences: Theory and applications. In *arXiv:1502.04897*, 2015.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Parallelised bayesian optimisation via thompson sampling. In *Proc. of International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP 2014*, 2014.

Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML*. Citeseer, 2014.

Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 2012.

Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian hyperparameter optimization for ensemble learning. *arXiv preprint arXiv:1605.06394*, 2016.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. *Proc. of ICLR*, 2017.

Odile Macchi. The coincidence approach to stochastic point processes. In *Advances in Applied Probability*, 1975.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013.

Aryan Mokhtari and Alejandro Ribeiro. First-order adaptive sample size methods to reduce complexity of empirical risk minimization. In *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. In *Proc. of International Journal of Computer Vision (IJCV)*, 2015.

Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: From theory to algorithms. In *Proc. of Cambridge University Press*, 2014.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2012.

Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. In *Proc. of Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1967.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, 2013.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. In *International Conference on Machine Learning (ICML)*, 2017.

Cheng Zhang, Hedvig Kjellström, and Stephan Mandt. Stochastic learning on imbalanced data: Determinantal point processes for mini-batch diversification. *UAI*, 2017. URL `http://arxiv.org/abs/1705.00607`.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *arXiv:1611.01578*, 2016.