Efficient Learning by Directed Acyclic Graph For Resource Constrained Prediction

Authored by:

Venkatesh Saligrama
Joseph Wang
Kirill Trapeznikov

Abstract

We study the problem of reducing test-time acquisition costs in classification systems. Our goal is to learn decision rules that adaptively select sensors for each example as necessary to make a confident prediction. We model our system as a directed acyclic graph (DAG) where internal nodes correspond to sensor subsets and decision functions at each node choose whether to acquire a new sensor or classify using the available measurements. This problem can be naturally posed as an empirical risk minimization over training data. Rather than jointly optimizing such a highly coupled and non-convex problem over all decision nodes, we propose an efficient algorithm motivated by dynamic programming. We learn node policies in the DAG by reducing the global objective to a series of cost sensitive learning problems. Our approach is computationally efficient and has proven guarantees of convergence to the optimal system for a fixed architecture. In addition, we present an extension to map other budgeted learning problems with large number of sensors to our DAG architecture and demonstrate empirical performance exceeding state-of-the-art algorithms for data composed of both few and many sensors.

1 Paper Body

Many scenarios involve classification systems constrained by measurement acquisition budget. In this setting, a collection of sensor modalities with varying costs are available to the decision system. Our goal is to learn adaptive decision rules from labeled training data that, when presented with an unseen example, would select the most informative and cost-effective acquisition strategy for this example. In contrast, non-adaptive methods [24] attempt to identify a common sparse subset of sensors that can work well for all data. Our goal is an adaptive method that can classify typical cases using inexpensive sensors while using expensive sensors only for atypical cases. We propose an adaptive

sensor acquisition system learned using labeled training examples. The system, modeled as a directed acyclic graph (DAG), is composed of internal nodes, which contain decision functions, and a single sink node (the only node with no outgoing edges), representing the terminal action of stopping and classifying (SC). At each internal node, a decision function routes an example along one of the outgoing edges. Sending an example to another internal node represents acquisition of a previously unacquired sensor, whereas sending an example to the sink node indicates that the example should be classified using the currently acquired set of sensors. The goal is to learn these decision functions such that the expected error of the system is minimized subject to an expected budget constraint. First, we consider the case where the number of sensors available is small (as in [19, 23, 20]), though the dimensionality of data acquired by each sensor may be large (such as an image taken in different 1

modalities). In this scenario, we construct a DAG that allows for sensors to be acquired in any order and classification to occur with any set of sensors. In this regime, we propose a novel algorithm to learn node decisions in the DAG by emulating dynamic programming (DP). In our approach, we decouple a complex sequential decision problem into a series of tractable cost-sensitive learning subproblems. Cost-sensitive learning (CSL) generalizes multi-decision learning by allowing decision costs to be data dependent [2]. Such reduction enables us to employ computationally efficient CSL algorithms for iteratively learning node functions in the DAG. In our theoretical analysis, we show that, given a fixed DAG architecture, the policy risk learned by our algorithm converges to the Bayes risk as the size of the training set grows. Next, we extend our formulation to the case where a large number of sensors exist, but the number of distinct sensor subsets that are necessary for classification is small (as in [25, 11] where the depth of the trees is fixed to 5). For this regime, we present an efficient subset selection algorithm based on sub-modular approximation. We treat each sensor subset as a new ?sensor,? construct a DAG over unions of these subsets, and apply our DP algorithm. Empirically, we show that our approach outperforms state-of-the-art methods in both small and large scale settings. Related Work: There is an extensive literature on adaptive methods for sensor selection for reducing test-time costs. It arguably originated with detection cascades (see [26, 4] and references therein), a popular method in reducing computation cost in object detection for cases with highly skewed class imbalance and generic features. Computationally cheap features are used at first to filter out negative examples and more expensive features are used in later stages. Our technical approach is closely related to Trapeznikov et al. [19] and Wang et al. [23, 20]. Like us they formulate an ERM problem and generalize detection cascades to classifier cascades and trees and handle balanced and/or multi-class scenarios. Trapeznikov et al. [19] propose a similar training scheme for the case of cascades, however restrict their training to cascades and simple decision functions which require alternating optimization to learn. Alternatively, Wang et al. [21, 22, 23, 20] attempt to jointly solve the decision learning problem by formulating a linear upper-bounding surrogate, converting the problem into a linear program (LP). Conceptually, our work is closely related to Xu et al. [25] and

Kusner et al.[11], who introduce Cost-Sensitive Trees of Classifiers (CSTC) and Approximately Submodular Trees of Classifiers (ASTC), respectively, to reducing test time costs. Like our paper they propose a global ERM problem. They solve for the tree structure, internal decision rules and leaf classifiers jointly using alternative minimization techniques. Recently, Kusner et al.[11] propose Approximately Submodular Trees of Classifiers (ASTC), a variation of CSTC which provides robust performance with significantly reduced training time and greedy approximation, respectively. Recently, Nan et al. [14] proposed random forests to efficiently learn budgeted systems using greedy approximation over large data sets.

Figure 1: A simple example of a sensor selection DAG for a three sensor system. At each state, represented by a binary vector indicating measured sensors, a policy ? chooses between either adding a new sensor or stopping and classifying. Note that the state sSC has been repeated for simplicity.

The subject of this paper is broadly related to other adaptive methods in the literature. Generative methods [17, 8, 9, 6] pose the problem as a POMDP, learn conditional probability models, and myopically select features based information gain of unknown features. MDP-based methods [5, 10, 7, 3] encode current observations as state, unused features as action space, and formulate various reward functions to account for classification error and costs. He et al. [7] apply imitation learning of a greedy policy with a single classification step as actions. Dulac-Arnold et al. [5] and Karayev et al. [10] apply reinforcement learning to solve this MDP. Benbouzid et al.[3] propose classifier cascades with an additional skip action within an MDP framework. Nan et al. [15] consider a nearest neighbor approach to feature selection, with confidence driven by margin magnitude.

2

2

Adaptive Sensor Acquisition by DAG

In this section, we present our adaptive sensor acquisition DAG that during test-time sequentially decides which sensors should be acquired for every new example entering the system. Before formally describing the system and our learning approach, we first provide a simple illustration for a 3 sensor DAG shown in Fig. 1. The state indicating acquired sensors is represented by a binary vector, with a 0 indicating that a sensor measurement has not been acquired and a 1 representing an acquisition. Consider a new example that enters the system. Initially, it has a state of [0, 0, 0]T (as do all samples during test-time) since no sensors have been acquired. It is routed to the policy function ?0 , which makes a decision to measure one of the three sensors or to stop and classify. Let us assume that the function ?0 routes the example to the state [1, 0, 0]T , indicating that the first sensor is acquired. At this node, the function ?1 has to decide whether to acquire the second sensor, acquire the third, or classifying using only the first. If ?1 chooses to stop and classify then this example will be classified using only the first sensor. Such decision process is performed for every new example. The system adaptively collects sensors until the policy chooses to stop and classify (we assume that when all

sensors have been collected the decision function has no choice but to stop and classify, as shown for ?7 in Fig. 1). Problem Formulation: A data instance, x ? X , consists of M sensor measurements, x = {x1 , x2 , . . . , xM }, and belongs to one of L classes indicated by its label y ? Y = {1, 2, . . . L}. Each sensor measurement, xm , is not necessarily a scalar but may instead be multi-dimensional. Let the pair, (x, y), be distributed according to an unknown joint distribution D. Additionally, associated with each sensor measurement xm is an acquisition cost, cm . To model the acquisition process, we define a state space S = {s1 , . . . , sK , sSC }. The states {s1 , . . . , sK } represent subsets of sensors, and the stop-and-classify state sSC represents the action of stopping and classifying with a current subset. Let Xs correspond to the space of sensor measurements in subset s. We assume that the state space includes all possible subsets1 , K = 2M . For example in Fig. 1, the system contains all subsets of 3 sensors. We also introduce the state transition function, T : S ? S, that defines a set of actions that can be taken from the current state. A transition from the current sensor subset to a new subset corresponds to an acquisition of new sensor measurements. A transition to the state sSC corresponds to stopping and classifying using the available information. This terminal state, sSC , has access to a classifier bank used to predict the label of an example. Since classification has to operate on any sensor subset, there is one classifier for every sk : fs1 , . . . , fsK such that fs : Xs ? Y. We assume the classifier bank is given and pre-trained. Practically, the classifiers can be either unique for each subset or a missing feature (i.e. sensor) classification system as in [13]. We overload notation and use node, subset of sensors, and path leading up to that subset on the DAG interchangeably. In particular we let S denote the collection of subsets of nodes. Each subset is associated with a node on the DAG graph. We refer to each node as a state since it represents the ?state-of-information? for an instance at that node. We define the loss associated with classifying an example/label pair (x, y) using the sensors in sj as X Lsj (x, y) = 1fsj (x)6=y + ck . (1) k?sj

Using this convention, the loss is the sum of the empirical risk associated with classifier fsj and the cost of the sensors in the subset sj . The expected loss over the data is defined

LD (?) = Ex,y?D L?(x) (x, y) . (2) Our goal is to find a policy which adaptively selects subsets for examples such that their average loss is minimized min LD (?), (3) ???

where ? : X ? S is a policy selected from a family of policies ? and ?(x) is the state selected by the policy ? for example x. We denote the quantity LD as the value of (3) when ? is the family of all measurable functions. LD is the Bayes cost, representing the minimum possible cost for any 1 While enumerating all possible combinations is feasible for small M , for large M this problem becomes intractable. We will overcome this limitation in Section 3 by applying a novel sensor selection algorithm. For now, we remain in the small M regime.

3

function given the distribution of data. In practice, the distribution D is unknown, and instead we are given training examples (x1 , y1), . . . , (xn , yn

) drawn I.I.D. from D. The problem becomes an n X empirical risk minimization: min ???

L?(xi) (xi , yi).

(4)

i=1

Recall that our sensor acquisition system is represented as a DAG. Each node in a graph corresponds to a state (i.e. sensor subset) in S, and the state transition function, T (sj), defines the outgoing edges from every node sj . We refer to the entire edge set in the DAG as E. In such a system, the policy ? is parameterized by the set of decision functions ?1 , . . . , ?K at every node in the DAG. Each function, ?j : X ? T (sj), maps an example to a new state (node) from the set specified by outgoing edges. Rather than directly minimizing the empirical risk in (4), first, we define a step-wise cost associated (P with all edges (sj , sk) ? E C(x, y, sj , sk) =

t?sk sj

1fsj (x)6=y

ct

if sk 6= sSC . otherwise

(5)

C(?) is either the cost of acquiring new sensors or is the classification error induced by classifying with the current subset if sk = sSC . Using this step-wise cost, we define the empirical loss of the system w.r.t a path for an example x: X R (x, y, ?1 , ..., ?K) =

C (x, y, sj , sj+1) ,

(6)

(sj ,sj+1) ? path(x,?1 ,...,?K)

where path (x, ?1 , . . . , ?K) is the path on the DAG induced by the policy functions ?1 , . . . , ?K for example x. The empirical minimization equivalent to (4) for our DAG system is a sample average n over all example specific path losses: X ? ?1? , . . . , ?K = argmin

?1 ,...,?K ??

R (xi , yi , ?1 , . . . , ?K) .

(7)

i=1

Next, we present a reduction to learn the functions ?1 , ..., ?K that minimize the loss in (7). 2.1

Learning Policies in a DAG

Learning the functions ?1 , . . . , ?K that minimize the cost in (7) is a highly coupled problem. Learning a decision function ?j is dependent on the other functions in two ways: (a) ?j is dependent on functions at nodes downstream (nodes for which a path exists from ?j), as these determine the cost of each action taken by ?j on an individual example (the cost-to-go), and (b) ?j is dependent on functions at nodes upstream (nodes for which a path exists to ?j), as these determine the distribution of examples that ?j acts on. Consider a policy ?j at a node corresponding to state sj such that all outgoing edges from j lead to leaves. Also, we assume all examples pass through this node ?j (we

are ignoring the effect of upstream dependence b). This yields the following important lemma: Lemma 2.1. Given the assumptions above, the problem of minimizing the risk in (6) w.r.t a single policy function, ?j , is equivalent to solving a k-class cost sensitive learning (CSL) problem.2 Proof. Consider the risk in (6) with ?j such that all outgoing edges from j lead to a leaf. Ignoring n the effect of other policy functions is: X upstream from j, the risk w.r.t ?jX C(x, y, sj , sk)1?j (x)=sk ? min

R(x, y, ?j) =

???

sk ?T (sj)

R(xi , yi , ?j).

i=1

Minimizing the risk over training examples yields the optimization problem on the right hand side. This is equivalent to a CSL problem over the space of ?labels? T (sj) with costs given by the transition costs C(x, y, sj , sk). In order to learn the policy functions ?1 , . . . , ?K , we propose Algorithm 1, which iteratively learns policy functions using Lemma 2.1. We solve the CSL problem by using a filter-tree scheme [2] for Learn, which constructs a tree of binary classifiers. Each binary classifier can be trained using regularized risk minimization. For concreteness we define the Learn algorithm as Learn ((x1 , w 1), ..., (xn , w n)) , F ilterT ree((x1 , w 1), ..., (xn , w n))

(8)

where the binary classifiers in the filter tree are trained using an appropriately regularized calibrated convex loss function. Note that multiple schemes exist that map the CSL problem to binary classification. 2 We consider the k-class CSL problem formulated by Beygelzimer et al. [2], where an instance of the problem is defined by a distribution D over X ?[0, inf)k , a space of features and associated costs for predicting each of the k labels for each realization of features. The goal is to learn a function which maps each element of X to a label {1, . . . , k} s.t. the expected cost is minimized.

4

A single iteration of Algorithm 1 proAlgorithm 1 Graph Reduce Algorithm ceeds as follows: (1) A node j is choInput: Data: (xi , yi)ni=1 , sen whose outgoing edges connect only DAG: (nodes S, edges E, costs C(xi , yi , e), ?e ? E), to leaf nodes. (2) The costs associated CSL alg: Learn ((x1 , w 1), . . . , (xn , w n))) ? ?(?) with each connected leaf node are found. while Graph S is NOT empty do (3) The policy ?j is trained on the entire (1) Choose a node, j ? S, s.t. all children of j are set of training data according to these leaf nodes costs by solving a CSL problem. (4) for example i ? {1, . . . , n} do The costs associated with taking the ac(2) Construct the weight vector w i of edge costs tion ?j are computed for each example, per action. and the costs of moving to state j are end for updated. (5) Outgoing edges from node (3) ?j ? Learn ((x1 , w 1), . . . , (xn , w n)) j are removed (making it a leaf node), (4) Evaluate ?j and update edge costs to node j: and (6) disconnected nodes (that were C(xi , yi , sn , sj) ? w ij (?j (xi)) + C(xi , yi , sn , sj) previously connected to node j) are re(5) Remove all outgoing edges from node j in E moved. The algorithm

iterates through (6) Remove all disconnected nodes from S. these steps until all edges have been reend while moved. We denote the policy functions Output: Policy functions, ?1 , . . . , ?K trained on the empirical data using Alg. n n 1 as ?1 , . . . , ?K . 2.2

Analysis

Our goal is to show that the expected risk of the policy functions ?1 , . . . , ?K learned by Alg. 1 converge to the Bayes risk. We first state our main result: Theorem 2.2. Alg. 1 is universally consistent, that is n lim LD (?1n , . . . , ?K) ? LD (9) n??

n are the policy functions learned using Alg. (1), which in turn uses Learn dewhere ?1n , . . . , ?K scribed by Eq. 8.

Alg. 1 emulates a dynamic program applied in an empirical setting. Policy functions are decoupled and trained from leaf to root conditioned on the output of descendant nodes. To adapt to the empirical setting, we optimize at each stage over all examples in the training set. The key insight is the fact that universally consistent learners output optimal decisions over subsets of the space of data, that is they are locally optimal. To illustrate this point, consider a standard classification problem. Let X 0 ? X be the support (or region) of examples induced by upstream deterministic decisions. d? and f ? , Bayes optimal classifiers w.r.t the full space and subset, respectively, are equal on the reduced support:

d? (x) = arg min E 1d(x)6=y —x = f ? (x) = arg min E 1f (x)6=y —x, x ? X 0 ? X ? x ? X 0 . d

f

From this insight, we decouple learning problems while still training a system that converges to the Bayes risk. This can be achieved by training universally consistent CSL algorithms such as filter trees [2] that reduce the problem to binary classification. By learning consistent binary classifiers [1, 18], the risk of the cost-sensitive function can be shown to converge to the Bayes risk [2]. Proof of Theorem 2.2 is included in the Supplementary Material. Computational Efficiency: Alg. 1 reduces the problem to solving a series of O(KM) binary classification problems, where K is the number of nodes in the DAG and M is the number of sensors. Finding each binary classifier is computationally efficient, reducing to a convex problem with O(n) variables. In contrast, nearly all previous approaches require solving a non-convex problem and resort to alternating optimization [25, 19] or greedy approximation [11]. Alternatively, convex surrogates proposed for the global problem [23, 20] require solving large convex programs with ?(n) variables, even for simple linear decision functions. Furthermore, existing off-the-shelf algorithms cannot be applied to train these systems, often leading to less efficient implementation. 2.3

Generalization to Other Budgeted Learning Problems

Although, we presented our algorithm in the context of supervised classification and a uniform linear sensor acquisition cost structure, the above framework holds for a wide range of problems. 5

In particular, any loss-based learning problem can be solved using the proposed DAG approach by generalizing the cost function (? y, sj , sk) = C(x,

c(x, y, sj , sk) if sk 6= sSC , D (x, y, sj) otherwise
(10)

where c(x, y, sj , sk) is the cost of acquiring sensors in sk sj for example (x, y) given the current state sj and D (x, y, sj) is some loss associated with applying sensor subset sj to example (x, y). This framework allows for significantly more complex budgeted learning problems to be handled. For example, the sensor acquisition cost, c(x, y, sj , sk), can be object dependent and non-linear, such as increasing acquisition costs as time increases (which can arise in image retrieval problems, where users are less likely to wait as time increases). The cost D (x, y, sj) can include alternative costs such as ‘2 error in regression, precision error in ranking, or model error in structured learning. As in the supervised learning case, the learning functions and example labels do not need to be explicitly known. Instead, the system requires only empirical performance to be provided, allowing complex decision systems (such as humans) to be characterized or systems learned where the classifiers and labels are sensitive information.

3

Adaptive Sensor Acquisition in High-Dimensions

So far, we considered the case where the DAG system allows for any subset of sensors to be acquired, however this is often computationally intractable as the number of nodes in the graph grows exponentially with the number of sensors. In practice, these complete systems are only feasible for data generated from a small set of sensors (10 or less). 3.1

Learning Sensor Subsets

Although constructing an exhaustive DAG for data with a large number of sensors is computationally intractable, in many cases this is unnecessary. Motivated by previous methods [6, 25, 11], we assume that the number of ?active? nodes in the exhaustive graph is small, that is these nodes are either not visited by any examples or all examples that visit the node acquire the same next sensor. Equivalently, this can be viewed as the system needing only a small number of sensor subsets to classify all examples with low acquisition cost.

Figure 2: An example of a DAG system using the 3 sensor subsets shown on the bottom left. The new states are the union of these sensor subsets, with the system otherwise constructed in the same fashion as the small scale system.

Rather than attempt to build the entire combinatorially sized graph, we instead use this assumption to first find these ?active? subsets of sensors and construct a DAG to choose between unions of these subsets. The step of finding these sensor subsets can be viewed as a form of feature clustering, with a goal of grouping features that are jointly useful for classification. By doing so, the size of the DAG is reduced from exponential in the number of sensors, 2M , to exponential in a much smaller user chosen parameter number of subsets, 2t . In experimental results, we limit t = 8, which allows for a diverse subsets of sensors to be found while preserving computational tractability and efficiency. Our goal is to learn sensor subsets with high classification performance and low acquisition cost (empirically low cost as defined in (1)). Ideally, our goal is to jointly learn the subsets which minimize the empirical risk of the entire system as defined in (4), however this presents a computationally intractable

problem due to the exponential search space. Rather than attempt to solve this difficult problem directly, we minimize classification error over a collection of sensor subsets ?1 , . . . , ?t subject to a cost constraint on the total number of sensors used. We decouple the problem from the policy learning problem by assuming that each example is classified by the best possible subset. For a constant sensor cost, the problem can be expressed as a set constraint problem: min

?1 ,...,?t

N t h i X B 1 X min 1f?j (xi)6=yi such that: —?j — ? , N i=1 j?{1,...,t} ? j=1

where B is the total sensor budget over all sensor subsets and ? is the cost of a single sensor. 6

(11)

Although minimizing this loss is still computationally intractable, consider instead the equivalent problem of maximizing the ?reward? (the event of a correct classification) of the subsets, defined as G=

N X i=1

max j?{1,...,t}

i h 1f?j (xi)=yi ? max

?1 ,...,?t

t X 1 B G(c1 , . . . , ct) such that: —?j — ? . N ? j=1

(12)

This problem is related to the knapsack problem with a non-linear objective. Maximizing the reward in (12) is still a computationally intractable problem, however the reward function is structured to allow for efficient approximation. Lemma 3.1. The objective of the maximization in (12) is sub-modular with respect to the set of subsets, such that adding any new set to the reward yields diminishing returns. Theorem 3.2. Given that the empirical risk of each classifier f?k is submodular and monotonically decreasing w.r.t. the elements in ?k and uniform sensor costs, the strategy in Alg. 2 is an O(1) approximation of the optimal reward in (12). Proof of these statements is included in the Supplementary Material and centers on showing that the objective is sub-modular, and therefore applying a greedy strategy yields a 1 ? 1e approximation of the optimal strategy [16]. 3.2

Algorithm 2 Sensor Subset Selection Input: Number of Subsets t, Cost Constraint B? Output: Feature subsets, ?1 , . . . , ?t Initialize: ?1 , . . . , ?t = ? (i, j) = argmaxi?{1,...,t} argmaxj??iC G(?1 , ..., ?i ? j, ..., ?t) PT while j=1 —?j — ? C? do ?i = ?i ? j (i, j) = argmaxi?{1,...,t} argmaxj??iC G(?1 , ..., ?i ? j, ..., ?t) end while

Constructing DAG using Sensor Subsets

Alg. 2 requires computation of the reward G for only O B? tM sensor subsets, where M is the number of sensors, to return a constant-order approximation to the NP-hard knapsack-type problem. Given the set of sensor subsets ?1 , . . . , ?t , we can now construct a DAG using all possible unions of these subsets, where each sensor subset ?j is treated as a new single sensor, and apply the small scale system presented in Sec. 2. The result is an efficiently learned system with

relatively low complexity yet strong performance/cost trade-off. Additionally, this result can be extended to the case of non-uniform costs, where a simple extension of the greedy algorithm yields a constant-order approximation [12]. A simple case where three subsets are used is shown in Fig. 2. The three learned subsets of sensors are shown on the bottom left of Fig. 2, and these three subsets are then used to construct the entire DAG in the same fashion as in Fig. 1. At each stage, the state is represented by the union of sensor subsets acquired. Grouping the sensors in this fashion reduces the size of the graph to 8 nodes as opposed to 64 nodes required if any subset of the 6 sensors can be selected. This approach allows us to map high-dimensional adaptive sensor selection problems to small scale DAG in Sec. 2.

4

Experimental Results

To demonstrate the performance of our DAG sensor acquisition system, we provide experimental results on data sets previously used in budgeted learning. Three data sets previously used for budget cascades [19, 23] are tested. In these data sets, examples are composed of a small number of sensors (under 4 sensors). To compare performance, we apply the LP approach to learning sensor trees [20] and construct trees containing all subsets of sensors as opposed to fixed order cascades [19, 23]. Next, we examine performance of the DAG system using 3 higher dimensional sets of data previously used to compare budgeted learning performance [11]. In these cases, the dimensionality of the data (between 50 and 400 features) makes exhaustive subset construction computationally infeasible. We greedily construct sensor subsets using Alg. 2, then learn a DAG over all unions of these sensor subsets. We compare performance with CSTC [25] and ASTC [11]. For all experiments, we use cost sensitive filter trees [2], where each binary classifier in the tree is learned using logistic regression. Homogeneous polynomials are used as decision functions in the filter trees. For all experiments, uniform sensor cost were were varied in the range [0, M] achieve systems with different budgets. Performance between the systems is compared by plotting the average number of features acquired during test-time vs. the average test error. 7

4.1

Small Sensor Set Experiments

0.4

0.27 LP Tree DAG

0.45 LP Tree DAG

0.26

0.35

LP Tree DAG

0.4

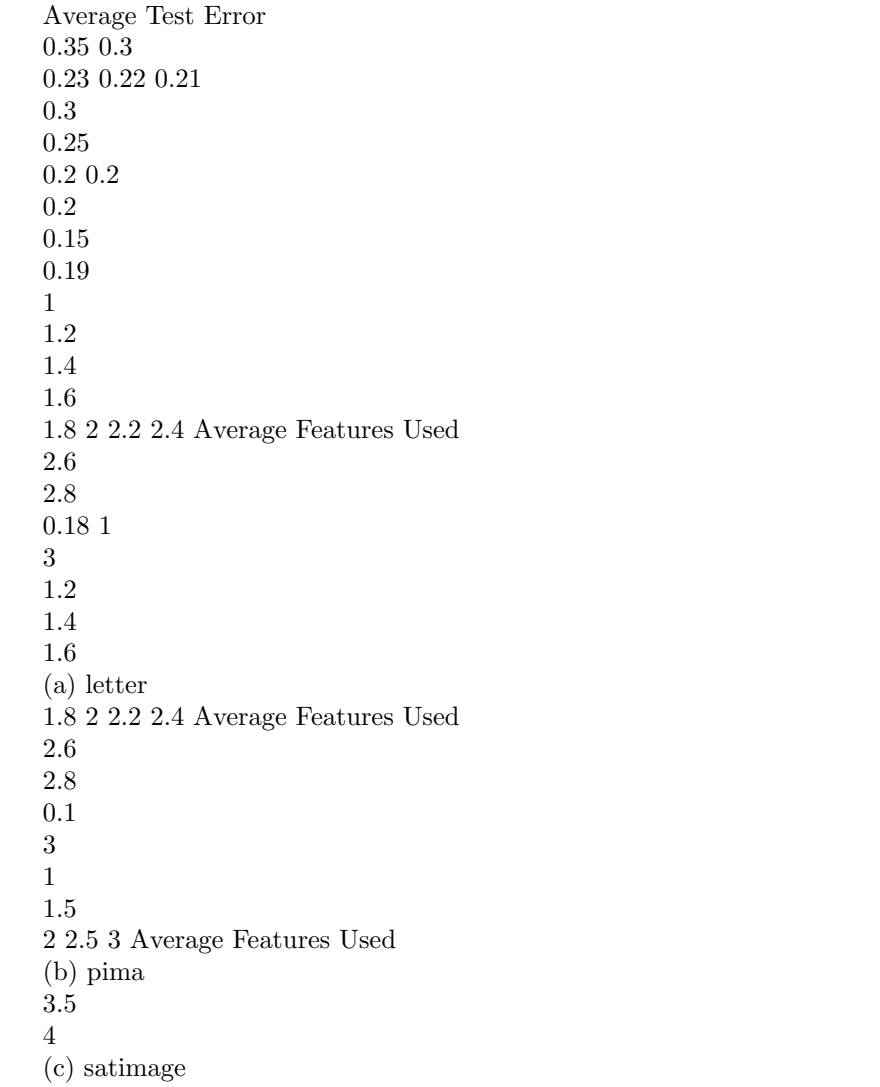
0.25

0.25

0.24

Average Test Error

Average Test Error



(a) letter

(b) pima

(c) satimage

Figure 3: Average number of sensors acquired vs. average test error comparison between LP tree systems and DAG systems.

We compare performance of our trained DAG with that of a complete tree trained using an LP surrogate [20] on the landsat, pima, and letter datasets. To construct each sensor DAG, we include all subsets of sensors (including the empty set) and connect any two nodes differing by a single sensor, with the edge directed from the smaller sensor subset to the larger sensor subset. By including the empty set, no initial sensor needs to be selected. 3rd -order homogeneous polynomials are used for both the classification and system functions in the LP and DAG. As seen in Fig. 3, the systems learned with a DAG outperform the LP tree systems. Additionally, the performance of both of the systems is significantly better than previously reported performance on these data sets for

budget cascades [19, 23]. This arises due to both the higher complexity of the classifiers and decision functions as well as the flexibility of sensor acquisition order in the DAG and LP tree compared to cascade structures. For this setting, it appears that the DAG approach is superior approach to LP trees for learning budgeted systems. 4.2

Large Sensor Set Experiments 0.28
0.3 ASTC CSTC DAG
0.44 ASTC CSTC DAG
0.27
ASTC CSTC DAG
0.42
0.25
0.2
0.15
0.26
0.4
0.25
0.38
0.24
0.36
0.23
0.34
0.1 0.22
0.05
0
5
10
15
20
25
30
35
40
45
50
0.21
0.32
0
5
10
15
20
25
30
35
40

45
50
0.3
0
50
100
150
200
250

(a) MiniBooNE (b) Forest (c) CIFAR Figure 4: Comparison between CSTC, ASTC, and DAG of the average number of acquired features (x-axis) vs. test error (y-axis).

Next, we compare performance of our trained DAG with that of CSTC [25] and ASTC [11] for the MiniBooNE, Forest, and CIFAR datasets. We use the validation data to find the homogeneous polynomial that gives the best classification performance using all features (MiniBooNE: linear, Forest: 2nd order, CIFAR: 3rd order). These polynomial functions are then used for all classification and policy functions. For each data set, Alg. 2 was used to find 7 subsets, with an 8th subset of all features added. An exhaustive DAG was trained over all unions of these 8 subsets. Fig. 4 shows performance comparing the average cost vs. average error of CSTC, ASTC, and our DAG system. The systems learned with a DAG outperform both CSTC and ASTC on the MiniBooNE and Forest data sets, with comparable performance on CIFAR at low budgets and superior performance at higher budgets. Acknowledgments This material is based upon work supported in part by the U.S. National Science Foundation Grant 1330008, by the Department of Homeland Security, Science and Technology Directorate, Office of University Programs, under Grant Award 2013- ST-061-ED0001, by ONR Grant 50202168 and US AF contract FA8650-14-C-1728. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the social policies, either expressed or implied, of the U.S. DHS, ONR or AF.

8

2 References

[1] P. Bartlett, M. Jordan, and J. Mcauliffe. Convexity, Classification, and Risk Bounds. Journal of American Statistical Association, 101(473):138?156, 2006. [2] A. Beygelzimer, J. Langford, and P. Ravikumar. Multiclass classification with filter trees. 2007. [3] R. Busa-Fekete, D. Benbouzid, and B. K?gl. Fast classification using sparse decision dags. In Proceedings of the 29th International Conference on Machine Learning, 2012. [4] M. Chen, Z. Xu, K. Weinberger, O. Chapelle, and D. Kedem. Classifier cascade: Tradeoff between accuracy and feature evaluation cost. In International Conference on Artificial Intelligence and Statistics, 2012. [5] G. Dulac-Arnold, L. Denoyer, P. Preux, and P. Gallinari. Datum-wise classification: a sequential approach to sparsity. In Machine

Learning and Knowledge Discovery in Databases, pages 375?390. 2011. [6] T. Gao and D. Koller. Active classification based on value of classifier. In Advances in Neural Information Processing Systems, volume 24, pages 1062?1070, 2011. [7] H. He, H. Daume III, and J. Eisner. Imitation learning by coaching. In Advances In Neural Information Processing Systems, pages 3158?3166, 2012. [8] S. Ji and L. Carin. Cost-sensitive feature acquisition and classification. Pattern Recognition, 40(5), 2007. [9] P. Kanani and P. Melville. Prediction-time active feature-value acquisition for cost-effective customer targeting. In Advances In Neural Information Processing Systems, 2008. [10] S. Karayev, M. Fritz, and T. Darrell. Dynamic feature selection for classification on a budget. In International Conference on Machine Learning: Workshop on Prediction with Sequential Models, 2013. [11] M. Kusner, W. Chen, Q. Zhou, Z. Xu, K. Weinberger, and Y. Chen. Feature-cost sensitive learning with submodular trees of classifiers. In Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014. [12] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In International Conference on Knowledge Discovery and Data Mining, 2007. [13] L. Maaten, M. Chen, S. Tyree, and K. Q. Weinberger. Learning with marginalized corrupted features. In Proceedings of the 30th International Conference on Machine Learning, 2013. [14] F. Nan, J. Wang, and V. Saligrama. Feature-budgeted random forest. In Proceedings of the 32nd International Conference on Machine Learning, 2015. [15] F. Nan, J. Wang, K. Trapeznikov, and V. Saligrama. Fast margin-based cost-sensitive classification. In International Conference on Acoustics, Speech and Signal Processing, 2014. [16] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions???i. Mathematical Programming, 14(1):265?294, 1978. [17] V. S. Sheng and C. X. Ling. Feature value acquisition in testing: A sequential batch test algorithm. In Proceedings of the 23rd International Conference on Machine Learning, pages 809?816, 2006. [18] I. Steinwart. Consistency of support vector machines and other regularized kernel classifiers. Information Theory, IEEE Transactions on, 51(1):128?142, 2005. [19] K. Trapeznikov and V. Saligrama. Supervised sequential classification under budget constraints. In International Conference on Artificial Intelligence and Statistics, pages 581?589, 2013. [20] J. Wang, T. Bolukbasi, K. Trapeznikov, and V. Saligrama. Model selection by linear programming. In European Conference on Computer Vision, pages 647?662, 2014. [21] J. Wang and V. Saligrama. Local supervised learning through space partitioning. In Advances in Neural Information Processing Systems, pages 91?99. 2012. [22] J. Wang and V. Saligrama. Locally-linear learning machines (L3M). In Asian Conference on Machine Learning, pages 451?466, 2013. [23] J. Wang, K. Trapeznikov, and V. Saligrama. An lp for sequential learning under budgets. In International Conference on Artificial Intelligence and Statistics, pages 987?995, 2014. [24] Z. Xu, O. Chapelle, and K. Weinberger. The greedy miser: Learning under test-time budgets. In Proceedings of the 29th International Conference on Machine Learning, 2012. [25] Z. Xu, M. Kusner, M. Chen, and K. Weinberger. Cost-sensitive tree of classifiers. In Proceedings of the 30th International Conference on Machine Learning, pages 133?141, 2013.

[26] C. Zhang and Z. Zhang. A Survey of Recent Advances in Face Detection. Technical report, Microsoft Research, 2010.

9