THE EFFECT OF ANISOTROPIC NODE SPLITS ON WAVELET DECOMPOSITION OF RF

ASAF ABAS URIA MOR

When building a decision trees (DT), at each node, the data is split by a hyper-plane in a way that minimizes a cost function which represent the learning process according to some criteria. The search space of all possible splits of the data is huge: if our data consists of n samples over d features, the number of all possible $\mathbb{R}^{d-1}$ hyper planes spitting the data has an upper bound of $\binom{n}{d}$ since d points define a $d-1$ dimensional subspace . Of course, this bound is super loose as the samples may be co-linear (and thus many choices will coincide) we did not came up with a better analytic bound, nor probabilistic one (on the expectation). We assume that the size of the search space is what drives prominent implementations of DT to restrict the search to subdivisions by an isotropic split (IS) , that is, splits parallel to one of the features. In this work we describe our attempt to implement a random forest (RF) regressor of decision trees (DT) using near optimal anisotropic split (AS) in each decision node.

The cost function we try to minimize in this work is the sum of variances: suppose we have a split s of domain Ω into two disjoint subdomains $\Omega = \Omega' + \Omega''$, we define the cost of split s as

$$C(s) \coloneqq \frac{1}{|\Omega'|}\sum_{x\in\Omega'}\left(x-\overline{\Omega'}\right)^2 + \frac{1}{|\Omega''|}\sum_{x\in\Omega'}\left(x-\overline{\Omega''}\right)^2$$

Where $|A|$ is the number of elements in set A, and $\overline{A}$ is the mean estimation of outcomes for all data points in set A.

Space size challenge. Examining all possible splits proved itself infeasible not only because of the number of examples to process, but mostly because each such examination requires to solve a linear equation which takes d^2 operations at best. So the natural approach we turn to is treating this search as a 'common' optimization problem, but since our cost function is not continuous, of-the-shelf optimization algorithms such as gradient-descent which require a smooth derivative are not relevant here. A quick experiment with a gradient free optimization method (Nedler-Mead) taught us the local minimas are abound and may be far worse than the best IS cost. Being unable to perform the only two things we actually know (iterating through database and optimizing a smooth function)[1] , we were forced to be creative and seek for some effective heuristics.

First attempt: ***Capture the geometry*** *as if there is* ***no problem***

```
Algorithm 1: 2MeansSVM
  Data: (Training) set $\Omega$ consists of $n$ samples represented by feature matrix $X \in \mathbb{R}^{n\times d}$ and an
        outcome matrix $Y \in \mathbb{R}^{n\times m}$
  Result: A set of parameters $\Theta \coloneqq \theta_1, ..., \theta_d \in \mathbb{R}^d$ and an offset $\theta_0$ such that the split
        $\Omega' = \{x \in \Omega; x \cdot \Theta \le \theta_0\}$ and $\Omega'' = \Omega \setminus \Omega'$ defining the near optimal split of $\Omega$
  begin
      $X \longleftarrow \frac{X-\bar{X}}{\max X - \min X}$;
      Compute KMeans with $k = 2$ on $Y$ and predict labeling of samples;
      Compute SVM on $X$ using predicted labels to find best split;
      return output of SVM;
```

This approach relies on the (almost always wrong) assumption that relations between features and outcomes are global w.r.t the dataset, that is, we assume the underlying function $f : X \to Y$ is *nice enough* in the sense that the direction of greatest variation in the outcome space will correspond to some noticeable change in the feature space. But since the 2Means predicted labels address the geometry of the outcomes space alone, the split in the feature space may be nonsensical, and as we observed - it mostly is.

Date: 02/09/2018.

[1]Joking, we also know all of saint David Bowie's masterpieces by heart

Second attempt: ***Solve the problem*** *as if it had* ***no geometry***

Algorithm 2: PCA

```
Data: (Training) set $\Omega$ consists of $n$ samples represented by feature matrix $X \in \mathbb{R}^{n\times d}$ and an
      outcome matrix $Y \in \mathbb{R}^{n\times m}$
Result: A set of parameters $\Theta \coloneqq \theta_1, ..., \theta_d \in \mathbb{R}^d$ and an offset $\theta_0$ such that the split
        $\Omega' = \{x \in \Omega; x \cdot \Theta \leq \theta_0\}$ and $\Omega'' = \Omega \setminus \Omega'$ defining the near optimal split of $\Omega$
begin
    $X \longleftarrow \frac{X-\bar{X}}{\max X - \min X}$;
    Compute $PCA$ of matrix $X$ ;
    best_var $\longleftarrow \infty$ ;
    for $p \in$ first 2 principal components do
        $\bar{\Theta} \longleftarrow InverseProjection\,(p)$ ;
        $\bar{\Theta}, \bar{\theta_0} \longleftarrow CostMinimization\,(\bar{\Theta}, 0)$ ;
        $C \longleftarrow$ cost of split defined by $\bar{\theta_0}, \bar{\Theta}$;
        if $C \leq$ best_var then
            best_var $\longleftarrow C$;
            $\theta_0 \longleftarrow \bar{\theta_0}$;
            $\Theta \longleftarrow \bar{\Theta}$ ;
    return $\Theta, \theta_0$;
```

This approach aims to find the most varying directions in the feature space of the data, and use these as starting points for classical gradient free optimization (we used the Nedler-Mead solver). While having no guaranties for this split to yield a near optimal cost value, a split that is on one of the first principal components of the features space should provide the best progress in terms of variance reduction in the features space and thus advance faster towards smaller regions in which, we assumed, capturing the local geometry will be easier.

Third attempt: ***Capture the geometry of the problem***

Algorithm 3: PLS

```
Data: (Training) set $\Omega$ consists of $n$ samples represented by feature matrix $X \in \mathbb{R}^{n\times d}$ and an
      outcome matrix $Y \in \mathbb{R}^{n\times m}$
Result: A set of parameters $\Theta \coloneqq \theta_1, ..., \theta_d \in \mathbb{R}^d$ and an offset $\theta_0$ such that the split
        $\Omega' = \{x \in \Omega; x \cdot \Theta \leq \theta_0\}$ and $\Omega'' = \Omega \setminus \Omega'$ defining the near optimal split of $\Omega$
begin
    $X \longleftarrow \frac{X-\bar{X}}{\max X - \min X}$;
    Compute $PLS$ of matrices $X, Y$ ;
    best_var $\longleftarrow \infty$ ;
    for $r \in$ first 2 columns of $X$'s rotation matrix do
        $\bar{\Theta} \longleftarrow r$ ;
        $x' \longleftarrow$ sort $(X\bar{\Theta})$ ;  /* this array will contain the mid points between each two
         adjacent points projections */
        $x' \longleftarrow \frac{x'[:-1]+x'[1:]}{2}$;
        for $i \in 0, ...,$ length of $x'$ do
            $\bar{\theta_0} \longleftarrow x'[i]$ $C \longleftarrow$ cost of split defined by $\bar{\theta_0}, \bar{\Theta}$;
            if $C \leq$ best_var then
                best_var $\longleftarrow C$;
                $\theta_0 \longleftarrow \bar{\theta_0}$;
                $\Theta \longleftarrow \bar{\Theta}$ ;
    return $\Theta, \theta_0$;
```

Here, we first use partial least squares regression (PLS) model that will try to find the multidimensional direction in the features space that explains the maximum multidimensional variance direction in the outcome space. The model uses orthogonal matrices in order to project X and Y into a latent space in which their projections have maximal covariance. We take the columns of the matrix used to rotate X, and search for best split along these directions. We didn't yet manage to make it work in the C# platform, but we have an RF implementation in python which performs well for the Parkinson dataset but as we didn't write the wavelet decomposition for this, it's hard to compare using results performance over train/test/validation datasets alone.

Results. The following tables show errors (mean and std) of the inspected datasets for the two methods we were able to implement, and the α coefficient.

	mean	std	dataset
method			
2MeansSVM	7.158	0.650	Concrete
Iso	1.831	0.542	Concrete
PCA	6.472	0.546	Concrete
2MeansSVM	2.611	0.281	AirFoil
Iso	0.564	0.215	AirFoil
PCA	3.724	0.207	AirFoil
2MeansSVM	3.747	0.139	Protein
Iso	1.049	0.317	Protein
PCA	2.860	0.217	Protein
2MeansSVM	118.624	10.641	ToyData
Iso	42.063	12.979	ToyData
PCA	78.573	11.230	ToyData
2MeansSVM	5.760	0.190	Parkinson
Iso	1.056	0.377	Parkinson
PCA	4.213	0.343	Parkinson

Table 1. Train error

	mean	std	dataset
method			
2MeansSVM	6.916	0.788	Concrete
Iso	1.587	0.559	Concrete
PCA	6.295	0.528	Concrete
2MeansSVM	2.600	0.345	AirFoil
Iso	0.564	0.252	AirFoil
PCA	3.558	0.249	AirFoil
2MeansSVM	3.718	0.138	Protein
Iso	1.073	0.302	Protein
PCA	2.842	0.213	Protein
2MeansSVM	116.359	10.804	ToyData
Iso	40.518	11.907	ToyData
PCA	77.152	11.444	ToyData
2MeansSVM	5.608	0.200	Parkinson
Iso	1.021	0.393	Parkinson
PCA	4.153	0.355	Parkinson

Table 2. Validation error

method	mean	std	dataset
2MeansSVM	8.461	0.985	Concrete
Iso	5.198	0.883	Concrete
PCA	8.867	1.046	Concrete
2MeansSVM	3.038	0.363	AirFoil
Iso	1.889	0.284	AirFoil
PCA	4.559	0.336	AirFoil
2MeansSVM	4.290	0.149	Protein
Iso	3.835	0.345	Protein
PCA	4.203	0.224	Protein
2MeansSVM	143.372	11.906	ToyData
Iso	122.088	16.475	ToyData
PCA	125.165	13.071	ToyData
2MeansSVM	6.641	0.210	Parkinson
Iso	3.631	0.493	Parkinson
PCA	6.393	0.368	Parkinson

Table 3. Test error

method	mean	std	dataset
2MeansSVM	0.119	0.002	Concrete
Iso	0.255	0.009	Concrete
PCA	0.116	0.003	Concrete
2MeansSVM	0.133	0.003	AirFoil
Iso	0.246	0.001	AirFoil
PCA	0.072	0.003	AirFoil
2MeansSVM	0.046	0.000	Protein
Iso	0.139	0.000	Protein
PCA	0.060	0.000	Protein
2MeansSVM	0.070	0.001	ToyData
Iso	0.153	0.013	ToyData
PCA	0.094	0.001	ToyData
2MeansSVM	0.037	0.000	Parkinson
Iso	0.199	0.004	Parkinson
PCA	0.057	0.002	Parkinson

Table 4. α