Kernel Matching Pursuit for Large Datasets

Vlad Popovici [a,*] Samy Bengio [b] Jean–Philippe Thiran [a]

[a] *Ecole Polytechnique Fédérale de Lausanne (EPFL)*
Signal Processing Institute
CH-1015 Lausanne, Switzerland

[b] *IDIAP Research Institute*
CP 592 rue du Simplon 4
CH-1920 Martigny, Switzerland

Abstract

Kernel Matching Pursuit is a greedy algorithm for building an approximation of a discriminant function as a linear combination of some basis functions selected from a kernel–induced dictionary. Here we propose a modification of the Kernel Matching Pursuit algorithm that aims at making the method practical for large datasets. Starting from an approximating algorithm, the Weak Greedy Algorithm, we introduce a stochastic method for reducing the search space at each iteration. Then we study the implications of using an approximate algorithm and we show how one can control the trade–off between the accuracy and the need for resources. Finally we present some experiments performed on a large dataset that support our approach and illustrate its applicability.

Key words:
kernel matching pursuit, greedy algorithm, sparse classifier

1 Introduction

Recently, a number of machine learning techniques that produce sparse classifiers have been proposed. We call a classifier sparse if it can be represented

*

Email addresses: `Vlad.Popovici@epfl.ch` (Vlad Popovici), `bengio@idiap.ch` (Samy Bengio), `JP.Thiran@epfl.ch` (Jean–Philippe Thiran).

URLs: `http://itswww.epfl.ch/~vlad` (Vlad Popovici),
`http://www.idiap.ch/~bengio` (Samy Bengio),
`http://itswww.epfl.ch/~thiran` (Jean–Philippe Thiran).

Article published in Pattern Recognition (in press) (2005)

as a combination, usually linear, of some basis functions that depend only on a small proportion of the training examples. Probably the most widely used and well known are the Support Vector Machines (SVMs) [1], while alternatives include the Relevance Vector Machine (RVM) [2] and the Kernel Matching Pursuit (KMP) [3]. The sparsity is achieved either as a result of the constraints imposed, like in the case of SVM, or as a consequence of explicit search for the sparsest model, like in KMP or RVM.

However, when faced with large datasets (in the order of tens of thousands or more examples) all these techniques become less practical, requiring huge amounts of resources, both in memory and CPU time. The main focus of this paper is to propose a modification of one of these algorithms – the Kernel Matching Pursuit – so that it becomes tractable to train using large datasets with a reasonable amount of resources. The basis of the proposed modification are set by an approximation of the Matching Pursuit, known as the Weak Greedy Algorithm. Using this framework, we introduce a stochastic method for constructing the classification function. We then study the implications of such a modification and, more importantly, the degradation of the accuracy, which is inherent in any approximating algorithm.

The paper is organized as follows: after reviewing in Section 2 the KMP algorithm and its approximated variant, we present the stochastic extension of KMP and empirically study its behavior on a large dataset in Sections 3 and 4 respectively. Finally, we draw some conclusions in Section 5.

2 Basic and Weak KMP

KMP is a new algorithm that has recently been introduced as an application of the Matching Pursuit method from signal processing domain to the case of pattern classification [3]. Its main advantages stem from its conceptual simplicity and the sparsity of the models it produces. Despite its deceiving simplicity, it achieves performance comparable with that of more complex classifiers, like SVMs [3], but using models that are much sparser.

The problem of learning classification functions from examples can be formally stated as an estimation problem of a function $\hat{f} : X \subseteq \mathbb{R}^p \to Y = \{-1, 1\}$ using the training set $Z_l = \{z_i = (\mathbf{x}_i, y_i) | i = 1, \ldots, l\} \subset Z = X \times Y$ generated by some unknown function f^*, such that $\hat{f}$ will correctly classify unseen examples $z = (\mathbf{x}, y)$, i.e. $\hat{f}(\mathbf{x}) = y$ for examples z that are drawn from the same underlying probability distribution $P(Z)$ as the training data. It is usually convenient to consider firstly a more general problem, where the goal is to find a function $f : X \to \mathbb{R}$ such that a suitably chosen optimality criterion is satisfied, and then to take $\hat{f} = \mathrm{sign}(f)$. For simplicity, we will consider only the case of square–error loss function, $\mathcal{L}_2[f] = \frac{1}{2}(y_i - f(\mathbf{x}_i))^2$, but the discussion remains valid for other types of loss functions as well. Furthermore, it is well known that without any restriction on the class of functions f can be chosen from, even if the performance on the training set is good (e.g. $\hat{f}(\mathbf{x}_i) = y_i, \forall i = 1, \ldots, l$) it does not mean $\hat{f}$ generalizes well to unseen examples and regularization techniques must thus be used when searching for $\hat{f}$ [4].

In the case of KMP, one builds an approximation of the classification function as a linear combination of some basis functions selected from a dictionary $\mathcal{D}$:

$$f_n(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k g_k(\mathbf{x}), \tag{1}$$

where $g_i \in \mathcal{D} = \{g_1, \ldots, g_m\}, \alpha_i \in \mathbb{R}, \forall i = 1, \ldots, n$ and n is the number of terms in the expansion. The generalization capabilities of the classifier are controlled through the structure of the dictionary and the number of terms in the summation of Eq. (1). This form of the classification function is not specific to KMP but can be found in other methods like SVMs, additive models, Radial Basis Functions, neural networks and so forth [5,6].

The general Matching Pursuit algorithm constructs the approximation in a greedy manner, iteratively improving the current solution by minimizing the norm of the *residual* $\|R_n\|^2 = \|f^* - f_n\|^2$. Then, for any $n \geq 0$ we define the new approximation f_{n+1} to be

$$f_{n+1} = f_n + \alpha_{n+1} g_{n+1}, \quad f_0 = 0, \tag{2}$$

where

$$(\alpha_{n+1}, g_{n+1}) = \arg\min_{\substack{\alpha \in \mathbb{R} \\ g \in \mathcal{D}}} \left\| f^* - \left(\sum_{k=1}^{n} \alpha_k g_k + \alpha g \right) \right\|^2. \tag{3}$$

In the space of functions, the value of g that minimizes (3) is the one that maximizes $|\langle g, R_n \rangle| / \|g\|$, in other words it is the function that is most correlated with the current residual. The corresponding value for α is $\alpha_{n+1} = \langle g_{n+1}, R \rangle / \|g_{n+1}\|^2$. Note that for our case the functions are seen as vectors of values representing the evaluation of the function on the training set, so the true labeling function f^* is merely the vector of training labels $(y_1, \ldots, y_l)^t$ and f_n should be seen as the vector $(f_n(\mathbf{x}_1), \ldots, f_n(\mathbf{x}_l))^t$. Finally, when the dictionary functions g_k are generated by some kernel–like function, $g_k(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}_k)$ we obtain the basic Kernel Matching Pursuit algorithm [3]. Keeping this observation in mind, the KMP method is described in Algorithm 1.

We notice that the bottleneck of the algorithm is represented by the search of the next element from the dictionary to be added in the function expansion (line 4 of Algorithm 1). Usually, this requires a full search over the whole

Algorithm 1: Kernel Matching Pursuit [3]

input : a dataset $Z_l = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, l\}$, a kernel function $\kappa(\cdot, \cdot)$ and the number of iterations N.

output: $f_N(\mathbf{x}) = \sum_{k=1}^{N} \alpha_k g_k(\mathbf{x})$

build the dictionary matrix: $[D_{ij}] = g_j(\mathbf{x}_i) = \kappa(\mathbf{x}_i, \mathbf{x}_j) \quad \forall i, j = 1, \ldots, l$ and let d_j be the j–th column of D;

initialize the residual: $R \longleftarrow (y_1, \ldots, y_m)^t$;

for $n = 1, \ldots, N$ **do**

$\quad k_n \longleftarrow \arg\max_{k=1,\ldots,l} \dfrac{|\langle d_k, R\rangle|}{\|d_k\|}$;

$\quad \alpha_n \longleftarrow \dfrac{\langle d_{k_n}, R\rangle}{\|d_{k_n}\|^2}$;

$\quad R \longleftarrow R - \alpha_n d_{k_n}$;

end

dictionary (with the computation of all inner products $\langle d_k, R\rangle$) and may necessitate a large number of floating point operations. An alternative is provided by the so–called *Weak Greedy Algorithms* (WGA) [7] which provide an approximation of the MP and related greedy algorithms. WGA and its different formulations have been analyzed in [7] and [8] and proofs of convergence of the algorithm exist for various conditions. Basically, WGA generates an approximant sequence

$$\tilde{f}_{n+1} = \tilde{f}_n + t_{n+1}\alpha_{n+1}g_{n+1} = \tilde{f}_n + \tilde{\alpha}_{n+1}g_{n+1}, \quad t_{n+1} \in [0, 1], \tag{4}$$

where α_{n+1} and g_{n+1} are defined as before, and $\tilde{\alpha}_{n+1} = t_{n+1}\alpha_{n+1}$. Clearly, for $t_n = 1, \forall n$, one retrieves the original algorithm. The sequence $\tau = \{t_n | n \geq 1\}$ is called *weakness sequence* and it must obey some constraints for the algorithm to converge. While different conditions on τ result in different guaranteed convergence rates, we will simply require that $\exists \tilde{t} > 0$ such that $t_n \geq \tilde{t}, \forall n \geq 1$ [8], which ensures the convergence. These modifications imply that we are no longer forced to produce the global maximum at each iteration of the algorithm, but just a value that represents a fraction of this maximum. It is obvious that the closer we are to the global maximum, the smaller is the degradation of the performance compared with the original algorithm. In the next section we will describe a strategy for constructing the sequence $\tilde{f}_n$ (and, implicitly, the weakness sequence τ) that exploits this approximate MP algorithm and which greatly reduces the computation times. As a final observation, we note the similarity between the term t_{n+1} from (4) and other regularization terms, like shrinkage or learning rate parameters, commonly found in machine learning algorithms [9].

3 Stochastic KMP

We start by showing that in order to find the approximate maximum of a sequence of numbers one can restrict the search space to a limited subsample of the sequence, and that the size of this subsample does not depend on the size of the original set.

Let us assume that we are given a sample $\{z_1, \dots, z_s\}$ of real values generated by a probability density function $p(z)$, and let $P(z)$ be the corresponding cumulative distribution function. Let now $z^{(k)}$ denote the k–th order statistic. Then the probability distribution function of $z^{(k)}$ is given by [10], p. 22:

$$p_k(z) = \frac{s!}{(k-1)!(s-k)!} \left[P(z)\right]^{k-1} \left[1 - P(z)\right]^{s-k} p(z), \tag{5}$$

for all $k = 1, \dots, s$. We note that

$$\max\{z_1, \dots, z_s\} = z^{(s)} \tag{6}$$

and it follows that the distribution of the maximum is given by

$$p_s(z) = s \left[P(z)\right]^{s-1} p(z) \tag{7}$$

Integrating, we obtain the cumulative distribution of $z^{(s)}$ as

$$P_s(z) = \left[P(z)\right]^s. \tag{8}$$

This result justifies the following

Proposition 1 *The distribution of the random variable* $\zeta = \max\{z_1, \dots, z_s\}$ *is given by* $[P(\zeta)]^s$*, where* $z_1, \dots, z_s$ *are* s *independent and identically distributed random variables and* P *is the cumulative distribution function.*

Assuming a uniform distribution for z_k we obtain the distribution of the maximum as being ζ^s. In the case of the dictionary for KMP algorithms, if we do not have any information about the distribution of the examples, we can assume that the values of $|\langle g, R\rangle|/\|g\|$ are uniformly distributed and we will empirically show in the experimental section that this is a pessimistic approximation. We are then interested in finding the number of elements that must be considered from the full dictionary such that their maximum has a quantile of at least, say, q, with a given probability. Using Proposition 1 and the fact that the cumulative distribution of the maximum is ζ^s, we can formulate the answer as

Proposition 2 *Assuming a uniform distribution of* z*, the maximum of a sample* $\{z_1, \dots, z_s\}$ *has a quantile of at least* $\varepsilon^{\frac{1}{s}}$ *with probability* $1 - \varepsilon$*.*

The practical consequence of this result is that we may take the maximum of $s = \lceil \log \varepsilon / \log q \rceil$ randomly chosen elements from the full dictionary and we still have $1 - \varepsilon$ probability of having a value that has a quantile of q. For example, assume that we have 10000 examples in the training set and we want a value that has a quantile $q = 0.95$ (within the largest 5% values) with probability 95%. Then we only need to take the maximum of a subsample of $\lceil \log 0.05 / \log 0.95 \rceil = 59$ elements, which means merely 0.59% of the original set.

We can now define the stochastic version of KMP. Let $I_n^{(s)} \subset \{1, \ldots, l\}$ be a set of $s \leq l$ indices, called *active set* at iteration n. $I_n^{(s)}$ is obtained by randomly sampling (without replacement) from the full set of indices $\{1, \ldots, l\}$ and we restrict the search for the maximum to the set $\{|\langle g_k, R\rangle| / \|g_k\|, k \in I_n^{(s)}\}$. Then the Stochastic KMP (SKMP) algorithm requires replacing line 4 of Algorithm 1 with

$$\text{generate active set } I_n^{(s)}; \quad k_n \longleftarrow \arg\max_{k \in I_n^{(s)}} \frac{|\langle d_k, R\rangle|}{\|d_k\|}.$$

It is clear that these modifications lead to a weak greedy algorithm which is convergent, for the weakness sequence produced contains only positive elements as long as $I_n^{(s)} \neq \emptyset$. Moreover, by controlling the values of ε and q one can control the trade–off between the speed (the smaller s the faster the algorithm) and the accuracy (the larger s the closer we are to the original KMP) of the algorithm.

Finally, we note that a similar method was proposed in [11] for speeding up the optimization of SVMs.

4 Experiments

The main goal of the experiments reported here is to investigate the behavior of the approximate version of the KMP and to compare it with the original KMP. In all our experiments we used the largest dataset available in the UCI repository [12] – the *Forest* dataset. We transformed the original multi–class problem into a binary classification task where the goal was to discriminate class 2 from all the other six classes, this kind of partitioning making the two new classes of roughly the same size. As we were also interested in analyzing the performance of the algorithm on training sets having different cardinality, we have created 4 different disjoint training sets of respectively 5000, 10000, 15000 and 50000 elements, by randomly selecting examples from the original set. Finally, we have selected an independent test set of 30000 examples.

When using the stochastic KMP one has to select before starting the number of elements in the active set. In our experiments we used two different settings, one with 59 elements and one with 228 elements, corresponding to ($\varepsilon = 0.05, q = 0.95$) and ($\varepsilon = 0.01, q = 0.98$), respectively. In order to analyze the influence of the randomness introduced in the algorithm, we have repeated 10 times each of the experiments involving the Stochastic KMP. Finally, we have also trained an SVM for each of the four different training sets (the dashed line in the plots) and use it as a baseline classifier for comparison. The results are depicted in Figure 1 where the rows correspond to the four different training sets, while the columns correspond to respectively 59 and 228 elements in the active set. On the horizontal axes are the iterations of KMP (from 200 to 2400) and on the vertical the error rates. The continuous line corresponds to the error rate of the classical KMP, while the dashed line correspond to the error rate of a SVM. The error rates of the stochastic KMP are given as boxplots.

The kernel used in all experiments was a Gaussian kernel, $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\gamma\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$, and we let $\gamma = 0.5$ for (S)KMP and $\gamma = 0.03185$ for SVM with the γ parameter chosen by 10–fold cross–validation on a 5000 example set. As one can notice from Figure 1, increasing the training set size results in significantly better performance of all classifiers. What is also interesting to note is that (S)KMP algorithms generally reach the same error rate as SVM but using much sparser models: for the four datasets (in increasing order of their cardinality) the SVMs had 2643, 5174, 7529 and 23048 support vectors respectively, while the (S)KMP had the number of terms upper bounded by the number of iterations (at most 2400). The case of the training set having 50000 examples requires more iterations for KMP to reach the same classification error as SVM.

Comparing the two cases for SKMP, corresponding to the two different dimensions of the active set, we see that in the second case the degradation of the performance is much less significant then in the first case, as predicted by Proposition 2. Another important observation is that the behavior of SKMP mimics the one of its deterministic counterpart. This means that an equivalent level of performance can be reached given more iterations, but this increase in the number of iterations is largely compensated by the reduced number of floating–point operations needed to compute the inner products $\langle d_k, R\rangle$.

The main advantage of SKMP over KMP is due to the smaller number of floating point operations needed: indeed, if one disregards the overhead caused by the sampling process for building the active set, the only difference between the two algorithms resides in the way they search for the next basis function: while KMP performs a full search – and the number of inner products computed equals the number of examples l – the SKMP computes just s inner products. This means that while the complexity of KMP is linear, the one of SKMP is constant. Moreover, if we approximate the gain of speed as a function of s/l, the gain will be more significant with the increase of the cardinality of the training set.

Finally, we have empirically observed that the distribution of the values $|\langle d_k, R\rangle|$ is not uniform – as initially assumed – but rather exponential, for the particular type of kernel and training set used. This means that our initial hypothesis gives a pessimistic approximation of the maximum: for an exponential density of the form $f(z) = \exp(-z)$ the equivalent of Proposition 2 would guarantee a quantile of at least $-\ln(1-\varepsilon^{1/s})$ with probability $1-\varepsilon$, which is a better quantile than $\varepsilon^{1/s}$ ($\varepsilon \in [0,1)$). Nevertheless, Proposition 2 gives a good conservative estimate.

5 Conclusions

We have presented a stochastic version of the KMP algorithm that has the advantage of greatly reducing the computational overhead. While being an approximation of the original KMP, the approach described produces comparable results: for example, the decrease in accuracy, for 59 elements in the active set, was around $0.5\% - 1\%$, and this error became less significant with the increase in the number of iterations. On the other hand, one can control the trade–off between accuracy and speed by choosing a suitable value for ε and q parameters that determine the size of the active set.

Acknowledgements

This work has been performed with the financial support of the IM2–NCCR project of the Swiss NSF.

References

[1] B. Boser, I. Guyon, V. Vapnik, An algorithm for optimal margin classifiers, Fifth Annual Workshop on Computational Learning Theory, (1992), 144–152.

[2] Michael E. Tipping, Sparse Bayesian Learning and the Relevance Vector Machine, Journal of Machine Learning Research 1 (2001) 211–244.

[3] P. Vincent, Y. Bengio, Kernel matching pursuit, Machine Learning Journal 48 (1) (2002) 165–187.

[4] V. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995.

[5] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[6] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Wiley & Sons, 2001

[7] V. Temlyakov, Weak greedy algorithms, Advances in Computational Mathematics 12 (2,3) (2000) 213–227.

[8] R. Gribonval, M. Nielsen, Approximate weak greedy algorithms, Advances in Computational Mathematics 14 (4) (2001) 361–378.
URL `http://www.math.sc.edu/~imip/00papers/0016.ps`

[9] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning. Data Mining, Inference and Prediction, Springer–Verlag, 2001

[10] K.V. Bury, Statistical distributions in engineering, Cambridge University Press, 1999.

[11] B. Schölkopf, A.J. Smola, Learning with Kernels–Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, 2002

[12] C. Blake and C. Merz. UCI repository of machine learning databases. Technical report, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.
URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`

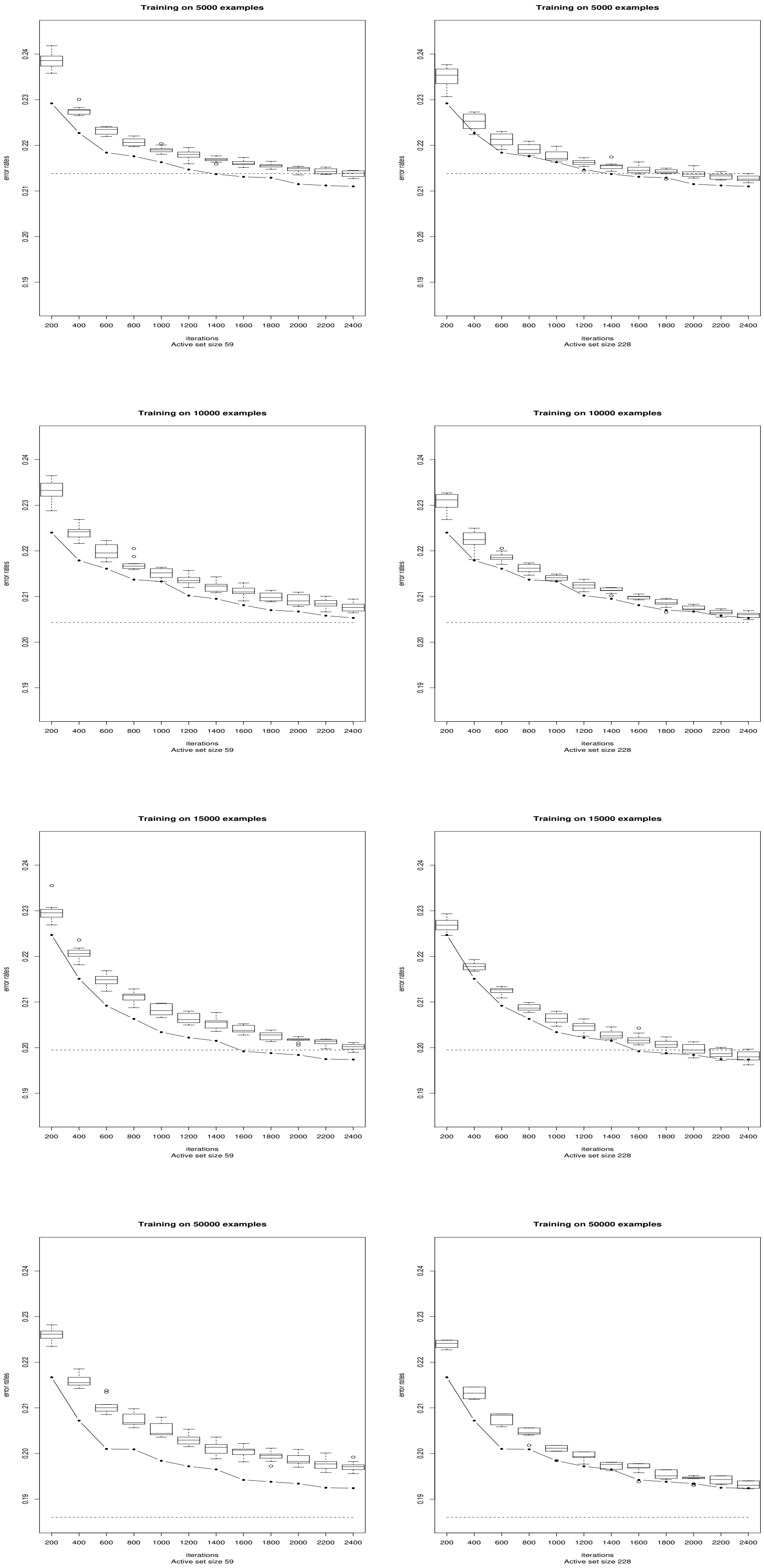

Fig. 1. Error rates on the testing set using models obtained by training on sets of different cardinality.