Kernel Matching Pursuit

Pascal Vincent

[vincentp@iro.umontreal.ca]

in collaboration with

Yoshua Bengio

Department of Computer Science and Operations Research

Université de Montréal

Matching Pursuit

- Matching Pursuit was introduced in the signal-processing community as an algorithm *"that decomposes any signal into a linear expansion of waveforms that are selected from a redundant dictionary of functions."* (Mallat and Zhang, 1993)
- Given a dictionary $\mathcal{D} = \{g_\gamma, \gamma \in \Gamma\}$ of functions in a Hilbert space $\mathcal{H}$ and a target function $f \in \mathcal{H}$ we are interested in expansions of the form

$$\tilde{f}_n = \sum_{k=1}^{n} \alpha_k g_{\gamma_k} \tag{1}$$

  where the α_k and $g_{\gamma_k} \in \mathcal{D}$ are chosen to minimize the squared norm of the residue $\|R_n\|^2 = \|f - \tilde{f}_n\|^2$.
- However finding the optimal solution for a given n is in general NP-complete. So we proceed in a suboptimal, greedy constructive fashion.

Simple Matching Pursuit

- The algorithm proceeds in an iterative fashion, starting at stage 0 with $\tilde{f}_0 = 0$
- Then, given the expansion at a stage $n-1$ we find the expansion at stage n:

$$\tilde{f}_n = \tilde{f}_{n-1} + \alpha_n g_{\gamma_n}$$

by searching for g_{γ_n} among the functions in the dictionary and for $\alpha_n \in \mathbf{R}$ that minimize the squared norm of the residue $\|f - \tilde{f}_n\|^2 = \|R_n\|^2 = \|R_{n-1} - \alpha_n g_{\gamma_n}\|^2$

$$(g_{\gamma_n}, \alpha_n) = \underset{(g \in \mathcal{D}, \alpha \in \mathbf{R})}{\arg\min} \| \underbrace{(\sum_{k=1}^{n-1} \alpha_k g_{\gamma_k})}_{\tilde{f}_{n-1}} + \alpha g - f \|^2 \qquad (2)$$

- The g_{γ_n} that minimizes this expression is the one that maximizes $|\frac{<g_{\gamma_n}, R_{n-1}>}{\|g_{\gamma_n}\|}|$ and the corresponding α_n is $\alpha_n = \frac{<g_{\gamma_n}, R_{n-1}>}{\|g_{\gamma_n}\|^2}$

Orthogonal Matching Pursuit

- In the simple version of the algorithm, not only is the set of basis functions $g_{\gamma_{1..n}}$ obtained at every step n suboptimal, but so are also their $\alpha_{1..n}$ coefficients. This can be corrected in a step called back-projection and the resulting algorithm is known as Orthogonal Matching Pursuit (OMP) (Pati, Rezaiifar and Krishnaprasad, 1993; Davis, Mallat and Zhang, 1994):
- After choosing g_{γ_n} as previously (equation 2), we compute the optimal set of coefficients:

$$\alpha_{1..n}^{(n)} = \underset{(\alpha_{1..n} \in \mathbf{R}^n)}{\arg\min} \|(\sum_{k=1}^{n} \alpha_k g_{\gamma_k}) - f\|^2 \qquad (3)$$

- This back-projection step has a geometrical interpretation:

  Let B_n the sub-space of $\mathcal{H}$ spanned by the basis $(g_{\gamma_1}, \ldots, g_{\gamma_n})$ and let $B_n^{\perp} = H - B_n$ be its orthogonal complement. Let P_{B_n} and $P_{B_n^{\perp}}$ denote the projection operators on these subspaces.

  Then, any $g \in \mathcal{H}$ can be decomposed as $g = P_{B_n} g + P_{B_n^{\perp}} g$

Orthogonal Matching Pursuit

- Ideally, we want the residue R_n to be as small as possible, so given the basis at step n, we want $\tilde{f}_n = P_{B_n} f$ and $R_n = P_{B_n^\perp} f$. This is what (3) insures.
- Whenever we append the next $\alpha_n g_{\gamma_n}$ found by (2) to the expansion, we actually add its two components:
  - $P_{B_{n-1}^\perp} \alpha_n g_{\gamma_n}$ contributes to reducing the norm of the residue.
  - $P_{B_{n-1}} \alpha_n g_{\gamma_n}$ which increases the norm of the residue.

  But as the latter part belongs to $P_{B_{n-1}}$ it can be compensated for by adjusting the previous coefficients of the expansion.
- (Davis, Mallat and Zhang, 1994) suggest maintaining an additional orthogonal basis of the B_n space to facilitate the back-projection.

Optimally Orthogonal Matching Pursuit

- The orthogonal version just described computes at every step the optimal expansion for the chosen basis $(g_{\gamma_1}, \ldots, g_{\gamma_n})$. Yet the choice of g_{γ_n} is made regardless of the later possibility to update the coefficients, and is thus sub-optimal.
- In the case where $\mathcal{H}$ has a finite dimension we suggest a further improvement that allows us to find the optimal

$$(g_{\gamma_n}, \alpha_{1..n}^{(n)}) = \underset{(g \in \mathcal{D}, \alpha_{1..n} \in \mathbf{R}^n)}{\arg\min} \|(\sum_{k=1}^{n-1} \alpha_k g_{\gamma_k}) + \alpha_n g - f\|^2 \qquad (4)$$

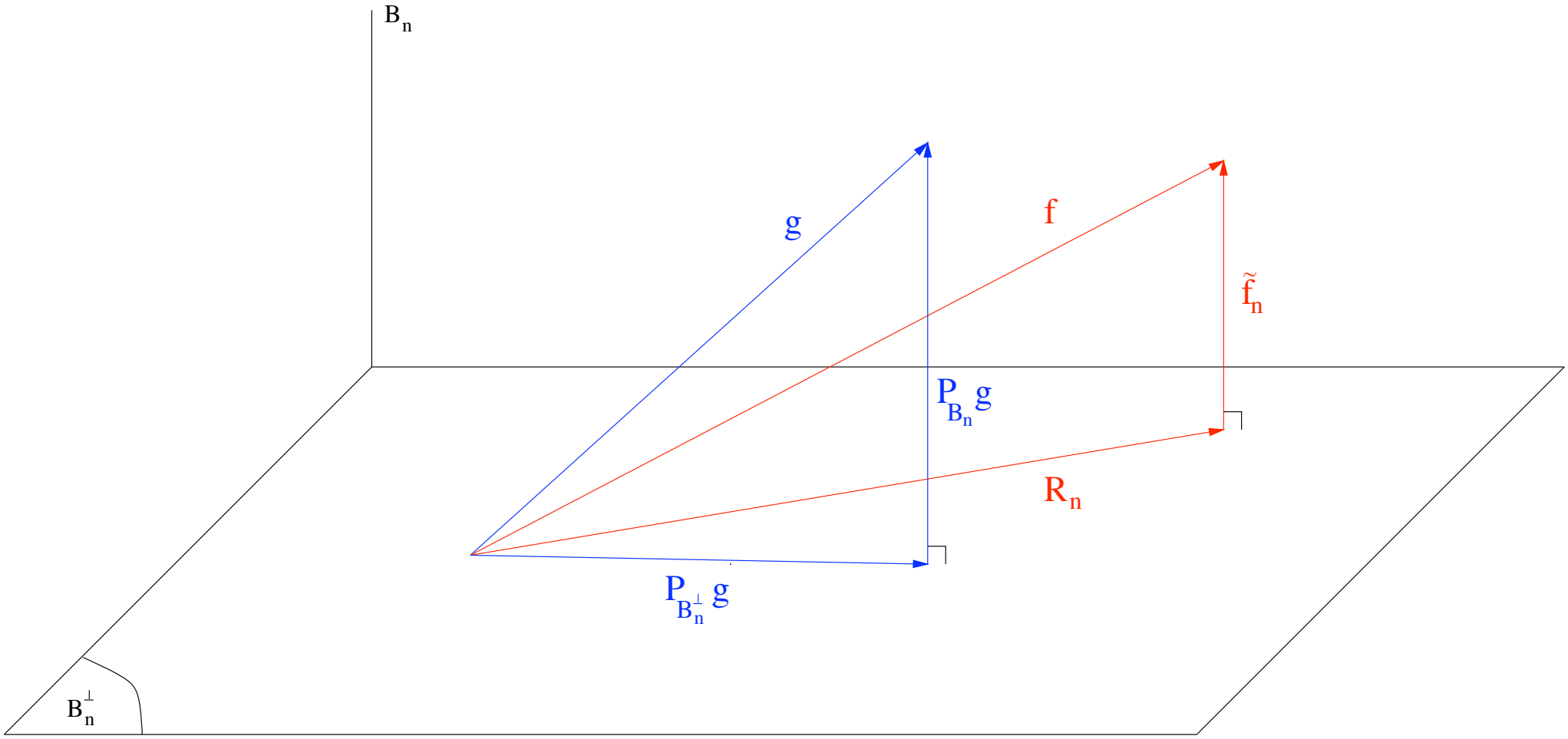
- The trick is to maintain, at every step, for every vector g of the dictionary, a decomposition into components $P_{B_n} g$ and $P_{B_n^\perp} g$, where $P_{B_n^\perp} g$ is expressed in the original coordinate system of $\mathcal{H}$, but $P_{B_n} g$ is expressed in the coordinate system of B_n (i.e. as a linear combination of the basis vectors $(g_{\gamma_1}, \ldots, g_{\gamma_n})$).
- We also maintain this representation for the target:

Optimally Orthogonal Matching Pursuit

$$f = \underbrace{P_{B_n} f}_{\tilde{f}_n} + \underbrace{P_{B_n^\perp} f}_{R_n}$$

- Now, when we search the next vector $n+1$ to append to the basis and its α_{n+1} coefficient, we consider only the components in $B_n^\perp$ (as only they can reduce the current residue R_n).
- We then cancel the negative effect of adding the B_n part of $\alpha_{n+1} g_{\gamma_{n+1}}$ by subtracting $P_{B_n} \alpha_{n+1} g_{\gamma_{n+1}}$ from the basis, (which is easy, since we kept $g_{\gamma_{n+1}}$ expressed in this basis' coordinates).
- We then update the representations of the dictionary vectors in a similar way, to fit the new basis (requires a single pass through the dictionary, just like the search).
- Note that it may be useful to add a penalty on large α_{n+1} when choosing the best vector, or we might be picking up noise that happens to be strongly collinear with the current residue.

Geometrical interpretation



Summary of Matching Pursuit

The three versions of matching pursuit differ only in the way the next function to append to the basis is chosen and the α coefficients are updated at each step n:

- **Simple version:** We find the optimal g_{γ_n} to append to the basis and its optimal α_n, while keeping all other coefficients fixed (equation 2).
- **Orthogonal version:** We find the optimal g_{γ_n} while keeping all coefficients fixed (equation 2). Then we find the optimal set of coefficients $\alpha_{1..n}^{(n)}$ for the new basis (equation 3).
- **Optimally orthogonal version:** We find at the same time the optimal g_{γ_n} and the optimal set of coefficients $\alpha_{1..n}^{(n)}$ (equation 4).

Kernel Matching Pursuit: Motivations

- Renewed interest in kernel-based methods due to *Support Vector Machines* (SVMs) (Boser, Guyon and Vapnik, 1992)
- There is a link between number of support vectors and generalization $\Longrightarrow$ how to control it directly?
- A solution expressed as a function of some distance to a few particularly relevant support points from the training set makes sense, regardless of any *"Kernel-trick"* inducing a mapping into an artificial higher-dimensional *feature-space*.
- Search for a more flexible framework to investigate influence of kernel shapes and other loss functions.
- Further exploring the links between SVMs, boosting techniques, and sparse approximation.

Matching pursuits with kernel-based dictionary

- Kernel Matching Pursuit (KMP) is simply the idea of applying the Matching Pursuit family of algorithms to problems in Machine-Learning (currently classification), using as the dictionary, kernels centered on the training data points $\{x_1, \ldots, x_m\}$:

$$g_i(x) = K(x; x_i)$$

- During training, both dictionary elements and target f are seen as m dimensional vectors, as we only consider their value at the m training points. Targets $(y_1, \ldots, y_m) = (f(x_1), \ldots, f(x_m))$ are typically given values $+1$ and -1 for binary classification problems.

Similarities and differences with SVMs

- The resulting functional form is very similar to the one obtained with SVMs:

$$\underbrace{\tilde{f}(x) = \sum_{i=1}^{n} \alpha_i K(x; x_{\gamma_i})}_{\textit{Kernel Matching Pursuit}} \qquad \underbrace{\tilde{f}(x) = b + \sum_{i=1}^{n} \alpha_i y_i K(x; x_{\gamma_i})}_{\textit{Support Vector Machines}}$$

- The bias term b of the SVM expansion can actually be obtained by simply appending the constant function to the dictionary.
- Capacity-control is achieved by directly specifying the number n of support points, as opposed to box-constraint C in SVMs
- What is being optimized, however, is not the same as for SVMs but typically the mean squared error, and in a greedy fashion.

Experimental comparison with SVMs

A first series of experiment was done with the Delve system[a] on the binary classification problem using the *mushroom* data set. We compared the preformance of SVMs and KMP (based on Optimally Orthogonal Matching Pursuit) with a Gaussian Kernel.

For each training, we first chose an appropriate setting of hyper parameters[b] based on K-fold cross-validation. We then retrained with this setting on the whole training-set, and applied the obtained function to the independent test-set.

---

[a] *Data for Evaluating Learning in Valid Experiments* developped at the University of Toronto

[b] σ of the gaussian among $\{.5, 1, 2, 3, 4\}$ and C for SVMs among $\{.1, 1, 2, 3, 4, 5, 6, 10, 100\}$ and the number of support points for K.M.P. as a percentage of the training-set size among $\{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%\}$

Experimental comparison with SVMs

The table below compares the classification error rate and the number of support vectors found by both algorithms for varying sizes of the training set [a]

size of train	KMP error	SVM error	significance (t-test)	KMP #s.v.	SVM #s.v.
64	6.28%	4.54%	0.24	17	63
128	2.51%	2.61%	0.82	28	105
256	1.09%	1.14%	0.81	41	244
512	0.20%	0.30%	0.35	70	443
1024	0.05%	0.07%	0.39	127	483

[a] for each size, the delve system did its estimations based on 8 disjoint training sets of the given size and 8 disjoint test sets of size 503, except for 1024, in which case it used 4 disjoint training sets of size 1024 and 4 test sets of size 1007

Experimental comparison with SVMs

We did some further experiments, using the UCI Machine Learning Databases *Breast Cancer* and *Letters*, also with a Gaussian Kernel. Hyper-parameters were chosen in a similar way as previously, using K-fold cross validation, and we used the resampled t-test (Nadeau and Bengio, 2000) to compare performance.

data set	# train	KMP error	SVM error	significance (p-value)	KMP #s.v.	SVM #s.v.
breast-cancer	500	2.00%	2.70%	0.10	4	121
letters C vs. G	605	0.75%	0.58%	0.32	242	317

As previously, the performance of both algorithms are comparable, but KMP uses far fewer support vectors.

Dictionary gives additional flexibility

This approach appears to be extremely flexible, as it allows to put any kind of functions in the dictionary. For instance:

- Absolutely no restriction on the shape of the kernel (can be asymmetrical, position dependent, ...)
- Dictionary could mix different kernel shapes to choose from at each point, allowing for instance the algorithm to choose among several widths of a Gaussian for each support point.
- Dictionary could constrain the algorithm to use a different kernel shape for each class, based on prior-knowledge.
- For huge data-sets, a reduced subset can be used as the dictionary to speed up the training.
- Dictionary can incorporate non-kernel based functions (we already mentioned the constant function to recover the bias term, but this could also be used to incorporate prior knowledge).

Extension to non squared-error loss

- The original versions of Matching Pursuit algorithms were designed to optimize a squared error loss but (Friedman, 1999) offers a way to extend the procedure to arbitrary loss-functions.
- Given a loss function $L(y_i, \tilde{f}_n(x_i))$ that computes the cost of predicting a value of $\tilde{f}_n(x_i)$ when the true target was y_i, we can use an alternative residue $\tilde{R}_n$ rather than the usual $R_n = f - \tilde{f}_n$ when searching for the next dictionary element to append to our basis at each step.

  This alternate target residue can be seen as the gradient from a gradient descent in function space:

$$\tilde{R}_n = \left\{ -\frac{\partial L(y_i, \tilde{f}_n(x_i))}{\partial \tilde{f}_n(x_i)} \right\}_1^m \tag{5}$$

  and gives the usual residue when L is the squared error loss.

Extension to non squared-error loss

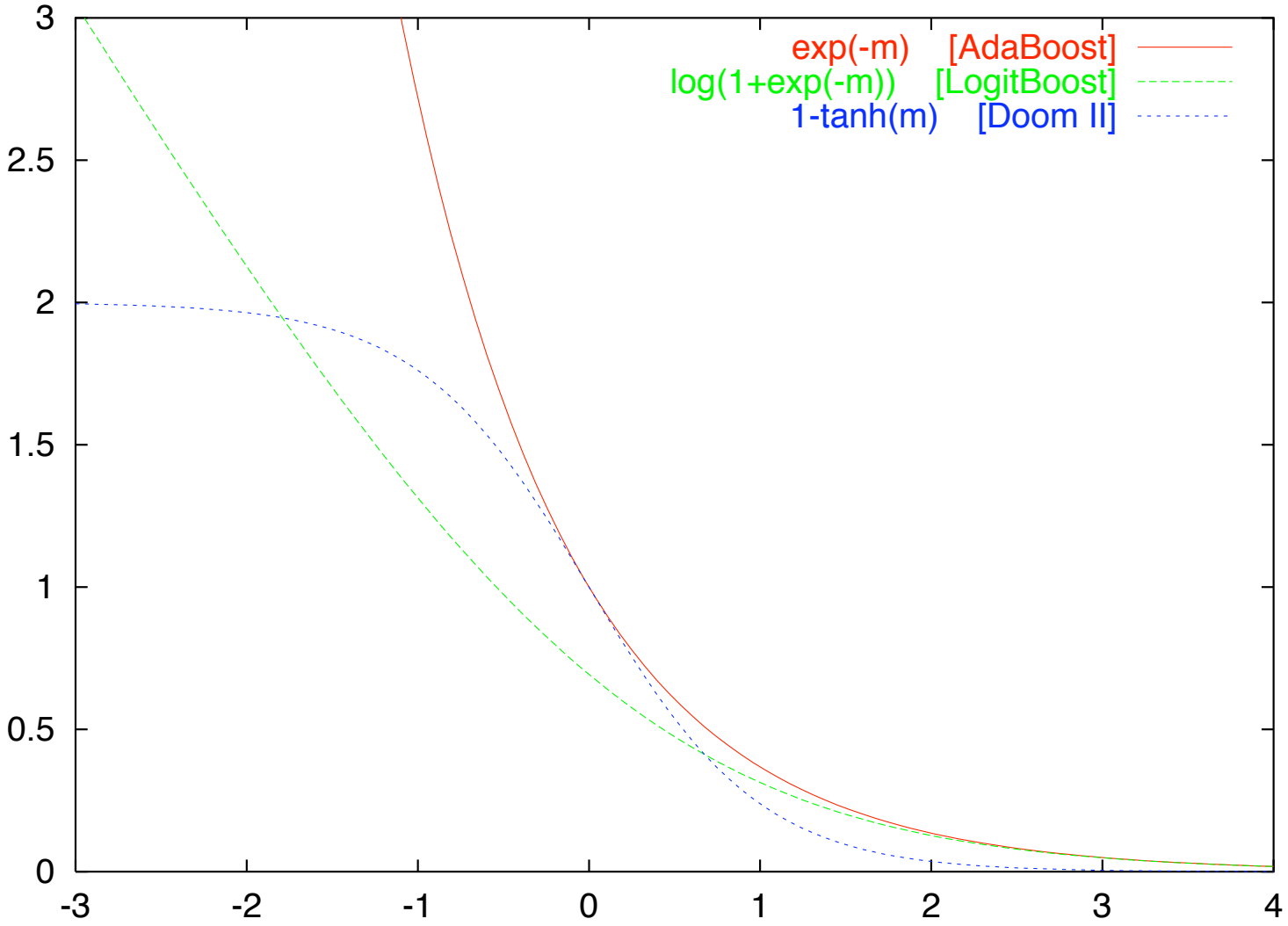
- Once the $g_{\gamma_{n+1}}$ is chosen that is most collinear with $\tilde{R}_n$, the corresponding α_{n+1} is found by directly minimizing (with conjugate gradient for instance):

$$\alpha_{n+1} = \arg\min_{\alpha \in \mathbf{R}} \sum_{i=1}^{m} L(f(x_i), \tilde{f}_n(x_i) + \alpha g_{\gamma_{n+1}}(x_i)) \qquad (6)$$

- This would be the equivalent of a simple matching pursuit. But we could also perform an OMP-like "back-projection" by re-optimizing all $\alpha_{1..n}$.
- Thus margin cost functions (Mason et al., 2000), could be used instead of squared-error loss, further closing the gap with SVMs

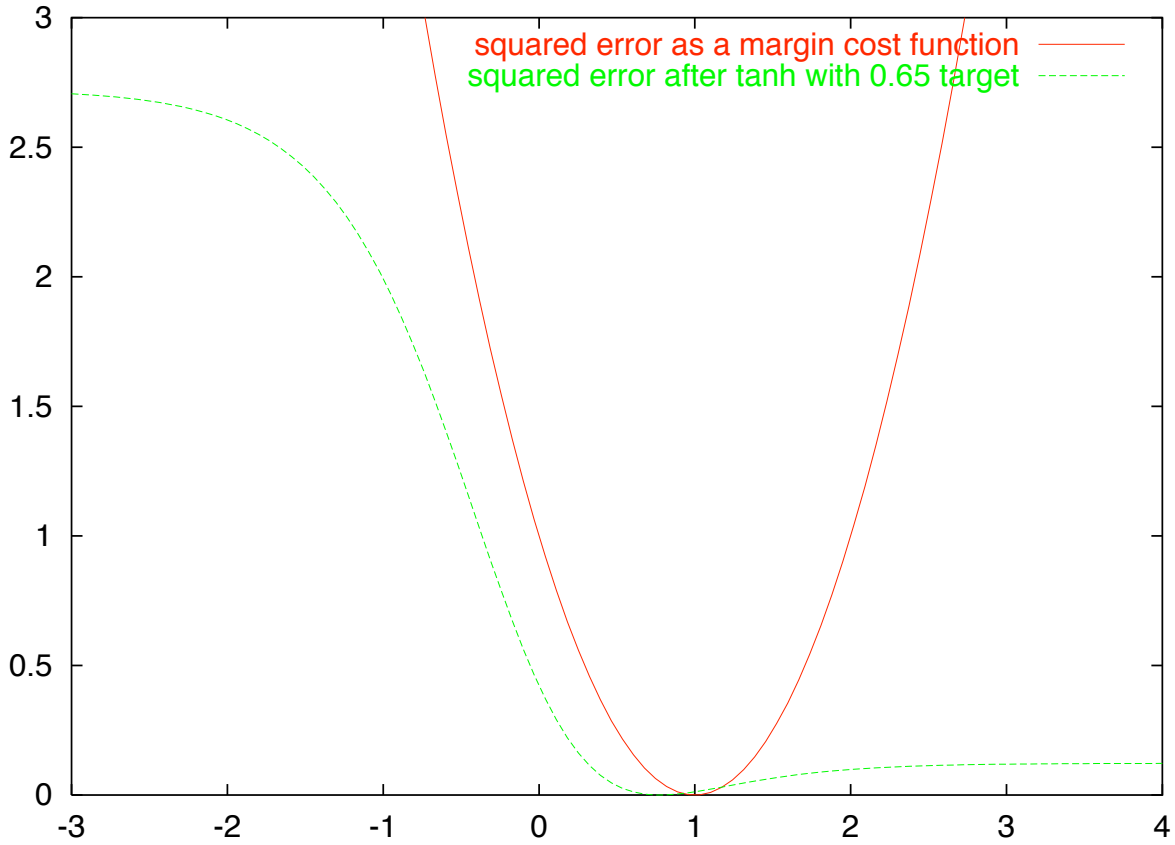
Margin cost functions

Margin $m = yf(x)$ with $y \in \{-1, +1\}$ can be seen as a confidence measure.



Margin cost functions

- Squared loss: $(f(x) - y)^2 = (1 - m)^2$
- Squared loss after tanh: $(tanh(f(x)) - 0.65y)^2 = (0.65 - tanh(m))^2$



Conclusion

In conclusion the Kernel Matching Pursuit family of algorithms provide an interesting alternate framework to explore properties of support-vector and kernel based solutions to machine-learning problems.

- The ability to directly specify the number of support points appears to be a more intuitive capacity-control parameter than the box-constraint C of SVM.
- It allows to enforce sparsity, and choose a trade-off between speed and accuracy
- Performance appears as good as SVM while often requiring far fewer support points.
- It is a very flexible framework that can be extended in many ways in a straight-forward manner, opening the way to further research.

Future work

In the future, we plan to explore the following:

- Influence of using different cost functions
- Mixing multiple kernel shapes
- Ways of reducing the training time (by using a support candidate subset).
- We will also attempt to derive theoretical bounds on the generalization error.

References

Boser, B., Guyon, I., and Vapnik, V. (1992). An algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh.

Davis, G., Mallat, S., and Zhang, Z. (1994). Adaptive time-frequency decompositions. *Optical Engineering*, 33(7):2183–2191.

Friedman, J. (1999). Greedy function approximation: a gradient boosting machine. Technical report, Dept. of Statistics, Stanford University.

Mallat, S. and Zhang, Z. (1993). Matching pursuit with time-frequency dictionaries. *IEEE Trans. Signal Proc.*, 41(12):3397–3415.

Mason, L., Baxter, J., Bartlett, P., and Frean, M. (2000). Boosting algorithms as gradient descent. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*. The MIT Press. Accepted for Publication.

Nadeau, C. and Bengio, Y. (2000). Inference for the generalization error. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*. The MIT Press. Accepted for Publication.

Pati, Y., Rezaiifar, R., and Krishnaprasad, P. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*.