Approximating the Kernel Trick with Random Feature Models

Shwetlana Jha,[1] Diego Miura,[2] and Yu-Ting Chang[2]

[1] *Yale University, Yale College*
[2] *Yale University, Department of Astronomy*

ABSTRACT

Kernel methods occupy an important place in machine learning because they allow simple linear algorithms to learn nonlinear patterns. This gives them both conceptual appeal and practical power. A classifier that would fail in the original input space can often succeed once the data is viewed through a richer feature representation. At the same time, the main strength of kernel methods becomes a weakness when the dataset grows large. The method relies on pairwise comparisons between training points, which leads to significant computational cost. Random feature models were introduced as a way to preserve much of the nonlinear power of kernels while avoiding the full burden of exact kernel computation. In that sense, they provide a useful compromise between expressive modeling and computational efficiency. In this work, we demonstrate this tradeoff numerically by comparing an exact Radial Basis Function (RBF) kernel, or Gaussian Kernel, with a random-feature approximation. Our results show that as the number of random features m increases, the approximation stability improves according to Monte Carlo scaling, achieving parity with exact kernel accuracy near $m = 500$ on a synthetic task. Ultimately, we show that random feature models serve as a practical approximation scheme that makes kernel methods more usable at a larger scale, provided that m is chosen to balance accuracy against increasing computational costs.

Keywords: Kernel Methods, Random Feature Models, Random Fourier Features, Nonlinear Classification

1. LINEAR CLASSIFIERS

In classical machine learning, a linear classifier attempts to draw a straight linear decision boundary in the original input space. Although this is computationally efficient, it is also fundamentally limited. It is effective only when the data is already close to linearly separable. In many realistic problems, that condition does not hold. A more flexible model is needed. One way to achieve that flexibility is to map each input x into a higher-dimensional feature space through some transformation $\phi(x)$. Doing so makes the the data significantly more likely to be linearly separable, according to Cover's theorem T. M. Cover (1965). In that transformed space, a linear boundary may become sufficient even when the original data looks highly nonlinear. In Figure 1, we see two classes of data points, red circles and blue triangles, that are not linearly separable. We can use a non-linear function to map the original points in two dimensions to three dimensions using the mapping $\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)$ to get Figure 2. The class of data points denoted using the blue triangles now lie lower in the three dimensional bowl and the points from the class of data points denoted using the red circles will be mapped to much higher z coordinates. Now, the data is linearly separable and can be visualized with a two dimensional plane cutting horizontally to separate all of the red circles from the blue triangles.

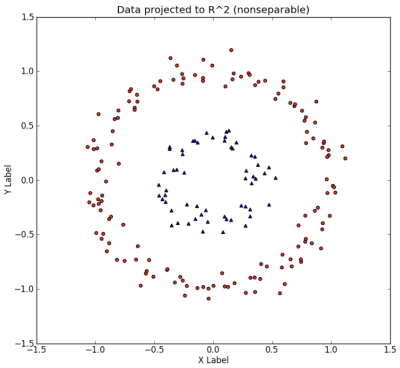

Figure 1. Two dimensional input space ($\mathbb{R}^2$) featuring two nonlinearly separable classes of data points. A circular decision boundary would be required here, which is unattainable for a standard linear classifier.E. Kim (2018)

We can go all the way to an infinite-dimensional Hilbert Space and linearize almost any distribution. The catch, of course, is that we can't explicitly compute an infinite vector. So, we must rely on the Kernel Trick.

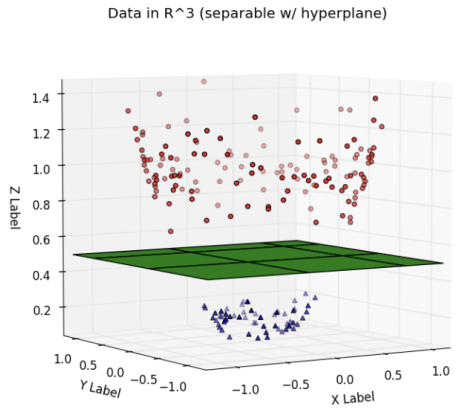

Figure 2. Lifting the data into a three dimensional feature space ($\mathbb{R}^3$) using the mapping $\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)$. The data points now lie on a paraboloid, allowing a two dimensional hyperplane to linearly separate the classes.E. Kim (2018)

2. KERNEL METHOD

The key idea behind kernel methods is that one often does not need to compute $\phi(x)$ explicitly. Instead, it is enough to know the inner product between transformed points:

$$k(x, x') = \phi(x)^\top \phi(x').$$

This quantity is called the kernel (B. Schölkopf & A. J. Smola 2002). It measures similarity in feature space without requiring the feature map itself to be written down. It lets a learning algorithm behave as though it is operating in a high-dimensional, or even infinite-dimensional, space while only working with kernel evaluations in the original space.

This kernel trick turns a difficult nonlinear problem into something that still looks linear, given the right similarity measure is chosen. Common kernels such as the Gaussian and polynomial kernels encode different notions of similarity, which gives the method a broad range of applications. Kernel methods combine the structure of linear learning with the expressive power of nonlinear feature spaces.

3. THE COMPUTATIONAL BOTTLENECK

The main practical limitation of kernel methods appears when the dataset becomes large. In a dataset with N training examples, kernel methods require the computation of an $N \times N$ kernel matrix. Each entry records the kernel evaluated on a pair of data points. For small datasets this is manageable. For large datasets it becomes expensive in both memory and runtime. This bottleneck motivates the search for an approximation that retains the central idea of kernel learning without requiring every pairwise interaction to be computed exactly.

4. RANDOM FEATURE MODELS

Random feature models address this problem by replacing the implicit feature map with an explicit finite-dimensional approximation. Rather than working with the exact kernel, we construct a random feature vector $z(x)$ such that

$$k(x, x') \approx z(x)^\top z(x').$$

The approximation converts the nonlinear kernel method into ordinary linear learning in the new feature space defined by $z(x)$. Instead of storing and using the full kernel matrix, we transform the data once and then apply a standard linear model.

For shift-invariant kernels, especially the Gaussian kernel, this construction has a natural theoretical basis through Bochner's theorem. The theorem says that certain kernels can be represented in terms of oscillatory building blocks. This leads to a random feature map of the form

$$z(x) = \sqrt{\frac{2}{m}} \begin{bmatrix} \cos(\omega_1^\top x + b_1) \\ \cos(\omega_2^\top x + b_2) \\ \vdots \\ \cos(\omega_m^\top x + b_m) \end{bmatrix},$$

where the vectors ω_i and phases b_i are sampled randomly. The integer m is the number of random features used in the approximation.

5. WHY THE APPROXIMATION WORKS

The kernel can be viewed as an average over many possible feature contributions. The random feature map estimates that average with a finite sample. Because of this, the inner product of the random features matches the kernel in expectation:

$$\mathbb{E}[z(x)^\top z(x')] = k(x, x').$$

While not exact for a fixed finite sample, it means the approximation is centered around the true kernel.

Additionally, due to the law of large numbers, as the number of sampled features increases, the average becomes more stable and the approximation improves. The error decreases as more random features are used, often with the familiar Monte Carlo scaling

$$\text{error} \sim \frac{1}{\sqrt{m}}.$$

6. NUMERICAL EXAMPLE

To highlight differences between the two, we compared an exact Radial Basis Function (RBF) kernel, or Gaussian kernel classifier with a random-feature approximation on a synthetic nonlinear classification task. We used a two-moons dataset, a fixed train-test split, an SVM with an RBF kernel as the exact model, and random Fourier features followed by a linear classifier as the approximate model. We varied the number of random features over

$$m \in \{50, 100, 200, 500, 1000, 2000\}.$$

Figure 3 shows how test accuracy changes as the number of random features increases. The exact kernel baseline achieved a test accuracy of 0.9442. The random-feature model reached similar accuracy once m was increased beyond roughly 100, and its best performance in our sweep occurred near $m = 500$, where the test accuracy was 0.9483. The gain was small, but it shows that a finite random feature map can capture much of the structure of the exact kernel.

We also measured training time. As m increased, the random-feature model became more expensive to fit because the transformed feature representation grew in size. On this simple two-dimensional problem, the exact RBF SVM remained faster than the random-feature pipeline over the dataset sizes we tested. These results highlight that the benefits of random feature approximations are not the same across all dataset scales. On small or moderate datasets, the overhead of creating this high-dimensional random mapping can actually be slower than just evaluating the exact kernel. The power of this method is in its scalability. Random features becomes essential once your dataset size N grows so large that storing the exact $O(N^2)$ matrix is no longer a physical possibility. We also want to note that this is a very limited example due to the 3-page limitation of this paper, but further exploration can be done using our code in the github repository.

7. LIMITATIONS AND TRADEOFFS

Random feature models are not a free replacement for kernel methods. Their performance depends on the quality of the approximation, and that quality depends on the number of random features m. If m is too small, the approximation may be poor and prediction accuracy may suffer.

This introduces a new design choice. In exact kernel methods, the main concern is the cost of the kernel matrix. In random feature models, one must decide how many features are enough. A larger value of m generally improves the approximation, but it also increases

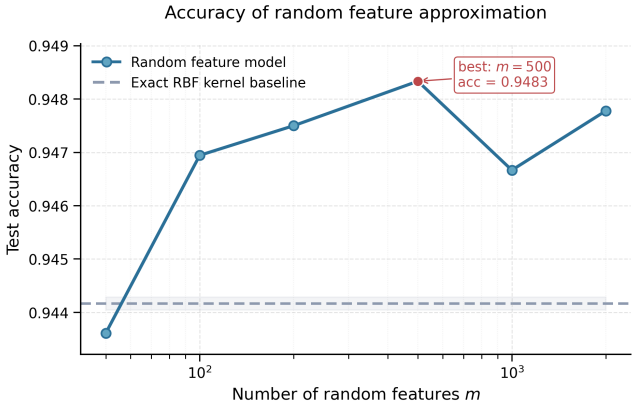

Figure 3. Test accuracy of the random-feature model as a function of the number of random features m. The dashed horizontal line marks the exact RBF kernel baseline, and the highlighted point shows the best result in our sweep, attained near $m = 500$. As m increases, the random-feature approximation approaches the exact kernel result.

computation and memory use. The method shifts the burden rather than eliminating it.

There are also limits on where the cleanest theory applies. Random Fourier features are especially natural for shift-invariant kernels. Other kernels may require different approximation schemes. On very small datasets, the advantage of random features may also be limited, since the exact kernel method may already be cheap enough.

8. CONCLUSION

The kernel trick extends linear learning into nonlinear settings by replacing explicit feature maps with kernel evaluations. This gives kernel methods much of their power, but it also creates a computational bottleneck when the dataset becomes large. Random feature models offer a way around that bottleneck by building a finite random feature map whose inner product approximates the original kernel.

Our numerical example reflects the main tradeoff clearly. As the number of sampled features increased, the random-feature model approached the accuracy of the exact kernel method. At the same time, the approximation became more expensive to compute. Random feature models are therefore best understood not as an automatic speedup, but as a practical approximation scheme that can make kernel methods more usable at larger scale.

APPENDIX

A. CODE AVAILABILITY

Here we provide the source code and instructions to reproduce our numerical example. Source Code: https://github.com/diegomiura/Random-Feature-Models

REFERENCES

Cover, T. M. 1965, IEEE Transactions on Electronic Computers, EC-14, 326, doi: 10.1109/PGEC.1965.264137

Kim, E. 2018, Everything You Should Know About the Kernel Trick,, https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Schölkopf, B., & Smola, A. J. 2002, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Cambridge, MA: MIT Press)