# Supplementary Material: À la Carte — Learning Fast Kernels

## A Experimental Details

To learn the parameters of the kernels, we optimize over the marginal likelihood objective function described in Section 3.1, using LBFGS.[1]

The datasets are divided into three groups: `SMALL` $n \leq 2000$ and `MEDIUM` $2,000 < n \leq 100,000$ and `LARGE` $100,000 < n \leq 2,000,000$. All methods – RBF, ARD, FSARD, GM, PWL and SSGPR – are tested on each grouping. For `SMALL` data, we use an exact RBF and ARD kernel. All the datasets are divided into 10 partitions. Every time, we pick one partition as test data and train all methods on the remaining 1 partitions. The reported result is based on the averaged RMSE of 10 partitions.

### Initialization

**RBF** We randomly pick $max(2000, n/5)$ pairs of data and compute the distance of these pairs. These distances are sorted and we pick the $[0.1 : 0.2 : 0.9]$ quantiles as length-scale (aka rescale) initializations. We run 20 optimization iterations starting with these initializations and then pick the one with the minimum negative log marginal likelihood, and then continue to optimize for 150 iterations. We initialize the signal and noise standard deviations as $std(y)$ and $std(y)/10$, respectively.

**ARD** For each dimension of the input, $X_d$, we initialize each length-scale parameter as $l_d = u(max(X_d) - min(X_d))$, where $u \sim \text{Uniform}[0.4, 0.8]$. We pick the best initialization from 10 random restarts of a 20 iteration optimization run. We multiply the scale by $\sqrt{d}$.

**FSARD** We use the same technique as in ARD to initialize the rescale parameters. For the $S$ matrix, we set it to be $||w||$, where $w$ is a $d$ dimensional random vector with standard Gaussian distribution.

**FSGBARD** The initialization is quite similar to that of FSARD. $B$ is drawn uniformly from $\{\pm 1\}$ and $G$ draws from standard Gaussian distribution.

**GM** We use a Gaussian distribition with diagonal covariance matrices to model the spectral density $p(\omega)$. Assuming there are $Q$ mixture components, the scale of each component is initialized to be $std(y)/Q$. We reuse the same technique to initialize the rescale matrices as that of ARD. We initialize the shift $\mu$ to be close to 0.

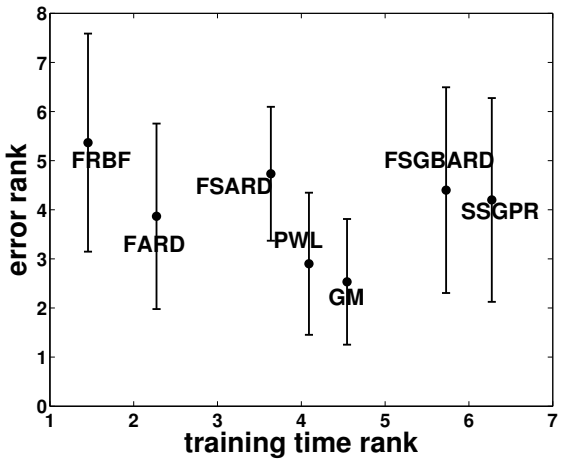
**PWL** We make use of a special case of a piecewise linear function, the hat function, in the experiments. The hat function is parameterized by $\mu$ and $\sigma$. $\sigma$ controls the width of the hat function and $\mu$ control the distance of the hat function from the origin. We also incorporate ARD in the kernel, and use the same initialization techniques. For the RBF kernel we compute the distance of random pairs and sort these distances. Then we get a distance sample $\lambda$ with a uniform random quantile within $[0.2, 0.8]$. $\lambda$ is like the bandwidth of an RBF kernel. Then $\sigma = 2/\lambda$ and $\mu = \max\{\sqrt{d-1} - 2, 0.01\}/\lambda$. We make use of this technique because for RBF kernel, the maximum points is at $\sqrt{d-1}/\lambda$ and the width of the radial distribution is about $2/\lambda$.

**SSGPR** For the rescale parameters, we follow the same procedure as for the ARD kernel. We initialize the projection matrix as a random Gaussian matrix.

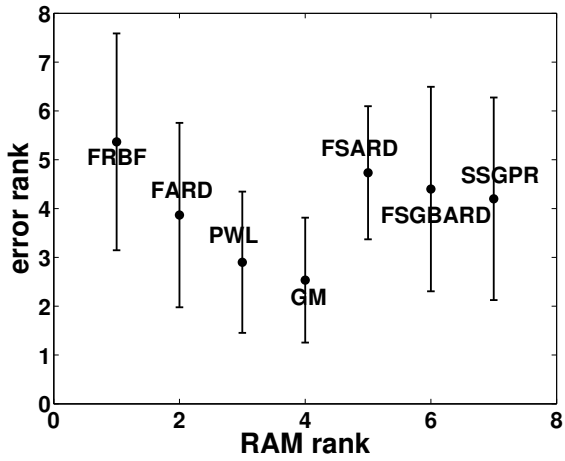
[1] http://www.di.ens.fr/~mschmidt/Software/minFunc.html

Table 1: Comparative RMSE performance on all datasets, with $n$ training points and $d$ the input dimensions. The results are averaged over 10 equal partitions of the data ± 1 standard deviation. We use exact RBF and ARD kernels on the small datasets, and Fastfood expansions of these kernels on the medium and large datasets. For GM and PWL, we set $Q = 5$ and use $m = 256$ sample points for each component. For FSARD and SSGPR, we set $m = 512$. For medium datasets, we set $Q = 3$ and $m = 256$ for GM and PWL. On the large datasets *3D Road network* and *Buzz*, GM and PWL use $Q = 6$ and $m = 16$, and $Q = 5$ and $m = 8$ for *Song* and *Household electric*. All other methods use $Qm$ basis functions. The terms 'small', 'medium' and 'large' are meant to be taken in the context of Gaussian process regression, which is typically intractable for $n > 2000$ when kernel learning is taking place.

| Datasets | n | d | RBF | ARD | FSARD | GM | PWL | FSGBARD | SSGPR |
|---|---|---|---|---|---|---|---|---|---|
| SMALL: | | | | | | | | | |
| Challenger | 23 | 4 | 0.63±0.26 | 0.63±0.26 | 0.72±0.34 | **0.61±0.27** | 0.68±0.28 | 0.77±0.39 | 0.75±0.38 |
| Fertility | 100 | 9 | **0.19±0.04** | 0.21±0.05 | 0.21±0.05 | **0.19±0.05** | **0.19±0.05** | 0.20±0.04 | 0.20±0.06 |
| Slump | 103 | 7 | 4.49±2.22 | 4.72±2.42 | 3.97±2.54 | **2.99±1.14** | 3.45±1.66 | 5.40±2.44 | 6.25±3.70 |
| Automobile | 159 | 15 | **0.14±0.04** | 0.18±0.07 | 0.18±0.05 | 0.15±0.0311 | **0.14±0.03** | 0.22±0.08 | 0.17±0.06 |
| Servo | 167 | 4 | 0.29±0.07 | 0.28±0.09 | 0.29±0.08 | **0.27±0.07** | 0.28±0.09 | 0.44±0.10 | 0.38±0.08 |
| Cancer | 194 | 34 | 32±4 | 35±4 | 43±9 | **31±4** | 35±6 | 34±5 | 33±5 |
| Hardware | 209 | 7 | 0.44±0.06 | 0.43±0.04 | 0.44±0.06 | 0.44±0.04 | **0.42±0.04** | **0.42±0.08** | 0.44±0.1 |
| Yacht | 308 | 7 | 0.29±0.14 | 0.16±0.11 | 0.13±0.06 | 0.13±0.08 | **0.12±0.07** | **0.12±0.06** | 0.14±0.10 |
| Auto MPG | 392 | 7 | 2.91±0.30 | 2.63±0.38 | 2.75±0.43 | **2.55±0.55** | 2.64±0.52 | 3.30± 0.69 | 3.19±0.56 |
| Housing | 509 | 13 | 3.33±0.74 | 2.91±0.54 | 3.39±0.74 | 2.93±0.83 | **2.90±0.78** | 4.62±0.85 | 4.49±0.69 |
| Forest fires | 517 | 12 | **1.39±0.15** | **1.39±0.16** | 1.59±0.12 | 1.40±0.17 | 1.41±0.16 | 2.64±0.25 | 2.01±0.41 |
| Stock | 536 | 11 | 0.016±0.002 | **0.005±0.001** | **0.005±0.001** | **0.005±0.001** | **0.005±0.001** | 0.006±0.001 | 0.006±0.001 |
| Pendulum | 630 | 9 | 2.77±0.59 | **1.06±0.35** | 1.76±0.31 | **1.06±0.27** | 1.16±0.29 | 1.22±0.26 | 1.09±0.33 |
| Energy | 768 | 8 | 0.47±0.08 | 0.46±0.07 | 0.47±0.07 | **0.31±0.07** | 0.36±0.08 | 0.39±0.06 | 0.42±0.08 |
| Concrete | 1,030 | 8 | 5.42±0.80 | 4.95±0.77 | 5.43±0.76 | **3.67±0.71** | 3.76±0.59 | 4.86±0.97 | 5.03±1.35 |
| Solar flare | 1,066 | 10 | **0.78±0.19** | 0.83±0.20 | 0.87±0.19 | 0.82±0.19 | 0.82±0.18 | 0.91±0.19 | 0.89±0.20 |
| Airfoil | 1,503 | 5 | 4.13±0.79 | 1.69±0.27 | 2.00±0.38 | **1.38±0.21** | 1.49±0.18 | 1.93±0.38 | 1.65±0.20 |
| Wine | 1,599 | 11 | 0.55±0.03 | **0.47±0.08** | 0.50±0.05 | 0.53±0.11 | 0.48±0.03 | 0.57±0.04 | 0.66±0.06 |
| MEDIUM: | | | | | | | | | |
| Gas sensor | 2,565 | 128 | 0.21±0.07 | **0.12±0.08** | 0.13±0.06 | 0.14±0.08 | **0.12±0.07** | 0.14±0.07 | 0.14±0.08 |
| Skillcraft | 3,338 | 19 | 1.26±3.14 | **0.25±0.02** | **0.25±0.02** | **0.25±0.02** | **0.25±0.02** | 0.29±0.02 | 0.28±0.01 |
| SML | 4,137 | 26 | 6.94±0.51 | 0.33±0.11 | **0.26±0.04** | 0.27±0.03 | 0.31±0.06 | 0.31±0.06 | 0.34±0.05 |
| Parkinsons | 5,875 | 20 | 3.94±1.31 | 0.01±0.00 | 0.02±0.01 | **0.00±0.00** | 0.04±0.03 | 0.02±0.00 | 0.08±0.19 |
| Pumadyn | 8,192 | 32 | 1.00±0.00 | 0.20±0.00 | 0.22±0.03 | 0.21±0.00 | **0.20±0.00** | 0.21±0.00 | 0.21±0.00 |
| Pole Tele | 15,000 | 26 | 12.6±0.3 | 7.0±0.3 | 6.1±0.3 | 5.4±0.7 | 6.6±0.3 | 4.7±0.2 | **4.3±0.2** |
| Elevators | 16,599 | 18 | 0.12±0.00 | 0.090±0.001 | 0.089±0.002 | 0.089±0.002 | 0.089±0.002 | **0.086±0.002** | 0.088±0.002 |
| Kin40k | 40,000 | 8 | 0.34±0.01 | 0.28±0.01 | 0.23±0.01 | 0.19±0.02 | 0.23±0.00 | 0.08±0.00 | **0.06±0.00** |
| Protein | 45,730 | 9 | 1.64±1.66 | 0.53±0.01 | 0.52±0.01 | 0.50±0.02 | 0.52±0.01 | 0.48±0.01 | **0.47±0.01** |
| KEGG | 48,827 | 22 | 0.33±0.17 | **0.12±0.01** | **0.12±0.01** | **0.12±0.01** | **0.12±0.01** | **0.12±0.01** | **0.12±0.01** |
| CT slice | 53,500 | 385 | 7.13±0.11 | 4.00±0.12 | 3.60±0.09 | 2.21±0.06 | 3.35±0.08 | 2.56±0.12 | **0.59±0.07** |
| KEGGU | 63,608 | 27 | 0.29±0.12 | **0.12±0.00** | **0.12±0.00** | **0.12±0.00** | **0.12±0.00** | **0.12±0.00** | **0.12±0.00** |
| LARGE: | | | | | | | | | |
| 3D road | 434,874 | 3 | 12.86±0.09 | 10.91±0.05 | 10.29±0.12 | 10.34±0.19 | 11.26±0.22 | **9.90±0.10** | 10.12±0.28 |
| Song | 515,345 | 90 | 0.55±0.00 | 0.49±0.00 | 0.47±0.00 | 0.46±0.00 | 0.47±0.00 | 0.46±0.00 | **0.45±0.00** |
| Buzz | 583,250 | 77 | 0.88±0.01 | 0.59±0.02 | 0.54±0.01 | **0.51±0.01** | 0.54±0.01 | 0.52±0.01 | 0.54±0.01 |
| Electric | 2,049,280 | 11 | 0.23±0.00 | 0.12±0.12 | 0.06±0.01 | **0.05±0.00** | 0.07±0.04 | **0.05±0.00** | **0.05±0.00** |

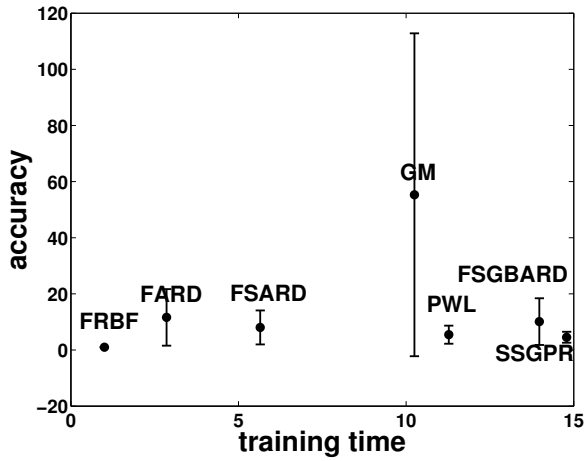



(a) error rank vs training time rank

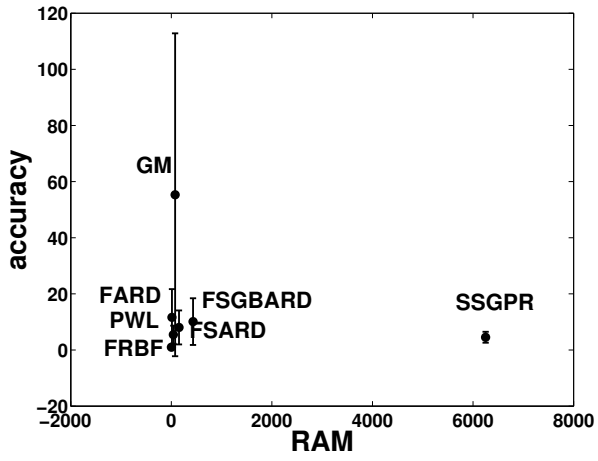



(b) error rank vs RAM rank

Figure 1: Fig. 1a and Fig. 1b compares all methods in by averaging the rank of all methods in terms of accuracy, training time and RAM.




(a) accuracy vs training time




(b) accuracy vs RAM

Figure 2: Fig. 2a and Fig. 2b compares all methods in term of accuracy, training time and RAM without using log scale.