1) $\Gamma(x,x)\Gamma(x',x') - |\Gamma(x,x')|^2 = \Gamma(x,x)\Gamma(x',x') - \Gamma(x,x')\Gamma(x',x)$

$$= \begin{vmatrix} \Gamma(x,x) & \Gamma(x,x') \\ \Gamma(x',x) & \Gamma(x',x') \end{vmatrix} = \det(\Gamma(A,A)) \text{ s.t. } A \subset X$$

$\Gamma$ PDS on $X \times X \Rightarrow \det(\Gamma(A,A)) \geq 0 \Rightarrow |\Gamma(x,x')|^2 \leq \Gamma(x,x)\Gamma(x,x') \;\forall x,x' \in X$ ■

2) $\bar{K}(x,x') = \begin{cases} 0, & K(x,x) = 0 \text{ or } K(x',x') = 0 \\ K(x,x')/(K(x,x)K(x',x'))^{1/2} & \text{else} \end{cases}$

If $K(x,x) = 0$ or $K(x',x') = 0$, $\bar{K}(x',x) = 0 = \bar{K}(x,x')$

Else $\bar{K}(x',x) = K(x',x)/(K(x',x')K(x,x))^{1/2} = K(x,x')/(K(x,x)K(x',x'))^{1/2} = \bar{K}(x,x')$

since $K$ PDS $\Rightarrow K(x',x) = K(x,x')$ and multiplication is commutative on $\mathbb{R}$

$\Rightarrow \bar{K}$ is symmetric

$K(x,x), K(x',x') \in \mathbb{R} \Rightarrow (K(x,x)K(x',x'))^{1/2} \geq 0$

Further, $K$ PDS $\Rightarrow K(x,x')$ is pos. def. $\Rightarrow K(x,x')/(K(x,x)K(x',x'))^{1/2}$ is pos. def.

$\Rightarrow \bar{K}$ is pos. def.

Thus, $\bar{K}$ is PDS

3) Let $\hat{k}$ be the linear kernel (proven PDS in-class)

Polynomial: $k(x,x') = (x^Tx'+c)^d \Rightarrow k(X,X) = (\hat{k}(X,X)+c)^d$

$c>0 \Rightarrow cI$ pos. def. $\Rightarrow \hat{k}(X,X)+cI$ PDS by closure under sums

$d \in \mathbb{N} \Rightarrow (\hat{k}(x,x)+cI)^d = (\hat{k}(x,x)+cI) \cdot \ldots \cdot (\hat{k}(x,x)+cI)$ PDS by closure under products

Exponential: $k(x,x') = \exp(x^Tx') \Rightarrow k(X,X) = \exp(\hat{k}(X,X)) = \sum_{j=0}^{\infty} 1/j!\, \hat{k}(x,x)^j$

$\hat{k}(X,X)^j$ PDS by closure under products

$\Rightarrow k(X,X) = \sum_{j=0}^{\infty} 1/j!\, \hat{k}(x,x)^j$ PDS by closure under sums

RBF: $k(x,x') = \exp(-\gamma^2 \|x-x'\|_2^2) = \exp[-\gamma^2(\|x\|_2^2+\|x'\|_2^2 - x^Tx' - x'^Tx]$

$= \exp[-\gamma^2(\|x\|_2^2+\|x'\|_2^2)] \cdot \exp(2\gamma^2 x^Tx')$

$2\gamma^2 \geq 0$, $\exp(\hat{k}(x,x))$ PDS by "Exponential" $\Rightarrow \exp(2\gamma^2 x^Tx')$ PDS

$\exp[-\gamma^2(\|x\|_2^2+\|x'\|_2^2)] > 0 \Rightarrow k(x,x')$ PDS

4) $K(x,x') = \sum_{j=1}^{n} \lambda_j \psi_j(x) \psi_j(x')$

$K(x',x) = \sum_{j=1}^{n} \lambda_j \psi_j(x') \psi_j(x) = \sum_{j=1}^{n} \lambda_j \psi_j(x) \psi_j(x') = K(x,x') \Rightarrow K$ symmetric

Consider $C \in \mathbb{R}^m$ s.t. $m = \dim(\Omega)$

Let $\langle \cdot, \cdot \rangle_{\mathbb{R}^n}$ denote the Euclidean IP and $\psi^i = \psi(x_i)$

$\lambda_j \geq 0 \ \forall j \leq n \Rightarrow \langle \psi, \psi \rangle_\lambda = \langle \psi, \lambda\psi \rangle_{\mathbb{R}^n} \geq 0$ is a well-defined weighted IP

$$C^T K(X,X) C = \sum_{a=1}^{m} \sum_{b=1}^{m} C_a C_b \left( \sum_{j=1}^{n} \lambda_j \psi_j(x_a) \psi_j(x_b) \right) = \sum_{a=1}^{m} \sum_{b=1}^{m} C_a C_b \langle \psi^a, (\lambda\psi)^b \rangle_{\mathbb{R}^n}$$

$$= \sum_{a=1}^{m} \sum_{b=1}^{m} (C_a \psi^a)^T C_b (\lambda\psi)^b = \langle \sum_{i=1}^{m} C_i \psi^i, \sum_{i=1}^{m} C_i (\lambda\psi)^i \rangle_{\mathbb{R}^m} \geq 0$$

since $\langle \cdot, \cdot \rangle_{\mathbb{R}^m}$ is a well-defined IP

Thus, $K$ is PDS on $\Omega$

5i) Let $\langle\cdot,\cdot\rangle$ be the IP associated with $\mathcal{H}$ on $X$ with induced norm $\|\cdot\|$

$K \in \mathcal{H}$ reproducing $\Rightarrow \langle K_x, f\rangle = f(x)\ \forall f \in \mathcal{H}$

$\Rightarrow \langle K_x, K_x\rangle = K(x,x),\ \langle K_x, K'_x\rangle = K'_x(x) = K'(x,x)$

$\text{dist}(K_x, K'_x)^2 = \|K_x - K'_x\|^2 = \langle K_x - K'_x, K_x - K'_x\rangle = \langle K_x, K_x\rangle + \langle K'_x, K'_x\rangle - \langle K_x, K'_x\rangle - \langle K'_x, K_x\rangle$

$= K(x,x) + K'(x,x) - K'(x,x) - K(x,x) = 0\ \ \forall x \in X \Rightarrow K \equiv K'$ ■

ii) Suppose $\mathcal{H}, \mathcal{H}'$ both have PDS Kernel $K$

$K_x \in \mathcal{H} \Rightarrow S = \text{span}(K_x)$ is a subspace of $\mathcal{H}$

For $f \in \mathcal{H}$, $0 = f(x) = \langle K_x, f\rangle \Rightarrow f \equiv 0 \Rightarrow \text{span}(K_x)$ is dense in $\mathcal{H}$

$\Rightarrow \exists$ Cauchy sequence $\{f_n\}_{n=1}^{\infty} \in \text{span}(K_x)$ s.t. $\|f_n - f\|_{\mathcal{H}} \to 0$

Consider $f' \in \mathcal{H}'$

The same arguments follow for $K_x \in \mathcal{H}'$ (i.e. $\|f'_n - f'\|_{\mathcal{H}'} \to 0$)

Further, $\text{span}(K_x)$ dense in $\mathcal{H}$ and $\mathcal{H}'$ $\Rightarrow \|\varphi\|_{\mathcal{H}} = \|\varphi\|_{\mathcal{H}'}\ \forall \varphi \in \text{span}(K_x)$

$\Rightarrow \|f_n - f'\|_{\mathcal{H}'} \to 0$ as well $\Rightarrow f_n \to f$ and $f_n \to f'$ pointwise $\forall x \in X$

$\Rightarrow f \in \mathcal{H}' \Rightarrow \mathcal{H} \subset \mathcal{H}'$

Repeating this for Cauchy seq. $\{f'_n\}_{n=1}^{\infty} \in \mathcal{H}' \Rightarrow \mathcal{H}' \subset \mathcal{H} \Rightarrow \mathcal{H} \equiv \mathcal{H}'$

Finally, $f_n \in \text{span}(K_x) \Rightarrow \|f\|_{\mathcal{H}} = \|f_n\|_{\mathcal{H}} = \|f_n\|_{\mathcal{H}'} = \|f\|_{\mathcal{H}'}\ \forall f \in \mathcal{H}$

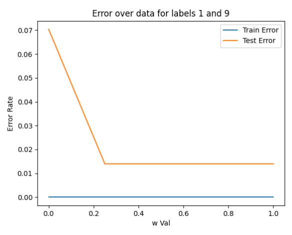
Thus, $(\mathcal{H}, \langle\cdot,\cdot\rangle_{\mathcal{H}}) \equiv (\mathcal{H}', \langle\cdot,\cdot\rangle_{\mathcal{H}'})$ ■

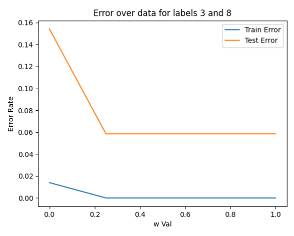
As a prelude to the experiments: running sklearn.decomposition.PCA with n_components=0.95 mapped the 784-feature training set to a 154-dimensional feature space. Therefore 154 PCA modes are needed to preserve 95% of the training set variance.

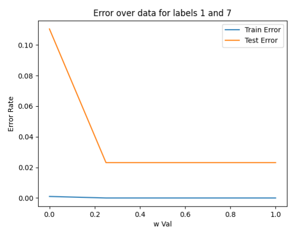
This report is split into two parts: The first is ridge regression directly implemented by myself. My inefficient programming made certain experiments infeasible, so the second part of the report shows results using scikit-learn tools.

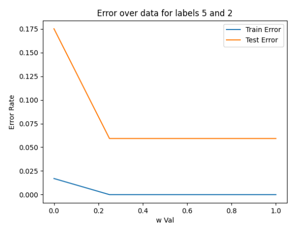
<u>Part 1: Handmade Ridge Regression</u>
The following experimentation was conducted  by designing a kernel regression classifier based on 3 kernels: linear, shiftless quadratic polynomial, and a hybrid kernel. The latter is a summation of the former two with a weight hyperparameter controlling their relative contributions. Specifically, w=0 corresponds to a linear kernel and w=1 corresponds to a quadratic polynomial, thus experimentation was performed only on the hybrid kernel (with these two cases included) for brevity. The nugget term and regularization parameter were held constant by their approximately optimal values as found by a grid search on a 1000-sample subset.

The structure of this section is a series of sub-experiments to gain insight on the kernels, culminating in a final run for the entire dataset. To begin, I examined the train and test errors for w values 0, 0.25, 0.5, 0.75, and 1 for a 1000-sample subset of the training data for each label pair:

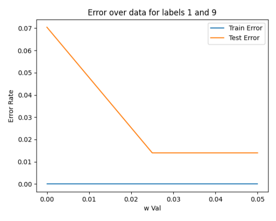


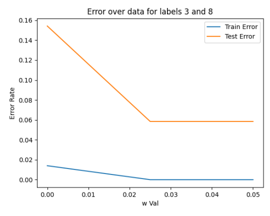


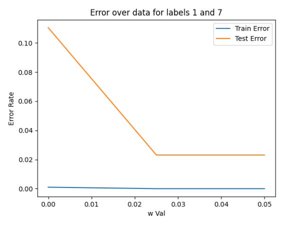


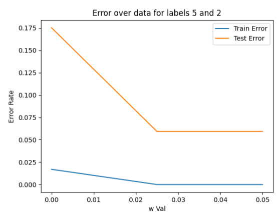


The quadratic polynomial kernel clearly dominates the linear kernel, however the hybrid kernel appears to be equivalent beyond 0.25. Repeating the experiment for w = 0, 0.025, and 0.05 yielded:



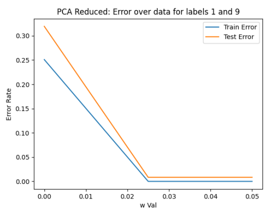


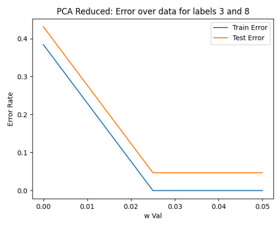


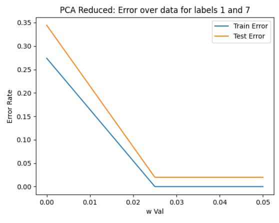


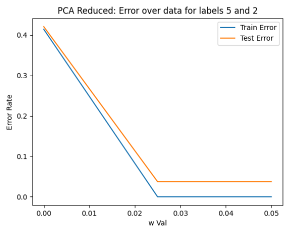
From these results, I conclude that incorporating any quadratic component to the kernel greatly improves its performance and that the weight hyperparameter appears to have little impact on the hybrid kernel (beyond being nonzero). Based on this preliminary experimentation, I determined tuning w to be a waste of time and opted to test these kernels on the entire dataset corresponding to each label with a hybrid model fixed arbitrarily at w=0.5.

For the first round of experimentation on the full dataset I attempted to use all 784 dimensions, however my CPU was not thrilled with the idea. Instead, I considered using the 154-dimensional output of PCA which –as demonstrated above-- preserves 95% of the training variance. To empirically check if my results aren't too distorted by this transformation, I first repeated the second experiment (with 1000 samples) with PCA dimensionality reduction:



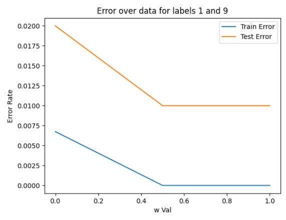


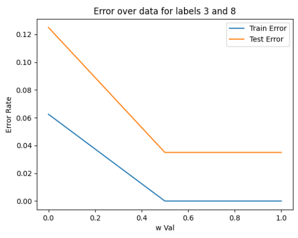


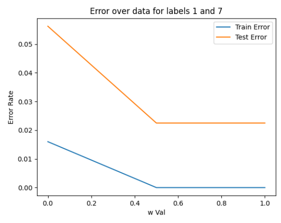


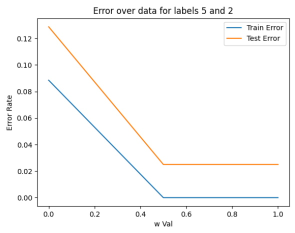
Regrettably, it appears that the linear kernel takes a disproportionate hit from the reduction, so I did not consider using PCA for the sake of generalizability. The train and test errors being more similar indicates that all kernels are underfitting the training data, further suggesting that the PCA representation is insufficient. The previous observation of nonlinearity improving the model almost equally in any amount still remains, however.

As a final test before tormenting my CPU, I retained the same fixed polynomial kernel on a subsample consisting of 1/3 of the data to gauge how each kernel type (w=0,0.5, and 1) scales with train set size 4000:



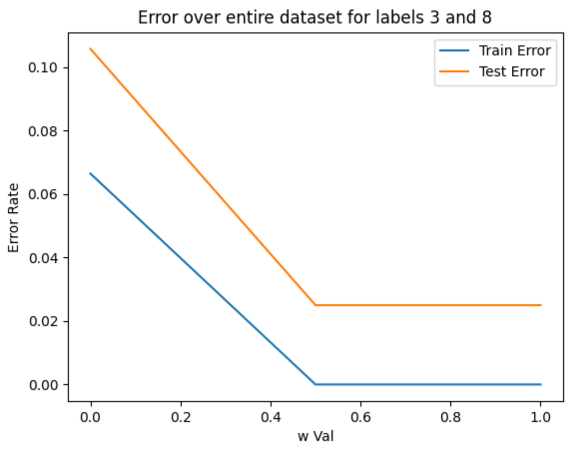






The larger sample greatly improved the linear kernel and modestly improved the polynomial and hybrid kernels, particularly with the "trickier" (3,8) and (5,2) labels. Additionally, these results support my hypothesis that any nonzero weight parameters yield an equally effective hybrid kernel.

Finally, this is the result of the "trickiest" label (3,8) for the full dataset (only 1 label pair was tested as training took very long):

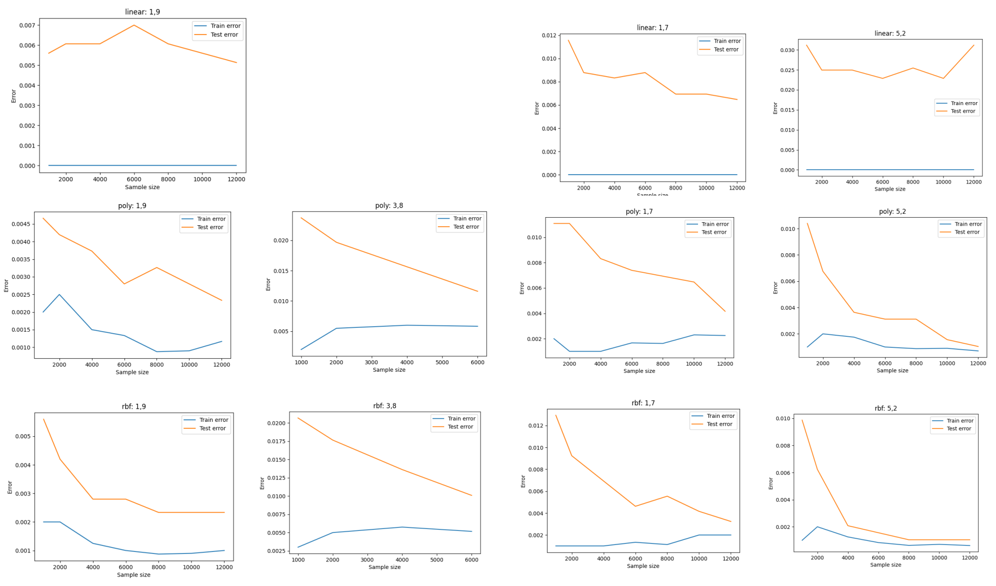


In conclusion, a polynomial and hybrid kernel were equally performant, with the linear kernel lagging behind. Further, the nonlinear kernels appear to approach their peak train and test accuracy with fewer training samples than the linear model. Thus, it appears that any sort of nonlinearity in the kernel allows the model to learn faster, but the linear kernel could potentially “catch up” given a large enough training set.

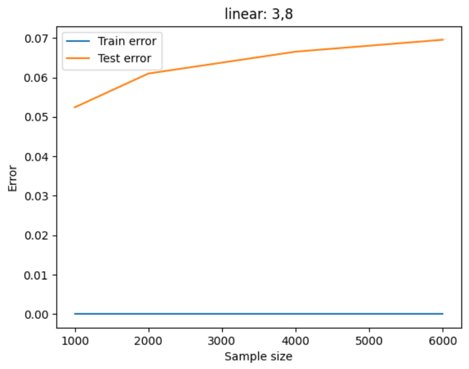
<u>Part 2: scikit-learn</u>

To contribute additional insight to my findings in part 1, I decided to run two scikit-learn experiments on the data.

For my first test, I considered linear, quadratic polynomial, and RBF kernels holding all other parameters constant. I attempted to use an SVM-based model --sklearn.svm.SVC-- over different sized subsets of the data:

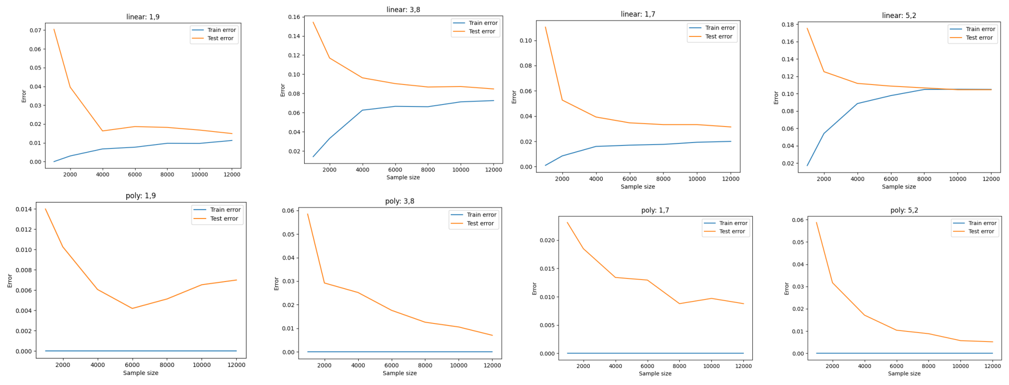


For all label pairs, it appears SVM equipped with these kernels handily learns the labels with somewhat similar errors and improvement rates by sample size as my manual kernel regression, but overall stronger performance in general. The one exception however was the linear kernel on (3,8), which performed worse as its training set increased and failed to converge to a separating hyperplane beyond 6000 samples:



This is particularly bizarre and unexpected since with ridge regression, the linear kernel struggled more with (5,2) than (3,8). Clearly kernel ridge regression and SVM are not interchangeable; despite somewhat worse performance for most labels, kernel ridge regression was able to effectively classify all of them. Additionally, my hypothesis from Part 1 that any amount of nonlinearity in the kernel lead to an approximately equal improvement is further supported by the RBF and polynomial kernels' domination over the linear kernel across the board to similar degrees.

To conclude my experimentation, I ran sklearn.kernel_ridge.KernelRidge with the linear and polynomial kernels for a direct comparison with my Part 1 results. Due to its more efficient implementation, this allows greater sample size granularity and use of the entire dataset:



The behaviours of the train and test error are approximately the same as the corresponding nonlinear hybrid kernels' for the 1000 and 4000 sample training set cases, leading me to believe that my earlier findings are representative of the model's true behaviour and that these plots are extrapolations of how the nonlinear hybrid kernels would perform if my CPU could handle larger sample sizes. The increasing training error at larger samples for the linear kernel demonstrates underfitting, which is not seen for the polynomial kernel. Although not included in detail due to space constraints, informal testing of a degree 3 polynomial kernel demonstrated overfitting for kernel ridge regression.

In conclusion, kernel regression is an effective classification technique for MNIST, given that 5% is typically considered "human error" for classification. Introducing quadratic nonlinearity greatly improved the model's performance without overfitting.