**STA414 – Assignment 3, Question 4**

1. **Findings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Linear Model** | **GP with Linear Cov** | **General GP on unscaled data** | **General GP on scaled data** |
| MSE | 0.2876439 | 0.443493 | 0.2942536 | 0.2407561 |
| Run Time | Elapsed 0.03 | Elapsed 3.24 | Elapsed 1995.87 | Elapsed 1919.87 |
| Optimal Hyperparameters | - | - | Gamma 10, rho 0.16  (C. MSE = 0.23488)  Refer to Appendix 1 | Gamma 4.1, rho 1  (C. MSE = .28309)  Refer to Appendix 2 |

1. **Discussion**

The models above bear a tradeoff between computational complexity and model fit. The quickest and second best fitting model is the linear model, in which optimal weights for each variable are assigned by minimizing the error function. It is reasonable to claim that such good fit is due to the dependent variable’s approximate linear relationship with all covariates in both training and test datasets. Moreover, a Q-Q plot suggests close fit to a Gaussian, following model assumptions.

Fitting the GP with linear covariance yielded the worst MSE and the second worst running time. Arguably, the kernel structure may have placed unreasonable amount of weight on covariates 1 and 7 due to their large magnitude.

In second place, in terms of MSE, came the general covariance structure GP, applied on the unscaled data. Note that the prediction MSE is only slightly worse than that of the linear model; however, computation time is significantly greater. The cross validation process for the selection of the optimal hyperparameters is the task that requires the most time. Similar to the GP with linear covariance, the suboptimal fit can be attributed to the difference in scales between covariates 1 and 7 and all other covariates.

Once the data is scaled – covariates 1 and 7 are divided by 10 to accommodate the scaled of all other covariates – General GP achieves the best fit of all other models. As with the general GP fit on the unscaled data, the computation time is also much greater than that of all non-GP models. It is worthwhile to point out that the General GP on unscaled data favors a larger gamma and a smaller rho, while the General GP on the scaled data favors a smaller gamma and a larger rho.

**Appendix 1 – Matrix of average MSE’s for a combination of hyperparameters gamma and rho for general GP on untransformed data**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Gammas** | | | | | | | | | | |
| **Rhos** |  | **0.1** | **0.6** | **1.1** | **1.6** | **2.1** | **2.6** | **3.1** | **3.6** | **4.1** | **4.6** | **5.1** |
| **0.01** | 0.700486 | 0.68046919 | 0.641102 | 0.59694 | 0.557687 | 0.526487 | 0.502765 | 0.48482 | 0.471022 | 0.460138 | 0.451312 |
| **0.06** | 0.681258 | 0.48483983 | 0.431115 | 0.408185 | 0.393191 | 0.381558 | 0.371711 | 0.362943 | 0.354908 | 0.347425 | 0.340398 |
| **0.11** | 0.647386 | 0.42853954 | 0.388294 | 0.36179 | 0.340136 | 0.322011 | 0.30695 | 0.294559 | 0.284437 | 0.276203 | 0.269518 |
| **0.16** | 0.614818 | 0.39972659 | 0.35341 | 0.322021 | 0.300094 | 0.284758 | 0.273896 | 0.266056 | 0.260292 | 0.255989 | 0.252743 |
| **0.21** | 0.590772 | 0.3796526 | 0.32957 | 0.300404 | 0.282854 | 0.271761 | 0.26444 | 0.259496 | 0.256147 | 0.25392 | 0.252508 |
| **0.26** | 0.575868 | 0.36541138 | 0.31489 | 0.289899 | 0.276325 | 0.268349 | 0.263523 | 0.260655 | 0.259081 | 0.258391 | 0.258313 |
| **0.31** | 0.568212 | 0.35516316 | 0.306841 | 0.285806 | 0.274839 | 0.268551 | 0.264924 | 0.262977 | 0.262139 | 0.262045 | 0.262449 |
| **0.36** | 0.565616 | 0.34857477 | 0.303302 | 0.284469 | 0.274517 | 0.268995 | 0.266166 | 0.265074 | 0.265125 | 0.265941 | 0.267277 |
| **0.41** | 0.566345 | 0.34505918 | 0.301827 | 0.283801 | 0.274668 | 0.27034 | 0.268919 | 0.269311 | 0.270858 | 0.273168 | 0.276007 |
| **0.46** | 0.569193 | 0.34362676 | 0.301117 | 0.283703 | 0.275848 | 0.273083 | 0.273211 | 0.275036 | 0.27789 | 0.28139 | 0.285311 |
| **0.51** | 0.573358 | 0.34344692 | 0.30083 | 0.284164 | 0.277595 | 0.27607 | 0.277204 | 0.279782 | 0.283157 | 0.286971 | 0.291022 |
| **0.56** | 0.578317 | 0.34402884 | 0.300889 | 0.284925 | 0.279302 | 0.2785 | 0.280075 | 0.282856 | 0.286254 | 0.289973 | 0.293862 |
| **0.61** | 0.583724 | 0.34515355 | 0.301326 | 0.285925 | 0.280944 | 0.280571 | 0.282386 | 0.285289 | 0.28876 | 0.292552 | 0.296551 |
| **0.66** | 0.589349 | 0.34676477 | 0.302234 | 0.287276 | 0.282782 | 0.282776 | 0.284881 | 0.288062 | 0.291847 | 0.296024 | 0.300495 |
| **0.71** | 0.595036 | 0.34887092 | 0.303682 | 0.289073 | 0.285007 | 0.285402 | 0.287918 | 0.291568 | 0.295916 | 0.300759 | 0.305993 |
| **0.76** | 0.600677 | 0.3514802 | 0.305676 | 0.291339 | 0.287683 | 0.288532 | 0.291575 | 0.295848 | 0.300916 | 0.306554 | 0.312632 |
| **0.81** | 0.606202 | 0.35457565 | 0.308179 | 0.294059 | 0.290809 | 0.292161 | 0.295801 | 0.300753 | 0.306551 | 0.312939 | 0.319758 |
| **0.86** | 0.611561 | 0.35811732 | 0.311141 | 0.297212 | 0.294377 | 0.29626 | 0.300511 | 0.306118 | 0.312577 | 0.319606 | 0.32703 |
| **0.91** | 0.616722 | 0.3620535 | 0.314522 | 0.300787 | 0.298386 | 0.300814 | 0.305664 | 0.31188 | 0.318928 | 0.32651 | 0.334447 |
| **0.96** | 0.621667 | 0.36633029 | 0.318289 | 0.304779 | 0.302839 | 0.30582 | 0.31125 | 0.318032 | 0.325611 | 0.333683 | 0.34207 |
| **1** | 0.625459 | 0.36996245 | 0.321562 | 0.308266 | 0.306714 | 0.310137 | 0.316011 | 0.323212 | 0.331176 | 0.339597 | 0.348303 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Gammas** | | | | | | | | | |
| **Rhos** |  | **5.6** | **6.1** | **6.6** | **7.1** | **7.6** | **8.1** | **8.6** | **9.1** | **9.6** | **10** |
| **0.01** | 0.443969 | 0.43771933 | 0.432298 | 0.427519 | 0.423251 | 0.419397 | 0.415884 | 0.412657 | 0.409674 | 0.40744 |
| **0.06** | 0.333773 | 0.32751431 | 0.3216 | 0.31601 | 0.310732 | 0.30575 | 0.301053 | 0.296629 | 0.292467 | 0.289318 |
| **0.11** | 0.264095 | 0.25969625 | 0.256127 | 0.253233 | 0.250887 | 0.24899 | 0.247461 | 0.246236 | 0.245261 | 0.244634 |
| **0.16** | 0.250279 | 0.24840939 | 0.246997 | 0.245941 | 0.245166 | 0.244616 | 0.244244 | 0.244016 | 0.243905 | **0.243884** |
| **0.21** | 0.251701 | 0.25135033 | 0.251348 | 0.251612 | 0.252084 | 0.252714 | 0.253466 | 0.254309 | 0.255219 | 0.255982 |
| **0.26** | 0.25866 | 0.25929682 | 0.260128 | 0.261084 | 0.262113 | 0.263179 | 0.264255 | 0.265325 | 0.266376 | 0.267198 |
| **0.31** | 0.263189 | 0.26415189 | 0.265264 | 0.266475 | 0.267756 | 0.269086 | 0.270456 | 0.271861 | 0.2733 | 0.274476 |
| **0.36** | 0.268978 | 0.27094554 | 0.27312 | 0.275466 | 0.277964 | 0.280604 | 0.283381 | 0.286294 | 0.289343 | 0.291881 |
| **0.41** | 0.279233 | 0.28276013 | 0.286534 | 0.290519 | 0.294691 | 0.299029 | 0.303518 | 0.308145 | 0.312896 | 0.31678 |
| **0.46** | 0.289511 | 0.29390305 | 0.298427 | 0.303044 | 0.307727 | 0.312458 | 0.317224 | 0.322019 | 0.326835 | 0.330701 |
| **0.51** | 0.2952 | 0.29944181 | 0.303718 | 0.308014 | 0.312328 | 0.316659 | 0.32101 | 0.325386 | 0.329789 | 0.333331 |
| **0.56** | 0.297851 | 0.30190972 | 0.306028 | 0.310205 | 0.314442 | 0.318743 | 0.323109 | 0.32754 | 0.332036 | 0.335677 |
| **0.61** | 0.300706 | 0.30499795 | 0.309419 | 0.313967 | 0.318641 | 0.323437 | 0.328353 | 0.333382 | 0.338518 | 0.342698 |
| **0.66** | 0.305215 | 0.3101616 | 0.315319 | 0.320673 | 0.326211 | 0.331918 | 0.337778 | 0.343774 | 0.349892 | 0.354862 |
| **0.71** | 0.311557 | 0.31740907 | 0.323512 | 0.329832 | 0.336341 | 0.343011 | 0.349817 | 0.356736 | 0.363748 | 0.369414 |
| **0.76** | 0.319061 | 0.32577433 | 0.332719 | 0.33985 | 0.347132 | 0.354535 | 0.362035 | 0.369611 | 0.377247 | 0.38339 |
| **0.81** | 0.326899 | 0.33428624 | 0.341862 | 0.349582 | 0.357414 | 0.365334 | 0.373322 | 0.381365 | 0.389451 | 0.395946 |
| **0.86** | 0.334736 | 0.34264677 | 0.35071 | 0.358887 | 0.367154 | 0.375491 | 0.383885 | 0.392327 | 0.400811 | 0.407624 |
| **0.91** | 0.342628 | 0.35098212 | 0.359462 | 0.368038 | 0.376688 | 0.385399 | 0.394159 | 0.402963 | 0.411803 | 0.418898 |
| **0.96** | 0.350668 | 0.35941013 | 0.368255 | 0.377176 | 0.386154 | 0.395176 | 0.404231 | 0.413312 | 0.42241 | 0.429698 |
| **1** | 0.357191 | 0.36619946 | 0.375288 | 0.384431 | 0.393609 | 0.402809 | 0.41202 | 0.421232 | 0.430439 | 0.437796 |

**Appendix 2 – Matrix of average MSE’s for a combination of hyperparameters gamma and rho for general GP on scaled data**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Gammas** | | | | | | | | | | |
| **Rhos** |  | **0.1** | **0.6** | **1.1** | **1.6** | **2.1** | **2.6** | **3.1** | **3.6** | **4.1** | **4.6** | **5.1** |
| **0.01** | 0.701085 | 0.700662 | 0.699639 | 0.698027 | 0.695843 | 0.693111 | 0.689858 | 0.686116 | 0.68192 | 0.677306 | 0.672314 |
| **0.06** | 0.700663 | 0.686135 | 0.655499 | 0.616534 | 0.575668 | 0.536857 | 0.502102 | 0.47211 | 0.446822 | 0.425782 | 0.408385 |
| **0.11** | 0.699645 | 0.65562 | 0.58265 | 0.513454 | 0.459176 | 0.419849 | 0.392078 | 0.372424 | 0.35832 | 0.348028 | 0.340401 |
| **0.16** | 0.698056 | 0.61703 | 0.513902 | 0.440043 | 0.394539 | 0.366968 | 0.349776 | 0.33868 | 0.331321 | 0.326347 | 0.322945 |
| **0.21** | 0.695928 | 0.576823 | 0.460215 | 0.394978 | 0.360976 | 0.342365 | 0.331597 | 0.325124 | 0.321143 | 0.318668 | 0.317127 |
| **0.26** | 0.693305 | 0.538907 | 0.421434 | 0.36763 | 0.342452 | 0.329594 | 0.322628 | 0.318729 | 0.316515 | 0.315259 | 0.314561 |
| **0.31** | 0.690239 | 0.505195 | 0.394044 | 0.350358 | 0.331367 | 0.322233 | 0.317592 | 0.315173 | 0.313906 | 0.313254 | 0.312934 |
| **0.36** | 0.686785 | 0.476299 | 0.374546 | 0.33885 | 0.324201 | 0.317543 | 0.314364 | 0.312811 | 0.312051 | 0.311678 | 0.311489 |
| **0.41** | 0.683004 | 0.452073 | 0.360339 | 0.33076 | 0.319222 | 0.314253 | 0.312 | 0.31094 | 0.31041 | 0.310104 | 0.309871 |
| **0.46** | 0.678956 | 0.432 | 0.349678 | 0.324788 | 0.315527 | 0.311714 | 0.310024 | 0.309191 | 0.308686 | 0.308275 | 0.307855 |
| **0.51** | 0.6747 | 0.415434 | 0.341432 | 0.320183 | 0.312604 | 0.309558 | 0.308154 | 0.307337 | 0.306689 | 0.306041 | 0.305331 |
| **0.56** | 0.670294 | 0.401739 | 0.334875 | 0.316491 | 0.31014 | 0.307553 | 0.306211 | 0.305241 | 0.304326 | 0.30336 | 0.302318 |
| **0.61** | 0.665791 | 0.390357 | 0.329534 | 0.313419 | 0.307928 | 0.305545 | 0.304085 | 0.302844 | 0.301599 | 0.300299 | 0.298955 |
| **0.66** | 0.661242 | 0.380825 | 0.325095 | 0.310765 | 0.305825 | 0.30344 | 0.301737 | 0.300168 | 0.298597 | 0.297018 | 0.29547 |
| **0.71** | 0.656692 | 0.372773 | 0.321338 | 0.308386 | 0.303738 | 0.3012 | 0.299189 | 0.297302 | 0.295471 | 0.293729 | 0.292124 |
| **0.76** | 0.652183 | 0.365914 | 0.318104 | 0.30618 | 0.301617 | 0.298832 | 0.296512 | 0.294375 | 0.292407 | 0.290655 | 0.289165 |
| **0.81** | 0.64775 | 0.360029 | 0.315275 | 0.304078 | 0.299447 | 0.296381 | 0.293808 | 0.291539 | 0.289585 | 0.287989 | 0.286784 |
| **0.86** | 0.643426 | 0.354947 | 0.312762 | 0.302038 | 0.297241 | 0.293917 | 0.291192 | 0.288933 | 0.287153 | 0.285874 | 0.2851 |
| **0.91** | 0.639237 | 0.350536 | 0.310496 | 0.30004 | 0.295035 | 0.291521 | 0.288771 | 0.286673 | 0.285214 | 0.284382 | 0.284149 |
| **0.96** | 0.635205 | 0.346695 | 0.308426 | 0.298082 | 0.292876 | 0.289271 | 0.286633 | 0.284832 | 0.283815 | 0.283525 | 0.283894 |
| **1** | 0.632106 | 0.343975 | 0.306886 | 0.296552 | 0.291215 | 0.28762 | 0.285164 | 0.283685 | 0.283085 | 0.283268 | 0.284133 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Gammas** | | | | | | | | | |
| **Rhos** |  | **5.6** | **6.1** | **6.6** | **7.1** | **7.6** | **8.1** | **8.6** | **9.1** | **9.6** | **10** |
| **0.01** | 0.666981 | 0.661347 | 0.655449 | 0.649324 | 0.643008 | 0.636532 | 0.629929 | 0.623228 | 0.616455 | 0.611002 |
| **0.06** | 0.394022 | 0.382141 | 0.372277 | 0.364049 | 0.357152 | 0.351341 | 0.346423 | 0.342243 | 0.338677 | 0.336199 |
| **0.11** | 0.334678 | 0.330343 | 0.327035 | 0.324499 | 0.32255 | 0.321051 | 0.3199 | 0.31902 | 0.318351 | 0.317938 |
| **0.16** | 0.320603 | 0.31899 | 0.317884 | 0.317136 | 0.31664 | 0.316326 | 0.316141 | 0.316048 | 0.31602 | 0.316032 |
| **0.21** | 0.316177 | 0.315607 | 0.315282 | 0.315117 | 0.315054 | 0.315057 | 0.3151 | 0.315165 | 0.315241 | 0.315305 |
| **0.26** | 0.314192 | 0.314017 | 0.313956 | 0.313958 | 0.313992 | 0.314038 | 0.314086 | 0.314126 | 0.314154 | 0.314166 |
| **0.31** | 0.312791 | 0.312737 | 0.312721 | 0.312715 | 0.3127 | 0.312666 | 0.312609 | 0.312525 | 0.312414 | 0.312305 |
| **0.36** | 0.311376 | 0.311281 | 0.311172 | 0.311033 | 0.310855 | 0.310636 | 0.310376 | 0.310077 | 0.30974 | 0.309447 |
| **0.41** | 0.309638 | 0.309369 | 0.309048 | 0.30867 | 0.308238 | 0.307754 | 0.307224 | 0.306656 | 0.306056 | 0.305557 |
| **0.46** | 0.30738 | 0.306837 | 0.306224 | 0.305549 | 0.304823 | 0.304056 | 0.303259 | 0.302444 | 0.30162 | 0.300961 |
| **0.51** | 0.304542 | 0.303681 | 0.302762 | 0.301802 | 0.30082 | 0.299831 | 0.298852 | 0.297895 | 0.296972 | 0.296266 |
| **0.56** | 0.301211 | 0.300062 | 0.298898 | 0.297743 | 0.29662 | 0.295548 | 0.294542 | 0.293616 | 0.29278 | 0.292181 |
| **0.61** | 0.297599 | 0.296266 | 0.294987 | 0.293791 | 0.292699 | 0.291729 | 0.290892 | 0.290198 | 0.289653 | 0.289327 |
| **0.66** | 0.293995 | 0.292634 | 0.291417 | 0.290367 | 0.289501 | 0.288829 | 0.288357 | 0.288086 | 0.288016 | 0.288103 |
| **0.71** | 0.290701 | 0.289494 | 0.288526 | 0.287809 | 0.287351 | 0.287151 | 0.287205 | 0.287506 | 0.288044 | 0.288639 |
| **0.76** | 0.287972 | 0.287097 | 0.286548 | 0.286324 | 0.286416 | 0.286813 | 0.287499 | 0.288458 | 0.289671 | 0.290814 |
| **0.81** | 0.285985 | 0.285592 | 0.285592 | 0.285969 | 0.286698 | 0.287755 | 0.289114 | 0.290748 | 0.292633 | 0.294306 |
| **0.86** | 0.284821 | 0.285012 | 0.28564 | 0.286669 | 0.28806 | 0.289777 | 0.291783 | 0.294046 | 0.296537 | 0.298676 |
| **0.91** | 0.28447 | 0.285297 | 0.286573 | 0.288247 | 0.290266 | 0.292584 | 0.29516 | 0.297956 | 0.30094 | 0.303444 |
| **0.96** | 0.284847 | 0.286307 | 0.2882 | 0.29046 | 0.293027 | 0.295847 | 0.298876 | 0.302075 | 0.305412 | 0.308163 |
| **1** | 0.285579 | 0.287514 | 0.289849 | 0.292512 | 0.295436 | 0.298567 | 0.301861 | 0.305279 | 0.308791 | 0.311652 |

**Appendix 3 – Code**

### STA414 - A3

### Prepping data

# Training data

trainx <- read.csv("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/train1x",

sep=" ",header=F)

trainy <- read.csv("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/train1y",

sep=" ",header=F)

train1y <- 0

for(i in 1:length(trainy[[1]])){

train1y <- c(train1y,as.numeric(trainy[[1]][i]))

}

train1y <- train1y[2:251]

train1y

x1 <- trainx[,1] ; x2 <- trainx[,2] ; x3 <- trainx[,3] ; x4 <- trainx[,4] ; x5<- trainx[,5] ; x6<- trainx[,6] ; x7<- trainx[,7]

x8<- trainx[,8]

train1x <- cbind(x1, x2, x3, x4, x5, x6, x7,x8)

# Use train1x and train1y for training

# Testing data

testx <- read.csv("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/testx",sep=" ",header=F)

testy <- read.csv("http://www.cs.toronto.edu/~rsalakhu/STA414\_2015/testy",sep=" ",header=F)

test1y <- testy[[1]]

test1y

ax1 <- testx[,1]; ax2 <- testx[,2]; ax3<- testx[,3]; ax4<- testx[,4]; ax5<- testx[,5]; ax6<- testx[,6]; ax7<- testx[,7]

ax8<- testx[,8]

test1x <- cbind(ax1, ax2, ax3, ax4, ax5, ax6, ax7,ax8)

testx <- 0; testy <- 0; trainx <- 0; trainy <-0

### Linear model ###

### Training model

ptm <- proc.time()

mod0 <- lm(train1y ~ train1x)

summary(mod0)

### Making predictions

betas <- mod0$coefficients

pred0 <- test1x %\*% betas[2:9]

pred <- pred0 + betas[1]

pred[1]

### Evaluating predictions

sse0 <- sum((pred - test1y)^2);sse0 # SSE 719.1098

mse0 <- sse0/length(test1y); mse0

# MSE .2876

proc.time() - ptm

### GP with linear covariance ###

### Setting up noise-free covariance

# Training set train1x - 8 by 250

ptm <- proc.time()

K <- matrix(as.numeric(0),nrow=250,ncol=250)

for (z in 1:250){

for (w in 1:250){

K[z,w] <- sum(train1x[z,] \* train1x[w,]) \* 100^2 + as.numeric(z==w)

}

}

invK <- solve(K) # Inverse

### Making predictions

pred1 <- numeric(2500)

kk <- function(testindex){

kk <- numeric(250)

for (ll in 1:250){

kk[ll] <- sum(train1x[ll,] \* test1x[testindex,]) \* 100^2

}

kk

}

k <- numeric(2500)

invKt <- invK %\*% train1y

for(ii in 1:2500){

k[ii] <- kk(ii) %\*% invKt

}

### Prediction vector

k

sse1 <- sum((k-test1y)^2); sse1 #SSE is 1108.733

mse1 <- sse1/2500; mse1 # MSE is .4435

proc.time() - ptm

#### Fitting GP models #####

install.packages("parallel");library(parallel)

# Note how new K is identical to MSE

# Crossvalidation 1

ctrx1 <- train1x[1:225,] ; ctex1 <- train1x[226:250,] ; ctry1 <- train1y[1:225] ;ctey1 <- train1y[226:250]

### Creating crossvalidation data

CROSSX <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

for(i in 1:10){

t <- i\*25

w <- ((i-1)\*25)+1

if (i < 2){

CROSSX[[i]] <- train1x[i:t,]

}

CROSSX[[i]] <- train1x[w:t,]

}

CROSSY <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

for(i in 1:10){

t <- i\*25

w <- ((i-1)\*25)+1

if (i < 2){

CROSSY[[i]] <- train1y[i:t]

}

CROSSY[[i]] <- train1y[w:t]

}

assembler.x <- function(listf){

final <- NA

outtest <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

outtrain <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

for (j in 1:10){

for (i in 1:10){

if (i < j){

outtrain[[j]] <- rbind(outtrain[[j]],listf[[i]])

}

if (i > j){

outtrain[[j]] <- rbind(outtrain[[j]],listf[[i]])

}

outtest[[j]] <- listf[[j]]

}

}

for (z in 1:10){

outtrain[[z]] <- outtrain[[z]][2:226,]

}

final <- list(outtrain,outtest)

final

}

assembler.y <- function(listf){

final <- NA

outtest <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

outtrain <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

for (j in 1:10){

for (i in 1:10){

if (i < j){

outtrain[[j]] <- c(outtrain[[j]],listf[[i]])

}

if (i > j){

outtrain[[j]] <- c(outtrain[[j]],listf[[i]])

}

outtest[[j]] <- listf[[j]]

}

}

for (z in 1:10){

outtrain[[z]] <- outtrain[[z]][2:226]

}

final <- list(outtrain,outtest)

final

}

# First training set, excluding obs 1-25: assembler(CROSSX)[[1]][[1]]

#train.CROSSX.2 <- rbind(CROSSX[[1]],CROSSX[[3]],CROSSX[[4]],CROSSX[[5]],CROSSX[[6]],CROSSX[[7]],CROSSX[[8]],CROSSX[[9]]

# ,CROSSX[[10]])

#train.CROSSY.2 <- c(CROSSY[[1]],CROSSY[[3]],CROSSY[[4]],CROSSY[[5]],CROSSY[[6]],CROSSY[[7]],CROSSY[[8]],CROSSY[[9]]

# ,CROSSY[[10]])

#test.CROSSX.2 <- CROSSX[[2]]

#test.CROSSY.2 <- CROSSY[[2]]

# Use rbind to put together training datasets

### Creating vector of potential hyperparameters

gamt <- numeric(20)

rhot <- numeric(20)

for (i in 1:20){

gamt[i] <- gamt[i] + i\*.5

}

gamt <- c(.1,gamt[1:19]+.1,gamt[20]);gamt

for(j in 1:20){

rhot[j] <- rhot[j]+.05\*j

}

rhot <- c(.01,rhot[1:19]+.01,rhot[20]);rhot

# Basic functions

K.basic <- function (x1,x2,r,g){

100^2 + g^2 \* exp(-r^2 \* sum((x1-x2)^2))

}

K.train <- function (xt, x, rh, gam){

K3 <- matrix(0,nrow(xt),nrow(x))

for(i in 1:nrow(xt) ){

for (j in 1:nrow(x)){

K3[i,j] <- K.basic(xt[i,],x[j,],rh,gam) + as.numeric(i==j)

}

}

K3

}

# Testing K builder

# K.train(ctrx1,ctrx1,rhot,gamt)

pred1 <- function(trx1,trx2,try1,rho,gamma){

predd <- numeric(25)

Kmed <- K.train(trx1,trx1,rho,gamma)

invKmed <- solve(Kmed)

invKmedt <- invKmed %\*% try1

for(jj in 1:25){

ip <- numeric(225)

for(ee in 1:225){

ip[ee] <- K.basic(trx1[ee,],trx2[jj,],rho,gamma)

}

predd[jj] <- ip %\*% invKmedt

}

predd

}

# Putting together cross validation sets

#### Final calculations

mscal\_ext <- function(X,Y,R,G){

# X is matrix of observations and covariates. dim(X)= NxD

# Y is vector of dependent variables. dim(Y)=Nx1

# R is vector of rhos

# G is vector of gammas

matx <- assembler.x(X) # List of 2 lists: (list of training matrices, list of corresponding test matrices)

maty <- assembler.y(Y) # List of 2 lists: (list of training matrices, list of corresponding test matrices)

holder <- numeric(10)

HH <- matrix(numeric(441),nrow=21,ncol=21)

for (a in 1:length(R)){

for (b in 1:length(G)){

print(c("gamma = ", G[b], " #### rho = ", R[a]))

for(c in 1:10){

print(c)

holder[c] <- (sum((pred1(matx[[1]][[c]],matx[[2]][[c]],maty[[1]][[c]],R[a],G[b]) -maty[[2]][[c]])^2))/25

}

HH[a,b] <- mean(holder)

holder <- numeric(10)

}

}

HH

}

#sum((pred1(ctrx1,ctex1,ctry1,rhot[1],gamt[1]) - ctey1)^2) # MSE = .4798

# Testing

#train.CROSSX.1 <- rbind(CROSSX[[2]],CROSSX[[3]],CROSSX[[4]],CROSSX[[5]],CROSSX[[6]],CROSSX[[7]],CROSSX[[8]],CROSSX[[9]]

# ,CROSSX[[10]])

#train.CROSSY.1 <- c(CROSSY[[2]],CROSSY[[3]],CROSSY[[4]],CROSSY[[5]],CROSSY[[6]],CROSSY[[7]],CROSSY[[8]],CROSSY[[9]]

# ,CROSSY[[10]])

#test.CROSSX.1 <- CROSSX[[1]]

#test.CROSSY.1 <- CROSSY[[1]]

#mscalc(train.CROSSX.1,test.CROSSX.1,train.CROSSY.1,test.CROSSY.1,rhot,gamt)

### Result ###

rhot

gamt

ptm <- proc.time()

BAMFM <- mscal\_ext(CROSSX,CROSSY,rhot,gamt); BAMFM # Cross validation matrix

# Optimal params: gamma 10, rho .16, MSE=.23488

# Generating MSE from predictions generated by optimal hyperparams

pred2 <- function(trx1,trx2,try1,rho,gamma){

predd <- numeric(25)

Kmed <- K.train(trx1,trx1,rho,gamma)

invKmed <- solve(Kmed)

invKmedt <- invKmed %\*% try1

for(jj in 1:2500){

ip <- numeric(250)

for(ee in 1:250){

ip[ee] <- K.basic(trx1[ee,],trx2[jj,],rho,gamma)

}

predd[jj] <- ip %\*% invKmedt

}

predd

}

pred\_final1 <- pred2(train1x,test1x,train1y,rhot[4],gamt[21]);pred\_final1

ssegp1 <- sum((pred\_final1-test1y)^2); ssegp1

msegp1 <- ssegp1/2500; msegp1

proc.time() - ptm

# Export BAMFM to Excel

write.csv(BAMFM,file="BAMFM.csv")

### Reconsidering all 3 methods for the transformed dataset

# Divide covariates 1 and 7 by 10

train1x; test1x; train1y ; test1y

newtrain1x <- cbind(train1x[,1]/10,train1x[,2:6],train1x[,7]/10,train1x[,8])

newtest1x <- cbind(test1x[,1]/10,test1x[,2:6],test1x[,7]/10,test1x[,8])

CROSSX2 <- list(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA)

for(i in 1:10){

t <- i\*25

w <- ((i-1)\*25)+1

if (i < 2){

CROSSX2[[i]] <- newtrain1x[i:t,]

}

CROSSX2[[i]] <- newtrain1x[w:t,]

}

### Linear model 2###

### Training model

ptm <- proc.time()

mod1 <- lm(train1y ~ newtrain1x)

summary(mod1)

### Making predictions

betas1 <- mod1$coefficients; pred1 <- newtest1x %\*% betas1[2:9]; pred2 <- pred1 + betas1[1]

### Evaluating predictions

sse1 <- sum((pred2 - test1y)^2);sse1 #

mse1 <- sse1/length(test1y); mse1

#

proc.time() - ptm

### GP with linear covariance 2 ###

### Setting up noise-free covariance

# Training set newtrain1x - 8 by 250

ptm <- proc.time()

Kale <- matrix(as.numeric(0),nrow=250,ncol=250)

for (z in 1:250){

for (w in 1:250){

Kale[z,w] <- sum(newtrain1x[z,] \* newtrain1x[w,]) \* 100^2 + as.numeric(z==w)

}

}

invKale <- solve(Kale) # Inverse

### Making predictions

pred\_linGP1 <- numeric(2500)

kk2 <- function(testindex){

kk <- numeric(250)

for (ll in 1:250){

kk[ll] <- sum(newtrain1x[ll,] \* newtest1x[testindex,]) \* 100^2

}

kk

}

kaley <- numeric(2500)

invKaleyt <- invKale %\*% train1y

for(ii in 1:2500){

kaley[ii] <- kk2(ii) %\*% invKaleyt

}

### Prediction vector

kaley

sselingp1 <- sum((kaley-test1y)^2); sselingp1 #SSE is 1108.733

mselingp1 <- sselingp1/2500; mselingp1 # MSE is .4435

proc.time() - ptm

### General GP 2 ###

ptm <- proc.time()

BAMFM2 <- mscal\_ext(CROSSX2,CROSSY,rhot,gamt);BAMFM2 # Cross validation matrix

### Optimal params gamma 4.1, rho 1, MSE=.28309

pred\_final2 <- pred2(newtrain1x,newtest1x,train1y,rhot[21],gamt[9]);pred\_final2

ssegp2 <- sum((pred\_final2-test1y)^2); ssegp2

msegp2 <- ssegp2/2500; msegp2

proc.time() - ptm

write.csv(BAMFM2,file="BAMFM2.csv")