**Paper: A Nonlinear Optimized PSO-SVR Hybrid System for Time Series Forecasting with ARIMA.**

Este documento describe un sistema híbrido para series de tiempo, que busca predecir fenómenos de tiempo real, basado en técnicas de machine learning y regresiones, aplicando optimizaciones como PSO.

Proponen un método no linear donde toman el dataset original, pasa por las predicciones del primer modelo, de este salen los datos residuales, estos pasan por un segundo modelo aplicando una optimización de PSO para encontrar los mejores parámetros, y luego pasa por un tercer modelo.

Las series de tiempo por lo general son combinaciones de modelos lineales y no lineales.

Se utiliza un 50% de los datos para entrenar y el otro 50% para probar los modelos.

Para los resultados, se comparan los errores cuadráticos, en total se calculan tres tipos de errores, donde mientras más bajo el valor, mejor es el modelo. Concluyen que el modelo propuesto obtiene mejores resultados en términos de precisión que los estudiados (ARIMA, SVR) lineales y no lineales.

La idea es primero trabajar con los datos lineales y luego utilizar técnicas de ML para los datos no lineales, donde se pueden utilizar optimizaciones.

**Machine learning intro in R: Support Vector Regression**[**https://rpubs.com/richkt/280840**](https://rpubs.com/richkt/280840)En este post, grafican unos datos random y realizan una regresión lineal simple sobre estos, lo cual genera una línea que muestra la pendiente de la tendencia de los datos.

Luego realizan una regresión con vectores de soporte (técnica de machine learning) a los datos, generando una curva que sigue la tendencia de los datos, se utiliza el paquete e1071 en R con la función svm() y predict().

Luego se afina el modelo con el método tune() que realiza una búsqueda grid de los mejores parámetros, luego se realiza una mejor búsqueda utilizando estos parámetros. (Se ha agregado un parámetro NU al tune y se modificaron los rangos, lo cual genera una demora mucho mayor en el método).

Se utilizan los $best.model de los resultados tune().

Se realiza predict a los resultados afinados y se visualizan, el gráfico muestra una mejor curva que se acerca a la curva de los datos reales.

Se comparan los errores de cada procedimiento.

**Conclusiones**

La idea es reemplazar la búsqueda grid por un método de optimización, para encontrar los hyper parámetros del modelo SVR, acortando el tiempo de procesamiento e igualando o mejorando los resultados obtenidos. El método de optimización propuesto es el PSO.

**Particle Swarm Optimization**[**https://rpubs.com/argaadya/intro-pso**](https://rpubs.com/argaadya/intro-pso)

Este blog primero presenta una introducción a los problemas y optimizaciones. Luego explica la idea general de la PSO. Luego expone algunos ejemplos en R. El primero es un problema del peso de un resorte, donde se presenta la función a optimizar, las restricciones y los posibles valores de las variables. Entonces define una función de evaluación, que descompone el vector dado en el argumento en las 3 variables del problema, luego define la función a minimizar utilizando estas variables, define las restricciones, y por último llama a la función que minimiza, donde si las variables violan las restricciones, se penaliza.

Se utiliza la función psoptim() del paquete pso, donde los argumentos son la dimensionalidad de la función a optimizar, la función a optimizar que se definió en el problema, las cotas inferiores y superiores de las variables, y por último parámetros de control de la PSO. Entonces se corre el procedimiento, realizando los cálculos necesarios para su análisis y comparándolo con un algoritmo genético, donde PSO es mejor.

Un ejemplo similar es presentado a continuación, esta vez con la compra de acciones en un portafolio dado la información obtenida de un dataset. Al igual que en el ejemplo anterior, los datos son procesados para crear una matriz de datos, luego se define la función a optimizar y se aplica el método PSO.

Por último, se realiza una optimización a una aplicación de machine learning, en este caso un random forest refinamiento de hyper parámetros. Se importan los datos, se separan para realizar cross validation y se pre procesan. Cuando se define la función de evaluación, se define el modelo de clasificación random forest, se define el motor del modelo y se afina utilizando fit y luego predict, retornando un vector de precisión. Finalmente se corre junto con PSO.

**Conclusiones**

Se podría hacer una función de evaluación, con la SVR vista en la publicación anterior, ¿dónde el afinamiento sea la búsqueda grid con predict? ¿Cuál sería el vector de precisión en este caso? Es decir, lo que debe retornar la función para que sea un argumento válido para psoptim?

**Paquete hydroPSO**[**https://cran.r-project.org/web/packages/hydroPSO/hydroPSO.pdf**](https://cran.r-project.org/web/packages/hydroPSO/hydroPSO.pdf)State-of-the-art version of the Particle Swarm Optimisation (PSO) algorithm (SPSO-  
2011 and SPSO-2007 capable). hydroPSO can be used as a replacement of the 'optim' R func-  
tion for (global) optimization of non-smooth and non-linear functions. However, the main fo-  
cus of hydroPSO is the calibration of environmental and other real-world mod-  
els that need to be executed from the system console. hydroPSO is model-independent, allow-  
ing the user to easily interface any computer simulation model with the calibration en-  
gine (PSO). hydroPSO communicates with the model through the model's own input and out-  
put files, without requiring access to the model's source code. Several PSO variants and control-  
ling options are included to fine-tune the performance of the calibration engine to different cali-  
bration problems. An advanced sensitivity analysis function together with user-friendly plot-  
ting summaries facilitate the interpretation and assessment of the calibration results. hy-  
droPSO is parallel-capable, to alleviate the computational burden of complex mod-  
els with ``long'' execution time.

**Conclusiones**Se podrá ocupar hydroPSO para encontrar los hyper parámetros de una nu SVR ? Un dato importante es que es paralelizable. No se encontraron ejemplos prácticos en Google de hydroPSO con machine learning, así que se debe utilizar la documentación del paquete para una investigación más profunda.

**e1071 (version 1.7-5)  
tune: Parameter Tuning of Functions Using Grid Search**

Description

This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges.

<https://www.rdocumentation.org/packages/e1071/versions/1.7-5/topics/tune>  
  
Esta es la función tune del paquete e1071, que realiza búsquedas grid.

Acepta como parámetro un method, el cual solo lo describe y no presenta los posibles valores, pero en su mayoría de usos es con svm, los otros parámetros son de datos y control. El final termina con un ejemplo.

**Conclusiones**

Muy ambiguo el funcionamiento interno de este método, no se encuentran fácilmente otras opciones en Google, tampoco mezcla o reemplazo de tune con una optimización PSO para hyper parámetros, ni con otra optimización, para SVM y SVR. Hay resultados en stackoverflow de preguntas sobre como mejorar el rendimiento de esta función: <https://stackoverflow.com/questions/34888062/r-improving-e1071-tuning-performance>.

**Próxima lectura:**

**Flexible and Robust Machine Learning Using mlr3 in R**[**https://mlr3book.mlr-org.com**](https://mlr3book.mlr-org.com)A simple vista, se ve muy interesante, ya que trata de optimización de hyper parámetros, utilizando un paquete en R, viene con un ejemplo práctico, utiliza SVMs y es una lectura extensa.

**Post-lectura:**

Es un ecosistema muy completo y complejo de utilizar, uno puede definir las clases y los objetos propios de los modelos y los aprendizajes, existen varios métodos, funciones, librerías involucradas en este proyecto, más abajo hay información más detallada de este ecosistema en R.

Documentación:

PDF: <https://mlr3book.mlr-org.com/Flexible-and-Robust-Machine-Learning-Using-mlr3-in-R.pdf>

Tuners: <https://mlr-org.com/tuners.html>

Tuning spaces: <https://mlr-org.com/tuning_spaces.html>

**TO DO (UPDATE 20/2):**

* **Investigar paquete CARET**
  + **El paquete CARET parece ser más efectivo realizando validaciones cruzadas y otras técnicas de machine learning que no han sido vistas aún. La librería e1071 parece más efectiva para el problema actual, además que puede realizar también una validación cruzada con el método tune()**
* **Escribir al profe sobre descubrimientos**
  + **Primer correo enviado, respuesta automática dice que estará disponible desde el 28 de feb. Contactar por esa fecha y tener reunión online para ver próximos pasos.**
* **Investigar sobre Cross-validation con series de tiempo**
  + **La validación cruzada divide el conjunto de datos en las partes que son especificadas por el investigador. Por ejemplo, se puede dividir el conjunto en 3 partes, donde 2/3 de los datos se utiliza para entrenar al modelo, y el otro 1/3 se utiliza para evaluarlo. Estas partes se llaman pliegues, y además, los conjuntos de entrenamiento y evaluación se intercambian para una mayor precisión y menor aleatoriedad en el modelo. En las series de tiempo, la señal se va dividiendo también en intervalos de tiempo.**
* **Investigar aún más sobre las SVR con NU**
  + **El nu es un hiperparámetro de la SVR, este puede ser optimizado, parece que al correr las pruebas sobre el ejemplo que se ha estado viendo, el hiperparámetro nu ofrece una mejor calidad en la predicción de los datos comparado con el hiperparámetro original.**
* **Leer otra vez mi propuesta de memoria, la memoria de Miranda y la memoria de Vallejos.**
* **Tratar de explorar el repositorio de datos de la autorregulación cerebral, para integrar los datos y los métodos ya utilizados.**
  + **Ya no tengo acceso al repositorio, pedir acceso.**
* **Investigar sobre computación paralela en R**
* **Escribir código en R con los descubrimientos ya realizados, con ejemplos de prueba.**
  + **Se han realizado casi todos los experimentos propuestos de prueba, excepto por los de validación cruzada, que esos parecen ser más complejos utilizando el conjunto de datos del ejemplo.**
* **Crear hábito de investigación, en promedio unos 3-4 días a la semana, 4-6 horas por cada día.**
  + **Se ha conseguido durante un par de semanas, pero cuando empecé a realizar pruebas con la validación cruzada, las dificultades me desmotivaron para seguir avanzando.**
* **Pensar constantemente en esta investigación.**
  + **Las primeras semanas si, luego me quedé sin ideas. Ampliar.**
* **Generar un esquema de métodos y datos.**
  + **He desarrollado más o menos un mapa mental de los datos y los modelos que debo utilizar, sin embargo, aún hay vacíos de información, quizás sería una buena idea realizar un diagrama más visual.**
* **Investigar en profundidad sobre métodos de optimización y machine learning.**
  + **He llevado a la práctica lo de las máquinas de vectores de soporte, lo de la optimización por enjambre de partículas y lo de la búsqueda grid. Debo aprender más sobre la teoría.**

**Lunes 6 de febrero 2023**

**Investigaciones con Chat GPT**

* Hay varias opciones para optimizar los hiperparámetros de una SVR que clasifica modelos de autorregulación cerebral en R. Algunas librerias populares son:

1. mlr3: Esta es una librería moderna y completa para aprendizaje automático en R que incluye una amplia variedad de algoritmos de aprendizaje supervisado y no supervisado, así como herramientas para optimizar los hiperparámetros.
2. caret: Esta es una librería popular para la construcción de modelos de aprendizaje automático en R, que incluye una amplia variedad de algoritmos de aprendizaje supervisado y no supervisado, así como herramientas para optimizar los hiperparámetros.
3. e1071: Esta es una librería que incluye algunas funciones para aprendizaje automático en R, incluida la implementación de la SVM. Incluye una función tune que puede utilizarse para optimizar los hiperparámetros de la SVM.

En general, cualquiera de estas librerías podría ser adecuada para tus necesidades, y la elección final dependerá de tus preferencias personales y del problema específico que deseas resolver. Te recomiendo que evalúes cada una de ellas y compares su funcionalidad y facilidad de uso antes de tomar una decisión.

* Para completar una tarea de optimización de hiperparámetros con una gran cantidad de datos, puedes considerar la utilización de métodos de optimización basados en búsqueda bayesiana, como la optimización bayesiana con mlr3 o la libreria BayesianOptimization en R. Estos métodos son generalmente más eficientes que la búsqueda grid y pueden producir soluciones de mejor calidad en menos tiempo.

Sin embargo, la eficiencia y la calidad de la solución también dependen de otros factores, como el tamaño y la complejidad de los datos, la capacidad de la máquina en la que se está ejecutando el código, y la implementación específica de los algoritmos. Por lo tanto, te recomiendo que evalúes y compares diferentes métodos y librerías para asegurarte de elegir la opción que mejor se adapte a tus necesidades específicas.

* El método "predict" es una función común en la mayoría de los paquetes de aprendizaje automático que se utiliza para realizar predicciones en nuevos datos a partir de un modelo previamente entrenado.

El método "predict" toma como entrada un modelo previamente entrenado y un conjunto de datos de entrada nuevos, y devuelve un vector de predicciones correspondientes a cada observación en los datos de entrada nuevos.

En el contexto de una SVM, "predict" se utiliza para hacer predicciones de la clase o valor numérico en los nuevos datos utilizando el modelo SVM previamente entrenado.

* En el ejemplo de mlr3, se instala el paquete "paradox" porque se utiliza para crear un objeto de "resampling", que es un método común en el aprendizaje automático para realizar validación cruzada. La validación cruzada es un método que permite evaluar el desempeño de un modelo en datos que no se han utilizado para entrenar el modelo.

El paquete "paradox" proporciona una variedad de métodos de resampling, incluida la validación cruzada, que se pueden utilizar para evaluar el desempeño de los modelos de aprendizaje automático. Al utilizar "paradox" para realizar la validación cruzada, se puede tener una evaluación más precisa y robusta del desempeño del modelo en los datos de validación.

**Conclusiones:**

Realicé varias preguntas a chat gpt sobre el problema en cuestión, librerías y código que puedo utilizar en R. Dejé funcionando un ejemplo en R donde se utiliza una PSO a los hiperparámetros de uns SVR, en este caso el costo y gamma (que en realidad tiene que ser sigma), a señas de series de tiempo. Faltaría agregarle el hiperparámetro Nu y la validación cruzada al problema, pero parece que chat gpt tiene problemas para mezclar estas soluciones, ya que daba soluciones dispersas que en su mayoría se caían en algún punto, por ejemplo, las llamadas a las funciones, y los tipos de datos de sus argumentos.

**Martes 7 de febrero 2023**

Traté de realizar una optimización PSO a la SVR que sale en el ejemplo de rpubs del presente documento en un nuevo script de R. Con la ayuda de chat gpt pude definir una función objetivo en base a minimizar los errores cuadráticos, y aplicando la función psoptim de la librería pso. Utilizando los valores de épsilon y costo encontrados en el tune() realizado anteriormente, como parámetros de entrada para las cotas inferiores y superiores. Luego se realiza una svm con los parámetros obtenidos y se realiza un predict de los datos, para luego visualizarlos. La curva que genera psoptim es muy parecida a la del segundo tune(), pero queda todavía realizar el análisis del error para comprar estos dos métodos.

En un principio, se mide el tiempo de ejecución del segundo tune() y de psoptim, donde tune() daba un promedio de 7-8 segundos de ejecución, mientras que psoptim daba un tiempo de ejecución de 40-50 segundos. Luego se buscaron que parámetos de control podían afectar el tiempo de ejecución de PSO, que luego fueron cambiados. Estos fueron el tamaño del enjambre, el número de iteraciones, el mejor resultado local y global, y el peso incercial. Jugando con estos valores, se llega a unos 4-5 segundos de ejecución, menor al valor del segundo tune(), y con la curva no cambiando casi nada.

Documentación de la librería PSO:

<https://cran.r-project.org/web/packages/pso/pso.pdf>

**Conclusiones**

Falta limpiar el código para comparar bien estos métodos, realizar una función de psoptim para ir variando los parámetros y ver que pasa con los resultados, e implementar este mismo método ojalá con las librerías caret y mlr3.

**Miércoles 8 de febrero 2023 (3 days streak!)**

En esta oportunidad, iba a dejar este día como de descanso, pero aproveché de limpiar un poco el código, lo ordené y apliqué la optimización PSO con 2 iteraciones consecutivas, en vez de realizar un tune() y luego PSO. Además, calculé la suma de los errores cuadráticos para cada modelo y también realicé la visualización demostrativa, además realicé un análisis del tiempo de ejecución de cada método por separado con la función nativa Sys.time()

**Conclusiones**

Los experimentos han dado resultados exitosos, se ve una mejora de aproximadamente un 60% del tiempo de ejecución, además que mejora el rendimiento del modelo.

Los siguientes pasos son:

* Agregar el hiperparámetro nu y sigma, quitar épsilon.
* Alimentar a los modelos con datos de señales de series de tiempo.
* Análisis de validación cruzada.
* Análisis de múltiples optimizaciones, para cada objetivo de estudio.

**Jueves 9 de febrero 2023 (4 days streak)**

Dupliqué el script de R de SVR para realizar el ejemplo anterior, pero ahora con los hiperparámetros sigma, costo y nu. Después de algunas modificaciones, agregué los 3 hiperparámetros, pero la segunda vez que entraba a la búsqueda grid, el sigma daba nulo en el mejor modelo, después de investigar un poco resulta que en teoría la SVM debería tener un Kernel de tipo radial para ocupar el hiperparámetro sigma. Por otro lado, se tiene que existen dos SVM, las lineales y las no lineales, las lineales utilizan solo costo y nu, y las no lineales le agregan el sigma. En la tabla de parámetros, existe el parámetro de costo lineal y exponencial, que es el único que cambia. Se realizaron las optimizaciones con los parámetros costo y nu, pero el rango de valores estaba un poco alterado, por lo que los resultados pueden estar afectados, con respecto a estos resultados, la búsqueda grid demoraba 60 segundos, y pso 30 segundos, pero no mejoró la calidad del modelo.

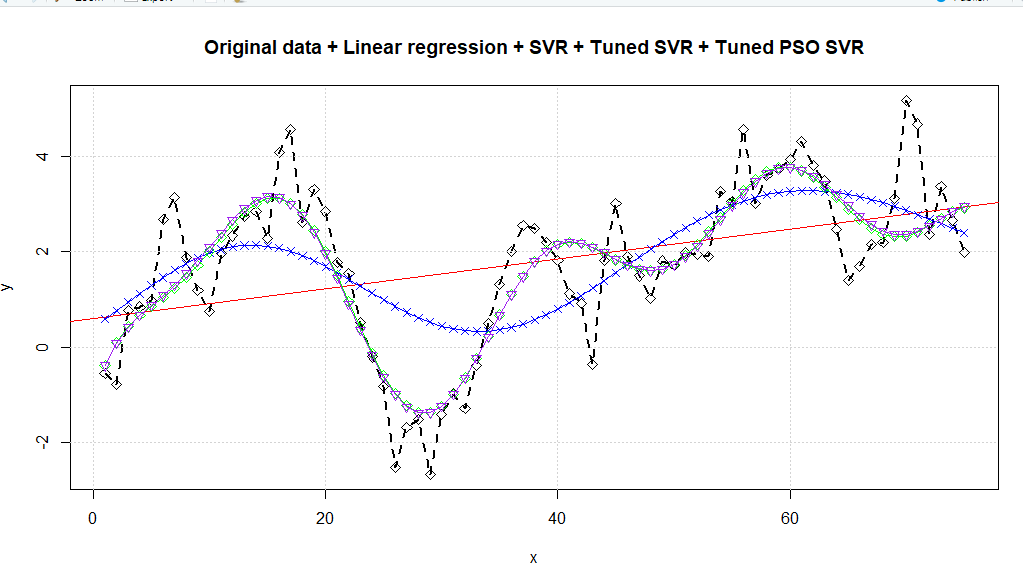
**Conclusiones**

Realizar el experimento del ejemplo con los modelos lineales y no lineales, los hiperparámetros con sus rangos de valores correctos, y comparar los resultados. Para esto definir el kernel de las svm, e investigar sobre los rangos de los hiperparámetros.

**Viernes 10 de febrero 2023 (5 DAYS STREAK)**

Primero comentar que chat gpt tiene una confusión con algunas librerías de R porque tienen nombres iguales con algunas funciones de otra librería, por ejemplo existe la librería pso con el método psoptim, pero también existe la librería psoptim por si sola.

También me he fijado en el valor del error al final de las predicciones del modelo, este error es cercano a 1 incluso al final de la segunda optimización, lo máximo que ha bajado ha sido 0.86 aproximadamente. Esto refleja lo difícil que es poder predecir señales de series de tiempo reales, ya que estoy ejemplos están hechos con datos de distribución normal aleatorios, pero que son muy parecidos en su naturaleza a las señales de AR.



Por ejemplo en esta curva se puede apreciar la línea punteada negra al fondo son los datos reales, mientras que la línea verde y morada intentan seguir a esta línea original, sin embargo esta es muy cambiante por lo tanto se le hace difícil poder predecir con exactitud el comportamiento de estos datos, lo cual hace que sea mayor el error de los modelos.

Por último, comentar que los rangos de valores, los parámetros, como se realiza la optimización y los modelos, depende de cada problema en particular, por eso en este ejemplo algunos valores funcionan bien y otros no tanto. Lo importante es ir ajustando estos modelos, de acuerdo al contexto del problema en cuestión. Esto es importante porque más adelante se tendrá que optimizar el problema de la predicción de señales de la AR.

Después de modificar el código, actualmente se realiza la PSO sobre la svr con los parámetros costo y NU lineales, todavía no se prueban los no lineales y no se ha agregado el sigma. También se han cambiado el rango de los valores, que se adecuan más a los valores del experimento del problema. Durante la ejecución del código, se fueron ajustando algunas variables y valores. Por ejemplo, cuando se cambiaron los hiperparámetros, la calidad de los modelos del grid y PSO ahora está igualada, pero lo curioso es que se quitaron los parámetros extra de control de la pso, y se redujo el tamaño del enjambre y número de iteraciones, lo cual produce un menor tiempo de ejecución del algoritmo. Además, uno espera que aumentar el tamaño del enjambre o el número de iteraciones aumentaría la calidad del modelo, pero esto no fue así ya que disminuía. Por otro lado, en la función objetivo de la PSO, se calcula el coeficiente de correlación, este se maximiza por lo que se debe multiplicar por -1. Este coeficiente se utiliza en el problema de AR. También se intentó agregar el coeficiente de variación, para que junto al de correlación afectaran al objetivo de la función, sumando estos valores, pero el experimento no salió del todo bien, ya que disminuyó la calidad del modelo, de 0.86 a 1.2 (el error) aproximadamente. Para mejorar esto, se modificó el rango de los valores de los parámetros, y se modificó la variable par de PSOPTIM que es la cantidad de la que parten los parámetros de entrada. Pero la calidad del modelo seguía igual, incluso cambiando algunos parámetros de control extra de PSOPTIM, y también influenciando en el tiempo de ejecución. Queda para futura investigación seguir por este camino.

**Conclusiones**

Se comenzó a implementar validación cruzada en el modelo. Se intentó utilizar el paquete caret, pero este no tiene SVR con parámetro NU. Queda también realizar el análisis de kernel radial (modelos no lineales). Alimentar los modelos con series de tiempo (aunque los datos que se utilizan en el ejemplo se parecen bastante) y poder optimizar más de un objetivo.

**Lunes 13 de febrero 2023**

Continuación de la implementación de validación cruzada, pero los parámetros costo y NU no pudieron ser optimizados, esto se puede deber a varias razones. Se dejó hasta ahí el código.

Por otro lado, se implementó una SVR no lineal con kernel radial, y se agregó el parámetro sigma a las optimizaciones, los resultados son comprometedores, ya que se pudo agregar con éxito el parámetro a las optimizaciones grid y pso. La optimización grid ahora toma aproximadamente 2 minutos, y como resultado la precisión del modelo a mejorado unos 0.4 puntos aprox. En cuanto a PSO, se cambiaron algunos valores de los parámetros de control, y se podía observar una disminución de la precisión del modelo, aproximadamente 0.6 puntos, pero el tiempo de ejecución era significativamente menor a la búsqueda grid. Ya que esta se realiza en pocos segundos. Al cambiar la función objetivo de la PSO de coeficiente de variación a coeficiente de correlación, se observo una igualdad en las precisiones del modelo con búsqueda grid y con PSO, y una reducción del tiempo de ejecución cambiando algunos parámetros de control. También se cambiaron los espacios del rango de valores de los hiperparámetros.

Por último, se intentó alimentar el modelo con una señal de tiempo real, no se pudo aún encontrar una librería con datos que simulen estas señales, pero se generaron datos a partir de la función seno, que realiza un gráfico sinusoidal con los datos. Al pasar los datos como inputs generaban error en los parámetros, uno de estos daba menor o igual a 0, lo cual no puede ser recibido por el método SVM.

**Conclusiones**

Los datos aleatorios de distribución normal del ejemplo visto parecen ser una simulación más aproximada a una señal de tiempo real que la función seno vista en este día. Así que la idea es quedarse con esos datos. Además, falta una investigación para realizar la validación cruzada a este ejemplo.

**16 de febrero 2023**

Intenté realizar una validación cruazada a los experimentos que estoy realizando, a la búsqueda grid y a la pso de la svr lineal con los hiperparámetros correspondientes.

Primero intenté incluir una validación cruzada dentro del método tune, que parece que acepta un argumento de validación cruzada, pero no funcionó por que no toma los datos de entrada (además había que definir una función auxiliar para entregarle a tune). Después intenté realizar una validación cruazada manual de 3 pliegues junto con el algoritmo PSO implementado, lo cual resulta en una precisión muy cercana a la de una regresión lineal simple, pero se demora más o menos 10 segundos. Se intentaron cambiar los parámetros de psoptim, y también el rango de valores de los hiperparámetros, pero no aumentaba la precisión del modelo. Además, al aumentar el número de partículas e iteraciones, el tiempo de ejecución aumenta de manera exponencial.

**Conclusiones**

Se está realizando una nueva validación cruzada con la búsqueda grid, sin embargo, este método contiene warnings en la ejecución del código, y además parece que el tiempo de ejecución ahora es mucho mucho mayor, ya que aún no termina de ejecutarse la primera iteración de la búsqueda grid y ya lleva un buen rato, siendo además la primera iteración.

Queda realizar una búsqueda pso en ves de la grid con este enfoque de validación cruzada, para ver si el tiempo de ejecución baja, y revisar los resultados de la precisión del modelo.

Actualización: Acaba de terminar de correr el código, se demoró 10 min y tiene la misma precisión que la regresión lineal (un poco mejor que esta). Esto se puede deber a varios factores. (En el método se muestra que fue una validación cruzada de 10 pliegues, y en el argumento dice cross = 5, averiguar porque)

**Lunes 20 de febrero 2023**

Se realiza un mini repaso por el capítulo de TODO de este documento, donde se describe y se detalla algunos pasos ya realizados. Se realiza una lectura a la memoria de Vallejos, entiendo mucho mejor el problema, y también la parte teórica en cuestión. Se intenta realizar una simulación de los datos en R con la ayuda de chat gpt, sin embargo, estos datos están muy dispersos a pesar de agregar una normalización y un posterior procesamiento a los datos. Las variables fueron la PAM y la etCO2. Luego se revisa de nuevo el documento y pasa que los datos son procesados con métodos experimentales para obtener las observaciones de las señales, así que hay que esperar a tener las muestras reales.

Chart, scatter chart

Description automatically generated

**Conclusiones**: Pareciera que la validación cruzada se utiliza como técnica de validación después del entrenamiento con las SVM. Luego se realiza la selección de parámetros con la OMO. En teoría se debería poder realizar la PSO junto con este entrenamiento SVM y la validación cruzada para encontrar los parámetros óptimos. Se observa también que dada la naturaleza de la OMO, se debieron establecer múltiples objetivos para el problema, lo cual resulta en la interrogante de que si se podrá optimizar solo uno de estos objetivos, como por ejemplo la correlación, con la PSO para así poder maximizar la eficiencia y la precisión de los modelos?

Lunes 13 de mar. de 23

Ya tengo acceso al repositorio en github del proyecto del profe Jara antiguo y su mezcla con Miranda con título parcial. En el original se encuentra una carpeta de nombre Data, donde hay 3 archivos .txt de muestras reales de sujetos. He realizado un script en R que lee unos de estos archivos (sujeto 1), las columnas de las observaciones son el tiempo, la PAM, las velocidades del flujo sanguíneo cerebral izquierda y derecha, y la etCO2. En teoría, las señales que se van a generar tienen que parecerse lo más posible a una de las velocidades, o a las 2 si es que se puede.

Como se sabe, se van a utilizar 4 tipos de modelos, los de respuesta de impulso finito (FIR) y los de ARX (autoregressive with exogenous variables), lineales y no lineales ambos. Además, estos se van a dividir en dos, aquellos que reciben la VFSC como retardos y los que no la reciben.

Bajo este marco, se deben normalizar los datos con la ecuación dada en las memorias de Miranda y Vallejos.

En la carpeta de mezcla con Miranda se encuentra varios tipos de archivos:

La carpeta VEP: contiene archivos get-models, que cargan funciones de otros archivos, como funciones de grid, lags, simulaciones y svr. Luego parece que definen unos parámetros de svr y dentro de un for hacen un grid lags and parameters.

El archivo ARSVM\_SIM: realiza simulaciones de svr autoregresivas.

Un archivo txt de escalón negativo.

Un script de descarte de señal.

Varios archivos de test, en algunos se ocupa señales simuladas.

Un Excel de modelos.

En la carpeta original, se tiene un lag manager, una validación cruzada de señales, grids, laggings, etc.

El enlace al repo: <https://github.com/jljara/CHsvm-antiguo/tree/main/original>

Primero se deberá trabajar con un modelo FIR univariado sin retardos, definición de FIR:

*Los modelos FIR (del inglés, "Finite Impulse Response") son un tipo de modelo matemático que se utiliza en el procesamiento digital de señales.*

*Estos modelos se caracterizan por tener una respuesta al impulso finita, lo que significa que su salida depende solo de un número finito de entradas anteriores. En otras palabras, el modelo FIR utiliza una ventana de tiempo fija para calcular la salida actual en función de las entradas pasadas.*

*La ecuación general de un modelo FIR es:*

*y(n) = b0x(n) + b1x(n-1) + b2x(n-2) + ... + bMx(n-M)*

*Donde "y" es la salida del modelo, "x" es la entrada, "b" son los coeficientes del modelo y "M" es el número de coeficientes utilizados en el modelo.*

*Los modelos FIR se utilizan comúnmente en aplicaciones de filtrado, ecualización, interpolación y predicción. Una de sus ventajas es que son fácilmente implementables en sistemas digitales y su respuesta al impulso finita los hace más estables y menos propensos a oscilaciones que otros tipos de modelos.*

Se le ha enviado un correo al profe debido a un bloqueo de lo que se debería hacer ahora, si no se tiene respuesta de aquí a mañana, se continuará investigando sobre estos modelos y la simulación de señales que están presentes en algunos archivos del repositorio. Se intentará entender alguno de los archivos que realiza alguna función main, importando y utilizando la demás.

Martes 14 de mar. de 23

Profe aún no responde el correo.

Intenté averiguar sobre los modelos que se utilizan en la investigación, como primera instancia, el univariado, con entrada PAM y salida VFSC, de vectores de soporte de regresión con señales FIR.

Se pueden mezclar estos dos tipos de modelos, FIR con SVR, para entrenar y luego ajustar sobre los sujetos, pero primero se debe entrenar a los modelos.

Se leyó el marco teórico de la memoria de Miranda, rápidamente. Dice que estas señales de autorregulación cerebral es un sistema dinámico, es decir, que depende de las entradas de tiempo anteriores, de ahí que se deben modelar utilizando FIR y ARX. Estas señales de los sujetos podrían ser transformadas a matrices de datos y coeficientes, por ejemplo.

También, se incluye información importante sobre las SVM, y que las de regresión son un tipo especial de SVM, y que además, la inclusión del parámetro NU reemplaza al parámetro épsilon, forzando una búsqueda más ajustada del modelo.

También se entendió que se trata más sobre un problema de optimización, que necesariamente tiene que ser convexo, para encontrar un mínimo óptimo, que es lo que se intenta encontrar, a pesar del ruido que pueden tener las observaciones. El problema es que no siempre se tiene certeza sobre que ruido pueden tener los datos, por eso se utiliza la v-SVR.

También se revisó otra vez el código, se cargó en el entorno de visual code studio. Se intentó buscar la palabra clave SVM(), que es el método perteneciente a la librería e1071. Pero se encuentran muy pocas ocurrencias en el código en general. Se encuentra en el archivo de ARSVM\_sim, que básicamente provee funciones para ajustar modelos AR y ARSVM, utilizando datos simulados. Los usos de estas funciones son variados, pero los principales son poder evaluar y entrenar estos modelos.

También se encuentran ocurrencias de svm() en los archivos de test1 y test2, test1 es casi lo mismo que el de ARSVM\_sim, test2 contiene más funciones que se deben analizar.

Se intentó buscar la palabra clave FIR dentro del código, pero se encontrar super pocas ocurrencias, y están dentro de comentarios muy básicos, no se entiende muy bien a que se refieren.

También se recordó que VEP significa variación espontánea de presión. Los archivos test-ARI y test-models son grandes y contienen mucho código.

Quizás deba revisar uno a uno los archivos de test, y el archivo de ar-cross-validation.

Martes 21 de mar. de 23

El viernes de la semana pasada tuve reunión con el profe, hablamos sobre la memoria y le pregunté sobre los pasos a seguir desde ahora.

Voy a trabajar con el modelo FIR univariado, solo con la presión media arterial como entrada y como salida la velocidad del flujo sanguíneo cerebral.

Para esto, el profesor Jose Luis me dijo que leyera las muestras del Sujeto número 1, el cual contiene sus respectivas señales cerebrales.

Los datos dentro del archivo .txt son los siguientes:

* + Tiempo: esta es la variable independiente (x) y aumenta en intervalos de 0.5 segundos.
  + MAPB: Esta es la presión arterial media, oscila entre los 90 y 115, pero al normalizarla, oscila entre 0 y 1.
  + VFSC: Es la velocidad del flujo real de la persona, del hemisferio derecho e izquierdo del cerebro.
  + ETCO2: Es la cantidad de dióxido de carbono en la sangre.

Se construyó un script en R que lee estos datos, y almacena el tiempo y la PAM en dos variables, luego estas variables se grafican, quedando de la siguiente manera:  
  
Chart

Description automatically generated

Luego, se comenzó a verificar la memoria de Miranda para seguir los mismos pasos que el realizó con estos datos.

Según Jose Luis, esta señal se debe dividir en 2, la primera mitad de los datos y la segunda. Se debe entrenar el modelo FIR univariado con la primera mitad, y se debe validar con la segunda mitad, luego se intercambian los pliegues de validación y de entrenamiento. Esta técnica se le conoce como validación cruzada balanceada de dos pliegues.

Luego para validar los modelos, se debe utilizar el coeficiente de correlación, y además se debe realizar un descarte de señales, para que los modelos no estén sobre entrenados.

Según Jose Luis, esta es la parte más difícil, así que al resolver esto se podrá continuar más fácilmente.

Entonces, dividimos la señal en 2 mitades en el script de R.

Luego se creó un modelo v-SVR con el paquete e1071 entrenada con la primera mitad, y se realizaron las predicciones con la segunda mitad, y se calculó el índice de correlación entre estas dos mitades. Se debe validar este procedimiento con el profe.

Luego, se definió el rango de valores de v y del costo, y se realizó una búsqueda grid de los hiperparámetros, entrenando y validando como en el procedimiento anterior, y finalmente se calcula el coeficiente de correlación.

Finalmente, se realiza una PSO sobre una v-SVR, optimizando el coeficiente de correlación. Y también se entreno y validó como en los pasos anteriores, calculando el índice de correlación entre las señales.

Al correr el algoritmo, se obtienen los siguientes resultados: la correlación entre la primera mitad y la segunda es de 0.17 para las 3 técnicas, una correlación muy baja. Además, el tiempo de ejecución de la búsqueda grid es de 1.8 min y la de PSO es de 1.9 min.

Estos resultados son muy diferentes a los casos de prueba vistos anteriormente, donde el error de las predicciones tanto de la búsqueda grid como la de PSO no era tan alto, además de que PSO tenía un tiempo de ejecución mucho más bajo que la búsqueda grid.

También se puede observar que la segunda búsqueda grid (más fina) se realiza en menos tiempo que la primera, y que la segunda de la prueba del ejemplo que vimos anteriormente.

También se observa que se la consola arroja alertas de máxima cantidad de iteraciones alcanzadas, 1 para la búsqueda grid y 7 para la PSO. Esto se puede deber a varios factores, entre estos, que las técnicas de búsqueda no pudieron encontrar una buena solución, esto debido quizás a varias razones, puede ser los valores de los hiperparámetros de entrada, o los parámetros de control de las técnicas utilizadas.

También, se cambió el rango de parámetros del costo a un rango más pequeño que el de prueba. También, tanto en la búsqueda grid como en PSO, el rango del costo en la segunda iteración cambió a dividido y multiplicado por 2.

También, en la técnica PSO, se comenzó con unos valores iniciales de 2000 y 0.5 para costo y v.

Queda todavía investigar porque esta correlación está dando tan baja, y porque las técnicas se están comportando tan extraño, como por ejemplo PSO que tiene más tiempo de ejecución, y que ambas técnicas alcancen los límites de iteraciones máximas.

Además, otro cambio que se hizo fue agregar el tipo de svm en los métodos del paquete.

ACTUALIZACIÓN:

Después de un rato mirando el código y los datos, realicé el grafico de las predicciones que se estaban realizando, y el gráfico de las predicciones correspondía a una línea que seguía la tendencia de los datos, como realizar una regresión lineal. Era raro si que la correlación diera tan baja, si igual había una tendencia bien marcada de los datos, además, la gráfica de la línea de la predicción no estaba bien posicionada con los datos reales, así que cambié las variables de entrenamiento y las de validación en los modelos, en las predicciones, en la correlación y en los gráficos.

Se pudo observar que, aunque la segunda mitad son los datos de validación, es la primera mitad la que tiene los datos que son predichos, así que la correlación y la gráfica se deben hacer con estos datos.

Al final, se obtiene una correlación de 0.6 aprox para las 3 técnicas de entrenamiento. Y el gráfico queda más o menos así:

Chart

Description automatically generated

Además, la técnica de la búsqueda grid es la que tiene más baja correlación de las 3 (aunque es un valor despreciable) y esta se demora aproximadamente 6 min de ejecución, mientras que PSO tiene la mayor correlación, aunque también despreciable, y su tiempo de ejecución fue de 1 minuto.

Además, se aumentó el rango de valores del costo, y aparecieron algunas advertencias de iteraciones máximas alcanzadas.

Luego, se cambiaron todos los kernels de las svm a radiales, y se ejecutó el código, dando como resultado el siguiente gráfico:  
  
Chart, histogram

Description automatically generated  
  
Mucho mejor que el kernel lineal, además, las correlaciones mejoraron a 0.74 para svm sola, y 0.81 a grid y pso, donde PSO tiene mayor correlación por valores muy pequeños.

Además, el tiempo de ejecución de grid fue de casi 6 minutos y PSO casi 2 minutos. También, no se vieron las advertencias de iteraciones máximas alcanzadas en la consola.

Queda todavía bajar la cantidad de valores del costo y correr nuevamente el algoritmo, además, no se consideran los retardos en esta pasada, por lo que deben ser agregados en la próxima iteración.

También hay que hacer lo de el descarte de señales, sería bueno investigar sobre el escalón negativo.