DSNBOOST

V.5

Report 10/12/2015

Author:

MANUEL MONTOYA CATALÁ

1. INTRODUCTION

El objetivo principal de este report es caracterizar el comportamiento cualitativo de DSNBoost para diferentes escenarios:

- 3 Diferentes funciones de énfasis
- Versión con inyección y sin inyección para comparar comportamientos y prestaciones.

En este report se analizarán cuantitativa y cualitativamente las propiedades del algoritmo DSNBoost respecto a 5 parámetros:

- Número de Capas. nL
- Número de Neuronas Ocultas. Nh
- Número de Épocas Nepoch
- Valor de alpha α
- Valor de beta β

Primeramente se analizarán los resultados obtenidos en los 3 reports anteriores para cada base de datos por separado, describiendo el comportamiento del algoritmo para cada una de ellas en función de los parámetros utilizados, viendo principalmente la evolución de la accuracy del error de test con nL, nH y alpha.

1.1 Funciones de énfasis utilizadas

Se han llevado a la práctica 3 funciones de énfasis fundamentalmente. Otras funciones también han sido probadas pero con pocas prestaciones por lo que se han abandonado. En todas las funciones de énfasis tenemos 2 componentes principales:

- **Componente uniforme**: Su objetivo es uniformizar el énfasis de las muestras limitando los valores máximos y mínimos que pueden tener. El parámetro α se encarga de esta componente. Este parámetro podría hacer por ejemplo que no nos fijemos demasiado en outliers ya que baja su énfasis a la par que sube los de las muestras muy bien clasificadas.

- **Componente variable:** Es la parte de la función de énfasis que depende de la salida del sistema. Las muestras bien clasificadas tienen una componente variable baja y las mal clasificadas alta. Esta parte suele tener también 2 componentes separadas que se combinan de forma convexa mediante el parámetro β.

1.1.1 NeoDSNBoost1

Esta es la función original que se ha estado utilizando.

$$p(\bar{x}_i) = \alpha + (1-\alpha)\left[\beta\frac{\left(O_{nor_i}(l) - T_i\right)^2}{4} + (1-\beta)\left(1 - O_{nor_i}^2(l)\right)\right]$$

Donde α, β son parámetros de validar y O_{norm_i} es la salida total del sistema normalizada para la muestra $\bar{x}_i$

$$O_{nor_i} = \frac{O_i}{\max\left(abs(O)\right)}$$

T_i es el valor deseado a la salida para la muestra $\bar{x}_i$. $T_i \in [-1,1]$

Esta función tiene el inconveniente de que el valor relativo de la componente uniformizadora "alpha" respecto al resto de la ecuación, depende de los bueno que sea el resultado total $O_{nor_i}(l)$.

- Si $O_{nor_i}(l)$ mejora, es decir $O_{nor_i} \rightarrow T_i$, entonces la componente no uniforme baja

Además no sabemos a priori cuán cerca de 1 estará.

1.1.1 NeoDSNBoost2

Esta función es idéntica a la anterior pero se asegura que el porcentaje de uniformización de la componente uniforme sea siempre la misma:

$$p(\bar{x}_i) = \frac{\alpha}{N} + (1-\alpha)\frac{1}{Z_l}\left[\beta\frac{\left(O_{nor_i}(l) - T_i\right)^2}{4} + (1-\beta)\left(1 - O^2_{nor_i}(l)\right)\right]$$

Siendo:

- N: El número de muestras de entrenamiento.
- Z_l: Constante de normalización de la componente no uniforme.

$$Z_l = \sum_{i=1}^{N} \beta\frac{\left(O_{nor_i}(l) - T_i\right)^2}{4} + (1-\beta)\left(1 - O^2_{nor_i}(l)\right)$$

De esta forma, α uniformiza el énfasis

1.1.2 NeoRAB-we

Esta función de énfasis tiene la siguiente forma:

$$D_l(\bar{x}_i) = \frac{\alpha}{N} + (1-\alpha)\underbrace{\frac{1}{Z_l}\left[D_{l-1}(\bar{x}_i)\, e^{-\left(\beta(O_l(\bar{x}_i)-T_i)^2 - (1-\beta)O_l^2(\bar{x}_i)\right)\gamma_l}\right]}_{RAB-we}$$

Siendo:

$$Z_l = \sum_{i=1}^{N} D_{l-1}(\bar{x}_i)\, e^{-\left(\beta(O_l(\bar{x}_i)-T_i)^2 - (1-\beta)O_l^2(\bar{x}_i)\right)\gamma_l}$$

Como se puede observar es una versión del RAB-we a la que se le ha añadido una componente uniforme.

Esta versión uniformizada de RAB-we funciona bastante bien, otras que se han probado no han funcionado como por ejemplo:

$$D_l(\bar{x}_i) = \frac{\alpha}{N} + (1-\alpha)\underbrace{\frac{1}{Z_l}\left[D_{l-1}(\bar{x}_i)\, e^{-\left(\beta(O_l(\bar{x}_i)-T_i)^2 - (1-\beta)O_l^2(\bar{x}_i)\right)\gamma_l}\right]}_{RAB-we}$$

1.2 Algoritmo de Aprendizaje de cada capa

El algoritmo de aprendizaje **para cada capa** es backpropagation. Los parámetros utilizados son:

- Inicialización de pesos: Variable aleatoria uniforme entre -ro y ro donde:

$$ro = \frac{\sqrt{N_{init}}}{\sqrt{Nin + Nout}}$$

  Se han realizado experimentos para $N_{init} = 1$
- Constante de aprendizaje Inicial: La constante de aprendizaje inicial es diferente para la capa oculta y para la capa de salida. La capa oculta tiene una constante inicial de $\eta_H = 0.01$ mientras que la de la salida tiene una constante R_{OH} veces menor $\boldsymbol{\eta_O = \frac{\eta_H}{R_{OH}}}$.
  Se han realizado experimentos para $R_{OH} = 1$

- Criterio de parada: Se ha utilizado un criterio de parada que no requiera utilizar error de validación (como se pidió). El algoritmo tiene un **número de épocas fijo** y se han realizado experimentos para diferentes números de épocas $Nepoch = 25, 50, 100, 150, 200$
  La constante de aprendizaje **decae linealmente a 0** durante el número de épocas.

- La función de coste a minimizar es el error cuadrático medio, MSE.
- La función de activación de las **neuronas ocultas** es tanh.
- La función de activación de la **neurona de salida** es tanh.

Dado que el entrenamiento es el mismo independientemente de nH vamos a hacer la suposición de que:

- **Mayor capacidad expresiva → Overfitting.**
- **A mayor nH: Menos se entrenan las neuronas → Underfitting**

Estas dos componentes lucharán y darán lugar a un valor de nH intermedio.

Podríamos también suponer que, por lo menos para AODSN:

- El sistema mejorará con nL mientras mejore O y no se produzca overfitting.
- Si el resultado de O empieza a decaer debido al underfitting, se acabó lo que se daba.

1.3 Parámetros de los experimentos

Para el cálculo del **error omnisciente** se han llevado a cabo **20 realizaciones** para cada conjunto de parámetros.

En las gráficas se visualiza la **accuracy** del algoritmo, es decir, el porcentaje de acierto.

Siempre que hablemos de error de entrenamiento o de test nos estamos refiriendo a los errores habiendo entrenado con todo el train y después mirando el error de test.

2. WAVEFORM

A continuación se describirán las propiedades de la BBDD Waveform

Datasets	Notations	#Train; C_1/C_{-1}	#Test; C_1/C_{-1}	Dimension (D)
Waveform	wav	400; 124/276	4,600; 1,523/3,077	21

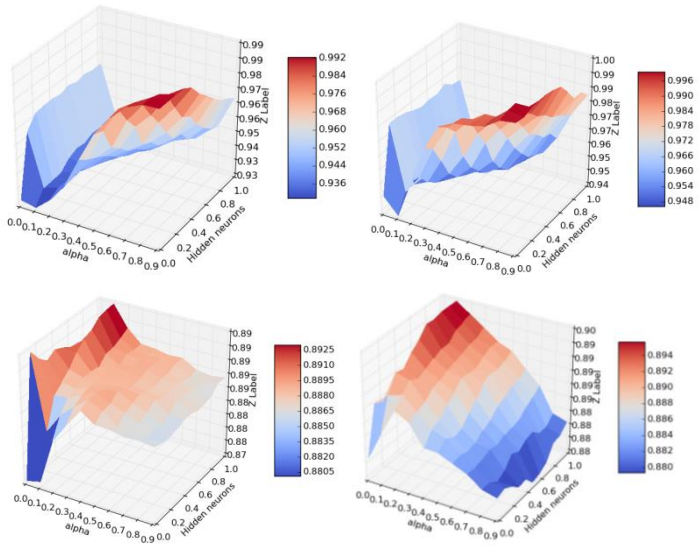
El valor inicial de gama = 1.4, AODSN evolución ascendente.

2.1 NeoRAB-we para DSN

Las siguiente graficas muestran la accuracy de Train y Test para $Nepoch = 100, Nh = 4$ en función de alpha y beta barridos para:

- Alpha: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
- Beta: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

(beta está etiquetado como "Hidden neurons", hay que modificar el código)



Observaciones:

- **Las curvas de tr y tst evolucionan de forma opuesta** con alpha y beta. Cuando la accuracy de train sube, la de test baja, lo cual quiere decir que hay overfitting para valores de tr muy altos.

- **Hay una discontinuidad para alpha = 0**. Esto ha venido pasando siempre, para alpha = 0, el error de tr mejora y el de tst empeora. Este caso equivale a RAB-We.

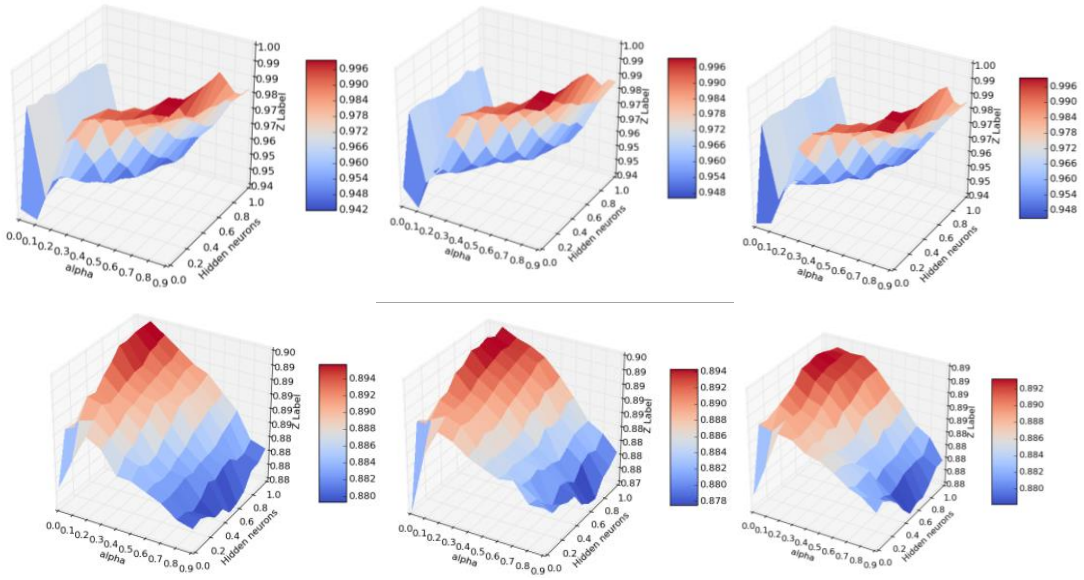
- Esta BBDD requiere un alpha pequeño, alpha = 0.1 o menor, debería explorarse alpha = 0.05 debido a la discontinuidad.

- Las curvas son parecidas conforme aumenta el número de épocas. **Al aumentar Nepoch**:
  - La accuracy para alpha grande se desploma debido al overfitting.
  - Para alphas bajos, la accuracy óptima sube.

- Esta BBDD requiere un beta alto.

2.2 Comportamiento respecto a nH NeoRAB-we

Cabe destacar que la forma de la curva respecto a nH se mantiene, si bien hay ligeros cambios. Las siguientes gráficas muestran la accuracy de train y tst para 4,8,12 neuronas con 100 épocas.

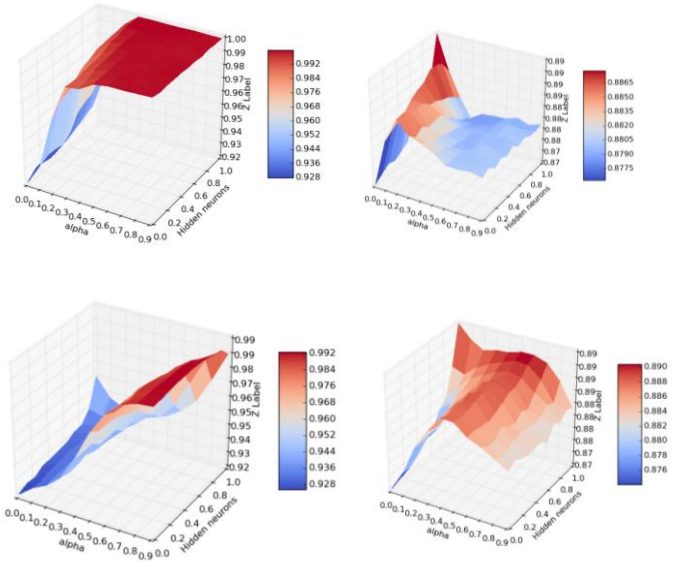


Observaciones:

- Las accuracies varían un poco de unas a otras pero no significativamente.
- Conforme subimos el número de nH la accuracy baja.
- Parece que el alpha óptimo crece con nH, esto puede ser debido a que al haber menos neuronas, se entrena menos la red (menos overfitting)

2.3 NeoDSNBoost1 y NeoDSNBoost2 (α, β)

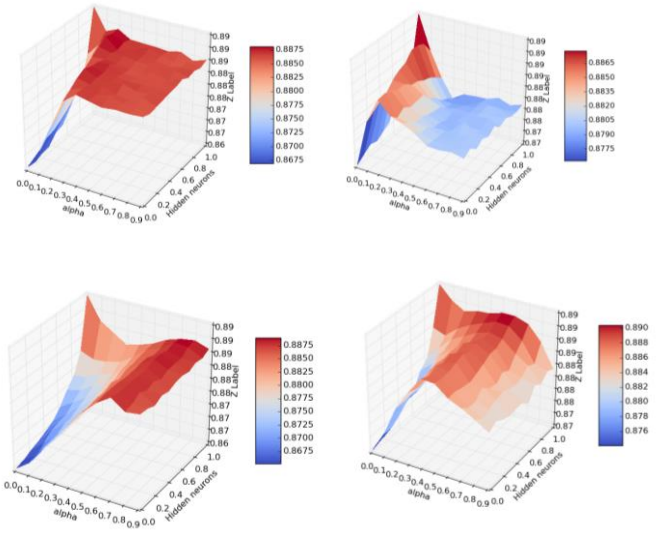
En este apartado compararemos las propiedades gráficas de NeoDSNBoost1 y NeoDSNBoost2 para $Nepoch = 100, Nh = 4$ y diferentes valores alpha y beta. Las primeras 2 gráficas son train y test para NeoDSNBoost1 y las siguientes 2 son las de NeoDSNBoost2.



Observaciones:

- NeoDSNBoost1 presenta mucho más overfitting que NeoDSNBoost2
- En este caso, el punto óptimo es alpha = 0, por lo que ambos esquemas tiene la misma accuracy opptima, sin embargo NeoDSNBoost2 alcanza mejores valores de accuracy a lo largo del resto de valores y podría obtener mejores valores que alpha = 0 si variamos Nepoch o nH.
- El efecto de alpha está mejor distribuído.

Poner aquí el error de tst para 50, 100 y 150 epochs



Observaciones:

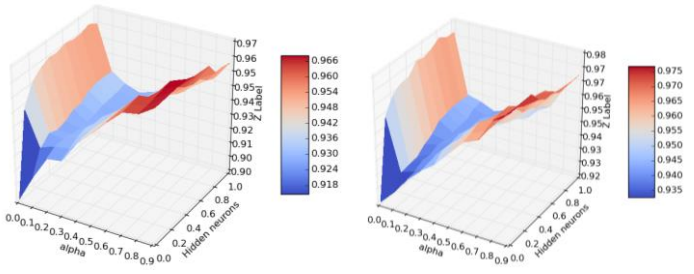
- NeoDSNBoost1 presenta mucho más overfitting que NeoDSNBoost2, saturándose para valores pequeños de alpha.
- El valor de alpha óptimo de NeoDSNBoost2 baja conforme aumentamos el número de nueronas y también aumenta la accuracy óptima.

2.4 Inyection VS NoInyection NeoRAB-we (α, β)

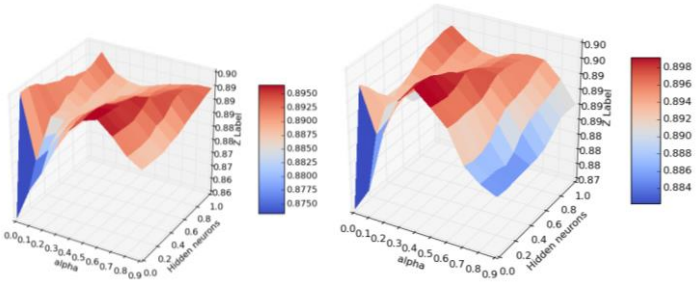
El objetivo de esta sección es presentar los resultados de la versión sin inyección, utilizando únicamente la versión modificada de la función de énfasis NeoRAB-we y compararlos con el caso de inyección DSN para ver cómo afecta la inyección al mismo.

Las siguiente graficas muestran la accuracy de Train y Test para $Nepoch = 100, Nh = 4$ en función de alpha y beta barridos para:

- Alpha: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
- Beta: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]



Train



Test

Observaciones:

- Las curvas de train de NoInyection y DSN **se parecen** en forma.
    - NoInyection alcanza menores valores de accuracy de tr (menos overfitting)

- Las curvas de test de NoInyection y DSN son diferentes.
    - Podría ser que quitando el overfitting extra que otorga la inyección, las curvas de train y test se parezcan más y no sean tan antagónicas.
    - A lo mejor en situaciones de menos overfitting la inyección ayuda y obtiene mejores valores. Pero alomejor el uso de alpha mediano o pequeño sólo sirve para casos en los que haya overfitting. ¿Podemos obtener mejores resultados partiendo del overfitting y aliviándolo con alpha pequeño que no haciendo overfitting y esperar que la evolución de correcta

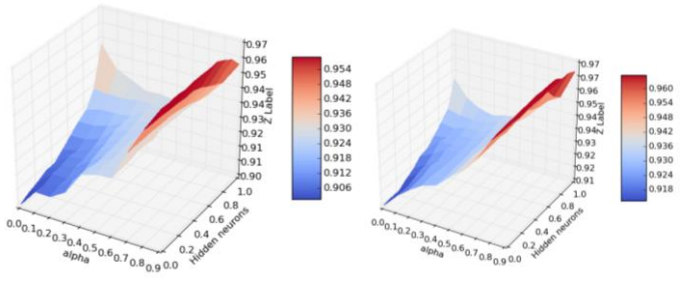
- NoInyection da mejores resultados que DSN, alomejor la inyección provoca **saturación de entrenamiento**, no overfitting. **No Inyection prefiere alphas mayores.**
- NoInyection puede soportar mayores Nepoch y mejorar su accuracy de tst. DSN empieza a empeorar si subimos Nepoch.
- Al aumentar Nepoch el alpha óptimo disminuye.
- La std de NoInyect es 4 veces menor que la de DSN.
- Destacar que **RAB-we (a = 0) funciona mucho mejor sin inyección** que con inyección en test.

2.5 Inyection VS NoInyection NeoBoosting2 (α, β)

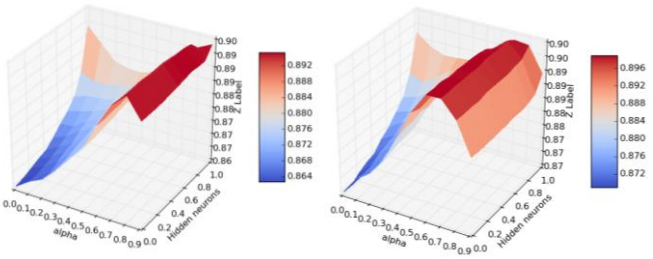
El objetivo de esta sección es presentar los resultados de la versión sin inyección, utilizando únicamente la versión modificada de la función de énfasis NeoBoosting2 y compararlos con el caso de inyección DSN para ver cómo afecta la inyección al mismo.

Las siguiente graficas muestran la accuracy de Train y Test para $Nepoch = 100, Nh = 4$ en función de alpha y beta barridos para:

- Alpha: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
- Beta: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]



Train



Test

Observaciones:

- Las curvas de train de NoInyection y DSN **se parecen** en forma.
  - NoInyection alcanza menores valores de accuracy de tr (menos overfitting)

- Las curvas de test de NoInyection y DSN son diferentes.
  - Podría ser que quitando el overfitting extra que otorga la inyección, las curvas de train y test se parezcan más y no sean tan antagónicas.
  - A lo mejor en situaciones de menos overfitting la inyección ayuda y obtiene mejores valores. Pero alomejor el uso de alpha mediano o pequeño sólo sirve para casos en los que haya overfitting. ¿Podemos obtener mejores resultados partiendo del overfitting y aliviándolo con alpha pequeño que no haciendo overfitting y esperar que la evolución de correcta

- NoInyection da mejores resultados que DSN, alomejor la inyección provoca **saturación de entrenamiento**, no overfitting. **No Inyection prefiere alphas mayores.**

- NoInyection puede soportar mayores Nepoch y mejorar su accuracy de tst.

- La std de NoInyect es 4 veces menor que la de DSN.
- Al aumentar Nepoch el alpha óptimo disminuye.

3. CONCLUSIONES DSNBOOST

Vamos a intentar sacar conclusiones del algoritmo y posibles variaciones del mismo para mejorar en base a estas 4 bases de datos mostradas para $beta = 0$. Se deja para el siguiente report el análisis conjunto de los estos primeros 3 reports para intentar sacar conclusiones utilizando **lógica inductiva** y a partir de esta, decidir qué experimentos realizar a continuación.

3.1 Conclusiones anteriores

Las **conclusiones de este report son muy parecidas a las del anterior**, sólo hemos variado un poco la forma de entrenamiento y establecido $beta = 1$ en vez de $beta = 0$. Resaltamos las algunas de ellas:

- **Valores pequeños de alpha parecen luchar contra el overfitting** produciendo underfitting conforme apilamos capas de una forma tal que el sistema generalice mejor. Mientras que valores grandes suelen mejorar la solución de train (si no hay overtitting también se mejora la de test y todos contentos)

- Las propiedades del sistema para **beta = 0 y beta = 1 son muy similares**. Para bases de datos cuyo alpha óptimo sea bajo merece la pena estudiar este parámetro (Waveform) sin embargo puede que no merezca la pena para otras (Abalone).

- Para las combinaciones óptimas de parámetros (alpha, Nepoch, nH) **el número de capas nL siempre mejora** o simplemente estabiliza el error. El mejor conjunto de parámetros suele ser el que mejor evoluciona con nL (no pasa que un conjunto sea el mejor porque directamente empezó siendo el mejor en las primeras capas).

- El número de **épocas Nepoch** juega un papel importante, el entrenamiento importa. El número óptimo de épocas depende grandemente de alpha (Wav $\alpha = 0$ y 0.3). Ambos parámetros tienen que ver con el overfitting.

- El número de neuronas óptimo nH depende del problema y del resto de parámetros.

- Las constantes de agregación **gamma** suelen tener una **evolución inicial** corta para luego saturarse en un valor fijo que depende del conjunto de parámetros (alpha, Nepoch, nH). Habría que estudiar **otras formas de agregación**.

- Para los problemas de **kwok y ima** (sobretodo) la evolución del algoritmo es muy pobre en todos los casos (apenas mejora con nL). Hay que seriamente **utilizar otra forma de entrenamiento o explorar más parámetros**.

- Hay que **disminuir los rangos de parámetros que barremos** y hacer un estudio del efecto de los parámetros en los problemas en función de las propiedades de los mismos (se hará en el siguiente report).

3.2 Conclusiones parciales

Las siguientes conclusiones son conclusiones parciales para el futuro:

- Cuando **no hay overfitting** en los aprendices se tiende a **alphas altos**. Las curvas de train y test evolucionan de forma similar para (alpha, beta, gamma). El sistema con inyección funciona mejor.

- Cuando **hay overfitting** en los aprendices se tiene a **alphas bajos**. Las curvas de train y test evolucionan de forma opuesta para (alpha, beta, gamma). El sistema sin inyección funciona mejor.

- Los **mejores errores se obtienen cuando hay algo de overfitting** que es solucionado por el alpha. La inyección es buena para el caso de no overfitting pero para el caso de overfitting parece disminuir las prestaciones haciendo que el sistema sature demasiado pronto.

- La curva del error respecto (alpha, beta) no varía mucho respecto de nH. Simplemente mejora o empeora un poco. Podría ser interesante para reducir barrido de nH.

- Beta baja suele entrenar mejor train, beta alta peor por lo que para overfitting. HACER OBSERVACIONES SOBRE BETA. Alomejor cada problema tiene la suya óptima.

- Hay discontinuidades para el caso de alpha = 0. Probablemente porque hay muestras muy buenas cuyo D(x) debería ser muy cercano a 0. Barrer para alpha = 0.05. Se podría hacer homologas observaciones para el resto de extremos de alpha y beta por lo que sería conveniente barrer alpha = [0.05, 0.95] y beta = [0.05, 0.95] también.

- NoInyection tiene valores de alpha menores para Waveform

- El número óptimo de Nepoch para No Inyection es mayor, probablemente porque un número muy grande estancaría el apredizaje del DSN. A lo mejor se podría comprobar viendo si en los casos de muchas Epoch en DSNBoost, el valor se estanca rápido y luego no baja (si bajase sería probablemente por underfitting lo cual invalidaría la teoría)

- Teorías absurdas parciales de evolucoin de alpha y beta optimos con nh y Nepoch.
    - Cuantas mas epocas, mayor es a,b optimo -> No parece pasar en el Inyection
    - Cuantas mas nh mayor es a,b optimo -> No parece pasar en el NoInyection
    - Alpha peques, Beta grandes -> Para Inyection
    - Alpha-Beta mas igualados para NoInyection

- NeoDSNBoost2 parece ser mejor que el NeoDSNBoost1 ya que no intensifica el valor de alpha en funcion de lo bueno que es el sistema. Tiene una evolución uniforme con alpha lo que redúcela búsqueda por ser más apropiada y no tener después que barrer más valores.
    - Si el sistema va mejorando, alpha es relativamente mayor -> Si hay overfitting la has cagado chaval.

  - Igual pero al reves para cuando el sistema empeora, alpha decrece -> Si hay underfitting peor.

- Beta se podría samplear sólo para 0,5,10 ya que el óptimo suele estar entre ellos.

- DSNBoost2 y DSNBoost tienen más o menos la misma forma pero DSNBoost2 es más uniforme respecto a alpha

- Si entrenamos poco, entonces alpha = 0 será relativamente bueno, conforme hacemos overfitting, en validación se va a la puta.

- Discusión de si No inyect debería tener más épocas o menos y más alpha o menos:
  - Como no tiene inyección, ante poco entrenamiento, el alpha debería ser o bien 0, para el RAB o bien alpha alto para que alphas intermedios no causen underfitting y al no tener inyección requeriría un alpha más grande. Un alpha tan grande en DSN alomejor saturaría.
  - Al no tener inyección se puede permitir alphas más grandes sin overfitting por lo que podemos subir Nepoch, bajando el alpha. Si el alpha fuera alto, entonces sería únicamente unir mismos learners lo que equivale a reducir el overfitting por normalización.
  - Si aumentamos epocas, alpha optimos disminuye en toeria. A hep le pasa algo raro ya que alomejor es demasiado overfitting.