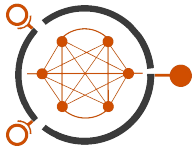
Inteligencia Artificial

Redes Neuronales Feedforward - Tips & Tricks

Martin Marchetta
martin.marchetta@ingenieria.uncuyo.edu.ar

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



LABSIN

Resumen

- Inicialización de pesos y biases
- Procesamiento de las entradas
- Prevención de overfitting
- Funciones de activación
- Loss functions para clasificación
- Optimización de hiperparámetros

Inicialización de pesos y biases

- En general: números aleatorios pequeños
  - Ejemplos:
    - Números aleatorios tomados de una distribución Gaussiana con $\mu = 0$ y $\sigma = 1$

```
W = 0.01 * np.random.randn()
```

    - El mismo caso, tomando números de una distribución uniforme

- En algunos casos la función de entrada de las neuronas inicializadas aleatoriamente tienen varianza que crece con la cantidad de entradas
  - Se puede mejorar esto escalando los pesos con 1/sqrt(n), donde n son las entradas:

```
W = np.random.randn() / sqrt(n)
```

- Biases: pueden inicializarse en 0 o una constante pequeña (ej: 0.01)

Procesamiento de datos de entrada

- Las 3 formas más comunes de procesar los datos son
  - Sustracción de la media (centrado)
  - Normalización
  - Principal Components Analysis (PCA) / Whitening

- Sustracción de la media
  - Consiste en calcular la media de cada feature (cada entrada/columna) a lo largo de todo el set de entrenamiento y luego restarlo a todos los valores de la columna
  - Esto equivale a centrar los datos alrededor de cada “eje” o “dimensión” de los datos

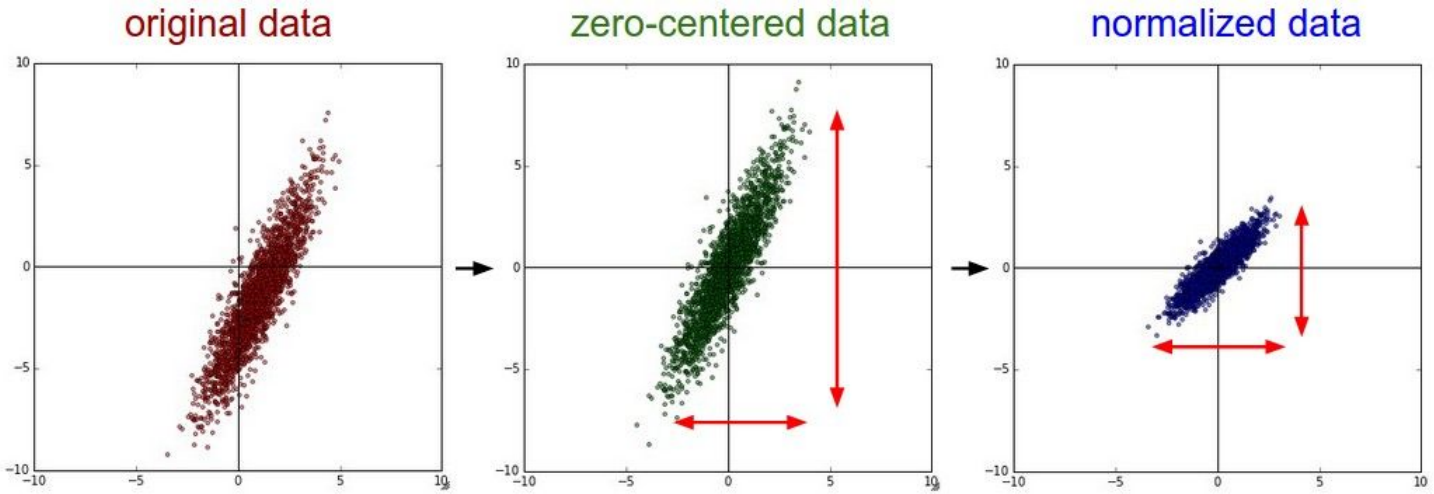
Procesamiento de datos de entrada

- Normalización
  - Consiste en normalizar las dimensiones (features/columnas) para que tengan escalas similares
  - Tiene sentido cuando las features tienen escalas distintas
  - Se puede hacer de 2 maneras
    - Dividir todos los valores de cada dimensión por la desviación estándar de la dimensión, una vez que los datos ya fueron centrados
    - Normalizar cada dimensión de manera que sus valores queden en el intervalo [-1, 1]

- En cualquier caso, la normalización se aplica a los datos de test/validación/ejecución con los mismos parámetros que los de training (ej: la media solo se calcula sobre los datos de training, y luego ésta misma se aplica a los datos de test/validación/ejecución)

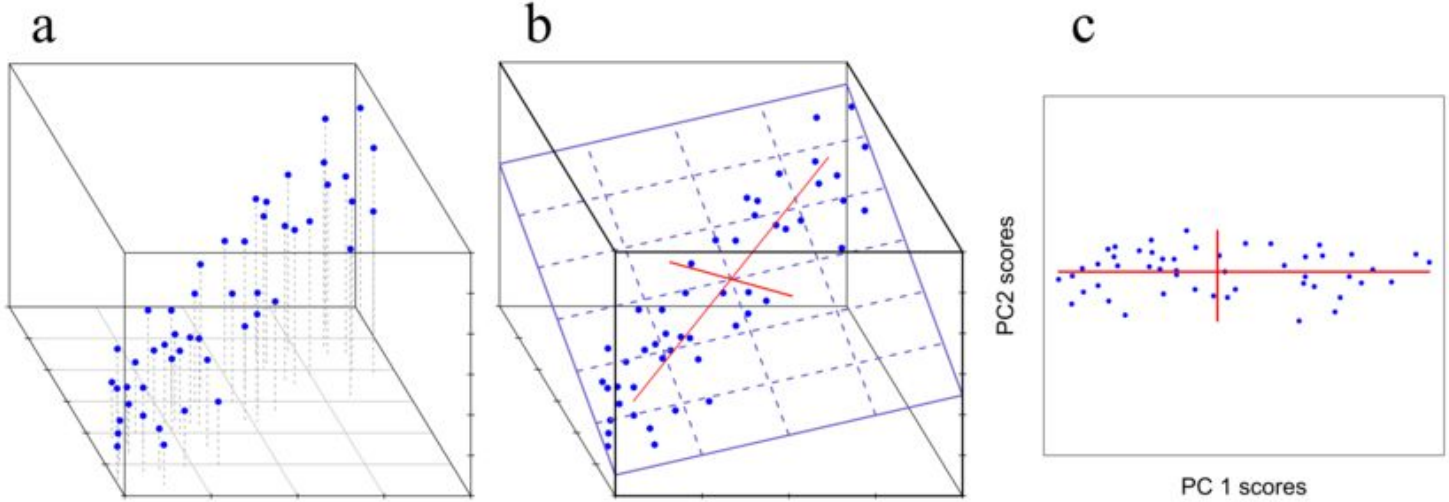
Procesamiento de datos de entrada

- Centrado y Normalización:



Procesamiento de datos de entrada

- PCA (Principal Components Analysis)
  - Es una transformación de los datos
  - Permite “ordenar” los atributos (dimensiones) en orden de mayor varianza a menor varianza

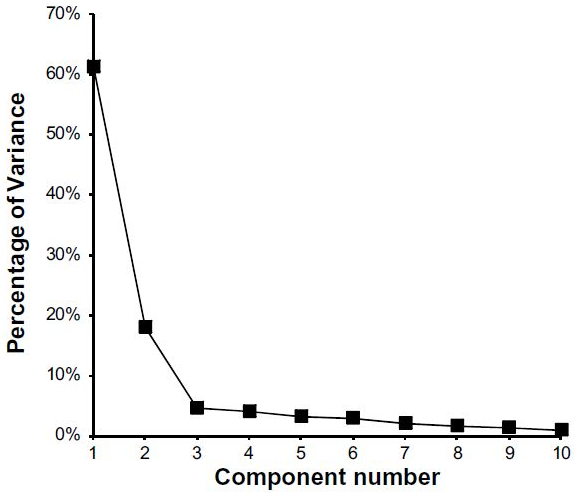


Procesamiento de datos de entrada

- PCA (Principal Components Analysis)
  - Luego de ordenarlos se pueden “filtrar” los atributos que aportan menos a la varianza de los datos

Axis	Variance	Cumulative
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100.0%

(A)



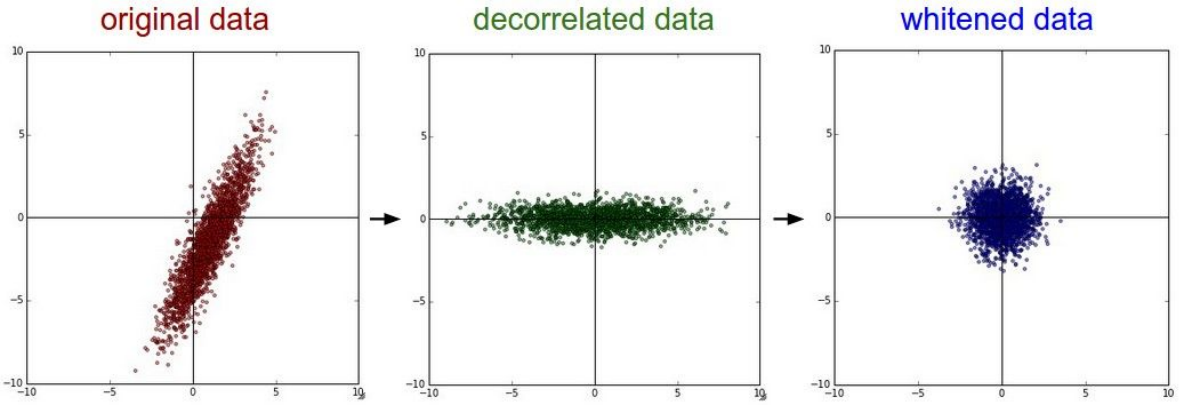
(B)

Procesamiento de datos de entrada

- PCA (Principal Components Analysis)
  - Cálculo
    - Aplicar centrado en la media (como se vio anteriormente)
    - Calcular matriz de covarianza
    - Diagonalizar: descomposición SVD (Singular Value Decomposition)
    - Las columnas de la matriz U resultante de SVD
      - Son una base ortonormal
      - Contienen los autovectores, ordenados de mayor a menor de acuerdo a sus autovalores
    - Proyectar los datos en U (producto matricial X . U, donde X son las entradas)
      - Si se quiere reducir la dimensionalidad, se pueden utilizar menos columnas de U antes de hacer el producto

Procesamiento de datos de entrada

- Whitening
  - Similar a PCA, pero divide las columnas de la matriz U (autovectores) por sus respectivos autovalores para escalar
  - Potencial problema: exagera el ruido



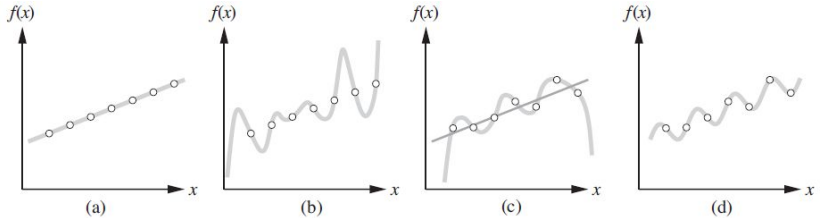
Prevención de overfitting

- Hay varias formas, entre las que se destacan
  - Cross-validation
  - Regularization
    - L2
    - Dropout

- Cross-validation: Evaluar el rendimiento cada cierta cantidad de ejemplos, verificando si el mismo sigue teniendo tendencia a la mejora

Prevención de overfitting

- Regularization
  - Dependiendo de la función de pérdida, pueden existir distintas configuraciones de pesos W que explican los datos de training y test



  - Según la *Navaja de Ockham*, buscamos la configuración más simple que explica los datos

Prevención de overfitting

- Regularization
  - Existen varias técnicas de regularización, entre las que se destacan recientemente: L2 y Dropout

- Regularización L2
  - Se incluye en la Función de Error (a.k.a. Función de Pérdida, Loss Function), un término llamado Penalidad de Regularización, que penalice los pesos grandes, de manera que el algoritmo BP tienda a generar pesos pequeños

Pérdida de datos | Pérdida de regularización

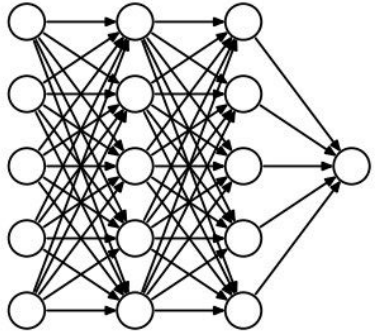
$$L = \frac{1}{N}\sum_{i} L_i + \frac{1}{2}\lambda \sum_{k}\sum_{l} W_{k,l}^2$$

λ: fuerza de regularización
L: Pérdida total
Li: Pérdida de 1 neurona de salida para 1 ejemplo
l: Subíndice de Capa
k: Subíndice de Neurona

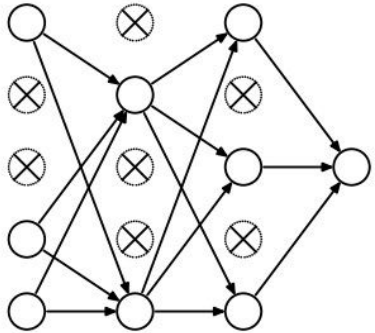
Prevención de overfitting

- Droput Regularization
  - Cómo funciona?
    - Consiste en “desactivar” neuronas aleatoriamente durante el entrenamiento
    - Una neurona se mantiene activa con una probabilidad *p*
    - En ejecución, todas las neuronas se mantienen activas pero sus salidas se escalan multiplicándolas por *p*
  - Se puede combinar con L2 u otros mecanismos de regularización

Sin Dropout



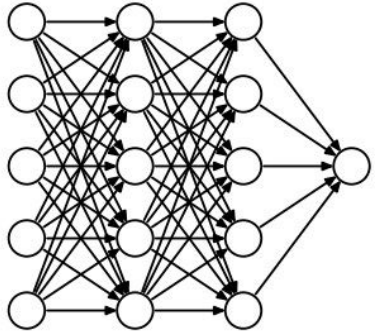
Con Dropout



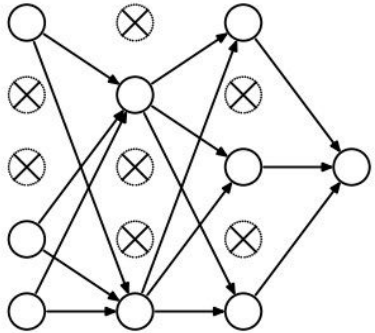
Prevención de overfitting

- Droput Regularization
  - La idea es forzar a las neuronas a “valerse por sí mismas” (reducir la dependencia de otras neuronas)
  - Conceptualmente es similar a entrenar varias redes neuronales distintas, que son subsets de la red completa, pero en un mismo proceso de entrenamiento.
    - Según esta concepción, al ejecutar la red entrenada se estaría obteniendo un resultado similar a un ensemble

Sin Dropout

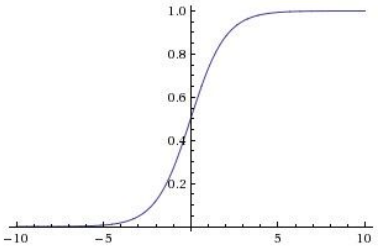


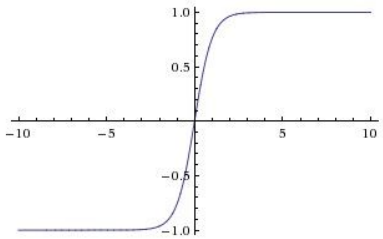
Con Dropout



Funciones de activación

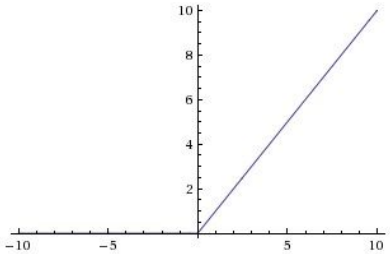
- Funciones “tradicionales” tienen el problema de generar gradientes pequeños: se dice que se saturan y anulan/matan el gradiente





- Hay varias alternativas. Actualmente el estado del arte recomienda el uso de funciones ReLU (Rectified Linear Unit)

$$f(x) = max\ (0,\ x)$$



Clasificación: función de error

- Para clasificación
  - Se suele utilizar una neurona de salida por cada clase
  - Se desea que la neurona cuya clase es la correcta tenga el mayor valor de salida
    - Se interpreta la salida de cada neurona como un puntaje o score
  - Las 2 loss functions más usadas para clasificación son
    - SVM Loss
    - Softmax Loss
  - Clasificador Softmax (a.k.a. Cross-entropy loss)

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

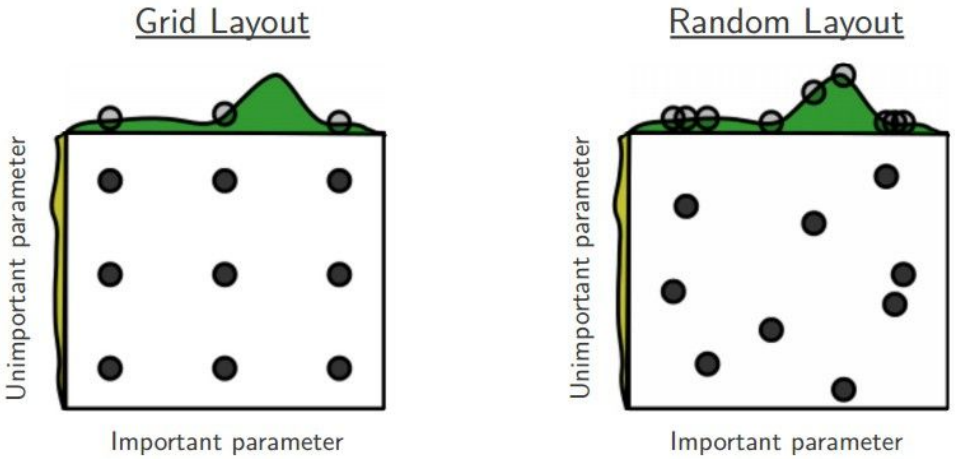
L_i: Pérdida de 1 neurona de salida para 1 ejemplo
f_j: Score de salida de la neurona j
f_{yi}: Score de salida de la neurona cuya clase es la correcta

Optimización de hiperparámetros

- Requiere una porción del set de entrenamiento independiente
  - Se divide el data set en 3 partes: training, validation y test
    - Training se utiliza para entrenar
    - Validation se utiliza para verificar la performance de los valores de los hiperparámetros
    - Una vez seleccionados los valores “óptimos” de los hiperparámetros, se prueba el rendimiento con el set de test
  - En ocasiones se utiliza cross-validation para evaluar los valores de los hiperparámetros (se calculan rendimientos promedio sobre distintos conjuntos de training y validation para cada selección de parámetros, para garantizar que los resultados son confiables)

Optimización de hiperparámetros

- Búsqueda random vs. Búsqueda en grilla
  - A veces conviene probar valores aleatorios de los parámetros, dentro de cierto rango, en lugar de probar sistemáticamente valores espaciados de manera uniforme:

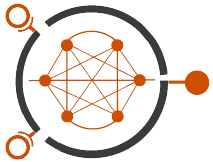


Optimización de hiperparámetros

- Se debe tener cuidado con valores óptimos para los parámetros que estén en los “bordes” de los intervalos definidos
- Hacer tests con distintos niveles de granularidad: probar intervalos más grandes, y luego focalizarse en intervalos más pequeños donde los mejores valores hayan sido encontrados
- Optimización Bayesiana de parámetros: área de investigación activa enfocada en obtener algoritmos para navegar mejor el espacio de hiperparámetros (aún se obtienen mejores resultados con las recomendaciones anteriores)

Preguntas?
Opiniones?

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



LABSIN