Reconocimiento de Patrones

Práctico 3

Entrega: jueves 6 de noviembre de 2018

Redes Neuronales Multicapa

Ejercicio 1

En este ejercicio se utiliza la red neuronal TensorFlow y en particular el framework Playground (https://playground.tensorflow.org/) para diseñar y entrenar redes neuronales. Se entrenarán algunas redes neuronals sencillas, a partir de datos simples, con el fin de familiarizarse con sus conceptos básicos.

1. Selección de atributos.

   a) Seleccionar el conjunto de atributos adecaudo para obtener la red neuronal más simple (con la menor cantidad de neuronas) de forma de obtener un error en el conjunto de test menor a 0.1 (obtener el modelo para cada conjunto disponible, Circle, Gaussian, Xor y Spiral). Utilizar noise=0 y percTrainData=50

2. En esta parte veremos como las redes neuronales nos brindan una forma de aprender modelos no lineales sin usar combinaciones de atributos explícitos: Utilizar el siguiente modelo: NN1 y responder:

   a) El modelo especificado combina dos atributos de entrada en una sola neurona. ¿Este modelo permite aprender una no linealidad? Ejecutar el modelo y justificar a partir de los resultados obtenidos.

   b) Incrementar el número de neuronas en la capa oculta de 1 a 2, probar cambiar de una activación lineal a una activación no lineal, como ReLU. ¿Es posible crear un modelo que aprenda no linealidades? Justificar

   c) Variar el número de capas ocultas y neuronas por capa de forma de obtener una clasificación aceptable (Se puede variar también la tasa de aprendizaje, la regularización y otras opciones de configuración del aprendizaje, pero no los atributos utilizados). ¿Cuál es la menor cantidad de nodos y capas que deben utilizarse para obtener un error en entrenamiento menor a 0.18? Detallar el modelo utilizado y verificar su correcto aprendizaje para varias inicializaciones de los parámetros de la red.

3. Inicialización de redes neuronales. Utilizar el modelo NN2

   a) Ejecutar el modelo especificado cuatro o cinco veces. Antes de cada prueba, presionar el botón Reiniciar la red para obtener una nueva inicialización aleatoria de los parámetros. ¿A qué forma converge el resultado de cada modelo? ¿Qué se puede inferir sobre el rol de la inicialización en la optimización?

   b) Agregar una capa y un par de nodos adicionales. Repitir las pruebas del punto anterior ¿Esta acción hace que los resultados sean más estables? Justificar

Ejercicio 2

Se considera la red neuronal de dos capas que se muestra en la figura 1.

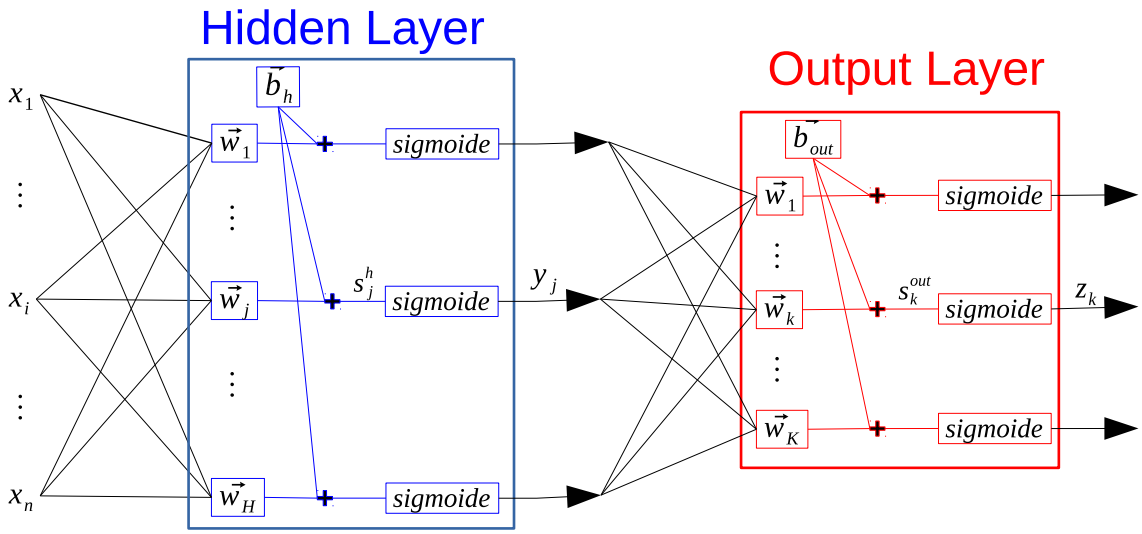

Figura 1: Red neuronal de dos capas

Para entrenar la red se utilizará como función de costo la entropía cruzada.

$$L = \sum_{k=1}^{K} -t_k \log(z_k) - (1 - t_k) \log(1 - z_k)$$

siendo $\mathbf{z} = (z_1, ..., z_K)$ las salidas de la red y $\mathbf{t} = (t_1, ..., t_K)$ el vector de etiquetas (codificación one-hot).

1. Mostrar que $\frac{\partial L}{\partial s_k^{out}} = z_k - t_k$
2. En el archivo `neuralNet.py` se implementa parcialmente una red neuronal de dos capas. Completar el código faltante en el método `fit()`. Para ello se deberán calcular los siguientes gradientes:
   - $\frac{\partial L}{\partial \mathbf{W_{out}}}$ y $\frac{\partial L}{\partial \mathbf{b_{out}}}$ de capa de salida
   - $\frac{\partial L}{\partial \mathbf{W_h}}$ y $\frac{\partial L}{\partial \mathbf{b_h}}$ de capa oculta
3. El archivo `ejemploDuda.py` genera los datos del problema no separable de la figura 6.8 del libro "Pattern Classification" Duda, Hart, Stork. Mostrar que con tres nodos en capa oculta es posible separar correctamente todos los patrones de entrenamiento.
4. Cambie el número de nodos en capa oculta a 2 y compare los resultados.
5. El archivo `ejemploMnist.py` carga los digitos de 0 a 9 y los muestra. Entrene una red que obtenga un accuracy de al menos el 90 % en el conjunto de validación. Comente sobre los errores cometidos.

Máquinas de Vectores de Soporte

Ejercicio 3

El algoritmo `C-SVM` da lugar al siguiente problema de optimización:

$$\begin{cases} \min\limits_{\boldsymbol{w},b,\boldsymbol{\xi}} & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum\limits_i \xi_i \\ \text{sujeto a} & \\ & y_i.(<\boldsymbol{w}, \Phi(x_i)> + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{cases} \tag{1}$$

1. Escribir el Lagrangiano del problema. Mostrar que el valor del Lagrangiano está acotado superiormente por la función de costo original.

2. Plantear el problema dual indicando explícitamente cuál es la función dual que se desea optimizar. La función dual se define como en la sección 5.1.2 del libro Convex Optimization disponible gratuitamente en `http://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`

3. Mostrar que el problema dual puede escribirse como:

$$\begin{cases} \min\limits_{\boldsymbol{\lambda}} & \frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{Q}\boldsymbol{\lambda} - \boldsymbol{e}^T\boldsymbol{\lambda} \\ \text{sujeto a} & \\ & 0 \leq \lambda_i \leq C \\ & \sum\limits_i \lambda_i y_i = 0 \end{cases}$$

   con $Q_{ij} = y_i y_j K(x_i, x_j)$

4. Mencione por qué puede resultar más conveniente resolver el problema dual.

Ejercicio 4

En este ejercicio se buscará ganar intuición en el uso de SVM.

1. Cargue el archivo `data_gaussian.train` disponible en la página del curso utilizando el comando y grafique con distinto color los patrones de ambas clases.

2. Entrene un clasificador SVM lineal utilizando un valor de $C = 1$. Para el modelo encontrado:

   a) Dibuje la superficie de decisión y las lineas de los márgenes superpuesta a los patrones de entrenamiento.

   b) Calcule el error de clasificación en los conjuntos de entrenamiento y test.

   c) Indique el número de vectores soporte utilizados

   d) Indique el valor del margen

3. Repita la parte anterior para otros valores de C y comente los resultados.

4. Comente que sucede si se entrena utilizando un clasificador SVM con un kernel de tipo RBF y parámetros $C = 1$ y $\gamma = 200$.

Aprendizaje no supervisado paramétrico

Ejercicio 5

El objetivo de este ejercicio es estudiar la convergencia del algoritmo EM. Más precisamente, se busca demostrar que si la verosimilitud del conjunto de parámetros con respecto a los datos buenos $l(\mathcal{D}_g;\boldsymbol{\theta}) = \log p(\mathcal{D}_g;\boldsymbol{\theta})$ no es un óptimo en $\boldsymbol{\theta}$, el algoritmo EM evoluciona maximizando esta cantidad. Demostrar los pasos siguientes:

1. Sea $\mathcal{D}_g$ y $\mathcal{D}_b$, respectivamente, el conjunto de características que son datos "buenos", y el conjunto de características "malas" o faltantes en los datos.
   - Observar que $l(\mathcal{D}_g;\boldsymbol{\theta}) = \log p(\mathcal{D}_g,\mathcal{D}_b;\boldsymbol{\theta}) - \log p((\mathcal{D}_b|\mathcal{D}_g;\boldsymbol{\theta})$.
   - Se define la función $\mathcal{E}^t(\cdot)$ como la esperanza con respecto a la distribución $p((\mathcal{D}_b|\mathcal{D}_g;\boldsymbol{\theta}^t)$, i.e. $\mathbb{E}_{\mathcal{D}_b}[\cdot\,|\mathcal{D}_g;\boldsymbol{\theta}^t]$ (t corresponde a la iteración en curso). Aplicar esta esperanza a $l(\mathcal{D}_g;\boldsymbol{\theta})$ y expresar la respuesta en términos de la función $Q(\boldsymbol{\theta};\boldsymbol{\theta}^t)$ del algoritmo EM.
2. Se define la función cociente de densidades
   $$\phi(\mathcal{D}_b) = \frac{p((\mathcal{D}_b|\mathcal{D}_g;\boldsymbol{\theta})}{p((\mathcal{D}_b|\mathcal{D}_g;\boldsymbol{\theta}^t)}.$$
   Demostrar que
   $$\mathcal{E}^t(\log\phi(\mathcal{D}_b)) \leq \mathcal{E}^t(\phi(\mathcal{D}_b)) - 1 = 0.$$
3. Usando el resultado de la parte anterior, demostrar que si $Q(\boldsymbol{\theta}^{t+1};\boldsymbol{\theta}^t) > Q(\boldsymbol{\theta}^t;\boldsymbol{\theta}^t)$, tal como se cumple en el **paso M** del algoritmo EM, entonces $l(\mathcal{D}_g;\boldsymbol{\theta}^{t+1}) > l(\mathcal{D}_g;\boldsymbol{\theta}^t)$.

Técnicas de agrupamiento

Ejercicio 6

En este ejercicio se verá k-means como técnica de agrupamiento de datos.

1. Implementar el algoritmo k-means y aplicarlo al conjunto de datos del archivo `data_4clusters`. La variable `X` contiene los patrones a agrupar. En este caso el número de clusters es 4.

```
def kmeans(X, k, seed=2):
    '''
    X - matriz de tamanho NxD que contiene los vectores de entrada
    k - numero de clusters a encontrar
    seed - semilla que se usa para inicializar los centros.
               elegir aleatoriamente k vectores de X
    '''

    # IMPLEMENTAR

    centros =    # centros de los clusters encontrados
    etiquetas =  # vector de largo N que contiene a que cluster
                 # fue asignada la muestra

    return centros, etiquetas
```

2. Verificar el correcto funcionamiento del algoritmo comparando los resultados obtenidos con los de la variable `y`.
3. Al variar el valor de la semilla que controla la inicialización ¿Siempre obtuvo resultados satisfactorios? En caso negativo, proponga un esquema para robustecer el método.
4. Utilice el algoritmo k-means para separar el conjunto de datos `data_4clusters_stretched` y comente los resultados.

Ejercicio 7

El objetivo de este ejercicio es implementar el esquema Expectation Maximization (EM) para encontrar los parámetros que maximizan la verosimilitud del modelo Mezcla de Gaussianas para un conjunto de datos X. Si se utilizan K componentes en la mezcla, el modelo está dado por:

$$p(\mathbf{x_n}|\mathbf{\Theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k}\right)$$

Se pide:

1. Implementar una función que utilice el esquema EM para encontrar los parámetros óptimos en el caso de Mezcla de Gaussianas. La función recibirá como parámetro los datos X, el número de gaussianas, una semilla que controle la inicialización de los $\mu_\mathbf{k}$ y condiciones de paradas (Ej: máximo de iteraciones). Dentro de la función el esquema sigue los siguientes pasos:

   a) Se inicializan las medias $\mu_\mathbf{k}$, las covarianzas $\mathbf{\Sigma_k}$ y los coeficientes de mezcla $\pi_\mathbf{k}$. Para facilitar la comparación con k-means se sugiere inicializar los $\mu_\mathbf{k}$ a K vectores de X elegidos aleatoriamente.

   b) **Expectation Step** Se calcula la probabilidad de que la *n-esima* muestra haya sido generada por la componente k de la mezcla. Para ello se utilizan los parámetros actuales.

   $$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}\left(\mathbf{x_n}|\mu_\mathbf{k}, \mathbf{\Sigma_k}\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x_n}|\mu_\mathbf{j}, \mathbf{\Sigma_j}\right)}$$

   c) **Maximization Step** Se encuentran los parámetros óptimos utilizando la distribución de $\gamma(z_{nk})$ actual

   $$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$
   $$\mu_\mathbf{k}^\mathbf{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x_n}$$
   $$\mathbf{\Sigma_k^{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x_n} - \mu_\mathbf{k}\right) \left(\mathbf{x_n} - \mu_\mathbf{k}\right)^T$$
   $$\pi_k^{new} = \frac{N_k}{N}$$

   d) Se evalúa la log-verosimilitud de los datos y se chequea que haya un avance respecto a la iteración anterior. Si el algoritmo no convergió, se vuelve al E-step

   La función deberá devolver los parámetros del modelo y además los valores de la log-verosimilitud a lo largo del entrenamiento.

2. Evaluar el algoritmo con los conjuntos de datos `data_4clusters` y `data_4clusters_stretched`. Para ambos casos:

   a) Indicar claramente los valores de los parámetros aprendidos y mostrar las curvas de log-verosimilitud en función de las iteraciones que dieron lugar a dichos parámetros.

   b) Graficar los patrones de entrenamiento dibujando con el mismo color a aquellos que asignaría al mismo cluster. Indique el criterio utilizado.

3. Compare los resultados obtenidos utilizando la mezcla de gaussianas con los obtenidos mediante k-means.