Proyecto 1: Parte A

Curso: Inteligencia Artificial

29 de abril de 2017

Instituto Tecnológico de Costa Rica
Sede Interuniversitaria de Alajuela
Prof. Fabián Fallas Moya

Introducción

En este proyecto usted implementará el algoritmo de backpropagation para redes neuronales y aplicarlo al reconocimiento de dígitos escritos a mano. En la tarea anterior ustedes implementaron feedforward para redes neuronales y hacer predicciones con los pesos que se les entregaron (thetas). En este proyecto ustedes implementarán el algoritmo de backpropagation para *aprender* los parámetros para la red neuronal.

Datos

En la primera parte de pp.m (nuestro template), el código se cargará y se desplegará en una gráfico de dos dimensiones, que es llamado por la función displayData. El data set será el mismo que se utilizó en la tarea anterior, hay 5000 imágenes en el data set data1.mat; donde cada fila (cada fila es de 400 datos) es una muestra de una imagen de 20x20 pixeles. Cada pixel está representado por un número flotante indicando la intensidad del pixel en escala de grises. Esto nos da una dimensión total de la matriz X de 5000x400, donde cada fila es una imagen desplegada en un vector de 400 puntos.

La segunda parte del training set es un vector de 5000 (vector y) datos que contiene las etiquetas de cada muestra. Para hacer las cosas más sencillas con Octave, que es un lenguaje indexado a partir del valor 1, se ha mapeado el dígito cero al valor de 10. Por lo tanto, el dígito 0 se mapea como 10, mientras que los dígitos del 1 al 9 son mapeados del 1 al 9, en su ordenamiento natural.

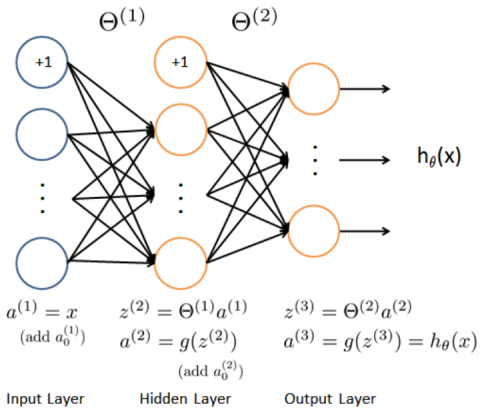

Figura 1: Arquitectura de la red neuronal

Modelo

Nuestra red neuronal como muestra la Figura 1. Tiene 3 capas: una capa input, una hidden y una output. Cabe resaltar que nuestra capa input es de 400 neuronas. También tendremos a disposición las variables Theta1 y Theta2,

los cuales tienen dimensiones con respecto a nuestra red neuronal (la hidden layer posee 25 unidades y el output layer posee 10 unidades), y que están contenidos en el archivo pesos.mat.

Feedforward y función de costo

Ahora se tendrá que programar la función de costo y los gradientes para nuestra red. Primero complete el archivo nnFuncionCosto.m para retornar el costo. Recuerde que la función de costo es:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log((h_\theta(x^{(i)}))_k) - (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right],$$

donde K = 10 (el número de posibles etiquetas). Tampoco olvide que debemos transformar nuestras etiquetas (1, 2, ..., 10) en vectores, donde por ejemplo el número 5 se representa con el vector [0 0 0 0 1 0 0 0 0 0]. Ustedes deben implementar el cómputo de feedforward para cada muestra y sumar el costo a través de todo el set de datos. Una vez que nnFuncionCosto.m está terminada usando los parámetros Theta1 y Theta2, debe tener un valor cercano a 0.287629.

Regularización

La función de costo con regularización para nuestra arquitectura es:

$$\begin{aligned}J(\theta) =& \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[-y_k^{(i)}\log((h_\theta(x^{(i)}))_k) - (1-y_k^{(i)})\log(1-(h_\theta(x^{(i)}))_k)\right] + \\ & \frac{\lambda}{2m}\left[\sum_{j=1}^{25}\sum_{k=1}^{400}(\Theta_{j,k}^{(1)})^2 + \sum_{j=1}^{10}\sum_{k=1}^{25}(\Theta_{j,k}^{(2)})^2\right].\end{aligned}$$

Recuerde que no debemos regularizar tomando en cuenta los ***bias units***, para el caso de Theta1 y Theta2 esto corresponde a sus primeras columnas. Agregue la regularización a la función de costo que está en nnFuncionCosto.m, debe darles un valor cercano a 0.383770.

Backpropagation

Ahora vamos a programar el algoritmo de backpropagation a nuestra función. Seguimos con la tarea de completar el archivo nnFuncionCosto.m, para que retorne el valor apropiado de grad. Observe que cuando tengamos el cálculo de la gradiente, vamos a poder minimizar la función de costo usando ***fmincg***. Primero implementaremos la versión normal de gradiente, luego vamos a regularizar (al igual que en el ejercicio anterior).

Sigmoide y random

Verifique que sigmoidGradient sirva, prueba un valor de 0, el gradiente debería dar 0.25. También verifique que ese método pueda recibir vectores y matrices.

Además verifique el archivo que se encarga de generar números aleatorios a los pesos, complete el código para que la generación de aleatorios esté dentro de un rango.

Algoritmo backpropagation

Ahora se programará el algoritmo de backpropagation. Recuerde que la idea es la siguiente: dado una muestra (x, y), primero vamos a ejecutar una pasada para generar todas las activaciones (a forward pass), incluyendo las de la última capa. Después para cada nodo j en la capa l, queremos calcular los errores (deltas) que va midiendo los errores cometidos en una muestra. Recuerde los pasos para nuestros algoritmo:

1. Hacer una pasada feedforward para obtener todas las activaciones (hasta la última capa).
2. En la última capa hacer el cáculo del error (utilizando y).
3. Calcular el error de la capa 2.
4. Acumular el gradiente para la muestra actual en delta (Δ), no olvide remover los bias.
5. Obtener los $\boldsymbol{D}$.

El algoritmo anterior debe estar en un ciclo (los pasos del 1 al 4) para ir corriendo muestra por muestra el backpropagation. El paso 5 divide entre m. Una vez que se tenga el algoritmo, se continuará con el gradient checking para comparar resultados.

Gradient Checking

En el archivo gradientChecking, está el código necesario para este cálculo (no tienen que hacerlo). Y el archivo verificarNNGradientes.m se encargará de crear y verificar que todo esté bien. Si la implementación de backpropagation está correcta, se debe observar una diferencia de 1e-9.

Regularización

Hasta este punto aún no se ha regularizado la gradiente. Ya puede agregar la regularización sobre el término D (recuerde que los bias no se regularizan). Al correr gradient checking, la diferencia debe mantenerse (menor a 1e-9).

fmincg

Ahora es momento de usar fmincg para el aprendizaje. Si todo está correcto se debe observar una precisión de 95.3 % (puede variar un poco por la inicialización aleatoria); también se puede mejorar este aspecto si se ponen más iteraciones y variar el parámetro lambda.