Curso de Optimización, 2022

Instituto de Matemática y Estadística (IMERL)

**Práctico 4: Optimización de una red neuronal**

Considere el problema del reconocimiento de un dígito manuscrito. Cada imagen de 20x20 pixeles está codificada en una matriz fila **x** de dimensión 400, donde cada componente  es un número que indica la intensidad de la escala de gris en un pixel.

Una red neuronal (RN) de ods capas está codificada en un programa **predict3**, cuyo resultado es un vector **h2** de dimensión 10 con las probabilidades de cada dígito (la décima posición corresponde al 0). La componente de mayor valor nos indica la predicción **p** (de 1 a 10) y **p0** (con el 10 convertido a 0).

function [p,p0,h2] = predict3(Theta1, Theta2, x)

Además del dato **x** con la imagen (matriz fila), la red usa matrices de parámetros Theta1 y Theta2.

El objetivo de la tarea es “entrenar” una red neuronal ya definida, mejorando iterativamente las matrices Theta1 y Theta2 para que la red neuronal haga una predicción correcta en la mayoría de una gran cantidad de ejemplos de entrenamiento disponibles.

**Ejercicio 4.1. Predicción con la RN aún no optimizada**

1. Cargue el archivo ThetasRN4.mat que contiene las matrices Theta1 y Theta2 correspondientes a una predicción mediocre (que buscaremos mejorar luego).

Cargue el archivo ejemplos43a, que contiene vectores x correspondientes a 4 dígitos: x8, x2, x6, x6b.

Estudie los resultados de la rutina predict3 en cada ejemplo, observando las probabilidades resultantes. Visualice la imagen con displayData .

Observe que la rutina puede llamarse con todos los ejemplos a la vez (en una matriz X donde cada ejemplo es una fila).

1. Cargue el archivo datos5.mat contiene los  datos de 5.000 ejemplos: una matriz **X** con una fila para cada imagen, y un vector **y** con los dígitos correspondientes.

Use la rutina displayData para mostrar varios ejemplos a la vez en un formato de tabla, p.ej. 100 figuras elegidas random enrte los m=5000 casos:

sel = randperm(m);

sel = sel(1:100);

displayData(X(sel, :));

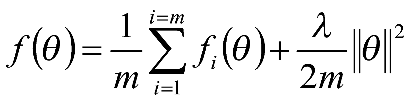
Compare los valores de **p** con **y** en todos los casos de la matriz X, calculando el promedio de aciertos de la predicción: accu=mean(p==y),

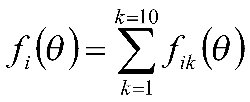
**Ejercicio 4.2.**  **Estudio de la función de costo a minimizar**

El entrenamiento tiene como objetivo minimizar una función de “costo”  asociado a la elección de las matrices Theta. El costo f y su vector gradiente g se calculan en la rutina[[1]](#footnote-1) funcosto3.m:

[f,g]= funcosto3(params)

La función  corresponde a la suma de un costo  para cada ejemplo, más un término de regularización para evitar el problema de overfitting:



El costo correspondiente al ejemplo *i* es de la forma      

donde cada sumando  penaliza la diferencia entre la predicción  y la etiqueta correspondiente a ese ejemplo (que es un dato y vale 0 o 1).

Cada función   es de la forma



La única variable de entrada es el vector params (de largo 25\*401+10\*26=10285), que contiene las dos matrices Theta:    params = [Theta1(:) ; Theta2(:)];

El sumando de regularización depende del parámetro  que debe ajustarse según el problema. En esta tarea definimos lambda = 1;

Otros datos necesarios para calcular el costo son variables globales, y deben definirse en el programa principal:

global n\_ils  n\_hls  n\_lab  X  y  lambda

n\_ils=400;  % input\_layer\_size=20x20

n\_hls=25;   % hidden\_layer\_size

n\_lab=10;   % number of labels, from 1 to 10

Nota: La rutina funcosto3.m necesita de sigmoid.m y de su gradiente en sigmoidGradient.m

**Se pide:**

1. Estudie la función , sabiendo que , analizando el mínimo cuando *v*=0 y cuando *v*=1.
2. Calcule el costo asociado a las matrices las matrices Theta1 y Theta2  del Ejercicio 4.1 (debe obtener un valor cercano a f=2.70).
3. Estime el tiempo de cálculo.
4. Calcule el costo sin el término de regularización.

**Ejercicio 4.3: Entrenamiento de la RN con gradiente conjugado**

**Se pide:**

1. Halle valores iniciales para la optimización con el comando **rand**, ajustado para que entregue valores distribuidos en forma uniforme en un intervalo centrado en 0:

alfa=0.12;   % valores con distrib uniforme en alfa\*[-1, 1]

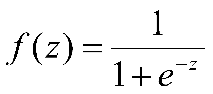
Theta10=alfa\*(2\*rand(n\_hls, n\_ils+1)-1);

Theta20=alfa\*(2\*rand(n\_lab, n\_hls+1)-1);

1. Calcule el costo f0 asociado a esta red neuronal (del orden de f0=7).
2. Use el método del gradiente conjugado para entrenar[[2]](#footnote-2) la red neuronal.
3. Con las mejores matrices Theta halladas, vuelva a hacer la comparación del promedio de aciertos, como en P4.3c.

**Opcionales**

**Ejercicio 4.4:** Estudio de la función sigmoide

Estudie la funcióny su derivada.  Use el programa para graficar en el intervalo [-10,10], observando que puede recibir como parámetro **z** de entrada una matriz.

function [f,g] = sigmoid(z)

f = (1.0)./(1+exp(-z));

g=f.\*(1-f); % derivada

En el ejercicio siguiente se usa esta función para convertir resultados a valores entre 0 y 1.

**Ejercicio 4.4:** Estudie el cálculo de la matriz jacobiana en la función de costo.

**Ejercicio 5.3** Entrenamiento de la RN con L BFGS

**Ejercicio 5.4** Entrenamiento con un algoritmo de descenso con gradiente estocástico (SGD). Una referencia puede ser:

https://en.wikipedia.org/wiki/Stochastic\_gradient\_descent

1. La rutina corresponde a la función NNCOSTFUNCTON del curso "Machine Learning" del **Prof. Andrew Ng**, Stanford University. [↑](#footnote-ref-1)
2. Como el tiempo de cada evaluación de la función de costo puede ser importante, se recomienda empezar probando con solo 3 o 4 iteraciones, para asegurarse que el algoritmo funciona (o sea que los valores del costo decrecen) antes de hacer una corrida con muchas iteraciones. [↑](#footnote-ref-2)