Clasificación de múltiples clases utilizando Regresión Logística y Redes Neurales

José Castro

1 Introducción

En este ejercicio, usted implementará un uno-contra-todos regresión logística y red neural para reconicer dígitos escritos a mano.

Para empezar el ejercicio debe utilizar el código correspondiente que se encuentra en el tecDigital.

2 Archivos en esta tarea

Los archivos incluidos en esta tarea son:

- `ej3.m` Un script de Octave que le ayuda con la primera parte de la tarea.
- `ej3_nn.m` Otro script para la segunda parte de esta tarea.
- `ej3data1.txt` Set de entrenamiento de dígitos escritos a mano.
- `ej3pesos.mat` Pesos iniciales para el ejercicio de redes neurales.
- `despliegueDatos.m` función para visualizar el set de datos.
- `fmincg.m` Función para minimizar (similar a fminunc).
- `sigmoide.m` función sigmoide.
- `[*] funcionDeCostoRL.m` función de costo de regresión logística.
- `[*] unoCotraTodos.m` Entrenamiento de clasificador ono-contra-todos.

- `[*] prediccionUnoCotraTodos.m` Predicción de clasificador uno-contra-todos.
- `[*] preciccion.m` Función de predicción de la red neural.

* indica un archivo que usted debe completar.

A través de este ejercicio usted estará utilizando los scripts `ex2.m` y `ex2_nn.m`. Estos preparan los datos para los problemas y llaman a las funciones que usted escribirá, no necesita modificarlos. Solo se requiere que modifique las funciones en otros archivos (los que tienen *).

3 Regresión logística

En este ejercicio usted utilizará regresión logística y redes neurales para reconocer digitos escritos a mano (del 0 al 9). Reconocimiento automático de dígitos es ampliamente utilizado hoy en día. Este ejercicio le dará los métodos necesarios para llevar a cabo esta tarea.

En la primera parte del ejercicio usted extenderá el ejercicio anterior de regresión logística para lograr la clasificación en varias categorías (uno contra todos).

3.1 Conjunto de Datos

Entre los archivos se le suministra el conjunto de datos `ex3data1.mat` que contiene 5000 ejemplos de entrenamiento de dígitos escritos a mano[1]

La extensión `.mat` significa que los datos han sido almacenados en el formato nativo para matrices de Octave/Matlab, y no en un archivo de texto (ASCII) tal como un archivo csv (Comma Separated Values). Estas matrices se pueden leer directamente en Octave con el comando `load`. Una vez cargadas, las matrices apareceran en la memoria. La matriz ya tendrá nombre, así que no necesita asignarles nombre.

```
>> load('ex3data1.mat');
```

Hay 5000 ejemplos de entrenamiento en `ex3data1.mat` donde cada ejemplo de entrenamiento es una imagen en escala de gris de 20 pixeles por 20

[1]Este es un subconjuto de la base de datos MNIST de dígitos escritos a mano (`http://yan.lecun.com/exdb/mnist/`)

pixeles. Cada pixel es representado por un número de punto flotante que indica la intensidad de la escala de gris en esa ubicación. La grilla de pixeles de 20 por 20 viene "desenrollada" en un vector 400-dimensional. Cada ejemplo de entrenamiento se convierte en una fila de la matriz X. Esto nos da una matriz X de tamaño 5000 por 400 donde cada fila es un ejemplo para un dítigo escrito a mano.

$$\begin{bmatrix} — \left(x^{(1)}\right)^T — \\ — \left(x^{(2)}\right)^T — \\ \vdots \\ — \left(x^{(m)}\right)^T — \end{bmatrix}$$

3.2 Visualizando los datos

Puede empezar visualizando los datos. En la parte 1 de `ex3.m` el código aleatoreamente selecciona 100 filas de `X` y los pasa por la función de `despliegueDatos`. Esta función mapea cada fila a una imagen en escala de gris de 20x20 pixeles y las despliega juntas. Se provee la función `despliegueDatos` que usted puede examinar. Después de correr este paso debería ver una figura como la siguiente:

3.2.1 Vectorizando la Regresión Logística

Usted estará utilizando múltiples modelos de regresión de uno-contra-todos para construir un clasificador multi-clase. Como hay 10 clases, tendrá que entrenar 10 clasificadores de regresión logística por separado. Para hacer eficientemente este entrenamiento, es importante que vectorice su código. En esta sección implementará una versión vectorizada de regresión logística sin ciclos `for`. Puede utilizar el código de su ejercicio anteriror como punto de partida, o si ya vectorizó su implementación, puede ignorar esta sección.

3.2.2 Vectorizando la función de costo

Empezaremos escribiendo una versión vectorizada de la función de costo. Recuerde que la función de costo (no regularizada) para la regresión logística es:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Para calcular cada elemento de la suma, debe calcular $h_\theta\left(x^{(i)}\right)$ para cada ejemplo i, donde $h_\theta\left(x^{(i)}\right) = g\left(\theta^T x^{(i)}\right)$ y $g(z) = \frac{1}{1+e^{-z}}$ es la función sigmoide. Resulta que podemos calcular esto rápidamente para todos los ejemplos utilizando multiplicación de matrices. Definamos X y θ como:

$$X = \begin{bmatrix} — \left(x^{(1)}\right)^T — \\ — \left(x^{(2)}\right)^T — \\ \vdots \\ — \left(x^{(m)}\right)^T — \end{bmatrix} \quad \text{y} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

Entonces, calculando el produco matricial $X\theta$ tenemos:

$$X\theta = \begin{bmatrix} — \left(x^{(1)}\right)^T \theta — \\ — \left(x^{(2)}\right)^T \theta — \\ \vdots \\ — \left(x^{(m)}\right)^T \theta — \end{bmatrix} = \begin{bmatrix} — \theta^T \left(x^{(1)}\right)^T — \\ — \theta^T \left(x^{(2)}\right)^T — \\ \vdots \\ — \theta^T \left(x^{(m)}\right)^T — \end{bmatrix}$$

En la última igualdad utilizamos la propiedad de que $a^T b = b^T a$ si a y b son vectores (el resultado es un escalar). Esto nos permite calcular $\theta^T x^{(i)}$ para todos los ejemplos i en una sola linea de código.

Su trabajo es escribir la función de costo `funcionDeCostoRL.m` Su implementación debe utilizar la estrategia presentada arriba para calcular $\theta^T x^{(i)}$. También debe utilizar código vectorizado para el resti de la función. Una vectorización completa de esta función no contendrá ningún loop (Pista: puede que le sirva utilizar la multiplicación por elementos (`.*`) y la operación de suma `sum` al escribir esta función).

3.2.3 Vectorizando el gradiente

Recuerde que el gradiente (no regularizado) del costo de la regresión logística es un vector donde el $j^{\text{ésimo}}$ elemento se define como

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \right)$$

Para vectorizar esto sobre el set de datos empezamos escribiendo todas las derivadas parciales

$$\begin{aligned}
\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \frac{\partial J(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} &= \frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x_0^{(i)} \right) \\ \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x_1^{(i)} \right) \\ \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x_2^{(i)} \right) \\ \vdots \\ \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x_n^{(i)} \right) \end{bmatrix} \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(\left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)} \right) \\
&= \frac{1}{m} X^T (h_\theta(x) - y).
\end{aligned}$$

donde

$$h_\theta(x) - y = \begin{bmatrix} h_\theta(x^{(1)} - y^{(1)} \\ h_\theta(x^{(2)} - y^{(2)} \\ \vdots \\ h_\theta(x^{(3)} - y^{(3)} \end{bmatrix}$$

Note que $x^{(i)}$ es un vector pero que $\left(h_\theta(x^{(i)}) - y^{(i)} \right)$ es un escalar. Para entender el último paso de la derivación haga $\beta_i = \left(h_\theta(x^{(i)}) - y^{(i)} \right)$ y observe

que:

$$\sum_i \beta_i x^{(i)} = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} = X^T \beta,$$

donde los $\beta_i = \left(h_\theta(x^{(i)}) - y^{(i)}\right)$.

La expresión anterior nos permite calcular todas las derivadas parciales sin utilizar loops. Ahora debe implementar su versión vectorizada del gradiente en el archivo `funcionDeCostoLR.m`.

3.2.4 Vectorizando la regresión logística regularizada

Después de haber implementado la vectorización de la regresión logística. Ahora debe agregar de nuevo la regularizacón a la función de costo. Reguerde que la regresión logística regularizada se define como:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Not que *no* debe regularizar θ_0 el cual es utilizado como coeficiente de $x_0 = 1$, término constante.

Correspondientemente, las derivadas parciales para la regresión logística regularizada para θ_j se definen como:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \qquad \text{para } j = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \quad \text{para } j \geq 1$$

Ahora modifique el código de la función `funcionDeCostoLR.m` para que tome en cuenta la regularización. De nuevo, no debe agregar ningun ciclo a su código.

Cuando implementa vectorización para la regresión logística, puede que le interese sumar solo ciertos elementos de θ. En Octave, puede indexar partes de las matrices. Por ejemplo `A(:,3:5) = B(:,1:3)` *remplazará las columnas 3 a 5 de A con las columnas 1 a 3 de B. Tambien puede usar el* **`end`**, `A(:,2:end)` *es la matriz A sin su primera columna. Si le interesa multiplicar una parte de un vector puede hacer cosas como*

```
sum(z(2:end).^2)
```

3.3 Clasificación de uno-contra-todos

En esta parte implementará clasificación de uno contra todos entrenando múltiples clasificadores de regresión logística, uno para cada una de las K clases. En el ejercicio concretamente hay $K = 10$ clases, pero su implementación debería funcionar con un valor arbitrario de K.

Ahora debe completar el codigo de `unoContraTodos.m` que entrena cada uno de los clasificadores . En particular, su código debe retornar los parámetros de todos los clasificadores en una matriz $\Theta \in \mathbb{R}^{K \times (N+1)}$, donde cada fila de Θ corresponde a los parámetros de la regresión logística aprendidos para una clase particular. Usted puede hacer esto con un "for" de 1 hasta K, entrenando cada clasificador independientemente.

Note que el argumento `y` de esta función es un vector de etiquetas de 1 a 10 donde se ha mapeado el dígito "0" a la etiqueta 10 para evitar confusión con los índices.

Cuando entrena un clasificador para una clase $k \in \{1, \ldots, K\}$ usted le puede interesar un vector m-dimensional de etiquetas y, donde $y_j \in 0, 1$ indica que el $j^{\text{ésimo}}$ par de entrenamiento pertenece a la clase k en cuyo caso $y_j = 1$, o pertenece a una clase distinta, en cuyo caso $y_j = 0$. Para esto puede encontrar útiles los arrays lógicos de Octave.

Pruebe en octave el siguiente código:

```
> a = 1:10;  % inicializamos los valores de a y b
> b = 3;
> a == b     % pruebe con distintos valores de b
```

Además estará utilizando `fmingc` en este ejercicio, es similar a `fminunc` pero es un poco más eficiente cuando se trabaja con un número grande de parámetros.

3.3.1 Predicción de uno-contra-todos

Una vez entrenado su clasificador de uno contra todos, lo puede utilizar para predecir el dígito contenido en una imagen. Para cada input, usted debe calcular la "probabilidad" de que pertenece a una clase utilizando sus clasificadores de regresión logística. Su predictor debe escoger la clase para la cual el clasificador de regresióin logística logró el valor más alto.

Debe completar `prediccionUnoContraTodos.m` y utilizarlo para hacer predicciones.

Una vez hecho, `ex3,m` llamará a su función y la probará. Debe ver una precisión en el test de entrenamiento cercana a 94.9% (clasifica correctamente al 94.9% de los ejemplos en el set de entrenamiento).

4 Redes Neurales

En la parte anterior de este ejercicio, usted implementó regresión logística para reconocer dígitos escritos a mano. Sin embargo, la regresión logística no puede lograr hipótesis muy complejas ya que es solo in clasficador lineal[2]

En esta parte del ejercicio usted implementara una red neural para reconocer dígitos escritos a mano con el mismo conjunto que el usado anteriormente. La red neural podrá representar modelos más complejos que conforman hipótesis no lineales. Su propósito por el momento es implementar nada más la propagación hacia adelante. En el próximo ejercicio hará el entrenamiento de los parámetros también.

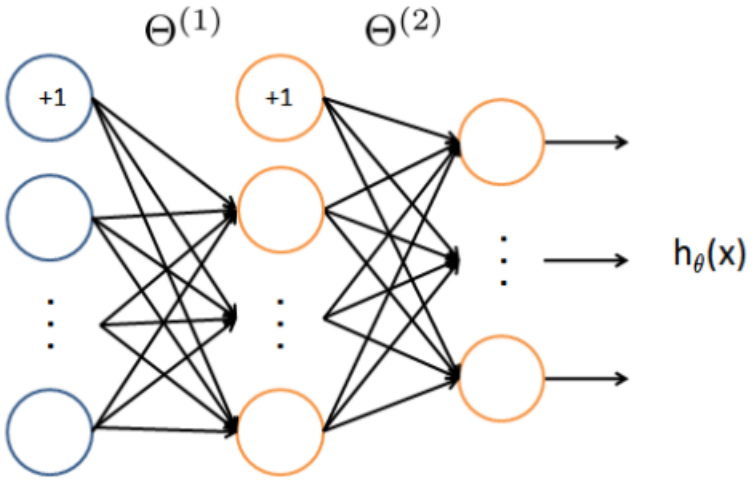
En esta parte estará utilizando el script `ex3_nn.m` como ayuda para efectuar la tarea.

4.1 Representación del modelo

4.1.1 Parámetros de aprendizaje utilizando `fminunc`

La red neural se presenta en la siguiente figura:

[2]Puede agregar más características (tal como utilizar polinomios de las características), pero esto puede salir muy caro de entrenar



$$a^{(1)} = x \qquad z^{(2)} = \Theta^{(1)} a^{(1)} \qquad z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$(+a_0^{(1)}) \qquad a^{(2)} = g(z^{(2)}) \qquad a^{(3)} = g(z^{(3)}) = h_\theta(x)$$
$$(+a_0^{(2)})$$

Tiene 3 capas – una capa de entrada, una capa oculta y una capa de salida. Recuerde que las entradas son valores de pixeles de las imágenes de los dígitos. Igual que antes, los datos de entrenamiento esta en las variables `X` y `y`.

Se le prove con los parámetros $(\Theta^{(1)}, \Theta^{(2)})$ ya entrenados. Estos se guardan en `ex3pesos.mat` y serán cargados por `ex_nn.m` en las variables `Theta1` y `Theta2`. Los parámetros tienen la dimensión necesaria para una red con 25 neuronas en la segunda capa y 10 en la tercera (correspondiente a las 10 clases de los dígitos).

4.2 Propagación hacia adelante y predicción

Ahora usted implementará la propagación hacia adelante de la red. Para ello necesita completar el código de `prediccion.m`.

Usted implementará el cómputo de propagación hacia adelante que calcula $h_\theta(x^{(i)})$ para cada ejemplo i y retorna la predicción asociada. Similar a la estrategia de clasificación de uno-contra-todos, la predicción de la red neural será la etiqueta de quien logre la salida más grande $(h_\theta(x))_k$

Nota de implementación La matrix `X` contiene los ejemplos en las filas. Cuando hata el código de `prediccion.m` necesitará agregar una columna de

1's a la matriz. Las matrices `Theta1` y `Theta2` contienen los parámetros de las neuronas pro filas. Específicamente la primer fila de `Theta1` corresponde a la primer neurona oculta. En Octave cuando calcule $z^{(2)} = \Theta^{(1)}a^{(1)}$, asegúrese de que indexa `X` correctamente (y la transpone si es necesario) tal que pueda obtener $a^{(l)}$ como un vector columna.

Una vez que logra terminar la predicción debería lograr una precisión del 97.5%.

5 Evaluación

- 30 pts – `funcionDeCostoLR.m`
- 20 pts – `unoCotraTodos.m`
- 20 pts – `prediccionUnoContraToos.m`
- 30 pts – `prediccion.m`