Deep Learning

Miguel Vilchis

03 de agosto de 2018

1. Definiciones generales

- Recordar que el objetivo es encontrar la función que logra clasificar correctamente cada entrada.
- Funciones de activación: Se usan funciones no lineales para poder aprender funciones no lineales.
- Parametrización: Proceso de definir que parametros son necesarios para un modelo dado.
- Función de perdida: Cuantifica que tan bien la clase de salida se apega a la verdadera clase. (Buscamos minimizar dicha función)
- Función de Scoring: Esta función acepta los datos como entrada y los mapea a una etiqueta de clase. Ejemplo:

  $$f(x_i, W, b) = Wx_i + b$$

  Donde:

  $$W = [K \times D], \quad x\_i = [D \times 1] \text{ y } b = [K \times 1]$$

  Siendo K el número de clases para clasificar.
- Bias: Permite recorrer o transladar nuestra función de scoring en una u otra dirección sin modificar la matriz de pesos.

2. Función de perdida

Es común usar funciones *hinge* pero se usa mas *cross-entropy loss* y *softmax* en el contexto de deep learning y redes convolucionales.

- Multi-class svm loss
  Agregando la columna de bias a la matriz de pesos, sea:

  $$s = f(x_i, W) = Wx_i$$

  Dado el punto *i-esimo*, el score predicho de la clase *jth* queda definido como:

  $$s_j = f(x_i, W)_j$$

  De aquí obtenemos la función *hinge loss function* (sumando las clasificaciones incorrectas de cada clase y comparandola con la salida de la función score):

  $$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

  Mientras que la función *squared hinge loss* (penaliza mas la perdida) se define como:

  $$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)^2$$

- Cross-entropy loss: Siendo softmax una función regresa probabilidades para cada clase mientras que la función *hinge* nos da el margen. El clasificador softmax es la generalización de la forma binaria de la regresión logística. $f(x)_i = e^{s_{x_i}} / \sum_j e^{x_j}$ y la función de perdida *cross-entropy*

  $$L_i = -ln(e^{s_{y_i}} / \sum_j e^{y_j})$$

  Entonces, la función de perdida debe minimizar el logaritmo negativo de la probabilidad de la clase correcta:

  $$L_i = -lnP(Y = y_i | X = x_i)$$

  Donde

  $$P(Y = y_i | X = x_i) = e^{s_{y_i}} / \sum_j e^{s_j}$$

  Por lo que la función de perdida *cross-entropy*

  $$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

3. Descenso de gradiente

Nosotros tratamos las superficies de perdida como convexas incluso si no lo son, porque hace buen trabajo. El algoritmo de descenso de gradiente tiene dos veritentes:

1. La implementación estandart *vanilla*
2. La versión optimizada *estocástica*

El pseudocodigo del descenso de gradiente es:

```
while True:
 Wgradient = evaluate_gradient(loss, data, W)
W += -alpha *Wgradient
```

Para el gradiente estocástico, en lugar de calcular el gradiente para todo el conjunto de datos, se hace sobre un sampleo de estos.

```
while True:
 batch = next_training_batch(data, 256)
 Wgradient = evaluate_gradient(loss, data, W)
W += -alpha *Wgradient
```

4. Momentum

Sea $W = W - \alpha \bigtriangledown_W f(W)$ el termino V momentum, ponderado por γ:

$$V = \gamma V_{t-1} + \alpha \bigtriangledown_W f(W) \longrightarrow W = W - V_t$$

Comunmente se usa $\gamma = 0.9$

5. Regularización

Es la técnica que asegura que nuestro modelo generalice bien, ayudandonos a controlar la capacidad de nuestro modelo. Sea la función de perdida *cross-entropy* L

$$L_i = -log(e^{s_{y_i}} / \sum_j e^{s_j}) \longrightarrow L = \frac{1}{N} \sum i = 1^N L_i$$

La función de regularización o decaemiento del peso se define como:

$$R(W) = \sum_i \sum_j W_{i,j}^2$$

Actualizando la función de perdida:

$$L = \frac{1}{N} \sum i = 1^N L_i + \lambda R(W)$$

Entonces:

$$W = W - \alpha \bigtriangledown_W f(W) - \lambda R(W)$$

Tipos de regularización:

1. L2 o decaemiento de peso: $R(W) = \sum_i \sum_j W_{i,j}^2$
2. L1: $R(W) = \sum_i \sum_j |W_{i,j}|$