

Deep Learning

Miguel Vilchis

03 de agosto de 2018

1. Definiciones generales

- Recordar que el objetivo que tiene deep learning 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:

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

Siendo K el número de clases para clasificar.

- Bias: Permite recorrer o trasladar 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
Se puede agregar la columna de Bias a la matriz de peso, con este truco, podemos dejar de preocuparnos por actualizar el bias, para solo parametrizar la matriz de pesos, haciendo esto nuestra función queda expresada como:

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

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 y se define como:

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

- Cross-entropy loss: Mientras la función softmax regresa probabilidades para cada clase. 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 pérdida *cross-entropy*

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

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

$$L_i = -\ln P(Y = y_i | X = x_i)$$

Donde

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

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

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

3. Descenso de gradiente

Nosotros tratamos las superficies de pérdida como convexas incluso si no lo son, ya que el hacerlo así da buenos resultados. El algoritmo de descenso de gradiente tiene dos variantes:

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

El pseudocódigo 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

La idea es que, cuando estas descendiendo, este descenso te otorgue impulso para llegar más rápido (en menos épocas) a tener menor pérdida y mejor precisión. Sea $W = W - \alpha \nabla_W f(W)$ el término V momentum, ponderado por γ :

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

Comunmente se usa $\gamma = 0.9$

5. Regularización

Es la técnica que asegura que nuestro modelo generalice bien, ayudándonos a controlar la capacidad de nuestro modelo. Lo cual logramos al penalizar altos valores de peso, ya que, estos influyen más en la predicción de salida. Sea la función de pérdida *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 decaimiento del peso se define como:

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

Actualizando la función de pérdida:

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

Entonces:

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

Tipos de regularización:

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