Optimización del entrenamiento

Redes profundas

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

23 de febrero de 2023

Entrenamiento en línea vs por lotes

Temas

Entrenamiento en línea

- Hablamos de *entrenamiento en línea* cuando se entrena a la red **ejemplar por ejemplar**.
- Los pesos son actualizados más rápidamente, pero la aproximación del gradiente es muy sensible al ejemplar, por lo que el entrenamiento oscila variando los parámetros en direcciones diferentes.

Algorithm Entrenamiento en línea.

1: **for each** x_i **in** X **do**
2: $h_i \leftarrow$ PROPAGACIÓN HACIA ADELANTE(x_i)
3: $\nabla_\Theta J \leftarrow$ PROPAGACIÓN HACIA ATRÁS(h_i, y_i)
4: $\Theta \leftarrow$ OPTIMIZA($\Theta, \nabla_\Theta J$)

Temas

Entrenamiento en lote

- Hablamos de *entrenamiento en lote* cuando se entrena a la red tras haber evaluado el gradiente **sobre todos los ejemplares** de entrenamiento.
- La estimación del gradiente es mejor, porque toma en cuenta más ejemplares de la función.
- Cada actualización de los parámetros requiere mucho más cómputo.
- No se recomienda cuando la red aún está lejos de valores óptimos.

Algorithm Entrenamiento en línea.

1: $H \leftarrow$ PROPAGACIÓN HACIA ADELANTE(X)
2: $\nabla_\Theta J \leftarrow$ PROPAGACIÓN HACIA ATRÁS(H, Y)
3: OPTIMIZA($\Theta, \nabla_\Theta J$)

Temas

Entrenamiento en minilotes

- Busca un compromiso entre el entrenamiento en línea y el entrenamiento en lote.
- Se divide el conjunto de entrenamiento X en bloque más pequeños X_k y se entrena para cada uno de ellos.

Algorithm Entrenamiento en minilotes.

1: **for each** X_k **in** X **do**
2: $H_k \leftarrow$ PROPAGACIÓN HACIA ADELANTE(X_k)
3: $\nabla_\Theta J \leftarrow$ PROPAGACIÓN HACIA ATRÁS(H_k, Y_k)
4: $\Theta \leftarrow$ OPTIMIZA($\Theta, \nabla_\Theta J$)

Normalización

Temas

Justificación

- Las funciones de optimización trabajan mejor si las curvas de nivel de la función de error tienen excentricidad baja, porque la dirección de máximo descenso no varía mucho durante el brinco discreto.

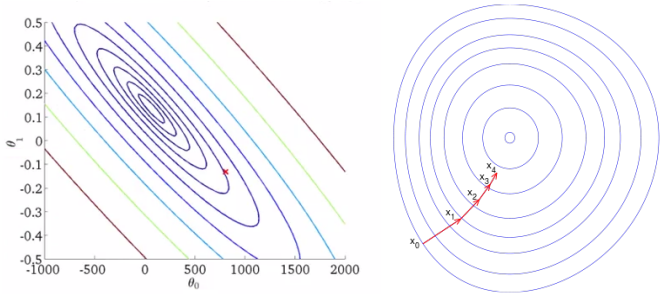


Figura: Izq: J con alta excentricidad. Der: J con curvas de nivel tendiendo a círculos.

- Este no es el caso si las magnitudes de los datos de entrada varían demasiado en magnitud.
  Por ejemplo:
  1. $x_1 \in [10000, 900000]$
  2. $x_2 \in [0.1, 1.33]$
  3. $x_3 \in [50, 200]$

  Lo que este tipo de datos hacen es producir elipses de altísima excentricidad cerca de los mínimos locales.
- Para compensar esto se propone normalizar los datos antes de ingresarlos a la red.

Normalización

- El objetivo es centrar los datos aproximadamente en el intervalo $[-1.0, 1.0]$.
- Existen varias fórmulas para realizar la normalización, se sugiere:

1. Calcular la media μ_i y varianza σ_i^2 para cada característica i en los datos del conjunto de entrenamiento X.

Sea X_i la columna con la i-ésima característica en los datos de entrenamiento X.

$$\mu_i = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{1}$$

$$\sigma_i^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu)^2 \tag{2}$$

$$X_i \leftarrow \frac{X_i - \mu}{\sigma^2} \tag{3}$$

- Cuando una red ha sido entrenada con datos normalizados, es necesario almacenar las medias y varianzas utilizadas durante el entrenamiento con el conjunto de entrenamiento X_E.
- Estos valores serán utilizados para normalizar otros datos que vayan a ser evaluados en la red, inclusive si se recibe un solo ejemplar.

Temas

Funcionamiento

- La normalización permite reducir la excentricidad en la función de error provocada por la disparidad entre los datos de entrada.
- Sin embargo, al calcular los valores para las capas intermedias, los valores pueden cambiar sus rangos para las neuronas intermedias.
  Ej:
  1. $x_1^l \in [0.003, 0.996]$
  2. $x_2^l \in [0.45, 0.65]$
  3. $x_3^l \in [0.68, 0.88]$
- La normalización por lotes aplica los beneficios de la normalización también a las capas intermedias.

- Los algoritmos de optimización podrán utilizar **tazas de aprendizaje** más altas, porque los brincos discretos del algoritmo no cambiarán drásticamente el comportamiento de la función de error durante un intervalo más largo.
- También cada capa aprenderá a calcular características un tanto independientes de algos sesgos en los datos con los cuales fue entrenada.

Normalización por lotes

1. Normalizar la salida de la capa de activación anterior restando su media y dividiendo entre la desviación estándar.
2. El *descenso por el gradiente estocástico*[1] modificará los pesos para optimizar J, probablemente contrarrestando el efecto de la normalización.
3. Para evitarlo, se añaden dos parámetros, también entrenables: γ una desviación estándar y β una media para corrimiento, la idea es que el algoritmo tienda a modificar estos dos para ese fin, en lugar de los pesos, de modo que los pesos produzcan un cómputo más estable.

[1] Es descenso por el gradiente, pero por lotes

Fórmulas

Sea $\mathcal{B}$ el minilote con x_i sus ejemplares, las entradas y_i para la capa siguiente se calculan con:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \tag{5}$$

$$x_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \tag{6}$$

$$y_i = \gamma x_i + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{7}$$

con ϵ una constante agregada para mantener la estabilidad numérica.

Inferencia

Cuando la red ya está entrenada y será utilizada para realizar *inferencias*, calcular la media y varianza **sobre los minilotes de entrenamiento**:

$$E[x] = E_{\mathcal{B}}[\mu_{\mathcal{B}}] \qquad Var[x] = \frac{m}{m-1} E_{\mathcal{B}}[\sigma^2_{\mathcal{B}}] \tag{8}$$

De modo que ahora los valores de activación saliendo de cada capa son:

$$y = \frac{\gamma}{\sqrt{Var[x] + \epsilon}} \cdot x + \left(\beta - \frac{\gamma E[x]}{\sqrt{Var[x] + \epsilon}} \right) \tag{9}$$

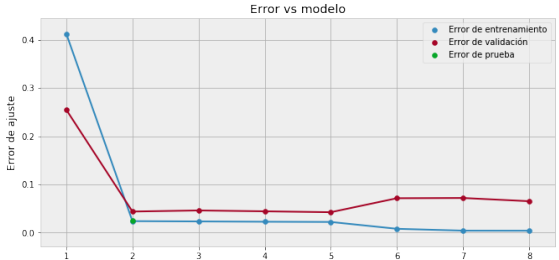
y dependen deterministicamente de los valores de entrada.

Regularización

Temas

Ajuste pobre y sobre ajuste

- Se dice que un modelo realiza un *ajuste pobre* (*underfitting*) si no logra reducir suficientemente el error sobre el conjunto de entrenamiento (menos aún sobre el de validación).
- Se dice que un modelo realiza un *sobre ajuste* (*overfitting*) si reduce muy bien el error sobre el conjunto de entrenamiento, pero es muy alto sobre el de validación, es decir la hipótesis **no generaliza** a datos no vistos previamente.



Estrategias de solución

Hay dos aspectos que se pueden variar para atender estos problemas:

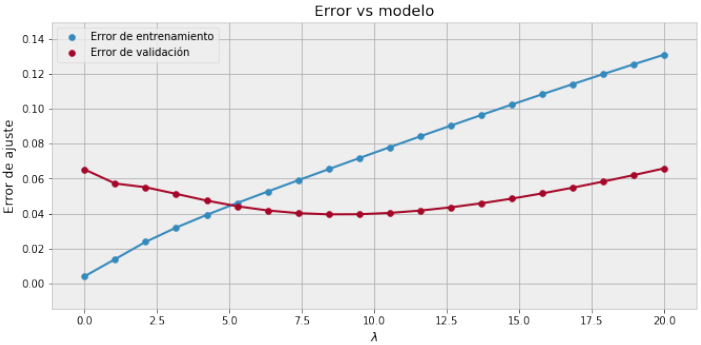
1. El **número de neuronas**:
   - Si el **ajuste es pobre**, quiere decir que el espacio de hipótesis **es muy simple** para poder aproximar la función.
     En redes neuronales, equivale a decir que **faltan neuronas**.
   - Si hay **sobreajuste**, el espacio de hipótesis es demasiado expresivo y ha logrado **memorizar** los datos de entrenamiento, evitándose abstraer la función que los genera.
     En redes neuronales, equivale a decir que **sobran neuronas**. Se caracteriza porque al reducir neuronas:
     - el desempeño de la red entrenada en el conjunto de entrenamiento X_E no empeora significativamente
     - el desempeño en el de validación X_V mejora.
2. Faltan datos de entrenamiento si, independientemente del número de neuronas, el error no baja en X_V, o incluso ni en X_E.

Regularización

- Para automatizar la búsqueda en un espacio de hipótesis suficientemente expresiva se utiliza también la estrategia de *regularización*.
- La función de pérdida a minimizar estará integrada por dos componentes:
  - La función de error
  - Un término de penalización sobre las magnitudes de los pesos de la red.
- El mínimo de esta función hallará un punto de compromiso tal que:
  1. Reduce el error lo más posible
  2. dejando crecer sólamente aquellos pesos que contribuyan mejor a reducir el error, *compensando por la contribución de su magnitud creciente*.

Ajuste probre o sobre ajuste según λ

$$J(\Theta) = \text{Error}(\Theta) + \frac{\lambda}{2m}\sum_{\Theta}\theta^2 \qquad \nabla_\Theta J^{(l)} = \nabla\text{Error}_\Theta^{(l)} + \frac{\lambda}{m}\begin{bmatrix} 0 \\ \Theta^{(l)}_{[:,1:]} \end{bmatrix} \tag{10}$$



Referencias I

Batch normalization in Neural Networks, Towards data science 2017,

https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

Licencia