Hyper Parameters Tuning:

Importance of Hyper-parameters:

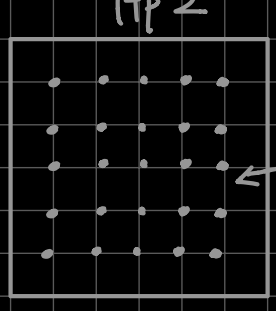
High
↓ importance
low

- α: learning rate
- β: momentum term (0.9)
- # hidden units
- mini-batch size
- # layers
- γ: learning rate decay

$\beta_1, \beta_2, \varepsilon$: momentum in the Adam optimization

The way to choose Hyper-parameters:

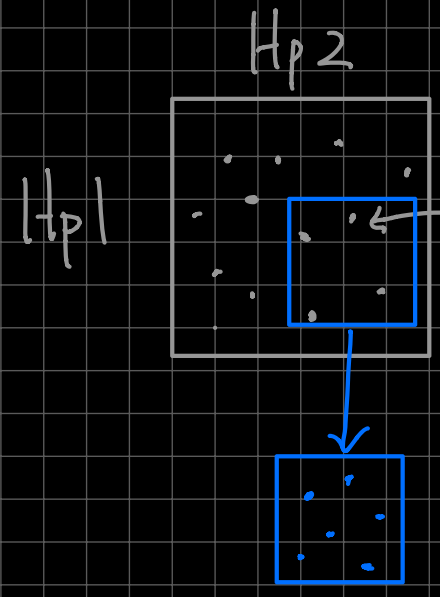
① In Machine Learning, when there are only a few hps, we use grid-search (网格化搜索) to systemetically choose their best value.



← values of Hp 1 and Hp 2

② In DeepLearning, there are a great num of hps and each group of hps need to be trained for a long time. So we don't use this method.

We use Random-Search method to choose.

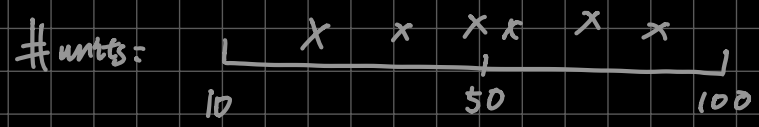


Reason: If use the grid-search method, 25 groups of hps only have 5 different Hp1 values, while in Random-search we have 25 different Hp1 values.

Using an appropriate scale ⇒

Random-choose doesn't mean that you choose points uniformly at random.

Linear Scale: #layers, # units



Log Scale: α

⇒ `r = -4 * np.random.rand()`
`α = 10^r`
($r \in [-4, 0]$)

Exponentially weighted averages : β

⇒ $1-\beta = 0.1, 0.01, 0.001$

`r = -3 * np.random.rand()`

$1-\beta = 10^r$

$\beta = 1 - 10^r$

△ Why we don't use linear scale for β:

β: 0.9000 → 0.9005 ⇒ Have no impact

Last Examples: ~10 ~10

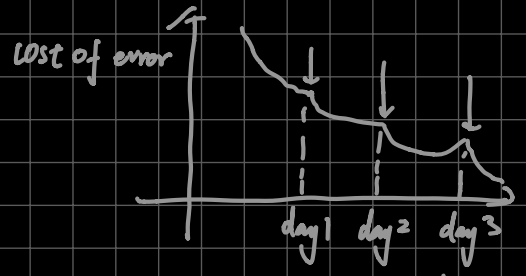
$\frac{1}{1-\beta}$ ←

β: 0.999 → 0.9995 ⇒

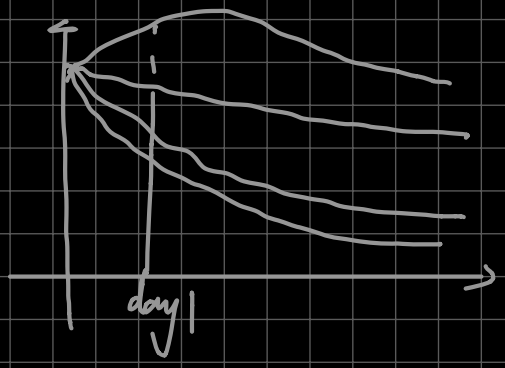
Last Examples: ~1000 ~2000

Hyper-parameters tuning method:

pandas: Babysitting one model



caviar: Training many models in parallels.



Implementing Batch Normalization:

Given some intermediate values: $Z^{(k)}_1, Z^{(k)}_2, \cdots Z^{(k)}_m$,

$$\Rightarrow \textcircled{1}\ \mu = \frac{1}{m}\sum_{i=1}^{m} Z_i^{(k)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(Z_i^{(k)} - \mu\right)^2 \Bigg\} \quad Z^{(i)}_{norm} = \frac{Z_i^{(k)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\textcircled{2}\quad \tilde{Z}^{(i)} = \gamma Z^{(i)}_{norm} + \beta$$

(γ, β: learnable parameters)

- Normalization
- Linear Transformation $\Rightarrow$ 弥补-损失的底层参数信息.

△ Batch norm is usually used accompanied by mini-batch.

△ Parameters during training: W, b, β, γ

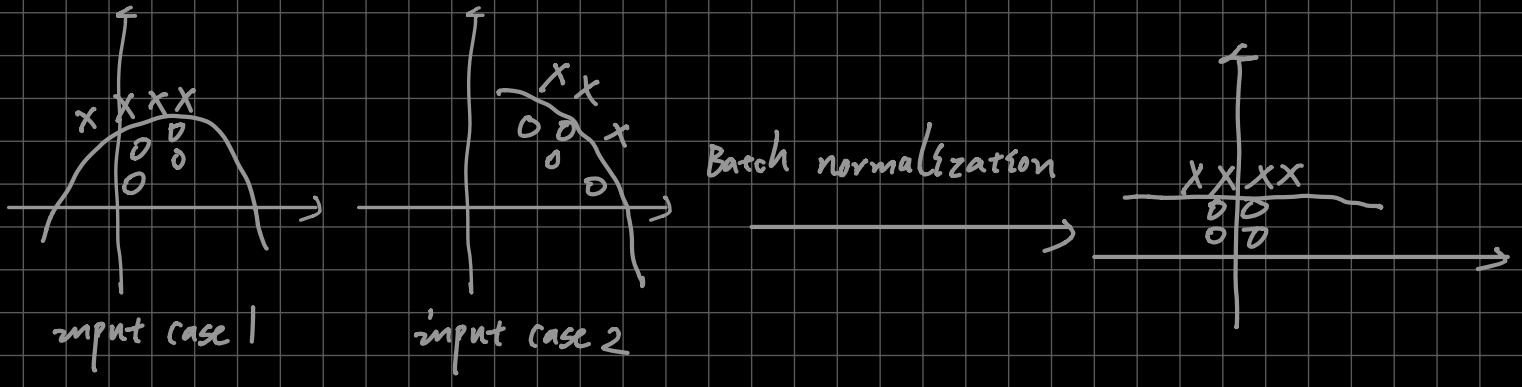
instead b.

$$\beta^{(l)} = \beta^{(l)} - \alpha \cdot d\beta^{(l)}$$

$$\gamma^{(l)} = \gamma^{(l)} - \alpha \cdot d\gamma^{(l)}$$

Why Batch-Normalization works?

1. Shift different input distribution to the same norm distribution. Increase the generalized ability of the model (⇒)



2. Avoid ICS (Interal Covariate Shift)

⇒ By ensuring the $u^{(k)}$. $z^{(k)}$ the same no matter how the $a^{(k-1)}$ changes. (The earlier layer changes)

Allow each layer to learn by itself, reduce the impact of earlier layers.

3. Avoid gradient saturation. ⇒ Similar to relu / dropout, Batch-normalization adds some noise to $Z^{[k]}$ within that minibatch. So it has a slight regularization effect.

△ mini-batch size ↑, regularization effect ↓

64 → 512, we may need to add drop out.

(because the noise we add into $Z^{[l]}$ reduces)

So don't expect to use Batch Normalization as a regularizer. The regularization effect of BN is unintended side-effect.

Batch norm at test time ⇒ In training step, we have mini-batch, which Batch norm relys on. while in the test step. we usually feed in the whole test set. Thus it need to do some changes.

Idea: Estimate u, σ^2 from training set.

⇒ Use exponentially weighted average across mini-batch as u, σ^2 estimation value.

Method: During training, keep track of each layer's $u^{(l)}, \sigma^{2(l)}$ and apply to test set in:

$$Z_{norm} = \frac{Z - \hat{u}}{\sqrt{\hat{\sigma}^2 + \varepsilon}}$$ when tests.

This method is quite robust.

Multi-class classification problem ⇒ Use softmax regression

Binary classification problem ⇒ Logistic regression

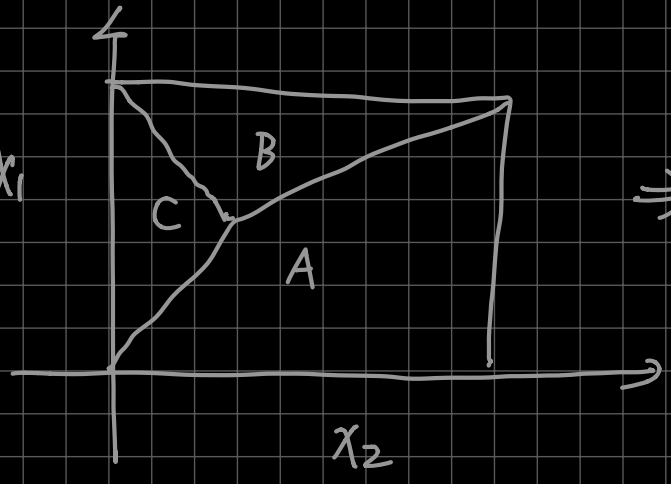
Regression problem ⇒ Linear regression / Multi-variate regression.

Softmax function:

$$a^{[L]} = g^{[L]}(z^{[L]})$$

$$t = e^{(z^{[L]})}, \quad t_i = e^{(z_i^{[L]})}$$

$$a^{[L]} = \frac{t}{\sum_{j=1}^{n} t_j}, \quad a_i^{[L]} = \frac{t_i}{\sum_{j=1}^{n} t_j}$$



⇒ After using softmax function, each class is seperated by straight lines.

Why it called softmax ?

Contrast to hard-max (one-hot). where there is : $\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}$ in the output.

Softmax Regression:

classes

Loss function: $L(y, \hat{y}) = -\sum_{j=1}^{n} y_j \cdot \log \hat{y}_j \Rightarrow \hat{y}_j \to 1, y_j \to 1$

or $\hat{y}_j \to 0, y_j \to 0$

均使得 $L \to 0$.

Cost function: $J(w^{[1]}, b^{[1]})$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\underbrace{y_i}_{n\times 1}, \underbrace{\hat{y}_i}_{n\times 1})$$

$$Y = \begin{bmatrix} \cdots \end{bmatrix}_{n\times m} \quad (\rightarrow m,\ \downarrow n)$$

$$\hat{Y} = \begin{bmatrix} \cdots \end{bmatrix}_{n\times m} \quad (\rightarrow m,\ \downarrow n)$$