# D76：優化器optimizers




PDF 下載 全螢幕

## Sample Code & 作業內容

請參閱作業範例 :Day76-Opimizer_example
作業1:以同一模型, 分別驗證 SGD, Adam, Rmsprop 的 accurancy
作業2:以, Adam, 為例, 調整 batch_size, epoch , 觀察aaurancy, loss 的變化

作業請提交Day76-Optimizer_HW
檔案 Day76-Optimizer_進階 額外提供給學員作為參考

檢視範例

## 參考資料

### 延伸閱讀

An overview of gradient descent optimization algorithms
**https://arxiv.org/pdf/1609.04747.pdf**

在很多機器學習和深度學習的應用中，我們發現用的最多的優化器是Adam，為什麼呢？
下面是TensorFlow中的優化器，**https://www.tensorflow.org/api_guides/python/train**

在keras中也有SGD，RMSprop，Adagrad，Adadelta，Adam等：**https://keras.io/optimizers/**

我們可以發現除了常見的梯度下降，還有Adadelta，Adagrad，RMSProp 等幾種優化器，都是什麼呢，又該怎麼選擇呢？**https://blog.csdn.net/qq_35860352/article/details/80772142**

Sebastian Ruder的這篇論文中給出了常用優化器的比較 **https://arxiv.org/pdf/1609.04747.pdf**

### 延伸閱讀:優化器是編譯Keras模型所需的兩個參數之一

```
from keras import optimizers
model = Sequential() model.add(Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(Activation('softmax'))
sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

您可以在傳遞優化器之前將其實例化model.compile()，如上例所示，或者您可以通過其名稱來調用它。
在後一種情況下，將使用優化程序的默認參數。

```
# pass optimizer by name: default parameters will be used
model.compile(loss='mean_squared_error', optimizer='sgd')
```

**所有Keras優化器通用的參數**
的參數clipnorm和clipvalue可以與所有優化可以用來控制限幅梯度

```
from keras import optimizers

# All parameter gradients will be clipped to
# a maximum value of 0.5 and
# a minimum value of -0.5.
sgd = optimizers.SGD(lr=0.01, clipvalue=0.5)
```

```
from keras import optimizers

# All parameter gradients will be clipped to
# a maximum norm of 1.
sgd = optimizers.SGD(lr=0.01, clipnorm=1.)
```

### 延伸閱讀:二階優化算法

**https://web.stanford.edu/class/msande311/lecture13.pdf**
二階優化算法使用了二階導數(也叫做**Hessian方法** )來最小化或最大化損失函數。由於二階導數的計算成本很高，所以這種方法並沒有廣泛使用。

The second-order information is used but no need to inverse it.

0) Initialization: Given initial solution $\mathbf{x}^0$. Let $\mathbf{g}^0 = \nabla f(\mathbf{x}^0)$, $\mathbf{d}^0 = -\mathbf{g}^0$ and $k = 0$.

1) Iterate Update:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k, \text{ where } \alpha^k = \frac{-(\mathbf{g}^k)^T \mathbf{d}^k}{(\mathbf{d}^k)^T \nabla^2 f(\mathbf{x}^k) \mathbf{d}^k}.$$

2) Compute Conjugate Direction: Compute $\mathbf{g}^{k+1} = \nabla f(\mathbf{x}^{k+1})$. Unless $k = n - 1$:

$$\mathbf{d}^{k+1} = -\mathbf{g}^{k+1} + \beta^k \mathbf{d}^k \quad \text{where} \quad \beta^k = \frac{(\mathbf{g}^{k+1})^T \nabla^2 f(\mathbf{x}^k) \mathbf{d}^k}{(\mathbf{d}^k)^T \nabla^2 f(\mathbf{x}^k) \mathbf{d}^k}$$

and set $k = k + 1$ and go to Step 1.

3) Restart: Replace $\mathbf{x}^0$ by $\mathbf{x}^n$ and go to Step 0.

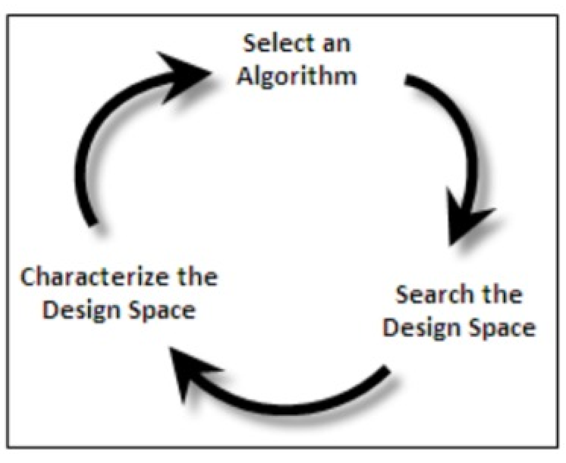
For convex quadratic minimization, this process end in no more than $1$ round.

### 延伸閱讀:自適應的算法

如果需要更快的收斂，或者是訓練更深更複雜的神經網絡，需要用一種自適應的算法。
**http://www.redcedartech.com/pdfs/Select_Optimization_Method.pdf**

定義一個設計問題 匹配給定搜索方法所需的功能 搜索的重要專業知識和經驗 方法和要解決的問題類型

目標是 確定哪種算法和相關的調整在最終解決方案中使用的參數




## 提交作業

請將你的作業上傳至 Github，並貼上該網網址，完成作業提交

https://github.com/ 確定提交

如何提交

## 到 Cupoy 問答社區提問，讓教練群回答你的疑難雜症

向專家提問

如何提問