Tareas de Apoyo en Proyectos de Deep Learning Parte 2

Juan Gamarra Moreno

Historia de entrenamiento

Introducción

Se puede aprender mucho sobre las redes neuronales y los modelos de aprendizaje profundo observando su rendimiento a lo largo del tiempo durante el entrenamiento. Es importante realizar tareas para revisar y visualizar el rendimiento de los modelos de aprendizaje profundo a lo largo del tiempo durante el entrenamiento.

Historia de entrenamiento

Introducción (cont.)

Es importante saber:

- Cómo inspeccionar las métricas del historial recopiladas durante el entrenamiento.
- Cómo trazar métricas de precisión en conjuntos de datos de entrenamiento y validación durante el entrenamiento.
- Cómo trazar métricas de pérdida del modelo en conjuntos de datos de entrenamiento y validación durante el entrenamiento.

Historia de entrenamiento

Acceso a la historia del entrenamiento en Keras

Keras proporciona la capacidad de registrar devoluciones de llamada al entrenar un modelo de aprendizaje profundo. Una de las devoluciones de llamada predeterminadas que se registra al entrenar todos los modelos de aprendizaje profundo es la devolución de llamada del historial. Registra métricas de entrenamiento para cada época. Esto incluye la pérdida (loss) y la exactitud (accuracy) para problemas de clasificación, así como la pérdida y exactitud del conjunto de datos de validación, si se establece uno.

Historia de entrenamiento

Acceso a la historia del entrenamiento en Keras (cont.)

El objeto history se devuelve de las llamadas a la función fit () utilizada para entrenar el modelo. Las métricas se almacenan en un diccionario en el miembro history del objeto devuelto. Por ejemplo, puede enumerar las métricas recopiladas en un objeto de history con print(historia.history.keys()).

Historia de entrenamiento

Acceso a la historia del entrenamiento en Keras (cont.)

Podemos utilizar los datos recopilados en el objeto de history para crear gráficos. Los gráficos pueden proporcionar una indicación de aspectos útiles sobre el entrenamiento del modelo, como:

- Su velocidad de convergencia sobre épocas (pendiente).
- Si el modelo puede haber convergido ya (meseta de la línea).
- Si el modelo puede estar sobreaprendiendo los datos de entrenamiento (inflexión para la línea de validación).

Y más.

Historia de entrenamiento

Visualizar el historial de entrenamiento del modelo en Keras

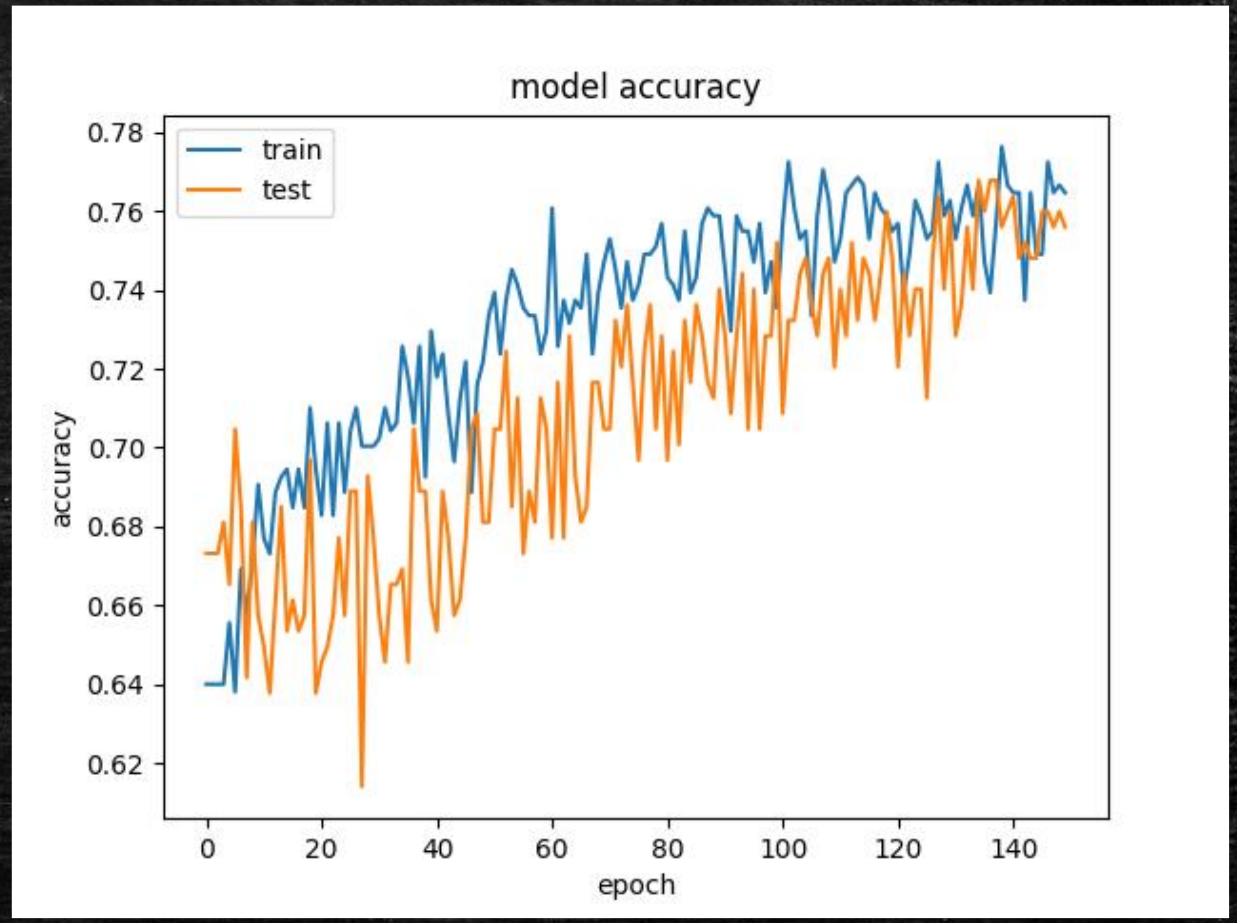
Podemos crear gráficos a partir de los datos históricos recopilados. Por ejemplo, se puede recopilar el historial, devuelto después de entrenar el modelo y crear dos gráficos:

1. Un gráfico de exactitud (accuracy) en los conjuntos de datos de entrenamiento y validación durante épocas de entrenamiento.
2. Una gráfica de pérdida (loss) en los conjuntos de datos de entrenamiento y validación durante épocas de entrenamiento.

Historia de entrenamiento

Visualizar el historial de entrenamiento del modelo en Keras (cont.)

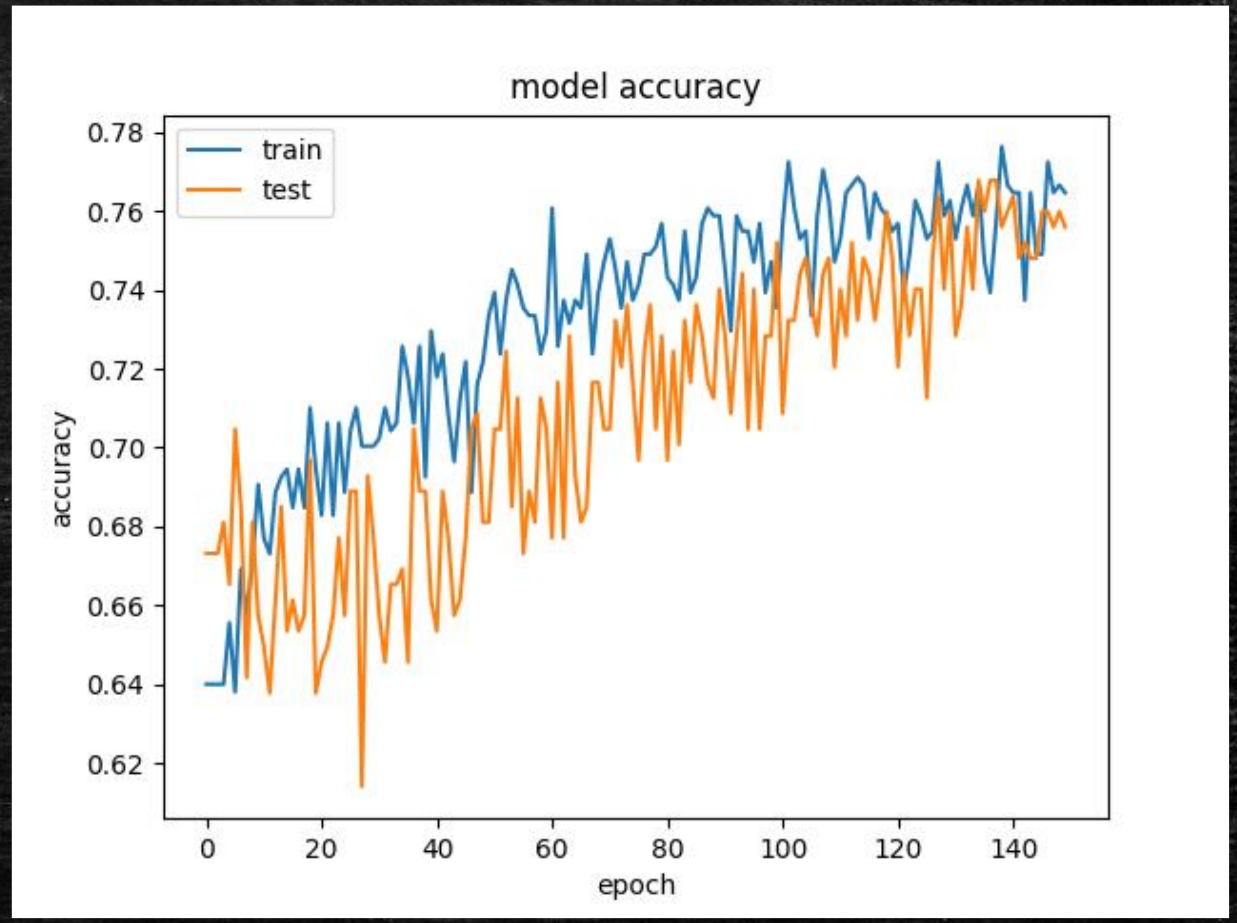
A partir de la gráfica de Accuracy, podemos ver que el modelo probablemente podría entrenarse un poco más, ya que la tendencia de precisión en ambos conjuntos de datos sigue aumentando durante las últimas épocas.



Historia de entrenamiento

Visualizar el historial de entrenamiento del modelo en Keras (cont.)

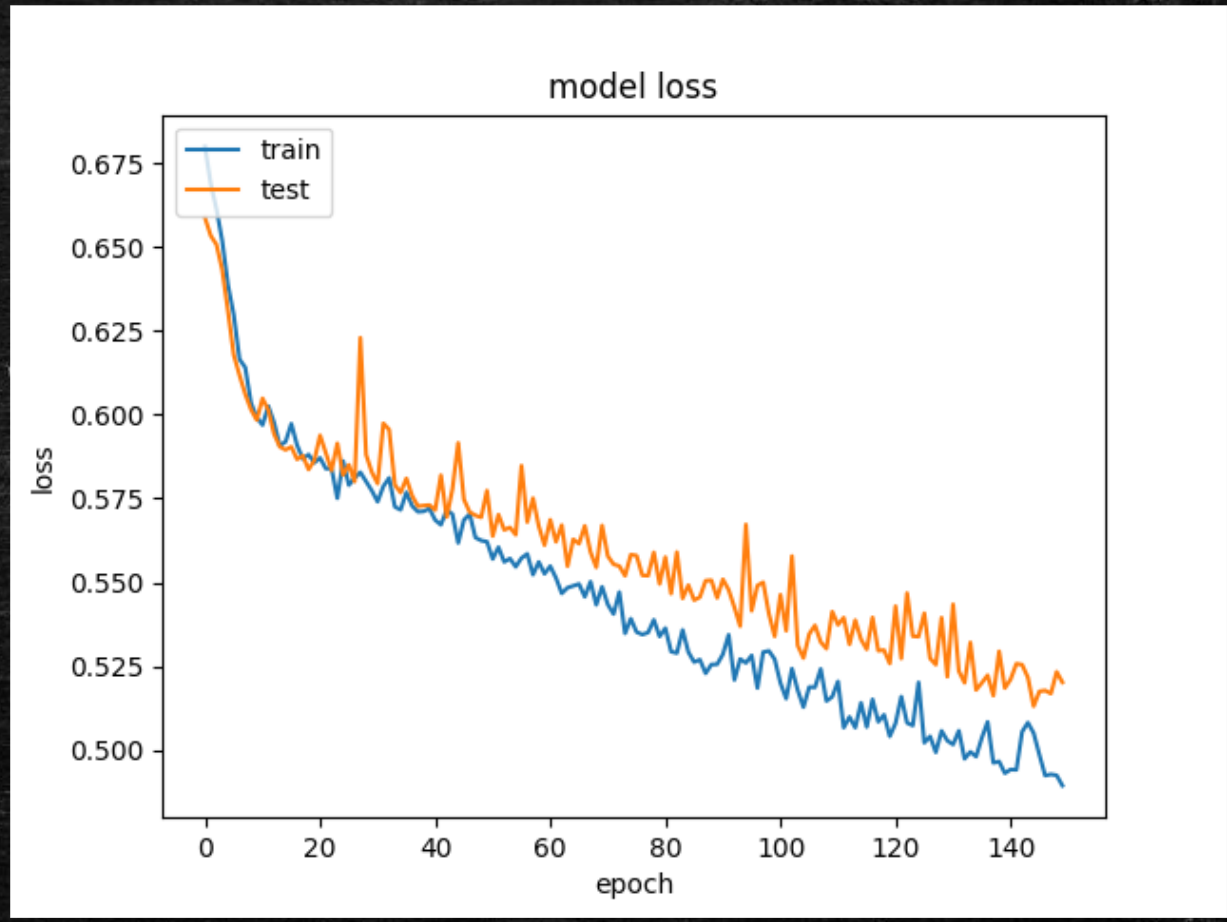
También podemos ver que el modelo aún no ha sobreaprendido el conjunto de datos de entrenamiento, mostrando habilidad comparables en ambos conjuntos de datos.



Historia de entrenamiento

Visualizar el historial de entrenamiento del modelo en Keras (cont.)

A partir de la gráfica de pérdida (loss), podemos ver que el modelo tiene un rendimiento comparable tanto en los datos de entrenamiento y validación. Si estas gráficas paralelas comienzan a desviarse de manera constante, podría ser una señal de dejar de entrenar en una época anterior.



Reducción de sobreajuste con Dropout

Una técnica de regularización simple y poderosa para redes neuronales y modelos de aprendizaje profundo es Dropout. Se puede aplicar la técnica de regularización Dropout en modelos en Python con Keras. Es importante saber:

- Cómo funciona la técnica de regularización Dropout.
- Cómo utilizar Dropout en las capas de entrada.
- Cómo utilizar Dropout en las capas ocultas.

Reducción de sobreajuste con Dropout

Una técnica de regularización simple y poderosa para redes neuronales y modelos de aprendizaje profundo es Dropout. Se puede aplicar la técnica de regularización Dropout en modelos en Python con Keras. Es importante saber:

- Cómo funciona la técnica de regularización Dropout.
- Cómo utilizar Dropout en las capas de entrada.
- Cómo utilizar Dropout en las capas ocultas.

Reducción de sobreajuste con Dropout

Dropout es una técnica de regularización para modelos de redes neuronales propuesta por Srivastava en su artículo de 2014 "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Dropout es una técnica en la que las neuronas seleccionadas al azar se ignoran durante el entrenamiento. Se "eliminan" al azar. Esto significa que su contribución a la activación de las neuronas que se encuentran más arriba se eliminan temporalmente en el pase hacia adelante y las actualizaciones de pesos no se aplican a la neurona en el pase hacia atrás.

Reducción de sobreajuste con Dropout

A medida que aprende una red neuronal, los pesos de las neuronas se asientan en su contexto dentro de la red. Los pesos de las neuronas se ajustan para características específicas que proporcionan cierta especialización. Las neuronas vecinas llegan a depender de esta especialización, que si se lleva demasiado lejos puede resultar en un modelo frágil demasiado especializado para los datos de entrenamiento. Esta dependencia del contexto de una neurona durante el entrenamiento se conoce como coadaptaciones complejas.

Reducción de sobreajuste con Dropout

Ahora puede imaginar que si las neuronas se eliminan aleatoriamente de la red durante el entrenamiento, otras neuronas tendrán que intervenir y manejar la representación requerida para hacer predicciones para las neuronas faltantes. Se cree que esto da como resultado que la red aprenda múltiples representaciones internas independientes.

El efecto es que la red se vuelve menos sensible a los pesos específicos de las neuronas. Esto, a su vez, da como resultado una red que es capaz de una mejor generalización y es menos probable que se ajuste en exceso a los datos de entrenamiento.

Reducción de sobreajuste con Dropout

Regularización Dropout en Keras

Dropout se implementa fácilmente seleccionando aleatoriamente los nodos que se "eliminarán" con una probabilidad determinada (por ejemplo, 20%) en cada ciclo de actualización de peso. Así es como se implementa Dropout en Keras. Dropout solo se usa durante el entrenamiento de un modelo y no se usa al evaluar la habilidad del modelo.

Reducción de sobreajuste con Dropout

Dropout en la capa visible

Dropout se puede aplicar a las neuronas de entrada, es decir en la capa visible. Por ejemplo, se agrega una nueva capa Dropout entre la entrada (o capa visible) y la primera capa oculta. La tasa de dropout se establece a un valor, por ejemplo 20%, lo que significaría que una de cada cinco entradas se excluirá aleatoriamente de cada ciclo de actualización.

Reducción de sobreajuste con Dropout

Dropout en la capa visible (cont.)

Además, como se recomienda en el paper original sobre Dropout, se podría imponer una restricción en los pesos para cada capa oculta, asegurando que la norma máxima de los pesos no exceda un valor de 3. Esto se hace estableciendo el argumento kernel_constraint en el Clase densa al construir las capas. La tasa de aprendizaje y el momento se pueden elevar. Estos aumentos en la tasa de aprendizaje también se recomiendan en el paper original.

Reducción de sobreajuste con Dropout

Dropout en capas ocultas

Dropout se puede aplicar a neuronas ocultas en el modelo de red. Por ejemplo, dropout se puede aplicar entre las dos capas ocultas y entre la última capa oculta y la capa de salida

Reducción de sobreajuste con Dropout

Recomendaciones para usar Dropout

El paper sobre Dropout proporciona resultados experimentales sobre un conjunto de problemas estándar de aprendizaje automático. Como resultado, proporcionan una serie de heurísticas útiles a considerar cuando se usa dropout en la práctica:

- Por lo general, use un valor de dropout pequeño, del 20% al 50% de las neuronas, el 20% constituye un buen punto de partida. Una probabilidad demasiado baja tiene un efecto mínimo y un valor demasiado alto da como resultado un bajo aprendizaje por parte de la red.

Reducción de sobreajuste con Dropout

Recomendaciones para usar Dropout (cont)

- Utilice una red más grande. Es probable que obtenga un mejor rendimiento cuando se utiliza dropout en una red más grande, lo que le da al modelo una mayor oportunidad para aprender representaciones independientes.
- Utilice dropout en la entrada (visible) y las capas ocultas. La aplicación del dropout en cada capa de la red ha mostrado buenos resultados.

Reducción de sobreajuste con Dropout

Recomendaciones para usar Dropout (cont)

- Utilice una tasa de aprendizaje grande con decadencia y un momento grande. Aumente su tasa de aprendizaje en un factor de 10 a 100 y utilice un valor de momento alto de 0,9 o 0,99.
- Restrinja el tamaño de los pesos de la red. Una tasa de aprendizaje alta puede resultar en pesos de red muy grandes. Se ha demostrado que la imposición de una restricción en el tamaño de los pesos de la red, como la regularización de max-norm con un tamaño de 4 o 5, mejora los resultados.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Entrenar una red neuronal o un gran modelo de aprendizaje profundo es una tarea de optimización difícil. El algoritmo clásico para entrenar redes neuronales se llama descenso de gradiente estocástico. Está bien establecido que puede lograr un mayor rendimiento y un entrenamiento más rápido en algunos problemas utilizando una tasa de aprendizaje que cambia durante el entrenamiento.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Se pueden usar diferentes tasas de aprendizaje para sus modelos de redes neuronales en Python usando Keras. Es importante saber:

- El beneficio de la planificación de la tasa de aprendizaje sobre el rendimiento del modelo durante el entrenamiento.
- Cómo configurar y evaluar una planificación de tasa de aprendizaje basado en el tiempo.
- Cómo configurar y evaluar una planificación de tasa de aprendizaje basado en la disminución.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Adaptar la tasa de aprendizaje para su procedimiento de optimización de descenso de gradiente estocástico puede aumentar el rendimiento y reducir el tiempo de entrenamiento. A veces, esto se denomina recocido de la tasa de aprendizaje o tasas de aprendizaje adaptativo. Aquí llamaremos a este enfoque, planificación de tasa de aprendizaje, donde el programa predeterminado es usar una tasa de aprendizaje constante para actualizar los pesos de la red para cada época de entrenamiento.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

La adaptación más simple y quizás más utilizada de las tasas de aprendizaje durante el entrenamiento son las técnicas que reducen la tasa de aprendizaje con el tiempo. Estos tienen la ventaja de realizar grandes cambios al comienzo del procedimiento de entrenamiento cuando se utilizan valores de tasa de aprendizaje más grandes, y disminuir la tasa de aprendizaje de manera que se realicen una tasa más pequeña y, por lo tanto, actualizaciones de entrenamiento más pequeñas a los pesos más adelante en el procedimiento de entrenamiento. Esto tiene el efecto de aprender rápidamente buenos pesos temprano y ajustarlos después.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Dos planificaciones de tasas de aprendizaje populares y fáciles de usar son las siguientes:

- Disminución de la tasa de aprendizaje gradualmente según la época.
- Disminución de la tasa de aprendizaje utilizando grandes disminuciones en épocas específicas.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Planificación de la tasa de aprendizaje basada en el tiempo

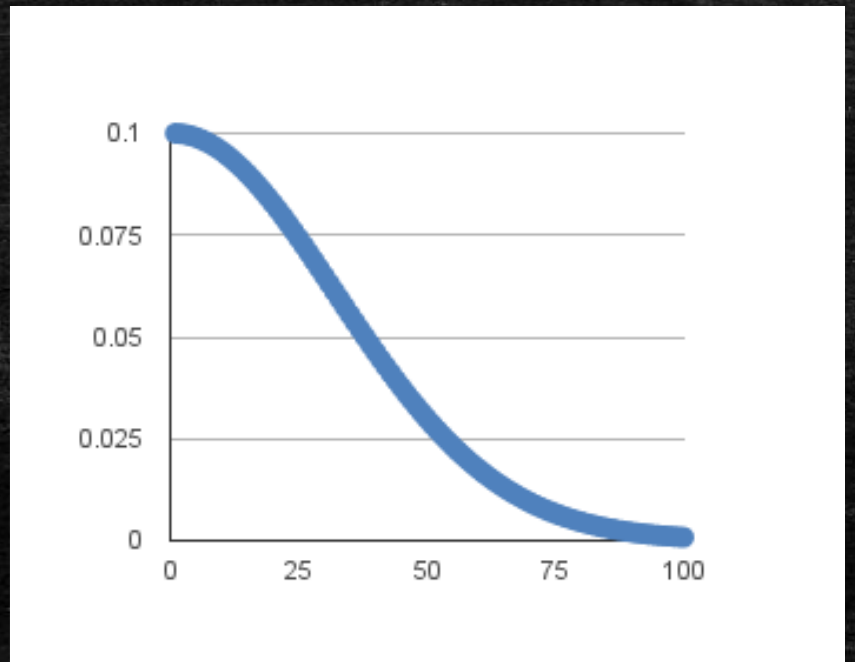
Keras tiene incorporado un programa de tasa de aprendizaje basado en el tiempo. La implementación del algoritmo de optimización de descenso de gradiente estocástico en la clase SGD tiene un argumento llamado decaimiento. Este argumento se utiliza en la ecuación para la disminución de la tasa de aprendizaje basada en el tiempo de la siguiente manera:

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Planificación de la tasa de aprendizaje basada en el tiempo (cont.)

$$TasaDeApr = TasaDeApr \frac{1}{1 + decaimiento * época}$$

Sugerencia:
Decaimiento = TasaDeApr / #Épocas



Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

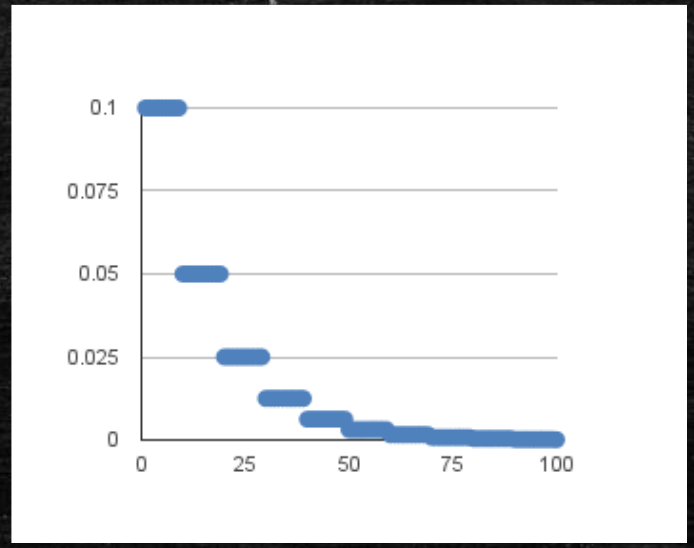
Planificación de la tasa de aprendizaje basada en la disminución

Otra planificación de la tasa de aprendizaje popular que se utiliza con los modelos de aprendizaje profundo es reducir sistemáticamente la tasa de aprendizaje en momentos específicos durante el entrenamiento. A menudo, este método se implementa reduciendo la tasa de aprendizaje a la mitad cada número fijo de épocas. Por ejemplo, podemos tener una tasa de aprendizaje inicial de 0,1 y reducirla en un factor de 0,5 cada 10 épocas. Las primeras 10 épocas de entrenamiento usarían un valor de 0.1, en las siguientes 10 épocas se usaría una tasa de aprendizaje de 0.05, y así sucesivamente.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Planificación de la tasa de aprendizaje basada en la disminución (cont.)

Si trazamos las tasas de aprendizaje para este ejemplo en 100 épocas, obtendrá el siguiente gráfico que muestra la tasa de aprendizaje (eje y) frente a la época (eje x).



Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Planificación de la tasa de aprendizaje basada en la disminución (cont.)

Podemos implementar esto en Keras usando LearningRateScheduler al ajustar el modelo. LearningRateScheduler nos permite definir una función que toma el número de épocas como argumento y devuelve la tasa de aprendizaje para usar en el descenso de gradiente estocástico. Cuando se utiliza, se ignora la tasa de aprendizaje especificada por el descenso del gradiente estocástico.

$$LearningRate = InitialLearningRate \times DropRate^{floor(\frac{1+Epoch}{EpochDrop})}$$

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Consejos para la planificación de la tasa de aprendizaje

- **Incrementar la tasa de aprendizaje inicial**. Debido a que la tasa de aprendizaje disminuirá, comience con un valor mayor desde el cual disminuir. Una tasa de aprendizaje mayor dará como resultado cambios mucho mayores en los pesos, al menos al principio, lo que le permitirá beneficiarse de un ajuste fino más adelante.
- **Utilice un gran momento**. El uso de un valor de momento mayor ayudará a que el algoritmo de optimización continúe realizando actualizaciones en la dirección correcta cuando su tasa de aprendizaje se reduzca a valores pequeños.

Planificación del ajuste de la tasa de aprendizaje para mejorar el desempeño

Consejos para la planificación de la tasa de aprendizaje

- **Experimente con diferentes planificaciones**. No estará claro qué planificación de aprendizaje usar, así que pruebe algunos con diferentes opciones de configuración y vea cuál funciona mejor en su problema. Pruebe también planificaciones que cambien exponencialmente e incluso planificaciones que respondan a la precisión de su modelo en los conjuntos de datos de entrenamiento o prueba.