Aprendizaje automático Modelado de canalizaciones en producción

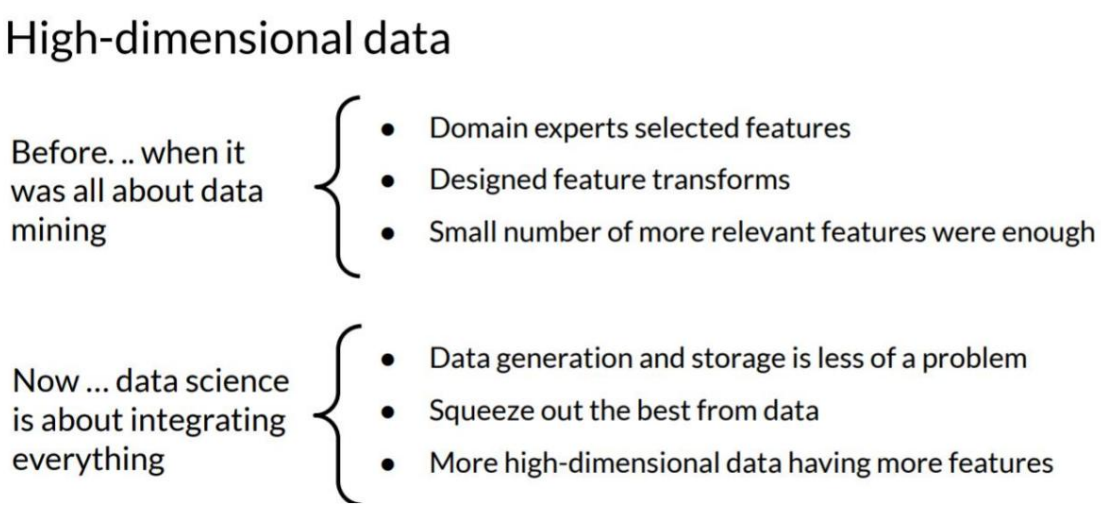
En el tercer curso de Ingeniería de aprendizaje automático para la especialización en producción, construirá modelos para diferentes entornos de servicio; implementar herramientas y técnicas para administrar de manera efectiva sus recursos de modelado y atender mejor las solicitudes de inferencia fuera de línea y en línea; y utilice herramientas de análisis y métricas de rendimiento para abordar la imparcialidad del modelo, los problemas de explicabilidad y mitigar los cuellos de botella.

Comprender los conceptos de aprendizaje automático y aprendizaje profundo es esencial, pero si está buscando desarrollar una carrera de inteligencia artificial efectiva, también necesita capacidades de ingeniería de producción. La ingeniería de aprendizaje automático para producción combina los conceptos fundamentales del aprendizaje automático con la experiencia funcional del desarrollo de software moderno y los roles de ingeniería para ayudarlo a desarrollar habilidades listas para la producción.

Semana 2: Técnicas modelo de gestión de recursos

Contenidos

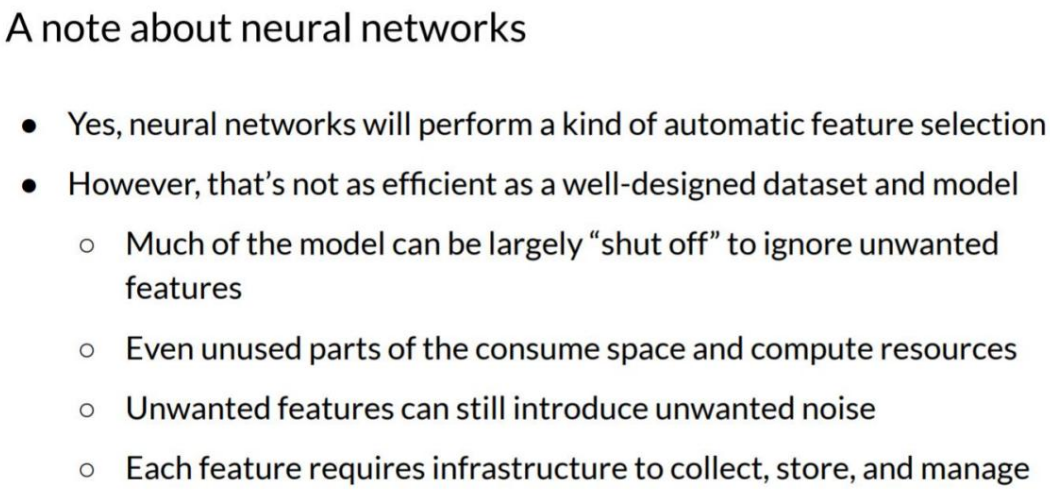
Reducción de dimensionalidad



- Los sistemas de cómputo, almacenamiento y IOT que requiere su modelo determinarán cuánto cuesta poner su modelo en producción y mantenerlo durante toda su vida útil. Esta semana analizaremos algunas técnicas importantes que pueden ayudarnos a administrar los requisitos de recursos
- del modelo. Comenzaremos analizando la dimensionalidad y cómo afecta el rendimiento de nuestros modelos y los requisitos de recursos. En un pasado no muy lejano, la generación de datos y, hasta cierto punto, el almacenamiento de datos era mucho más costoso de lo que es hoy. En
- aquel entonces, muchos expertos en dominios consideraban cuidadosamente qué características o variables medir antes de diseñar sus experimentos y transformaciones de características.
-

- Como resultado, se esperaba que los conjuntos de datos estuvieran bien diseñados y potencialmente contuvieran solo una pequeña cantidad de características relevantes.
- Hoy en día, la ciencia de datos tiende a tratarse más de integrar todo de principio a fin, generar y almacenar datos es cada vez más rápido, más fácil y menos costoso. - Entonces, hay una tendencia de que las personas midan todo lo que puedan e incluyan características cada vez más complejas.

  transformaciones.
- Como resultado, los conjuntos de datos a menudo tienen muchas dimensiones y contienen una gran cantidad de características, aunque la relevancia de cada característica para analizar estos datos no siempre está clara.

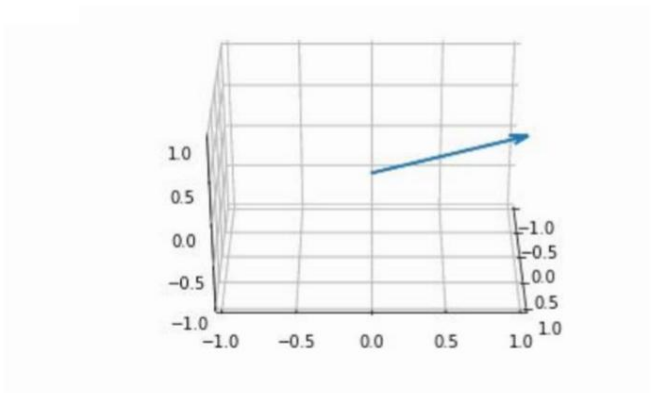


- Antes de profundizar, analicemos un concepto erróneo común sobre las redes neuronales. Muchos desarrolladores asumen correctamente que cuando entrenan sus modelos de redes neuronales, el propio modelo, como parte del proceso de entrenamiento, aprenderá a ignorar las funciones que no brindan información predictiva al reducir sus pesos a cero o cerca de cero.

- Bueno, si bien esto es cierto, el resultado no es un modelo eficiente.
- Gran parte del modelo puede terminar apagándose cuando se ejecuta la inferencia para generar predicciones, pero esos las partes no utilizadas del modelo todavía están allí.
- Ocupan espacio y consumen recursos informáticos a medida que el servidor modelo atraviesa el gráfico de cálculo. Esas funciones no deseadas también podrían introducir ruido no deseado en los datos, lo que puede degradar el rendimiento del modelo.

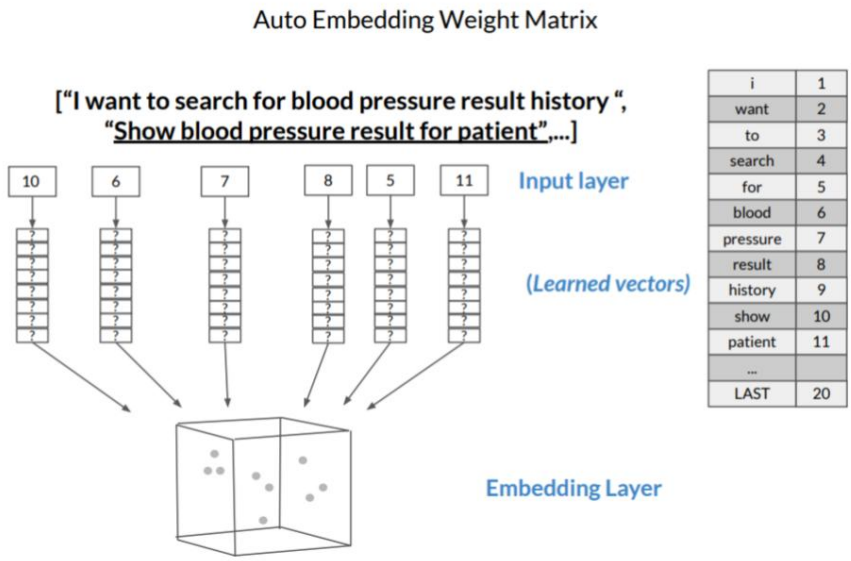
- Y fuera del modelo en sí, cada función adicional aún requiere sistemas e infraestructura para recopilar esos datos, almacenarlos, administrar actualizaciones, etc., lo que agrega costos y complejidad al sistema general. - Eso incluye el monitoreo de problemas con los datos y el esfuerzo para solucionar esos problemas si y cuando suceder.
- Esos costos continúan durante la vida útil del producto o servicio que está implementando, lo que podría ser fácilmente años.
- Existen técnicas para optimizar modelos con pesos cercanos a cero, que aprenderá más más adelante en este curso.
- Pero, en general, no debe limitarse a arrojar todo sobre su modelo y confiar en su proceso de entrenamiento para determinar qué características son realmente útiles.

High-dimensional spaces



- En el aprendizaje automático, a menudo tenemos que lidiar con datos de alta dimensión.
- Considere un ejemplo en el que tratamos de registrar 60 métricas diferentes para cada uno de nuestros compradores, lo que significa que estamos trabajando en un espacio con 60 dimensiones.
- En otros casos, si está tratando de analizar imágenes en escala de grises de 50 por 50, está trabajando en un área con 2500 dimensiones. Si las dimensiones o si las imágenes son más bien RGB, la dimensionalidad aumenta a 7500 dimensiones.
- En este caso tenemos una dimensión para cada canal de color en cada píxel de la imagen.

Word embedding - An example



- Algunas representaciones de características, como una codificación en caliente, son problemáticas para trabajar con texto en espacios de gran dimensión, ya que tienden a producir representaciones muy dispersas que no se escalan bien. - Una forma de superar esto es usar una capa de incrustación que tokenice las oraciones y asigne un flotante valor a cada palabra.
- Esto conduce a una representación vectorial más poderosa que respeta el tiempo y la secuencia de las palabras en una oración dada. Esta representación se puede aprender automáticamente durante el entrenamiento. La capa de incrustación etiquetada como cubo en esta figura es una representación conceptual de esos vectores en un espacio dimensional alto.

Initialization and loading the dataset

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
from keras.datasets import reuters
from keras.preprocessing import sequence
num_words = 1000

(reuters_train_x, reuters_train_y), (reuters_test_x, reuters_test_y) =
                    tf.keras.datasets.reuters.load_data(num_words=num_words)
n_labels = np.unique(reuters_train_y).shape[0]
```

- Veamos un ejemplo concreto de incrustación de word usando Keras. Comencemos cargando las bibliotecas y módulos necesarios para encontrar algunos parámetros importantes.
- El conjunto de datos de noticias de Reuters con el que trabajaremos contiene 11.228 noticias etiquetadas sobre 46 temas.
- Los documentos ya están codificados de tal manera que cada palabra está indexada por un número entero, su total frecuencia en el conjunto de datos.
- Mientras cargamos el conjunto de datos, especificamos la cantidad de palabras con las que trabajaremos (1000), de modo que la menor las palabras repetidas se consideran desconocidas.

Further preprocessing

```
from keras.utils import np_utils
reuters_train_y = np_utils.to_categorical(reuters_train_y, 46)
reuters_test_y = np_utils.to_categorical(reuters_test_y, 46)

reuters_train_x =
     tf.keras.preprocessing.sequence.pad_sequences(reuters_train_x, maxlen=20)
reuters_test_x = tf.keras.preprocessing.sequence.pad_sequences(reuters_test_x,
                 maxlen=20)
```

- Preprocesemos aún más los datos, de modo que estén listos para entrenar un modelo.
- Primero, esto convierte el vector de entrenamiento Y en una variable categórica tanto para entrenar como para probar. A continuación, el código segmenta el texto de entrada en secuencias largas de 20 palabras.

Using all dimensions

```
from tensorflow.keras import layers
model2 = tf.keras.Sequential(
    [
      layers.Embedding(num_words, 1000, input_length= 20),
      layers.Flatten(),
      layers.Dense(256),
      layers.Dropout(0.25),
      layers.Activation('relu'),
      layers.Dense(46),
      layers.Activation('softmax')
    ])
```

- Construir la red es el siguiente paso lógico. Así que aquí la opción es incrustar un vocabulario de 1000 palabras usando todas las dimensiones, aquí estamos usando todas las dimensiones de los datos.

- La última capa es densa con dimensión 46 ya que la variable objetivo es un vector de categorías de 46 dimensiones.

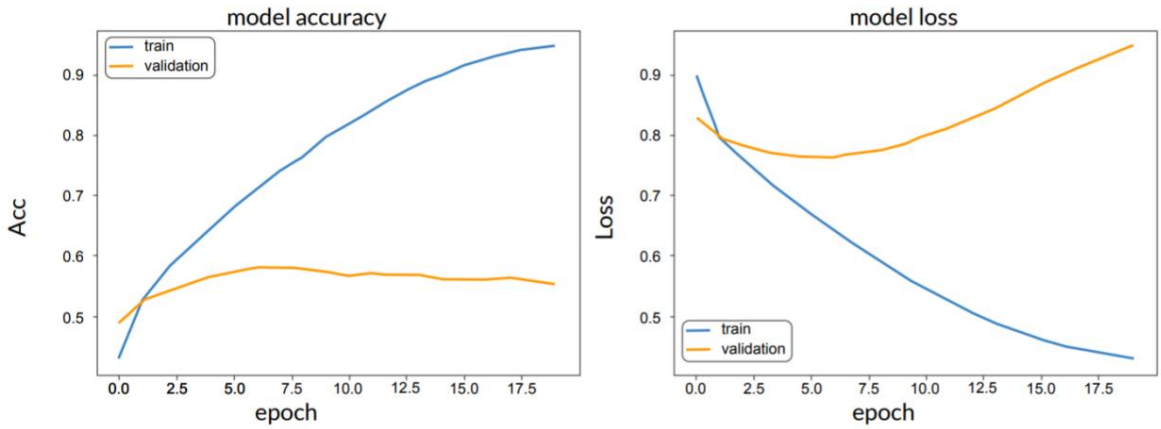
Model compilation and training

```
model.compile(loss="categorical_crossentropy", optimizer="rmsprop",
metrics=['accuracy'])


model_1 = model.fit(reuters_train_x, reuters_train_y,
                    validation_data=(reuters_test_x , reuters_test_y),
                    batch_size=128, epochs=20, verbose=0)
```

- Con la estructura del modelo lista, compilemos el modelo especificando el optimizador de pérdida y la métrica de salida.

- Para este problema, las opciones naturales son pérdida de entropía cruzada categórica, optimización rmsprop y precisión es la métrica.
- Ahora todo está listo para hacer un ajuste de modelo. Especificaremos el conjunto de validación, el tamaño del lote y el número de épocas. para entrenamiento.

Example with a higher number of dimensions

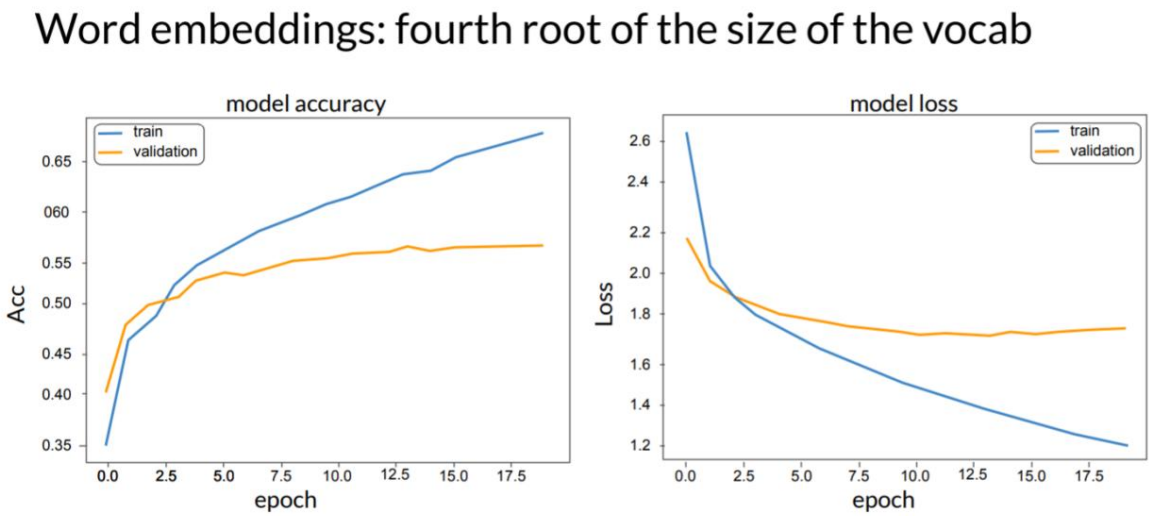


- Aquí está la precisión en función de las épocas de entrenamiento y la pérdida del modelo en función de las épocas de entrenamiento.
- Tenga en cuenta que después de aproximadamente dos épicas, los datos de entrenamiento dan como resultado precisiones significativamente más altas y pérdidas más bajas en comparación con el conjunto de validación.
- Esta es una clara indicación de que el modelo está severamente sobreajustado, y esto puede ser el resultado de usar todos los dimensiones de los datos.
- Entonces, el modelo está captando matices en el conjunto de entrenamiento que no se generalizan bien.

Word embeddings: 6 dimensions

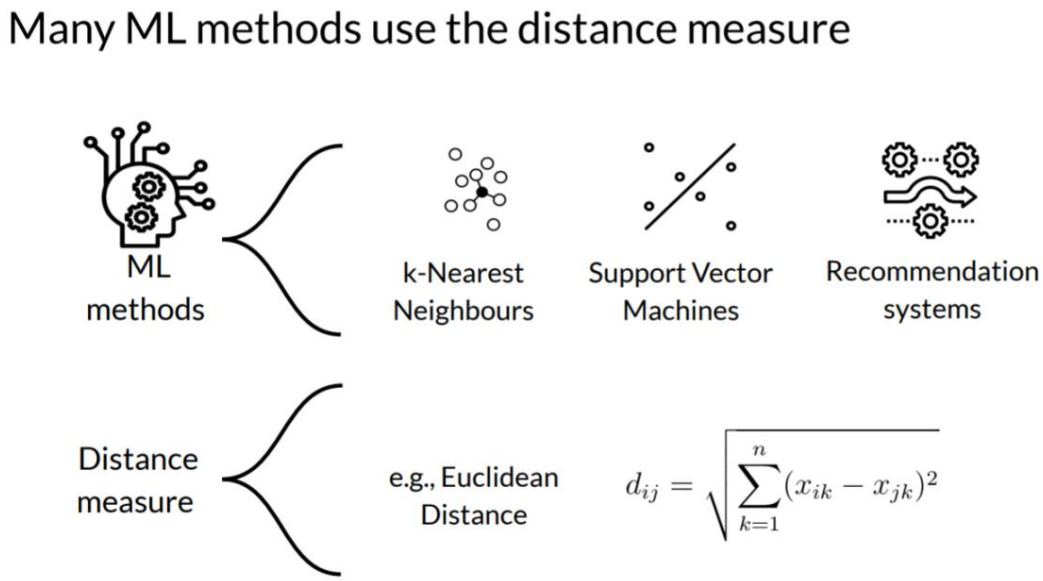
```
from tensorflow.keras import layers
model = tf.keras.Sequential(
    [
      layers.Embedding(num_words, 6, input_length= 20),
      layers.Flatten(),
      layers.Dense(256),
      layers.Dropout(0.25),
      layers.Activation('relu'),
      layers.Dense(46),
      layers.Activation('softmax')
    ])
```

- Intentemos reducir la dimensionalidad y veamos cómo afecta esto al rendimiento del modelo. Para eso vamos a insertar un Vocabulario de 1000 palabras en 6 dimensiones, esto es aproximadamente una reducción de un cuarto factor raíz.



- El modelo permanece sin cambios por lo demás. Todavía puede haber algo de sobreajuste, pero con ese cambio, este modelo funciona significativamente mejor que la versión de 1000 dimensiones.

Maldición de dimensionalidad



- Hablemos de la maldición de la dimensionalidad y por qué este es un tema muy importante al construir modelos. – - Muchas tareas comunes de aprendizaje automático, como la segmentación y el agrupamiento, se basan en el cálculo de distancias entre observaciones.
- Por ejemplo, la clasificación supervisada utiliza la distancia entre las observaciones para asignar una clase, la más cercana vecinos es un ejemplo básico de esto.
- Las máquinas de vectores de soporte o SVM se ocupan de proyectar observaciones utilizando núcleos basados en la distancia entre las observaciones después de la proyección.
- Otro ejemplo son los sistemas de recomendación que utilizan una medida de similitud basada en la distancia entre el usuario y los vectores de atributos del artículo. Incluso podría haber otras formas de distancia que se utilicen. - Entonces, la distancia juega un papel importante en la comprensión de la dimensionalidad. - Una de las métricas de distancia más comunes es la distancia euclidiana, que es simplemente una distancia lineal entre dos puntos en un espacio multidimensional.
- La distancia euclidiana entre dos vectores dimensionales con coordenadas cartesianas se calcula utilizando este fórmula familiar.

Why is high-dimensional data a problem?

- More dimensions → more features
- Risk of overfitting our models
- Distances grow more and more alike
- No clear distinction between clustered objects
- Concentration phenomenon for Euclidean distance

- Pero, ¿por qué es importante la distancia? Veamos algunos problemas relacionados con la medición de distancias en espacios de grandes dimensiones.
- Por ejemplo, es posible que se pregunte por qué los datos de gran dimensión pueden ser un problema.
- En casos extremos donde tenemos más características y observaciones, corremos el riesgo de sobreajustar masivamente nuestro modelo.

- Pero en el caso más general, cuando tenemos demasiadas características, las observaciones se vuelven más difíciles de agrupar. - Demasiadas dimensiones pueden hacer que cada observación en su conjunto de datos parezca equidistante de todas las demás.
- Y porque la agrupación usa medidas de distancia como la distancia euclidiana para cuantificar la similitud entre observaciones, este es un gran problema.
- Si las distancias son todas aproximadamente iguales, todas las observaciones aparecen igualmente parecidas y no se pueden formar grupos significativos .
- A medida que crece la dimensionalidad, disminuye el contraste que proporcionan las métricas habituales. En otras palabras, la distribución de normas en una determinada distribución de puntos tiende a concentrarse. Eso puede causar un comportamiento inesperado en espacios de alta dimensión.

Curse of dimensionality

"As we add more dimensions we also increase the processing power we need to train the model and make predictions, as well as the amount of training data required"

Badreesh Shetty

- Este fenómeno se llama la maldición de la dimensionalidad. Incluye
- situaciones en las que se observan propiedades no intuitivas de los datos en estos espacios de gran dimensión. Esto está específicamente relacionado con la usabilidad e interpretación de distancias y volúmenes.
- Cuando se trata de la maldición de la dimensionalidad, hay dos cosas a considerar. - Por un lado, el aprendizaje automático es bueno para analizar datos cuando se trata de muchas dimensiones. - Sin embargo, los humanos no somos expertos en encontrar patrones y datos que pueden estar dispersos en varios dimensiones, especialmente si esas dimensiones están interrelacionadas de manera contraria a la intuición.
- Por otro lado, a medida que agregamos más dimensiones, también aumentamos la potencia de procesamiento que necesitamos para analizar los datos y, al mismo tiempo, también aumentamos la cantidad de datos de entrenamiento necesarios para crear modelos significativos.

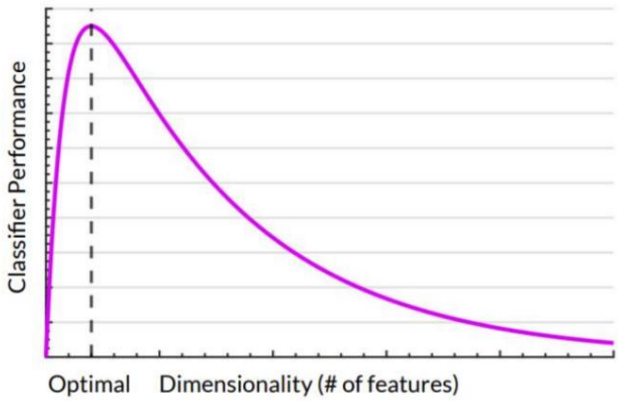
- Si tiene curiosidad, Richard Bellman acuñó por primera vez el término maldición de la dimensionalidad hace más de medio siglo en 1961 en su libro Adaptive Control Processes, A Guided Tour.

Why are more features bad?

- Redundant / irrelevant features
- More noise added than signal
- Hard to interpret and visualize
- Hard to store and process data

- Entonces, agregar más funciones puede crear problemas fácilmente. Esto podría incluir características redundantes o irrelevantes que aparece en los datos.
- Además, se agrega ruido cuando las características no brindan poder predictivo para nuestros modelos. - Además de eso, más funciones hacen que sea más difícil para uno interpretar y visualizar datos.
- Finalmente, más funciones significan más datos, por lo que necesita tener más almacenamiento y más poder de procesamiento para procesarlos. En última instancia, tener más dimensiones a menudo significa que nuestro modelo es menos eficiente.
-

The performance of algorithms ~ the number of dimensions



- Cuando tiene problemas para que su modelo funcione, a menudo se siente tentado a intentar agregar más y más características.
- Pero a medida que agrega más funciones, llega a un cierto punto en el que el rendimiento de su modelo se degrada. Este gráfico lo muestra bien. - Aquí se ve que a medida que aumenta la dimensionalidad, aumenta el rendimiento de los clasificadores hasta alcanzar el número óptimo de características.

- Un punto clave que debe comprender aquí es que está aumentando la dimensionalidad sin aumentar la cantidad de muestras de entrenamiento y eso da como resultado una disminución constante en el rendimiento del clasificador después del óptimo. Veamos otro problema con la
- dimensionalidad para descubrir qué hay detrás de este comportamiento.

Adding dimensions increases feature space volume

1-D

1	2	3	4	5

2-D

(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)
(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)
(3, 1)	(3, 2)	(3, 3)	(3, 4)	(3, 5)
(4, 1)	(4, 2)	(4, 3)	(4, 4)	(4, 5)
(5, 1)	(5, 2)	(5, 3)	(5, 4)	(5, 5)

... ...

- Miremos más profundamente para tratar de entender por qué más dimensiones pueden dañar sus modelos.
- Comencemos por comprender por qué la cantidad de parámetros de una función afecta la dificultad de aprender esa función.

- Tomemos por ejemplo, los parámetros de una función de línea. En este caso la lista es finita. Simplemente significa discretizar Las características. Simplifiquemos esto usando números a partir del 15.
- Suponiendo que tiene una función con un solo parámetro, entonces solo hay cinco valores posibles que esto el parámetro puede tomar.
- ¿Qué pasa si agregas un segundo parámetro que también puede tomar cinco valores?
- Ahora hay 5 veces 5 o 25 pares posibles. Este es un ejemplo simple usando valores de parámetros discretos, pero por supuesto, es aún peor con variables continuas.
- ¿Qué pasa si tuvieras un tercer parámetro? Ahora que la representación es un cubo, probablemente veas la fórmula detrás de este razonamiento. Si tenemos n valores que puede tomar un parámetro y m parámetros, terminas con n a los m valores de parámetro posibles.

- El número de valores de parámetros crece exponencialmente. - ¿Qué tan grande es el problema? Bueno, es un gran problema, por eso se llama la maldición de la dimensionalidad.

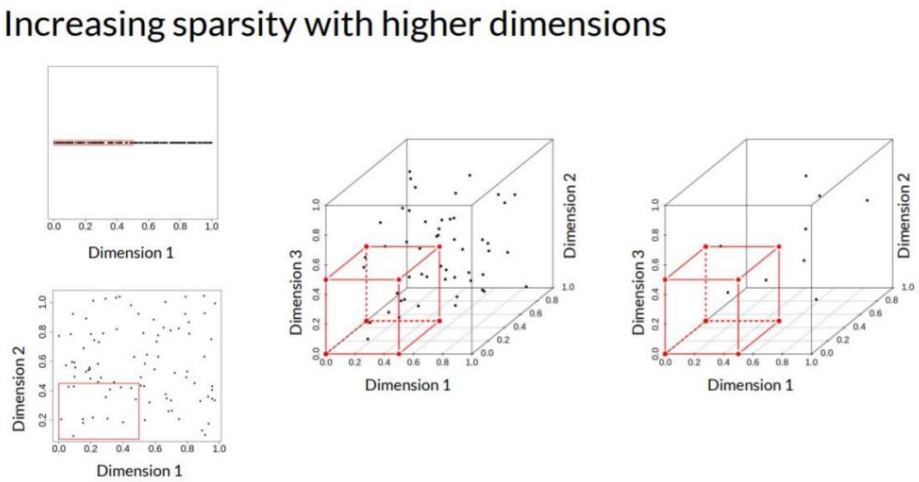
Curse of dimensionality in the distance function

Euclidean distance $$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

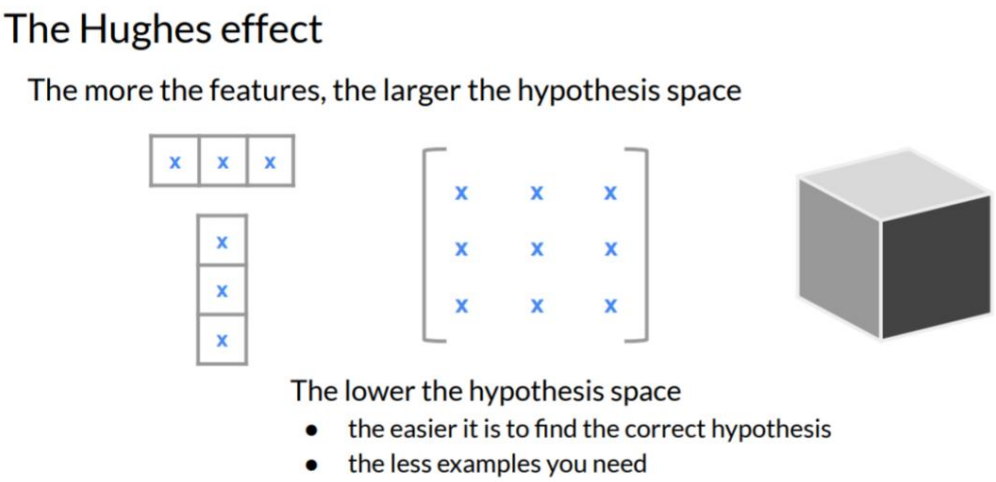
- New dimensions add non-negative terms to the sum
- Distance increases with the number of dimensions
- **For a given number of examples**, the feature space becomes increasingly sparse

- Un aumento en el número de dimensiones de un conjunto de datos significa que hay más entradas en el vector de características representando cada ejemplo de entrenamiento.
- Centrémonos en un espacio euclidiano y una medida de distancia euclidiana. Cada nueva dimensión agrega un término no negativo a la suma, por lo que la distancia aumenta con el número de dimensiones para distintos factores. En otras palabras, a medida que crece la
- cantidad de funciones para una cantidad determinada de ejemplos de entrenamiento, el espacio de funciones se vuelve cada vez más escaso con más distancia entre los ejemplos de entrenamiento. - Debido a eso, la menor densidad de datos requiere más ejemplos de entrenamiento para mantener igual la distancia promedio entre los puntos de datos.

- También es importante que los ejemplos que agregue sean significativamente diferentes de los ejemplos que ya tiene o que ya están presentes en la muestra.
- Aquí el argumento se construye utilizando la distancia euclidiana, pero es cierto para cualquier medida de distancia definida correctamente.



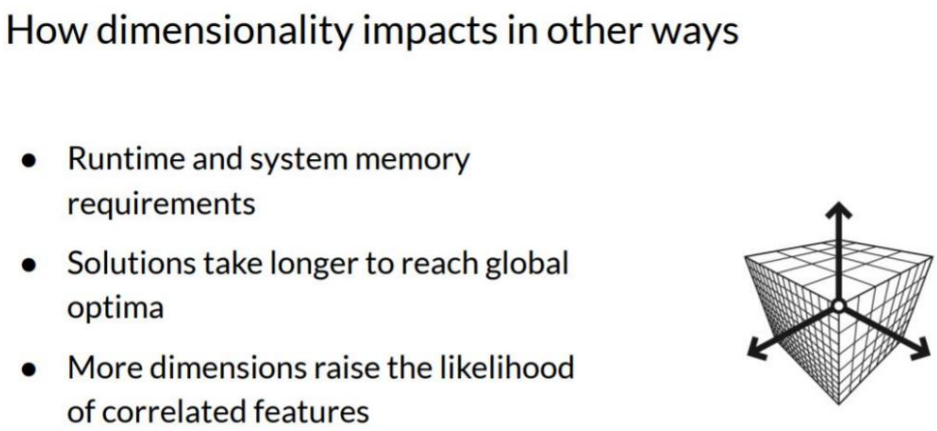
- Cuando crece la distancia entre las observaciones, el aprendizaje supervisado se vuelve más difícil porque es menos probable que las predicciones para nuevas muestras se basen en el aprendizaje de ejemplos de entrenamiento similares.
- El tamaño del espacio de funciones crece exponencialmente a medida que aumenta el número de funciones, lo que lo hace mucho más difícil generalizar eficientemente.
- La varianza aumenta y hay una mayor posibilidad de sobreajuste al ruido en más dimensiones, lo que da como resultado bajo rendimiento de generalización. En la
- práctica, las características también pueden estar correlacionadas o no mostrar mucha variación.
- Por estas razones, existe la necesidad de reducir la dimensionalidad. El desafío es mantener la mayor cantidad posible de información predictiva utilizando la menor cantidad de funciones posible.



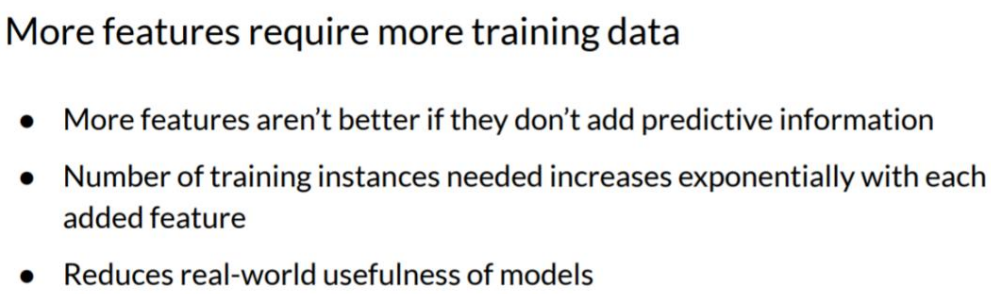
- Independientemente del enfoque de modelado que esté utilizando, aumentar la dimensionalidad tiene otro problema especialmente para la clasificación que se conoce como el efecto Hughes. Este es un
- fenómeno que demuestra la mejora en el rendimiento de la clasificación a medida que aumenta el número de funciones hasta que alcanzamos un punto óptimo en el que tenemos suficientes funciones.
- Agregar más funciones mientras se mantiene el conjunto de entrenamiento del mismo tamaño degradará el rendimiento de los clasificadores. Vimos esto anteriormente en nuestro gráfico.
- En la clasificación, el objetivo es encontrar una función que discrimine entre dos o más clases.
- Puedes hacer esto buscando hiperplanos en el espacio que separen estas categorías.

- Cuantas más dimensiones tenga, más fácil será encontrar un hiperplano durante el entrenamiento, pero al mismo tiempo más difícil será igualar ese rendimiento al generalizar a datos no vistos. - Y cuantos menos datos de entrenamiento tenga, menos seguro estará de identificar las dimensiones importantes para discriminar entre categorías.

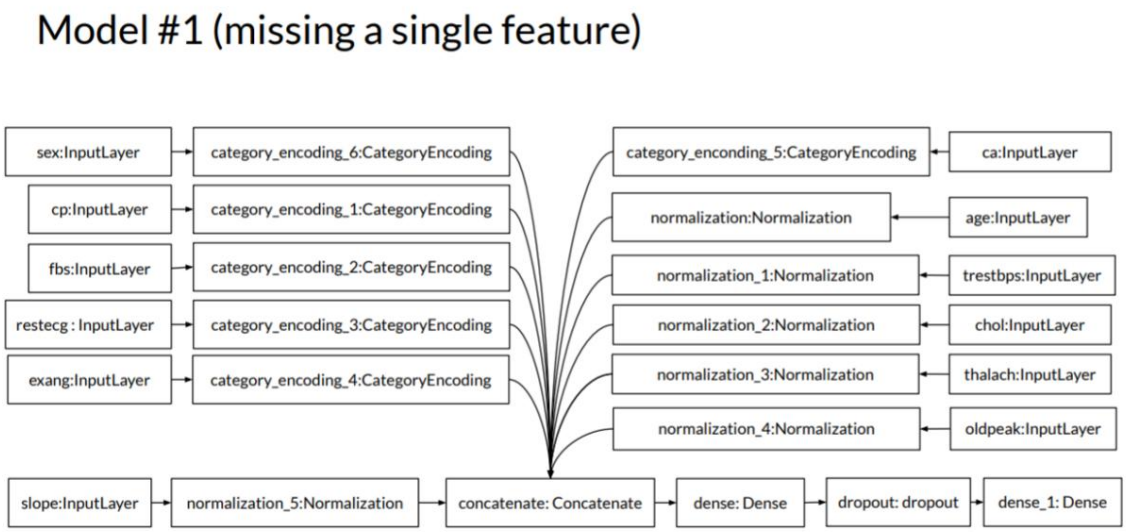
Maldición de la dimensionalidad: un ejemplo



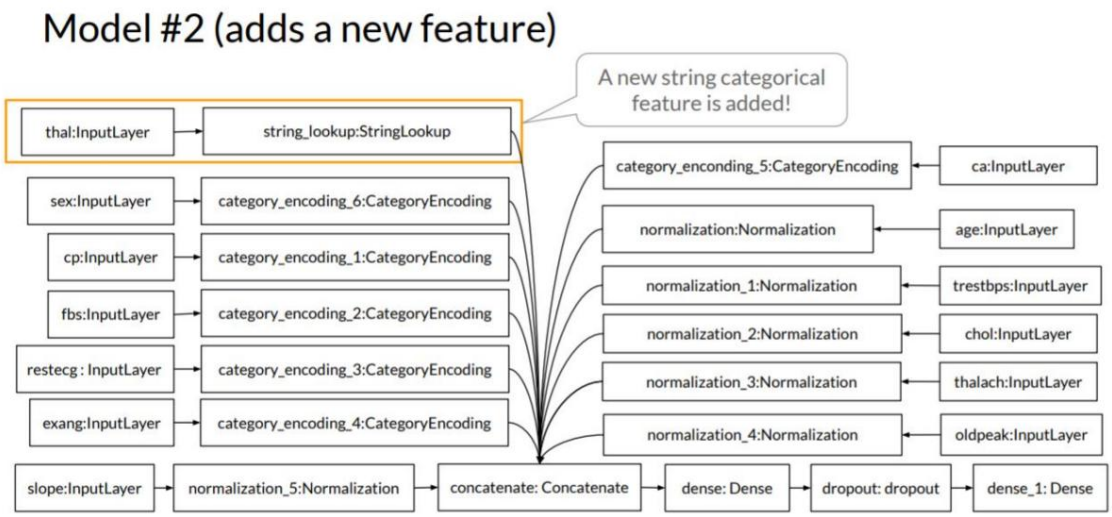
- Veamos un ejemplo de cómo la reducción de la dimensionalidad puede ayudar a que nuestros modelos funcionen mejor
- Además de las distancias y los volúmenes, aumentar el número de dimensiones puede generar otros problemas.
- Los requisitos de memoria y procesador a menudo escalan de forma no lineal con un aumento en el número de dimensiones, debido a un aumento exponencial en las soluciones factibles, muchos métodos de optimización no pueden alcanzar un óptimo global y se atascan en un óptimo local. - Más dimensiones también suelen aumentar la probabilidad de tener características y parámetros correlacionados . la estimación a menudo puede ser un desafío en los modelos de regresión.
- Entonces, veamos cómo más funciones pueden hacer que entrenar un modelo sea más difícil, con un ejemplo



- Cuando crea un modelo, lo diseña para una cierta cantidad de funciones o dimensiones, es posible que tenga la tentación de agregar más funciones para obtener un mejor modelo, pero más funciones pueden dañar su modelo.
- Cada función contiene información que puede o no ayudar a su modelo a predecir con precisión
- A medida que agrega más y más funciones, debe agregar más y más ejemplos de capacitación a lo largo del rango de valores para esas características.
- La cantidad de datos de entrenamiento necesarios aumenta exponencialmente con cada función añadida. - Eso significa que el volumen de datos de entrenamiento aumenta exponencialmente, y debemos asegurarnos de que los datos de entrenamiento cubran las mismas regiones del espacio de características que las solicitudes de predicción que recibirá Todo esto puede reducir la capacidad de generalización de su modelo . Bueno, junto con esto, la cantidad de variables entrenables en el modelo también aumenta. Para demostrar esto, veamos un ejemplo de un modelo de clasificación binaria para el conjunto de datos de enfermedades cardíacas de Cleveland cuando se agrega una sola característica adicional.



- Comencemos por crear un modelo de clasificación estructurado para el conjunto de datos de enfermedades cardíacas de Cleveland. Este conjunto de datos tiene 14 características para predecir si un paciente tiene o no una enfermedad cardíaca. Este es un problema de clasificación
- binaria. Este primer modelo omite una de las características originales denominadas thal.



- Volvamos a agregar esa única característica adicional que eliminamos en el primer modelo.
- Para esto, codifiquemos esto como una característica de cadena categórica usando capas de preprocesamiento de Keras, que está disponible en TensorFlow.
- A continuación, veamos cómo su adición impacta en su modelo original en términos de la cantidad de parámetros entrenables.

Comparing the two models' trainable variables

```
from tensorflow.python.keras.utils.layer_utils import count_params

# Number of training parameters in Model #1
>>> count_params(model_1.trainable_variables)
    833


# Number of training parameters in Model #2 (with an added feature)
>>> count_params(model_1.trainable_variables)
    1057
```

- Si compara la cantidad de parámetros entrenables entre los dos modelos, incluso agregar solo una característica da como resultado un aumento del 27%.
- Eso es un crecimiento considerable en la cantidad de parámetros, por decir lo menos. Eso hará que el entrenamiento sea más lento y más costoso. También deberá aumentar el tamaño del conjunto de datos de entrenamiento, lo que hará que el entrenamiento sea aún
- más lento y costoso.

What do ML models need?

- No hard and fast rule on how many features are required
- Number of features to be used vary depending on
- Prefer uncorrelated data containing information to produce correct results

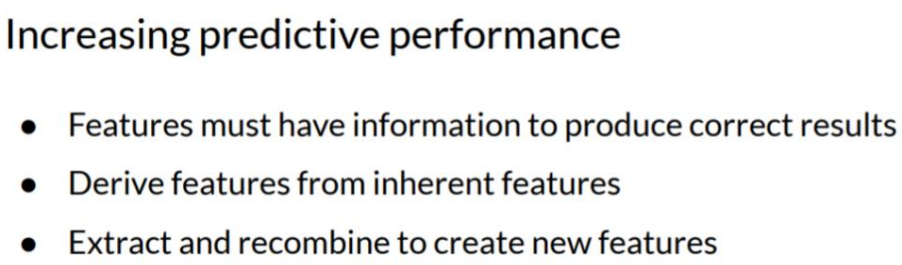
- El punto principal en esta sección es que cuando la dimensionalidad de los datos se vuelve demasiado grande, el desempeño de un clasificador disminuye y la demanda de recursos aumenta.

- La pregunta, es decir, ¿qué significa realmente demasiado grande? -

Desafortunadamente, no existe una regla fija sobre cuántas funciones se deben usar en un problema de aprendizaje automático. De hecho,

- esto depende de la cantidad de datos de entrenamiento disponibles, la varianza de esos datos, la complejidad de la superficie de decisión y el tipo de clasificador utilizado. También depende de qué funciones contienen realmente información predictiva que ayudará
- a entrenar su modelo. - Quiere suficientes datos con las mejores características y suficiente variedad en los valores de esas características y suficiente información predictiva en esas características para maximizar el rendimiento de su modelo mientras lo simplifica tanto como sea posible.

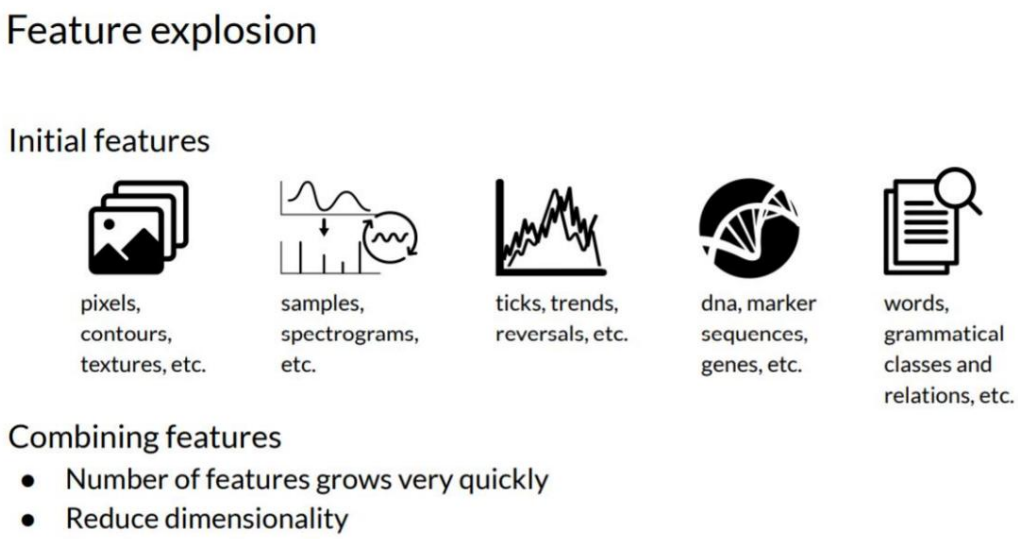
Reducción de dimensionalidad manual



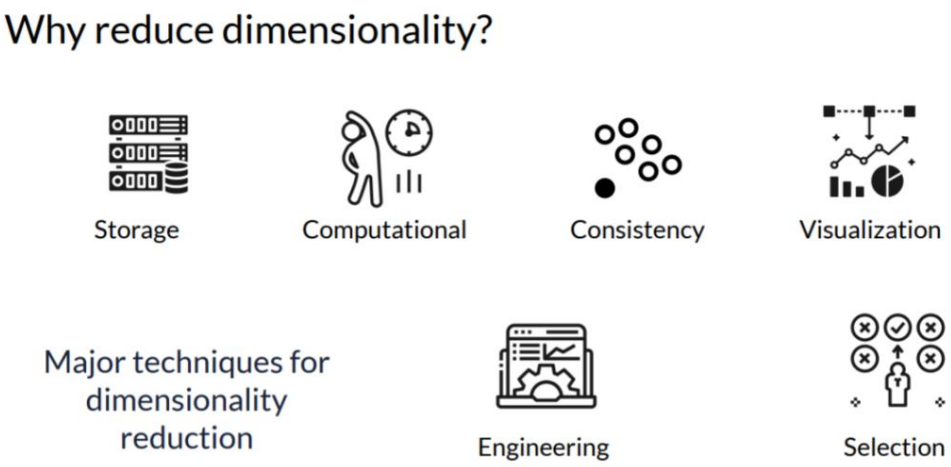
- Ahora que hemos discutido cómo sus datos tienen un impacto en el rendimiento de sus modelos y los recursos necesarios para entrenar y servir esos modelos, veamos algunas técnicas manuales para realizar la reducción de dimensionalidad.

- Al preprocesar un conjunto de características para crear un nuevo conjunto de características, es importante conservar la mayor cantidad de información predictiva posible. - Sin información predictiva, todos los datos del mundo no ayudarán a su modelo a aprender. - Las características deben ser representativas de la información predictiva del conjunto de datos. Esta información también debe estar en un formato que ayude a su modelo a aprender. - Si bien algunas características inherentes se pueden obtener directamente de los datos sin procesar, a menudo se necesitan características derivadas.

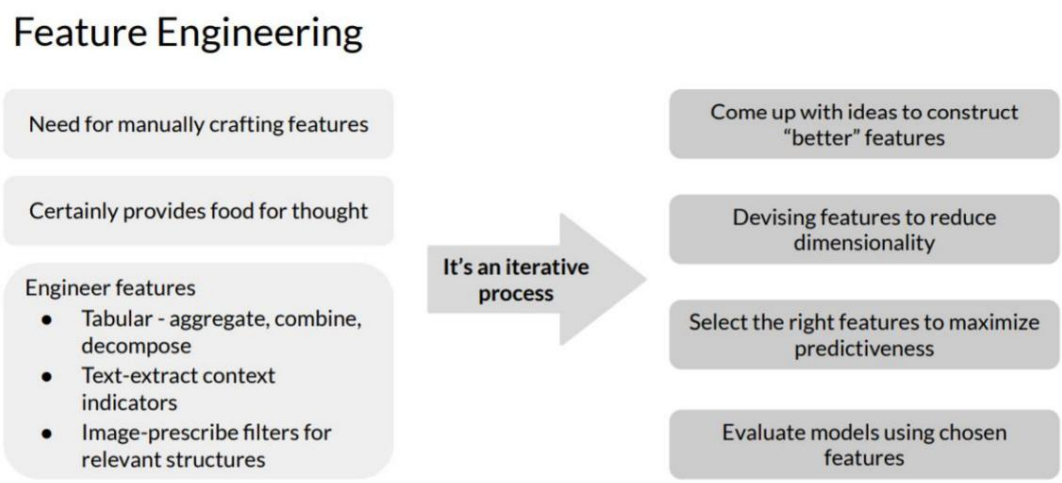
Funciones normalizadas, diseñadas o integradas. - Un modelo pobre alimentado con características importantes funcionará mejor que un modelo fantástico alimentado con baja calidad o malas características.



- Muchos dominios implican un gran número de características y dimensiones. - A menudo, la primera selección de funciones es una expresión de conocimiento del dominio, lo que a menudo puede resultar en más características de las que realmente necesitamos o queremos.
- Como vimos en nuestra discusión sobre la maldición de la dimensionalidad, esto tiene inconvenientes inherentes. Esto significa que necesita reducir la dimensionalidad, o más precisamente, la cantidad de características que está incluyendo en su conjunto de datos mientras retiene o mejora la cantidad de información predictiva contenida en los datos.



- La reducción de dimensionalidad busca patrones y datos y utiliza estos patrones para reexpresar los datos en un forma dimensional inferior.
- Esto hace que el cálculo sea mucho más eficiente, lo que puede ser un factor importante en un mundo de grandes modelos y grandes conjuntos de datos.
- Sin embargo, la función más esencial de la reducción de dimensionalidad es reducir el conjunto de datos a su esencia, descartando características ruidosas que causan problemas significativos para tareas de aprendizaje supervisado como regresión y clasificación.
- En muchas aplicaciones del mundo real, es la reducción de la dimensionalidad lo que hace posible las predicciones.
- Su infraestructura de recopilación y gestión de datos también se simplificará. Otro factor a considerar es que algunos algoritmos no funcionan bien cuando tenemos grandes dimensiones. También reduce
- la multicolinealidad al eliminar características redundantes. Ayuda cuando estamos tratando de
- visualizar los datos. Y como comentamos anteriormente, no es fácil visualizar datos en dimensiones más altas, por lo que reducir nuestro espacio a 2D o 3D puede ayudarnos a trazar y observar patrones con mayor claridad.
- La ingeniería de funciones ayuda a cumplir estos requisitos. Crea información valiosa a partir de datos sin procesar reformateando, combinando y transformando las características principales en otras nuevas hasta que produce un nuevo conjunto de datos que da como resultado un mejor modelo.
- Además, la selección de características examina un conjunto de características potenciales, selecciona algunas de ellas y descarta el resto. La selección de funciones se aplica para evitar la redundancia o para eliminar la irrelevancia en las funciones originales, o simplemente para llegar a un número limitado de funciones para evitar problemas. Como ya hemos visto varias técnicas de selección de
- características, veamos en su lugar la ingeniería de características.



- Los mejores resultados se reducen a usted, el profesional que elabora las características. Esta es una de las áreas donde La ingeniería de aprendizaje automático es algo así como una forma de arte .

- La importancia de las funciones y la selección de funciones pueden ayudar a informarle sobre la utilidad objetiva de las funciones, pero esas características tienen que venir de alguna parte.
- A menudo es necesario crearlos manualmente. Esto requiere pasar mucho tiempo con los datos de muestra reales y pensar en la forma subyacente del problema, las estructuras en los datos y la mejor manera de expresarlos para los algoritmos de modelado predictivo.

- Con datos tabulares, a menudo significa una mezcla de agregar o combinar características para crear nuevas características y descomponer o dividir características para crear también nuevas características. - Con datos textuales, a menudo significa diseñar documentos o indicadores específicos de contenido relevantes para el problema.
- Con datos de imagen, a menudo puede significar usar filtros para seleccionar estructuras relevantes como píxeles, contornos, formas y texturas.
- Tiende a ser un proceso iterativo que implica la selección de datos y la evaluación del modelo una y otra vez. - El proceso generalmente comienza con funciones de lluvia de ideas. Aquí realmente te metes en el problema, mira mucho de datos, estudie ingeniería de características en otros problemas y vea lo que puede aprender.
- Luego pasas a idear nuevas características. Depende de su problema, pero puede usar la función automática extracción, ingeniería de características manual o una combinación de ambas.
- A continuación, elija las funciones correctas utilizando métodos de puntuación y selección de funciones importantes para preparar una o más vistas de sus datos. Finalmente,
- mide el rendimiento del modelo en datos no vistos utilizando las características elegidas.

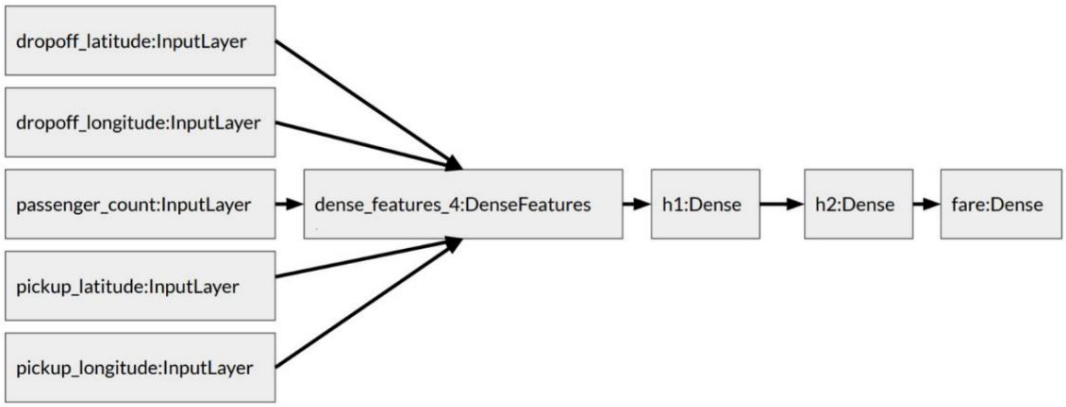
Reducción manual de la dimensionalidad: estudio de caso

Taxi Fare dataset

```
CSV_COLUMNS = [
    'fare_amount',
    'pickup_datetime', 'pickup_longitude', 'pickup_latitude',
    'Dropoff_longitude', 'dropoff_latitude',
    'passenger_count', 'key',
]

LABEL_COLUMN = 'fare_amount'
STRING_COLS = ['pickup_datetime']
NUMERIC_COLS = ['pickup_longitude', 'pickup_latitude',
                'dropoff_longitude', 'dropoff_latitude',
                'passenger_count']

DEFAULTS = [[0.0], ['na'], [0.0], [0.0], [0.0], [0.0], [0.0], ['na']]
DAYS = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
```

- Ahora ha visto que la dimensionalidad de sus datos tiene un impacto en el rendimiento de sus modelos. y los recursos necesarios para entrenar y servir esos modelos. Esto es aún
- más importante cuando se trata de escenarios de limitaciones de recursos, como las implementaciones móviles.
- Por lo tanto, debe administrar cuidadosamente la dimensionalidad de sus datos, lo que a menudo significa buscar formas de reducirlo.
- Veamos ahora algunas técnicas manuales para hacer la reducción de dimensionalidad. Veamos un ejemplo concreto utilizando el conjunto de datos de tarifas de taxi.
- El conjunto de datos consta de 106.545 viajes en taxi. El objetivo es predecir las tarifas de cada viaje en base a un variedad de características como la hora y el lugar de recogida, el tiempo de viaje y la distancia, el número de pasajeros, etc.
- Como de costumbre, los primeros pasos son descargar el conjunto de datos que está en formato CSV separando las variables en cadenas y tipos numéricos, y definir constantes y parámetros útiles.

Build the model in Keras



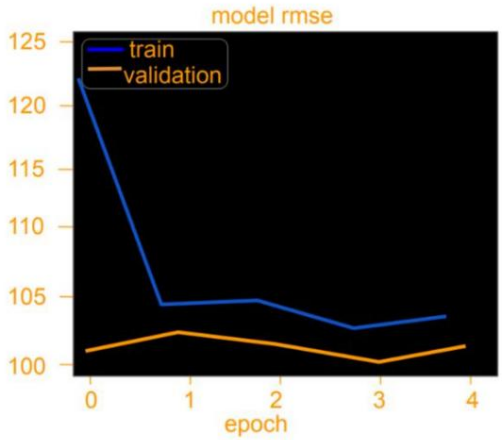
- Construyamos un modelo de referencia para predecir la tarifa. Intentaremos usar estas características: Regreso, latitud, Recuento de pasajeros de longitud de regreso, Recogida de latitud y Recogida de longitud. - La red consiste en una concatenación de densas capas ocultas con la última produciendo una predicción de tarifa producción.

Build a baseline model using raw features

```
from tensorflow.keras import layers
from tensorflow.keras.metrics import RootMeanSquared as RMSE

dnn_inputs = layers.DenseFeatures(feature_columns.values())(inputs)

h1 = layers.Dense(32, activation='relu', name='h1')(dnn_inputs)
h2 = layers.Dense(8, activation='relu', name='h2')(h1)

output = layers.Dense(1, activation='linear', name='fare')(h2)
model = models.Model(inputs, output)
model.compile(optimizer='adam', loss='mse',
              metrics=[RMSE(name='rmse'), 'mse'])
```

- Construiremos el modelo en Keras utilizando la API funcional : a diferencia de la API secuencial, deberá especificar las capas de entrada y ocultas.
- Tenga en cuenta que está creando un modelo de referencia de regresión lineal sin ingeniería de características - Como recordatorio rápido, un modelo de referencia es una implementación ingenua que ayuda a establecer expectativas para rendimiento del modelo.

Train the model



- Después de configurar el modelo para el entrenamiento y crear los conjuntos de datos, está listo para entrenar el modelo de referencia. Para entrenar el modelo, simplemente llame a
- model.fit. Veamos el rendimiento del entrenamiento y la validación utilizando la pérdida de error cuadrático medio durante las épocas de entrenamiento. Idealmente, desea que el RMSE de validación esté cerca del conjunto de entrenamiento.

Increasing model performance with Feature Engineering

- Carefully craft features for the data types
  - Temporal (pickup date & time)
  - Geographical (latitude and longitude)

- Ahora, intentemos mejorar el modelo. Para mejorar el rendimiento del modelo, creemos dos nuevas características, tipos de ingeniería, temporal y geográfica.

Handling temporal features

```
def parse_datetime(s):
    if type(s) is not str:
        s = s.numpy().decode('utf-8')
    return datetime.datetime.strptime(s, "%Y-%m-%d %H:%M:%S %Z")

def get_dayofweek(s):
    ts = parse_datetime(s)
    return DAYS[ts.weekday()]

@tf.function
def dayofweek(ts_in):
    return tf.map_fn(
        lambda s: tf.py_function(get_dayofweek, inp=[s],
                  Tout=tf.string),
        ts_in)
```

- Por ejemplo, trabajaremos con la característica temporal, tomaremos la fecha y la hora como una cadena y necesitaremos manejar esto dentro del modelo.
- Primero, incluirá la fecha y hora de recogida como una característica y luego deberá modificar el modelo para manejarlo como una característica de cadena.

Geolocational features

```
def euclidean(params):
    lon1, lat1, lon2, lat2 = params
    londiff = lon2 - lon1
    latdiff = lat2 - lat1
    return tf.sqrt(londiff * londiff + latdiff * latdiff)
```

- Los datos de longitud y latitud de recogida o entrega son cruciales para predecir el monto de la tarifa desde la tarifa
  Las cantidades en los taxis de la ciudad de Nueva York están determinadas en gran medida por la distancia recorrida.
- Como tal, necesitamos enseñar el modelo de la distancia euclidiana entre los puntos de recogida y entrega.
- Recuerde que la latitud y la longitud nos permiten especificar cualquier ubicación en la Tierra usando un conjunto de coordenadas. El conjunto de datos contiene información sobre las coordenadas de recogida y entrega.
- Sin embargo, no hay información sobre la distancia entre los puntos de recogida y entrega. - Así que vamos a crear una nueva función que calcule la distancia entre cada par de puntos de recogida y entrega. - Puedes hacer esto usando la distancia euclidiana, que es una distancia en línea recta entre dos coordenadas
  puntos, pero tenga en cuenta que esto solo será un indicador aproximado de la distancia real de conducción.

Scaling latitude and longitude

```
def scale_longitude(lon_column):
    return (lon_column + 78)/8.


def scale_latitude(lat_column):
    return (lat_column - 37)/8.
```

- Es muy importante que las variables numéricas se escalen antes de que se introduzcan en la red neuronal.
- Usemos la escala minmax, también llamada normalización, en las funciones de geolocalización. Más
- adelante en nuestro modelo, verá que estos valores se desplazan y reescalan para que terminen variando de 0 a 1. Usaremos el
- conocimiento del dominio para crear una función llamada scale_longitude donde pasa todos los valores de longitud y suma 78 a cada uno. valor. Tenga en cuenta que los valores de longitud de escala van desde 72 negativo a 78 negativo. Por lo tanto, el valor 78 es el valor longitudinal máximo. La diferencia o delta entre 70 negativo y 78 negativo es ocho. - La función suma 78 a cada valor longitudinal y luego divide por ocho para devolver un valor escalado.

- De manera similar, creemos una función llamada scale_latitude en la que pasas todos los valores de latitud y restas 37 de cada valor. Tenga en cuenta que el rango de latitud de la escala es de 37 a 45.
- Por tanto, el valor 37 es el valor latitudinal mínimo. El delta o diferencia entre negativo 37 y negativo
  45 también pasa a ser ocho.
- La función que resta 37 de cada valor latitudinal y luego divide por ocho para devolver un valor escalado.