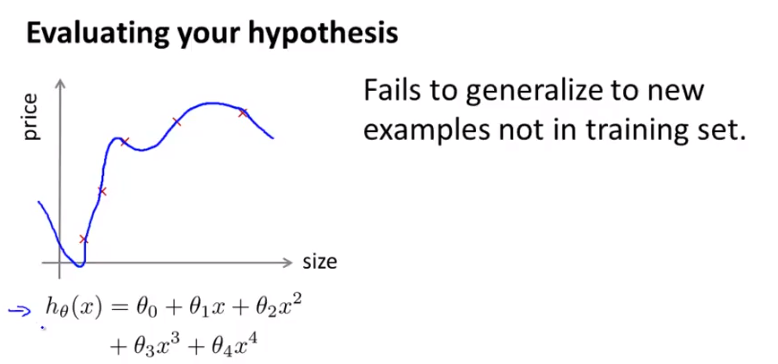
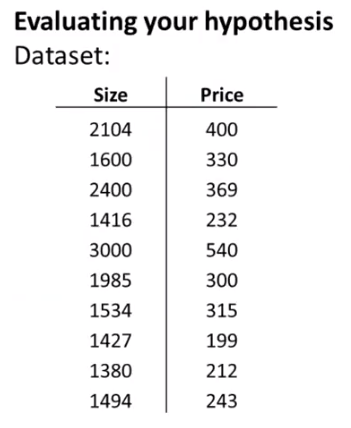
# Evaluando la hipótesis

En este video me gustaría hablar de cómo evaluar una hipótesis aprendida por nuestro algoritmo. En los siguientes videos profundizaremos para hablar de cómo prevenir los problemas de sobreajuste (*overfitting*) y de subajuste(*underfitting*).

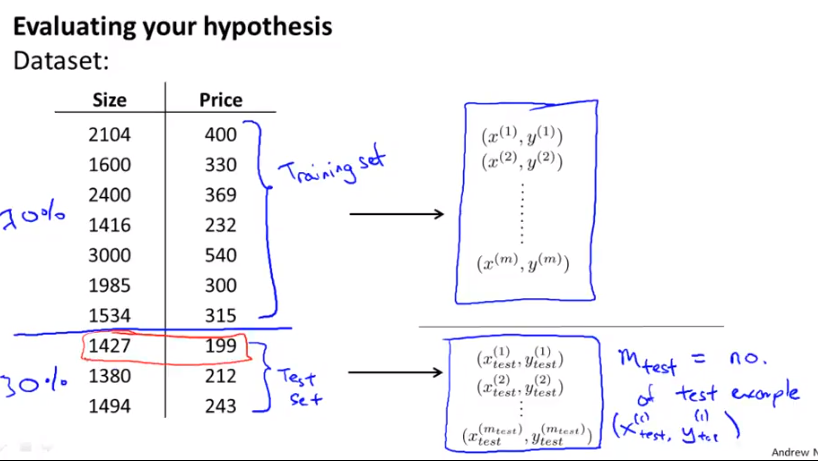
Cuando ajustamos los parámetros de nuestro algoritmo de aprendizaje pensamos en elegir los parámetros para minimizar el error de entrenamiento. Pudiéramos pensar que obtener un valor realmente bajo de error de entrenamiento es bueno, pero ya hemos visto que una hipótesis con un error de entrenamiento bajo no es necesariamente una buena hipótesis. También hemos visto ya un ejemplo de cómo la hipótesis puede causar un sobreajuste e impedir, por lo tanto, la generalización de nuevos ejemplos que no se incluyen en el conjunto de datos de entrenamiento.



Entonces ¿cómo podemos saber si una hipótesis está sobreajustada? En este ejemplo simple podemos trazar la hipótesis h(x) y ver qué sucede. Pero, en general, para problemas con más de una variable de entrada; es decir, en problemas con un gran número de funciones, se vuelve difícil o quizá imposible trazar cómo luce la función de la hipótesis. Por lo tanto, necesitamos otra manera para evaluar nuestra hipótesis. El método estándar para evaluar una hipótesis aprendida es el siguiente. Supongamos que tenemos un conjunto de datos de aprendizaje como este:



Aquí sólo se muestran 10 ejemplos de entrenamiento, pero podemos tener docenas, cientos, o quizá miles de ellos. Para asegurarnos de poder evaluar nuestra hipótesis, lo que haremos es dividir los datos que tenemos en dos porciones. La primera porción será nuestro conjunto de entrenamiento y la segunda porción será nuestro conjunto de prueba. Una división muy típica de los todos datos que tenemos en un conjunto de entrenamiento y en un conjunto de test puede ser de aproximadamente el 70% y 30%, respectivamente. Donde tenemos más datos en el conjunto de entrenamiento y relativamente menos en el conjunto de prueba.Ahora, si tenemos un conjunto de datos asignaremos solamente el 70% de los datos para que sean nuestro conjunto de entrenamiento donde “m” es, como de costumbre, nuestro número de ejemplos de entrenamiento y el resto de los datos será asignado para formar parte de nuestro conjunto de prueba. Aquí, utilizaré la notación “m” subíndice “prueba” para denotar el número de ejemplos de prueba. En general, este subíndice “prueba” denotará los ejemplos que resulten del conjunto de prueba de manera que “x1” subíndice “prueba” coma “y1” subíndice “prueba” es mi primer ejemplo de prueba.



Finalmente, un último detalle: Aunque he separado estos datos con el primer 70% asignado al conjunto de entrenamiento y el 30% restante al conjunto de pruebas, si hay algún tipo de orden en los datos, sería mejor asignar un 70% aleatorio de los datos al conjunto de entrenamiento y un 30% aleatorio de los datos al conjunto de prueba. Así que, con los datos ya asignados de manera aleatoria, podemos tomar el primer 70% y el último 30%. Pero si los datos no estuvieran ordenados aleatoriamente, sería mejor aleatorizar o reordenar de manera aleatoria los ejemplos en tu conjunto de entrenamiento. Antes de enviar el primer 70% al conjunto de entrenamiento y el último 30% al conjunto de prueba.

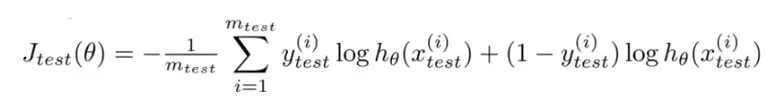
Aquí presento un procedimiento típico para entrenar y probar el algoritmo de aprendizaje y la regresión lineal.

1. Primero, es necesario aprender los parámetros theta de tu conjunto de entrenamiento, para que minimices el error de aprendizaje usual objetivo de “J” de theta, donde “J” de theta se definió utilizando el 70% de todos los datos que tenemos. Es decir, sólo los datos de entrenamiento.
2. Después es necesario calcular el error de prueba. Denotaré el error de prueba como “J” subíndice “test”. Lo que haremos es tomar los parámetros theta que hemos aprendido del conjunto de entrenamiento y lo insertaremos en los datos test. Luego calcularemos el error, que escribiré como sigue. Esto es, básicamente, el error cuadrático promedio como se midió en el conjunto de prueba. Es lo que uno se esperaría. Si aplicamos cada ejemplo de prueba a la hipótesis con el parámetro theta y medimos el error cuadrático de la hipótesis en “m” subíndice “test”, es decir, en los ejemplos de prueba.

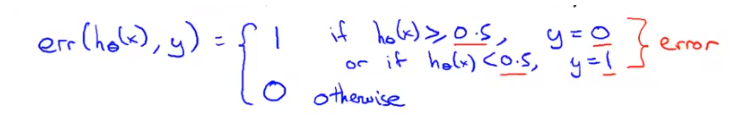


Y, por supuesto, esta es la definición del error del conjunto de prueba si utilizamos la regresión linear y utilizamos la métrica del error cuadrático. Pero, ¿qué pasa si estamos resolviendo un problema de clasificación utilizando, en cambio, la regresión logística? En este caso, el procedimiento para entrenar y probar la regresión logística es muy similar.

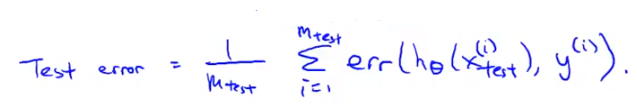
1. Primero obtendremos los parámetros de los datos de entrenamiento; es decir, el 70% de los datos.
2. Después calcularemos el error de prueba.Es la misma función objetiva que utilizamos siempre para la regresión logística, pero esta vez la definiremos usando “m” subíndice “test”; es decir, los ejemplos de prueba. Aunque esta definición del error del conjunto de prueba “J” subíndice “test” es perfectamente razonable.



A veces hay una alternativa de medición de los conjuntos de prueba que puede resultar más fácil de interpretar, y ese el error de mala clasificación. También se le llama error de clasificación cero-uno en donde cero y uno denotan que puedes obtener ya sea un ejemplo correcto o un ejemplo erróneo. Esto es lo que quiero decir. Permítanme definir el error de una predicción. Es decir error(h(x)), con un valor asignado a “y” igual a 1 si los resultados de mi hipótesis arrojan un valor mayor o igual a 5 y “Y” es igual a 0, o si mi hipótesis arroja un valor menor a 0.5 y “y” es igual a 1, bien, ambos casos responden, básicamente, si la hipótesis clasificó mal el ejemplo asumiendo que asignaste un umbral de 0.5. Así que, pensamos que era más probable que fuera 1, pero resultó ser 0, o si la hipótesis era más propensa al 0, pero el valor asignado era en realidad 1. De otra manera, definimos esta función de error como 0, si tu hipótesis clasificó el ejemplo “Y” de manera correcta.



Entonces, podríamos definir este error de prueba utilizando la medición del error de mala clasificación como uno de las pruebas m de la suma de “i” igual a 1 a “m” subíndice “prueba” del error de “h” de “x(i) prueba” coma “y(i)”.



Esta es mi manera de escribirlo y es exactamente la fracción de ejemplos en el conjunto de prueba mal asignados por mi hipótesis. Esta es la definición del error del conjunto de prueba utilizando el error de mala clasificación de la medición del error de mala clasificación 0 1.

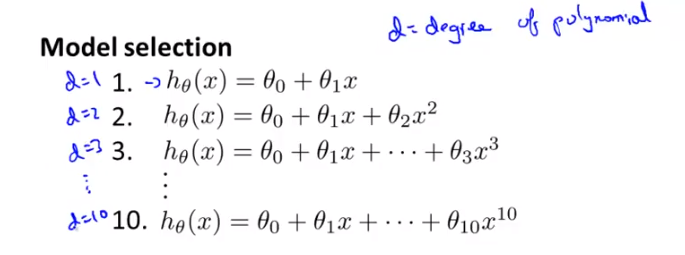
Esta es la técnica estándar para evaluar qué tan buena es una hipótesis. En el siguiente video adaptaremos estas ideas para ayudarnos a hacer cosas como elegir las características, como el grado polinomial que se utilizarán en el algoritmo de aprendizaje o a elegir los parámetros de regularización para un algoritmo de aprendizaje.

# Selección de modelo Entrenamiento/validación/Test

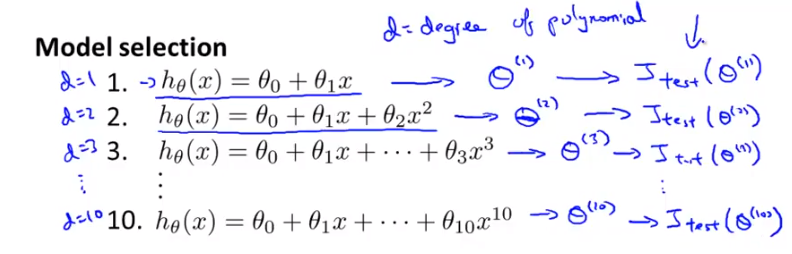
Imaginemos que te gustaría decidir qué grado de polinomio ajustar a un conjunto de datos, o qué funciones incluir para utilizar tu algoritmo de aprendizaje. O supongamos que quieres elegir el parámetro de regularización «lambda» para el algoritmo de aprendizaje. ¿Cómo lo harías? Estos se llaman problemas de selección de modelo. En nuestra discusión de cómo resolverlos, hablaremos no sólo sobre cómo dividir los datos en los conjuntos de prueba y aprendizaje, sino también sobre cómo dividir los datos en lo que llamaremos conjuntos de validación y de prueba. En este video veremos qué son estos conjuntos y cómo podemos utilizarlos para seleccionar un modelo.

Ya hemos visto en muchas ocasiones el problema del sobreajuste, que nos dice que no sólo porque el algoritmo de aprendizaje se ajusta bien un conjunto de entrenamiento, la hipótesis será buena. De manera más general, esta es la razón por la cual el error del conjunto de entrenamiento no es una buena variable predictiva para saber qué tan efectiva será la hipótesis con nuevos ejemplos. Específicamente, si aplicas un conjunto de parámetros, «theta» 0, «theta» 1, «theta» 2, y sucesivamente, a tu conjunto de entrenamiento, el hecho de que la hipótesis sea efectiva en el conjunto de entrenamiento no significa mucho en cuanto a la predicción de qué tan bien se generalizará la hipótesis con ejemplos que no se han visto antes en el conjunto de entrenamiento. El principio más general es que, una vez que los parámetros fueron ajustados a un conjunto de datos, ya sea el conjunto de entrenamiento o cualquier otro, el error de la hipótesis, como se midió en el mismo conjunto de datos, tal como el error de entrenamiento, tal vez no resulta un estimador adecuado del error de generalización real; es decir, qué tan bien se generalizará la hipótesis con los nuevos ejemplos.

Ahora consideremos el problema de selección de modelo. Digamos que intentas elegir qué grado de polinomio ajustar a los datos. ¿Debes elegir una función lineal, una función cuadrática o una cúbica? ¿Qué función, hasta un polinomio del 10 orden, debes elegir? Entonces, hay un parámetro adicional en este algoritmo, que denotaré con una “d”, que indica qué grado de polinomio quieres utilizar. De manera que además de los parámetros theta, hay un parámetro más (d) que intentamos determinar utilizando el conjunto de datos. La primera opción es “d” igual a 1, que corresponde a la función lineal. Podemos elegir “d” igual a 2, “d” igual a 3, y, hasta “d” igual a 10.



Queremos, entonces, ajustar este parámetro adicional, que denotaremos como parámetro “d”. Y digamos que quieres elegir un modelo, es decir, elegir un grado de polinomio o uno de estos diez modelos, y ajustarlo para obtener un estimado de qué tan bien se generalizará la hipótesis ajustada a nuevos ejemplos. Aquí hay algo que podrías hacer: podrías tomar tu primer modelo y minimizar el error de entrenamiento para que arroje el parámetro vector «theta». Después puedes tomar tu segundo modelo, la función cuadrática, y ajustarla al conjunto de entrenamiento. Esto te dará otros parámetros en el vector «theta». Para distinguir entre los diferentes parámetros de vector, utilizaré el superíndice 1 y el superíndice 2 donde «theta» superíndice 1 se refiere al parámetro que obtendré ajustando este modelo a mis datos de entrenamiento y «theta» superíndice 2 se refiere al parámetro que obtendré al ajustar la función cuadrática a mis datos de entrenamiento. Cuando ajuste el modelo cúbico obtendré el parámetro «theta» 3, y así sucesivamente hasta «theta» 10. Podemos tomar estos parámetros y buscar el error del conjunto de prueba. Puedo calcular en mi conjunto de prueba “j prueba” de «theta» 1, “j prueba” de «theta» 2, “j prueba” de «theta» 3, y así sucesivamente.



Luego tomaré cada una de mis hipótesis con sus parámetros correspondientes y mediré su desempeño en el conjunto de prueba. Para seleccionar uno de estos modelos, una de las cosas que puedo hacer es ver qué modelo tiene el error del conjunto de prueba más bajo.

Digamos, que para este ejemplo elegimos el polinomio de quinto orden. Esto parece ser razonable hasta el momento, pero qué pasa si quiero tomar mi hipótesis ajustada; es decir, el modelo del quinto orden, y preguntarme qué tan bien se generaliza este modelo. Para responder esto puedo analizar qué tan bien se desempeñó la hipótesis del polinomio de quinto orden en mi conjunto de prueba. El problema de esto es que no será un estimado justo de qué tan bien se generaliza mi hipótesis. La razón es que, lo que hemos hecho es ajustar este parámetro adicional “d”, que es el grado del polinomio, utilizando el conjunto de prueba “d”. En otras palabras, elegimos el valor de d que nos dio el mejor desempeño posible en el conjunto de prueba.



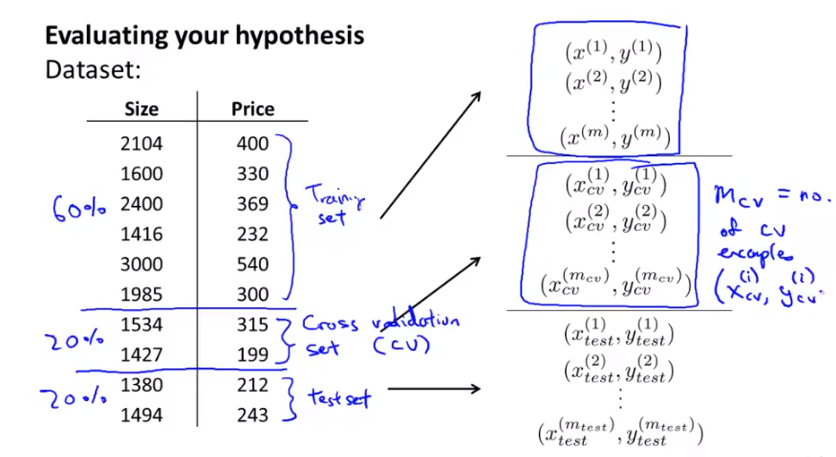
El desempeño de mi parámetro vector «theta» cinco del conjunto de prueba probablemente dará un estimado muy optimista del error de generalización. ¿Sí? Debido a que he ajustado este parámetro “d” a mi conjunto de prueba, ya no es justo evaluar mi hipótesis en este conjunto de prueba. Esto es porque he ajustado mis parámetros al conjunto de prueba y he elegido el grado “d” del polinomio utilizando el conjunto de prueba. De modo nuestra hipótesis, probablemente, será más efectiva en este conjunto de prueba que en nuevos ejemplos que no se han visto antes.

Así que solo para reiterar, en la diapositiva anterior, vimos que, si encajamos algún conjunto de parámetros, ya sabes, decir theta0, theta1, y así sucesivamente, a un conjunto de entrenamiento, entonces el rendimiento del modelo ajustado en el conjunto de entrenamiento no es predictivo de qué tan bien la hipótesis se generalizará a nuevos ejemplos. Es porque estos parámetros se ajustaron al conjunto de entrenamiento, por lo que es probable que les vaya bien en el conjunto de entrenamiento, incluso si los parámetros no funcionan bien en otros ejemplos. Y, en el procedimiento que acabo de describir en esta línea, simplemente hicimos lo mismo. Y específicamente, lo que hicimos fue, ajustamos este parámetro d al conjunto de prueba. Y al ajustar el parámetro al conjunto de prueba, esto significa que el rendimiento de la hipótesis en ese conjunto de prueba puede no ser una estimación justa de cómo de buena es la hipótesis, es probable que no lo haga tan bien en ejemplos que no hemos visto antes.

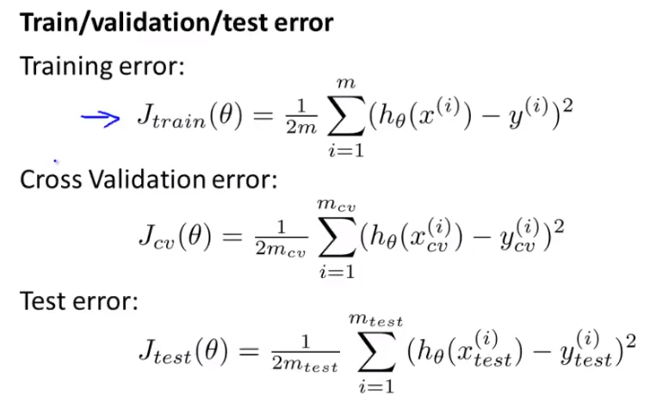
Para solucionar este problema, en una configuración de selección de modelo, si queremos evaluar una hipótesis, esto es lo que solemos hacer en su lugar. Dado el conjunto de datos, en lugar de simplemente dividir en un conjunto de prueba de entrenamiento, lo que vamos a hacer es dividirlo en tres partes.

1. Y la primera pieza se llamará el conjunto de entrenamiento como de costumbre.
2. Y la segunda parte, voy a llamar al conjunto de validación cruzada (cross validation). Validación cruzada. Y la validación cruzada, como CV. A veces también se llama el conjunto de validación en lugar del conjunto de validación cruzada.
3. Y luego la parte restante puede ser llamar al conjunto de prueba habitual (test set).

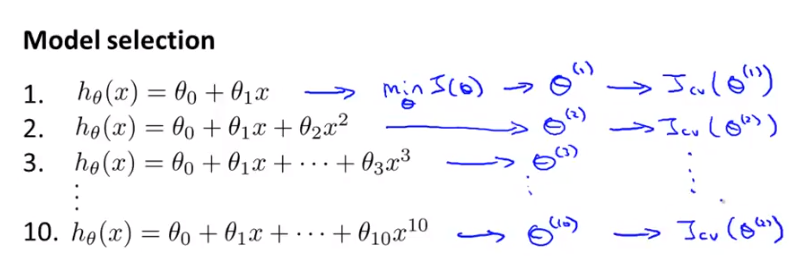
Y la proporción típica en la que dividir estas cosas será para enviar el 60% de tus datos, tu conjunto de entrenamiento, tal vez un 20% para su conjunto de validación cruzada, y un 20% para su conjunto de prueba. Y estos números pueden variar un poco, pero esta integración es bastante típica. Y entonces nuestros conjuntos de entrenamiento ahora serán solo el 60% de los datos, y nuestro el conjunto de validación cruzada, o nuestro conjunto de validación, tendrá algunos ejemplos. Voy a denotar ese m subíndice cv. Entonces esa es la cantidad de ejemplos de validación cruzada. Siguiendo nuestra convención de notación temprana, voy a usar xi cv comma yi cv, para denotar el ejemplo de validación cruzada. Y finalmente también tenemos un conjunto de pruebas.



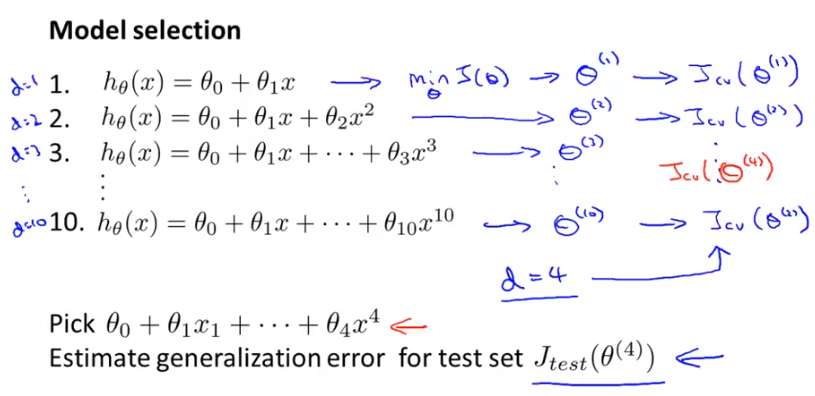
Entonces, ahora que hemos definido la validación del entrenamiento o conjuntos cruzados de validación y prueba. También podemos definir el error de entrenamiento, el error de validación cruzada y el error de prueba.



Entonces, cuando nos enfrentamos con un problema de selección de modelos como este, ¿qué vamos a hacer?. En lugar de usar el conjunto de prueba para seleccionar el modelo, estamos en su lugar va a usar el conjunto de validación o el conjunto de validación cruzada para seleccionar el modelo. Concretamente, primero vamos a tomar nuestra primera hipótesis, tomar este primer modelo, y digamos, minimizar la función de perdidas, y esto me daría algún vector de parámetro theta para el nuevo modelo. Y, como antes, voy a poner un superíndice 1, solo para denotar que este es el parámetro para el nuevo modelo. Hacemos lo mismo para el modelo cuadrático. Obtenga un vector de parámetro theta dos. Obtener algunos para, parámetro vector theta tres, y así encendido, abajo a theta diez para el polinomio. Y lo que voy a hacer es, en lugar de probar estas hipótesis en el conjunto de prueba, En su lugar voy a probarlos en el conjunto de validación cruzada para ver qué tan bien hacen cada una de estas hipótesis en mi conjunto de validación cruzada.



Y luego voy a elegir la hipótesis con el menor error de validación cruzada. Entonces, para este ejemplo, digamos por el bien de la discusión, que era mi polinomio de cuarto orden, que tenía el menor error de validación cruzada. Entonces, en ese caso, elegiré este modelo polinómico de cuarto orden. Y finalmente, lo que esto significa es que ese parámetro d, recuerda d era el grado de polinomio. Lo que hemos hecho es ajustar el parámetro d y diremos que d es igual a cuatro. Y lo hicimos usando el conjunto de validación cruzada. Y entonces este grado de polinomio, por lo que el parámetro ya no se ajusta a la prueba establecido, por lo que no hemos guardado el conjunto de prueba, y podemos usar el conjunto de prueba para medir o estimar el error de generalización del modelo que se seleccionó. Por el de ellos. Entonces, esa fue la selección del modelo y cómo puedes tomar tus datos, divídalo en un conjunto de entrenamiento, validación y prueba. Y use sus datos de validación cruzada para seleccionar el modelo y evaluarlo en el conjunto de prueba.



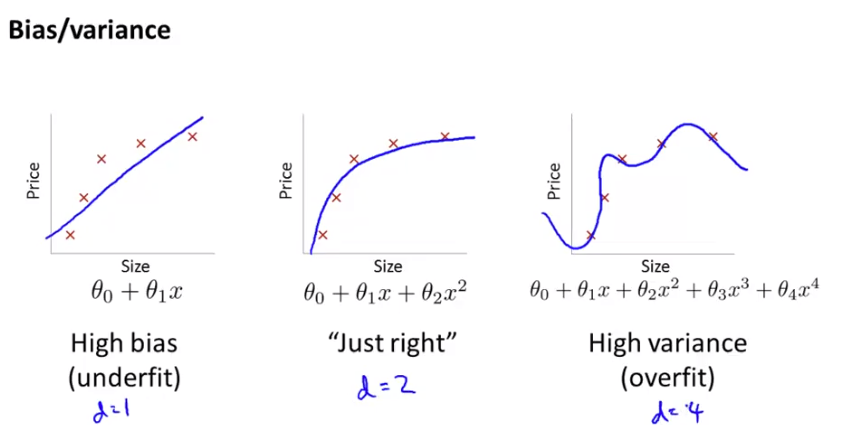
Una última nota. El aprendizaje automático tal como se práctica a dia de hoy, no hay mucha gente que hará primeramente de lo que hablé. No es una buena idea seleccionar tu modelo usando este conjunto de prueba. Y luego usar el mismo conjunto de prueba para informar del error como si seleccionando su grado de polinomio en el conjunto de prueba, y luego informando el error en la prueba se estableció como si fuera una buena estimación del error de generalización. Ese tipo de práctica es lamentablemente muchas, muchas personas lo hacen. Si tienes una prueba masiva y masiva que quizás no sea algo terrible de hacer, pero muchos practicantes, la mayoría de los practicantes tienden a desaconsejarlo. Y se considera una mejor práctica tener un conjunto de validación cruzada por separado y conjuntos de prueba. Acabo de advertirle que a veces las personas deben hacer, ya saben, usar los mismos datos para el propósito del conjunto de validación, y para el propósito del conjunto de prueba. Necesitas un set de entrenamiento y un set de prueba, y eso está bien, eso es práctica, aunque verá que algunas personas lo hacen. Pero, si es posible, recomendaría no hacerlo tú mismo.

# Diagnosticando Bias y Variance

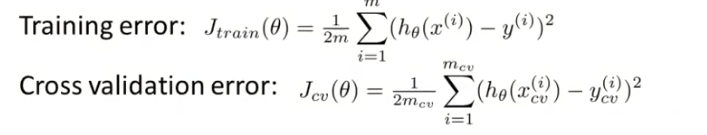
Si ejecutas el algoritmo de aprendizaje y no resulta tan bien como esperabas, la mayoría de las veces será porque existe ya sea un problema de sesgo alto (“*high bias*”) o de varianza alta (“*high variance*”). En otras palabras, un problema de sobreajuste o de subajuste.   
  
En este caso es muy importante entender cual de estos dos problemas tenemos: de sesgo o varianza, o un poco de ambos. Ya que saber cuál de ellos sucede nos daría un indicador certero de de las maneras útiles y prometedoras para mejorar tu algoritmo.

En este video, me gustaría desarrollar más el asunto de sesgo y varianza para entenderlos mejor y para encontrar la manera de ver un algoritmo de entrenamiento y evaluar o diagnosticar si tenemos, o no, un problema de varianza o de sesgo. Esto es crítico para entender cómo mejorar el desempeño del algoritmo de aprendizaje que intentas implementar.

Ya has visto esta figura varias veces. En ella puedes aplicar

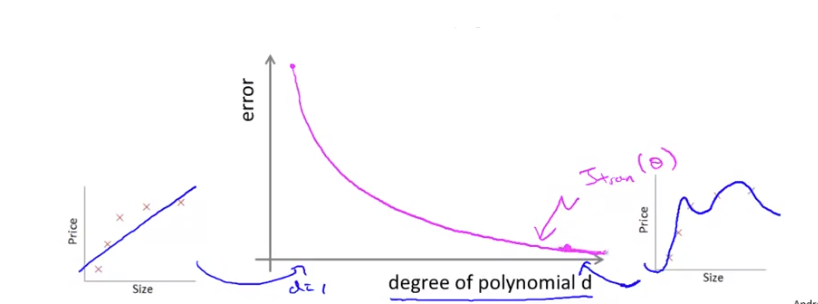


1. una hipótesis simple que resulta en una línea recta que subajusta los datos.
2. Si aplicas una hipótesis compleja, entonces quizá se ajustará al conjunto de entrenamiento perfectamente, pero sobreajustará los datos
3. hipótesis de complejidad media con polinomios de segundo grado; es decir, de un grado no muy alto y no muy bajo, sino justo.

Ahora que tenemos el conocimiento de entrenamiento y validación en los conjuntos de prueba, podemos entender los conceptos de sesgo y varianza un poco mejor. Definamos nuestro error de entrenamiento y error de validación como en los videos anteriores. Es decir, el error cuadrático o el error cuadrático promedio como se calculó en el conjunto de entrenamiento o en el conjunto de validación cruzada.   


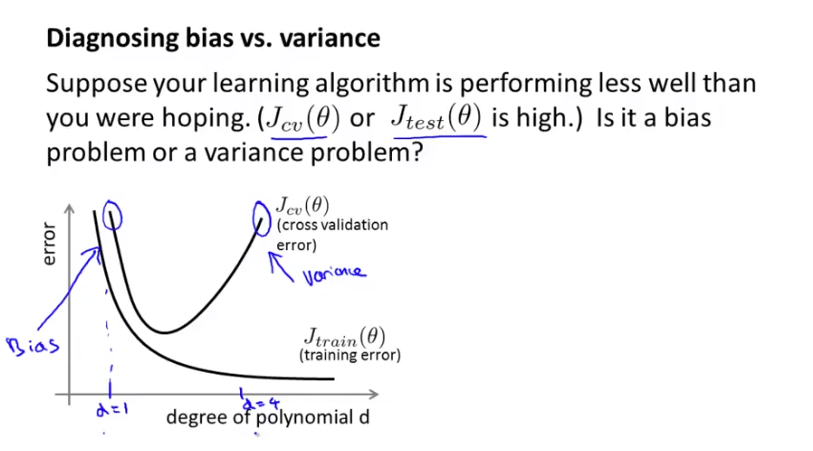
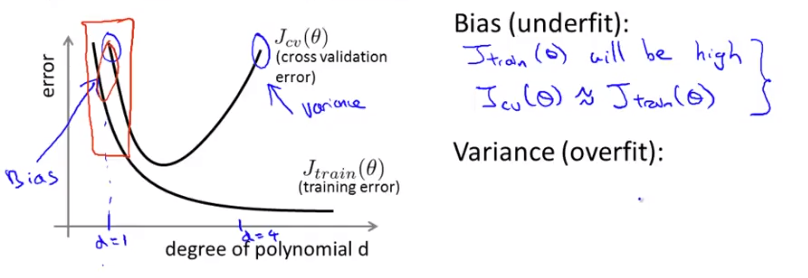
Ahora, tracemos la siguiente figura. En el eje horizontal trazaré el grado del polinomio. Conforme me acerque a la derecha ajustaré polinomios de un orden cada vez más grande.

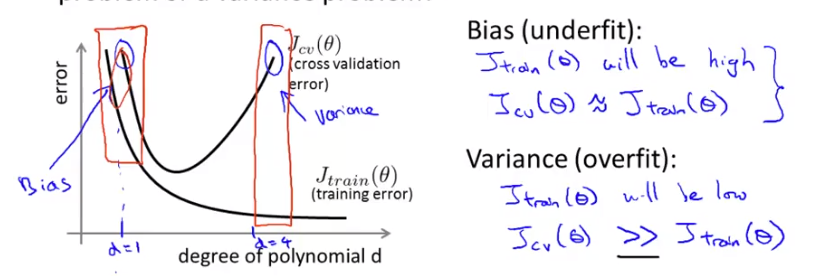
Haremos esto con esta figura donde “d” es quizá igual a 1 y ajustaremos funciones muy simples, mientras que a la derecha del esto podría ser, d es igual a 4 o podría ser un número aún mayor. Ajustaré órdenes de polinomios muy complejos que podrían ajustarse al conjunto de entrenamiento con variables mucho más complejas mientras que estamos en el extremo derecho del eje horizontal tendremos valores de “d” mayores o polinomios de grado más alto. Y esto corresponderá al ajuste de funciones más complejas a tu conjunto de entrenamiento. Ahora veamos el error de entrenamiento y el error de validación cruzada y tracémoslo en esta figura. Empecemos con el error de entrenamiento. A medida que aumenta el grado del polinomio, podremos ajustar el conjunto de entrenamiento cada vez mejor. Si “d” es igual a 1   
tendremos un error de entrenamiento relativamente alto. Mientras que si tenemos un polinomio de grado alto nuestro error de entrenamiento será realmente bajo. Quizá incluso sea cero porque se ajustará muy bien al conjunto de entrenamiento.



Entonces, mientras aumentamos el grado del polinomio encontramos típicamente que el error de entrenamiento disminuye. Escribiré aquí “J” subíndice “entrenamiento” de theta, porque nuestro error de entrenamiento tiende a disminuir con el grado del polinomio que ajustamos a los datos.

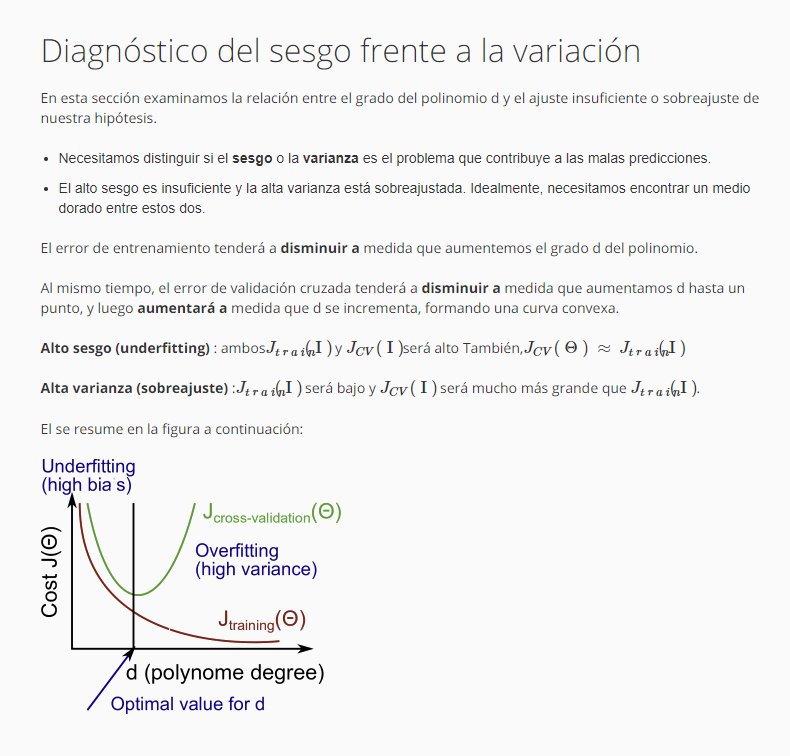
A continuación veremos el error de validación cruzada. O, para el caso, si vemos al error del conjunto de prueba obtendremos un resultado muy similar como si trazáramos el   
error de validación cruzada. Sabemos que si “d” es igual a 1, estaremos ajustando una función muy simple y quizá subajustaremos el conjunto de entrenamiento y tendremos un error de validación muy alto. Si ajustamos un polinomio de grado intermedio, de un polinomio de grado intermedio como “d” igual a 2 que presentamos en el ejemplo de la diapositiva anterior, tendremos un error de validación cruzada mucho menor porque estamos encontrando un valor más adecuado para los datos. un ajuste mucho mejor para los datos.   
Por el contrario, si “d” fuera muy alta, con un valor de cuatro, entonces estaremos sobreajustando de nuevo y acabaremos con un valor alto de error de validación cruzada.   
  
Si se mantuviera una variación uniforme y se trazara la curva, obtendríamos una curva como esta, donde esta sería “Jtest” de theta. Y, de nuevo, si trazamos “J” “prueba” de theta obtendríamos algo muy similar.

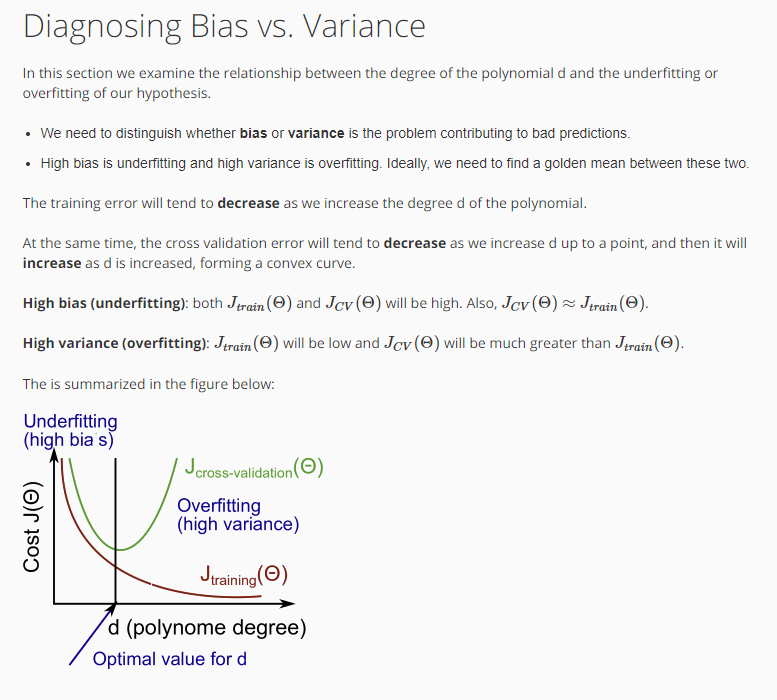
Este tipo de trazos nos ayudan a entender mejor las nociones de sesgo y varianza. supongamos que tienes un algoritmo de aprendizaje que no funciona tan bien como quisieramos que lo hiciera, ¿cómo saber si tu algoritmo de aprendizaje tiene problemas?   
  
Ahora, supongamos que has aplicado un algoritmo de aprendizaje que no funciona tan bien como quieres porque ya sea el error del conjunto de validación o el error del conjunto de prueba son muy altos.   
¿Cómo podemos averiguar si el algoritmo de aprendizaje sufre de un sesgo alto o de una varianza alta?   
La determinación de un error de validación alto corresponde a, ya sea este régimen o este otro régimen.   
  
El régimen de la izquierda corresponde a un problema de sesgo alto; es decir, cuando se ajusta un polinomio de orden muy bajo, como “d” igual a 1, en donde en realidad se necesita un polinomio de orden más alto para ajustar los datos. Por el contrario, el régimen de la derecha corresponde a un problema de varianza alta. Es decir, si “d”, el grado del polinomio, fuera muy grande para nuestro conjunto de datos. Esta figura nos da una pista para distinguir entre los dos casos.   
De forma concreta, para el caso de sesgo alto, es decir, el caso de subajuste, lo que encontramos es que tanto el error de validación cruzada como el error de entrenamiento serán altos. Por lo tanto, si tu algoritmo sufre de un problema de sesgo, el error del conjunto de entrenamiento sería alto y se podrá ver que el error de validación cruzada también sería alto y cercano o quizá un poco más alto que el error de entrenamiento. Y, si observas esta combinación tienes un signo de que tu algoritmo quizá sufra de un sesgo alto.   
  
Por el contrario, en el caso de que tu algoritmo sufra de una varianza alta, si miramos aquí nos daremos cuenta de que “J” subíndice “entrenamiento”; que es el error de entrenamiento, será bajo. Esto indica que estás ajustando bien el conjunto de entrenamiento. En cambio, en el error de validación cruzada, asumiendo que es un error cuadrático que intentamos minimizar, Este, el error en el conjunto de validación cruzada o la función de costo del conjunto de validación cruzada, será mayor que el error del conjunto de entrenamiento.



Este signo doble de mayor es el símbolo matemático para “mucho mayor que”, Así que esto es un doble mayor que el signo, que es el símbolo matemático para mucho mayor y se denota con dos símbolos de “mayor que”.

Así que, verás que esta combinación de valores es una pista de que tu algoritmo de aprendizaje sufre de una varianza alta y puede estar sobreajustado.   
Y la clave para distinguir estos dos casos es el hecho de que, si tienes un problema de sesgo alto, el error de tu conjunto de entrenamiento también será alto dado que la hipótesis no se está ajustando bien al conjunto de entrenamiento.   
Y si tienes un problema de varianza alta, tu error del conjunto de entrenamiento será generalmente bajo; es decir, mucho más bajo que el error de validación cruzada.

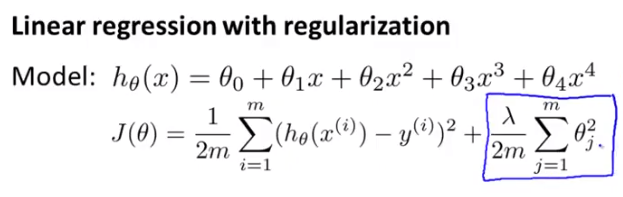




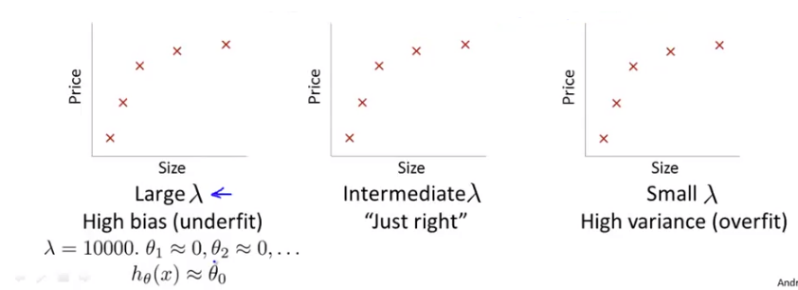
Ojalá esto te de un mejor entendimiento de los problemas de varianza y sesgo. Aún te tengo más información del sesgo y la varianza en los siguientes videos. Pero lo que veremos después es que, al diagnosticar, veremos si un algoritmo de aprendizaje sufre de una varianza o un sesgo alto. Les mostraré más detalles de cómo hacer esto en los videos siguientes. Veremos que al encontrar si un algoritmo de aprendizaje sufre de un sesgo alto o una varianza alta o una combinación de ambos, tendremos una mejor guía de cuáles son las cosas más prometedoras que podemos intentar para mejorar el desempeño de un algoritmo de aprendizaje.

# Regularización del sesgo/varianza

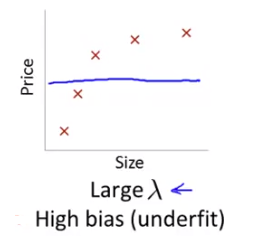
Ya has visto cómo la regularización nos puede ayudar a evitar el sobreajuste, pero ¿cómo afecta la oscilación y la varianza de un algoritmo de aprendizaje? En este video, me gustaría ahondar en el tema de oscilación y varianza y hablar acerca de cómo interactúan y se ven afectadas por la regularización de tu algoritmo de aprendizaje.   
  
Supongamos que ajustamos un modelo modelo de regresión lineal con un polinomio de alto grado, pero para evitar el sobreajuste, vamos a regularizarlo como se muestra aquí. Imaginemos que estamos ajustando un polinomio de alto orden como el que se muestra aquí, pero para evitar el sobreajuste, utilizaremos la regularización como la que se muestra aquí.



De manera que tenemos un término de regularización para intentar mantener bajos los valores de los parámetros. Consideremos estos tres casos.

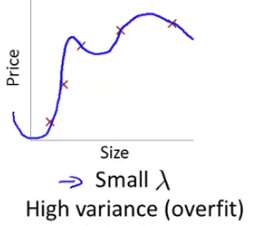


En el primero es el caso con un valor muy alto del parámetro de regularización «lambda», como si «lambda» fuera igual a 10,000. Es un valor enorme. En este caso, todos estos parámetros, «theta» 1, «theta» 2, «theta» 3, etc., estarán muy penalizados y terminaremos con la mayoría de estos parámetros cercanos a cero y la hipótesis será aproximadamente h(x)  justo igual o aproximadamente igual a «theta» 0. Terminaremos con una hipótesis que lucirá más o menos así; como una línea más o menos recta y constante.

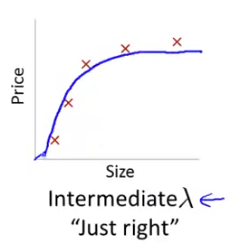


Entonces, esta hipótesis tiene una oscilación alta y subajusta el conjunto de datos. Esta línea recta horizontal no es un buen modelo para este conjunto de datos.

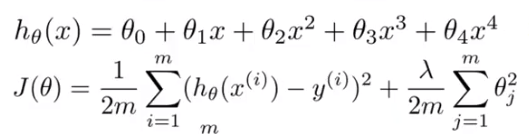
En el otro extremo tenemos un valor muy pequeño de «lambda», como si «lambda» fuera igual a 0. En este caso, debido a que estamos ajustando un polinomio de alto orden, resulta generalmente en una situación de sobreajuste. En este caso, debido a que estamos ajustando un polinomio de alto orden, sin regularización o con una regularización mínima, terminaremos con nuestra situación usual de varianza alta o de sobreajuste, porque «lambda» es igual a cero y estamos ajustandola con nuestra regularización para que sobreajuste la hipótesis.



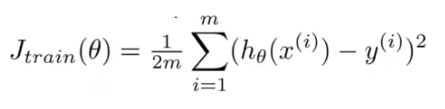
Y sólo si tenemos un valor intermedio de «lambda», es decir, ni muy alto ni muy bajo, tendremos unos parámetros «theta» que nos darán un ajuste razonable con estos datos.



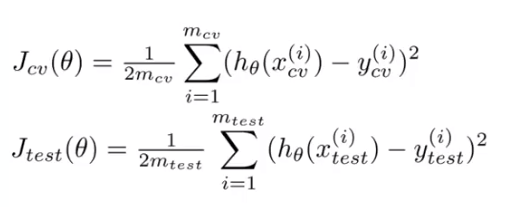
Entonces, ¿Cómo podemos elegir automáticamente un buen valor para el parámetro de regularización «lambda»?   
  
Sólo para recapitular, aquí tenemos nuestro modelo y nuestro objetivo del algoritmo de aprendizaje.



Para la situación en la que utilizamos la regularización, definiré “J entrenamiento” de «theta» como algo diferente como el objetivo de optimización pero sin el término de regularización.

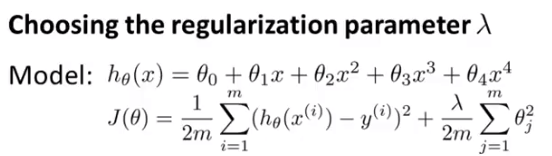


Anteriormente, en otro video, cuando no utilizamos la regularización, definí “J entrenamiento” de «theta» como igual a “J” de «theta» o la función de costo, pero cuando utilizamos la regularización con este término «lambda» adicional definiremos “j entrenamiento” o el error del conjunto de aprendizaje como la suma de los errores cuadráticos en el conjunto de aprendizaje o el error cuadrático promedio del conjunto de entrenamiento, sin tomar en cuenta el término de regularización. De manera similar, definiré también el error de validación cruzada o el error del conjunto de prueba como antes, para que sea la suma promedio de los errores cuadráticos en los conjuntos de prueba y de validación cruzada.

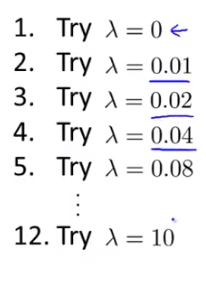


Para resumir, mis definiciones de “J entrenamiento”, “Jcv” y “J prueba” son sólo el error cuadrático promedio o un medio del error cuadrático promedio de mi conjunto de validación, entrenamiento y conjuntos de prueba sin el término de regularización adicional. Así es como podemos elegir automáticamente el parámetro de regularización «lambda».

Lo que hago usualmente es tener un rango de valores de «lambda» para probarlos.

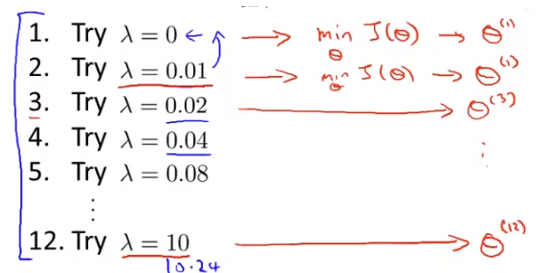


Puedo considerar no utilizar la regularización o intentar algunos valores que he considerado como:

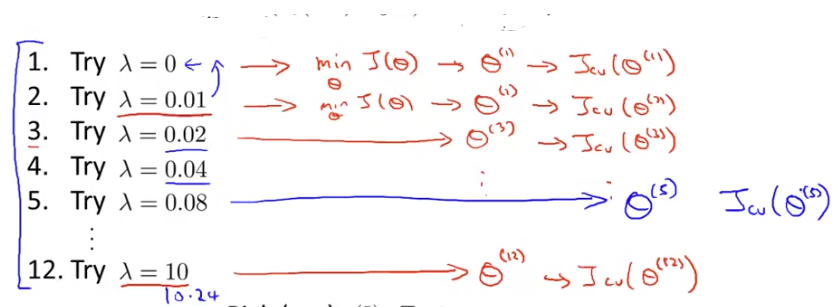


Usualmente escalo en múltiplos de dos hasta llegar a un valor más alto. Si lo hago con múltiplos de dos terminaré con 10.24 en vez de 10 exactamente, pero está suficientemente cerca y unos puntos decimales no afectarán mucho el resultado.   
  
Esto me arroja, tal vez, doce modelos diferentes que intentaré seleccionar en correspondencia con 12 valores diferentes del parámetro de regularización «lambda». Por supuesto, puedes tener valores menores de 0.01 o valores mayores a 10. pero yo lo he reducido por conveniencia.   
Con estos 12 modelos, podemos hacer lo siguiente:   
Podemos tomar este primer modelo donde «lambda» es igual a 0, y minimizar la función de costos de “J” de «theta». Esto nos daría un parámetro vector «theta». Al igual que en el video anterior, denotaré esto como «theta» superíndice 1.

Luego puedo tomar mi segundo modelo con «lambda» igual a 0.01 y minimizar la función de costo usando ahora «lambda» igual a 0.01. para obtener un vector parámetro «theta» diferente que denotaremos como «theta» 2. Con esto, terminaré con «theta» 3 si es correcto para mi tercer modelo, y así sucesivamente, hasta mi último modelo, designado como «theta» 12, donde «lambda» está determinado como 10, o 10.24.

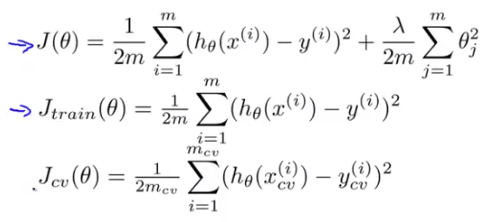


Ahora, puedo tomar todas estas hipótesis, todos estos parámetros, y utilizar mi conjunto de validación cruzada para evaluarlos. Puedo ver mi primer modelo o mi segundo modelo ajustado a los diferentes valores del parámetro de regularización y evaluarlos en mi conjunto de validación cruzada. Aquí básicamente mido el error cuadrático promedio de cada uno de los parámetros vector «theta» en mi conjunto de validación cruzada. Después elegiré de entre estos 12 modelos el que me de el error más bajo en el conjunto de validación cruzada.   
Digamos que, por ejemplo, elijo «theta» 5, el polinomio del quinto orden, porque tiene el error de validación más bajo.



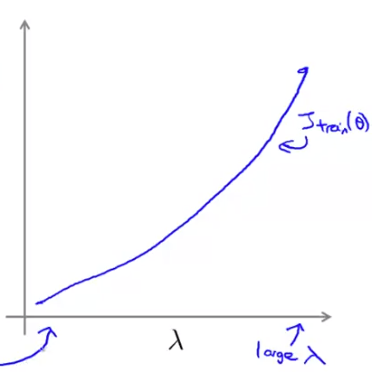
Una vez hecho esto, lo que haré, finalmente, si quiero reportar el error del conjunto de prueba es tomar el parámetro «theta» 5 que seleccioné y evaluar qué tan bien se desempeña en mi conjunto de prueba. Una vez más, aquí ajustamos este parámetro «theta» al conjunto de validación cruzada. Por esto guardamos otro conjunto de prueba aparte que utilizaremos para obtener un mejor estimado de qué tan bien se generalizará mi parámetro vector «theta» con ejemplos que no se han visto anteriormente.   
  
Esto fue la selección de modelos aplicada a la elección del parámetro de regularización «lambda». Lo último que quisiera hacer en este video es obtener un mejor entendimiento de cómo varían los errores de validación cruzada y de entrenamiento a medida que variamos el parámetro de regularización «lambda».

Sólo como resumen, esta era nuestra función de costo original “J” de «theta», pero para este caso definiremos el error de entrenamiento sin utilizar el parámetro de regularización y el error de validación cruzada sin utilizar el parámetro de validación.



Lo que me gustaría hacer ahora es trazar “J entrenamiento” y trazar este “Jcv”; es decir, ver qué tan bien se desempeña mi hipótesis en el conjunto de entrenamiento y qué tan bien se desempeña mi hipótesis en el conjunto de validación cruzada a medida que varío mi parámetro de regularización «lambda».

Entonces, como vimos anteriormente, si «lambda» es baja, entonces no estaremos aplicando una gran regularización y correremos un riesgo más alto de sobreajuste. Mientras que si «lambda» es alta; es decir, si estuviéramos en el extremo derecho de este eje horizontal, correremos un riesgo alto de tener un problema de oscilación, por el valor alto de «lambda». Si trazas “J entrenamiento” y “Jcv”, lo que encontrarás es que, con valores pequeños de «lambda», puedes ajustar el conjunto de entrenamiento relativamente bien porque no estás usando regularización. Con valores pequeños de «lambda», el término de regularización básicamente desaparece y minimizas, solamente, el error cuadrático. Así que, cuando «lambda» es pequeña, terminaremos con un valor pequeño de “J entrenamiento” mientras que si «lambda» es alta, entonces tendremos un problema de alta oscilación y no estaremos ajustando bien nuestro conjunto de entrenamiento. Así que terminan con un valor acá, hasta arriba. Así, “J entrenamiento” de «theta» tenderá a aumentar cuando «lambda» aumenta, porque un valor alto de «lambda» corresponde a una oscilación alta y quizá no puedas ajustar bien tu conjunto de entrenamiento, mientras que un valor bajo de «lambda» corresponde a que puedes ajustar libremente polinomios de un grado alto a tu conjunto de datos.



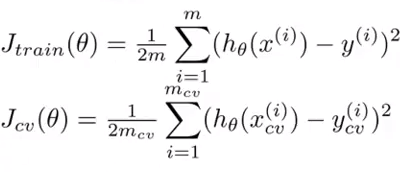
En cuanto al error de validación cruzada, nos encontramos con una figura como esta.   
Dónde si tenemos un valor alto de «lambda» aquí a la derecha, quizá generaremos un subajuste. Por lo tanto, este es un régimen de oscilación.



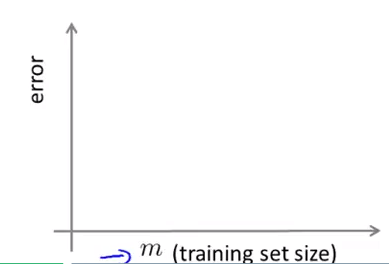
Aquí, el error de validación cruzada será alto. Entonces, con una oscilación alta no tendremos buenos resultados en el conjunto de validación cruzada. A la izquierda tenemos un régimen de varianza alta donde si tenemos un valor muy pequeño de «lambda», sobreajustaremos los datos. Al sobreajustar los datos, el error de validación también será alto.   
Así es como se pueden ver los errores de validación y de entrenamiento en un conjunto de entrenamiento a medida que variamos el parámetro «lambda» o el parámetro de regularización «lambda». Una vez más, será un valor intermedio de «lambda» el que funcione bien, o se adapte mejor cuando tenemos un error de validación o un conjunto de prueba pequeños. Las curvas que he dibujado son caricaturescas e idealizadas, de alguna manera, pero en un conjunto de datos real las curvas que obtendría serían un poco más irregulares o con más ruido que estas. En algunos conjuntos de datos podrás ver estos cuatro tipos de tendencias y al mirar el trazo del error de validación cruzada podrás seleccionar, manual o automáticamente, un punto que minimice el error de validación cruzada y seleccionar el valor de «lambda» que corresponda al error de validación cruzada más bajo. Cuando intento elegir el parámetro de regularización «lambda» para un algoritmo de aprendizaje, seguido veo que trazar una figura como esta que acabo de hacer, me ayuda a entender mejor qué es lo que está pasando y a verificar que de hecho estoy eligiendo un buen valor para el parámetro de regularización «lambda». Con suerte, esto te dará un mayor entendimiento de la regularización y sus efectos en la oscilación y la varianza de un algoritmo de aprendizaje.   
  
Para este momento ya has visto la oscilación y a la varianza desde muchas perspectivas diferentes. Ahora, lo que me gustaría hacer en el video siguiente es tomar el conocimiento que hemos adquirido y elaborarlo para generar un diagnóstico llamado curva de aprendizaje que es una herramienta que utilizo para diagnosticar si un si el algoritmo tiene un de un problema de oscilación o de varianza o un poco de ambos.

# Learning curves

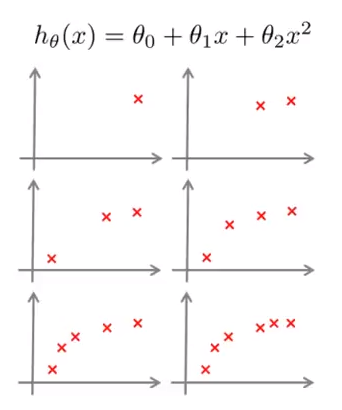
En este video, me gustaría hablarles de las curvas de aprendizaje.   
Trazar una curva de aprendizaje es, a menudo, muy útil. Si quieres verificar que tu algoritmo funcione de manera correcta o si quieres mejorar el desempeño del algoritmo.   
  
Una curva de aprendizaje es una herramienta que utilizo mucho para diagnosticar si un algoritmo de aprendizaje en particular sufre de alta oscilación , varianza alta o un poco de ambos.   
  
A continuación definiré la curva de aprendizaje. Para trazar una curva de aprendizaje, lo que hago generalmente es trazar “J entrenamiento” que es, digamos, el error cuadrático promedio de mi conjunto de entrenamiento, y “Jvc” que es el error cuadrático promedio en mi conjunto de validación cruzada.



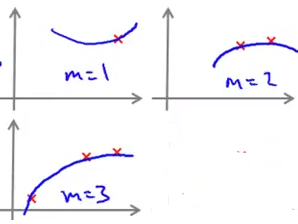
Luego trazaré esto como una función de “m”; es decir, una función del número de mis ejemplos de entrenamiento. “M” será, generalmente, una constante, como, quizá, 100 ejemplos de aprendizaje. Lo que haré realmente es reducir artificialmente el tamaño de mi conjunto de entrenamiento; es decir, limitarme intencionalmente a usar sólo, digamos, 10, 20, 30 o 40 ejemplos de entrenamiento y trazar el error de entrenamiento y el error de validación cruzada para los conjuntos de entrenamiento más pequeños.



Ahora, veamos cómo se ven estos trazos.

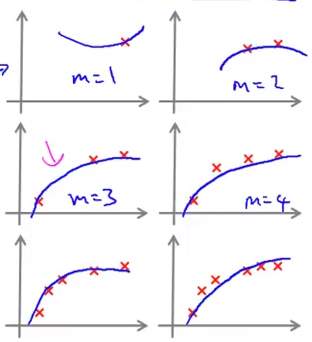


Supongamos que sólo tenemos un ejemplo de entrenamiento, como el que se muestra en este primer ejemplo, y digamos que estoy ajustando una función cuadrática. Bien, sólo tengo un ejemplo de entrenamiento. Seré capaz de realizar un ajuste perfecto ¿si? Si ajusto la función cuadrática Tendré un error de 0 en este único ejemplo de entrenamiento. Si tuviera dos ejemplos de entrenamiento la función cuadrática también se ajustaría bien. Aún   
si estuviera usando la regularización, probablemente podría ajustar la función bastante bien o aún sin utilizar la regularización, probablemente ajustaría la función perfectamente. Ahora, si tuviera tres ejemplos de entrenamiento podría ajustar la función cuadrática perfectamente como se ve en la siguiente figura:

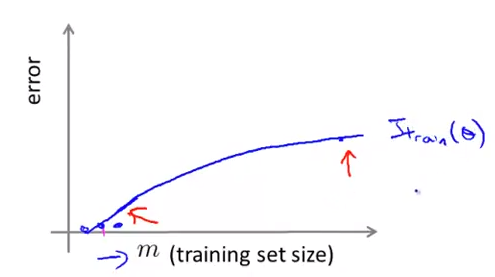


Entonces si “m” es igual a 1, o “m” es igual a 2 o “m” es igual a 3, mi error de entrenamiento en mi conjunto de entrenamiento será de 0, suponiendo que no estoy utilizando la regularización, o un poco más alto que 0 si suponemos que estoy utilizando la regularización. Si tengo un conjunto de entrenamiento grande y limito artificialmente el tamaño de mi conjunto de entrenamiento para trazar “J entrenamiento”. Si establezco “m” igual a 3 y si lo entreno en sólo tres ejemplos solamente, entonces para esta figura, mediré mi error de entrenamiento sólo en los tres ejemplos en los que ajusté mis datos.   
Y aún si tuviera 100 ejemplos de entrenamiento pero quiero trazar el error de entrenamiento con “m” igual a tres lo que haré es medir mi error de entrenamiento solamente en los tres ejemplos a los que ajusté para mi hipótesis. Y en todos los otros ejemplos que deliberadamente omití del proceso de entrenamiento hasta llegar a un error lo suficientemente estable.

Para recapitular, hemos visto que si el tamaño del conjunto de entrenamiento es pequeño, entonces el error de entrenamiento también será pequeño porque si tenemos un conjunto de entrenamiento pequeño será más fácil ajustar bien, o incluso perfectamente, el conjunto de entrenamiento. En cambio, si tenemos que “m” es igual a 4 entonces la función cuadrática ya no se ajustará perfectamente a este conjunto de datos y si tenemos que “m” es igual a 5, entonces quizá la función cuadrática no se ajuste tan bien. A medida que el conjunto de entrenamiento se hace más grande será cada vez más difícil asegurarse de que encontraré una función cuadrática que se ajuste perfectamente a todos mis ejemplos.

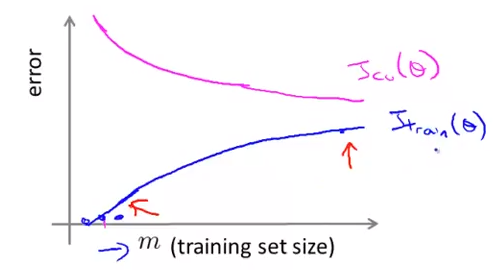


Ahora, conforme crece el conjunto de entrenamiento encontraremos que el error de entrenamiento promedio en realidad aumenta y si trazas esta figura, lo que encontrarás será que el error del conjunto de entrenamiento; es decir, el error promedio en la hipótesis, crece a medida que “m” crece.

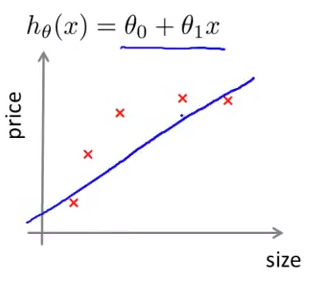


Para aclarar, el concepto, cuando “m” es pequeña o cuando tenemos pocos ejemplos de entrenamiento, es muy fácil ajustar perfectamente cada uno de los ejemplos de entrenamiento por lo que el error será pequeño, mientas que a medida que "m" crece, será más difícil ajustar perfectamente todos los ejemplos de entrenamiento y, por lo tanto, el error del conjunto de aprendizaje se hace más grande.

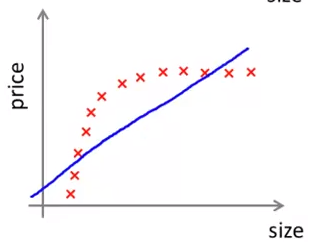
¿Qué pasa con el error de validación cruzada? El error de validación cruzada es el error en el conjunto de validación cruzada que ya hemos visto. Cuando tenemos un conjunto de entrenamiento muy pequeño, no podrá generalizarse bien. Simplemente no se desempeñará bien y esta hipótesis de aquí no se verá como una buena hipótesis. Sin embargo, si tenemos un conjunto de aprendizaje más grande empezaremos a tener hipótesis que se ajustan mejor a los datos. De manera que el error de validación cruzada y el error del conjunto de prueba tenderán a disminuir a medida que aumenta el tamaño del conjunto de entrenamiento porque, entre más datos tengamos, mejor será la generalización con ejemplos nuevos. Es decir, entre más datos tengamos, la hipótesis se ajustará mejor. Si trazas “J entrenamiento” y “Jcv”, este es el tipo de resultado que obtienes.



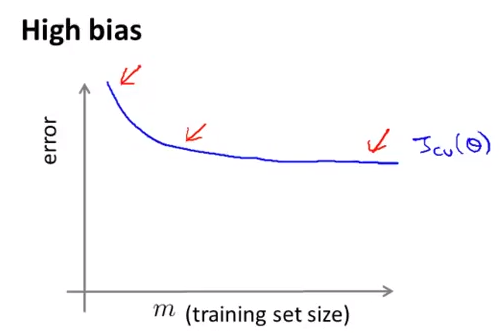
Ahora veamos cómo se verían las curvas de aprendizaje si tenemos ya sea un problema de alta oscilación o alta varianza. Supongamos que tu hipótesis tiene un alta oscilación. Para explicar esto pondré un ejemplo ajustando una línea recta a datos que realmente no se pueden ajustar con una línea recta. Así, terminaremos con una hipótesis que quizá se vea como esta.



Ahora pensemos qué pasaría si aumentamos el tamaño del conjunto de entrenamiento. Imaginemos que en vez de cinco ejemplos, como lo he trazado aquí, tenemos muchos más.   
Lo que pasaría o lo que encontrarías si ajustas una línea recta a esto, sería una línea igual de recta. Una línea recta simplemente no puede ajustar estos datos y obtener una tonelada de datos adicionales simplemente no cambiará mucho la situación. Esta es la línea recta que mejor se ajusta a estos datos, pero una línea recta simplemente no puede ajustar bien estos datos.

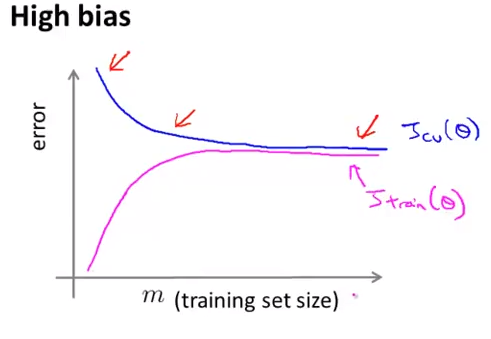


Aún si trazas el error de validación cruzada, se vería así.



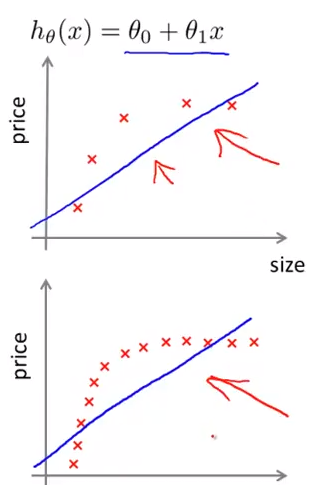
Aquí arriba a la izquierda, si tienes un conjunto de entrenamiento minúsculo con, digamos, un sólo ejemplo de entrenamiento, no resultará bien. Para cuando alcances un cierto número de ejemplos o de ejemplos de entrenamiento, has ajustado casi la mejor línea recta posible y aún si terminas con un conjunto de entrenamiento mayor o un valor de “m” mucho mayor, obtendrás básicamente la misma línea recta. Por lo tanto, el error de validación cruzada, o el error del conjunto de prueba se aplanarán o se harán rectos muy pronto una vez que has pasado cierto número de ejemplos de aprendizaje o has ajustado la mejor línea recta posible.

Y ¿qué hay el error de aprendizaje? El error de aprendizaje será pequeño de nuevo. Lo que encontrarás en un caso de alta oscilación es que el error de entrenamiento terminará muy cerca del error de validación. Debido a que tenemos muy pocos parámetros y muchos datos, por lo menos cuando “m” tiene un valor alto, el desempeño del conjunto de entrenamiento y el conjunto de validación cruzada serán muy similares. Así es como lucirán las curvas de aprendizaje si tenemos un algoritmo con alta oscilación.



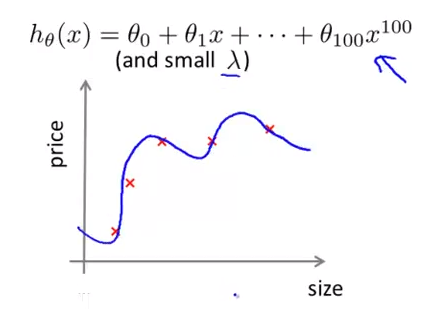
Finalmente, el problema de alta oscilación se ve reflejado en el hecho de que ambos, el error de validación cruzada y el error de entrenamiento son altos y terminaremos con un valor relativamente alto de “Jcv” y de “j entrenamiento”.   
  
Esto también implica algo muy interesante: Si un algoritmo de aprendizaje tiene un alta oscilación , a medida que tenemos más ejemplos de entrenamiento, es decir, que nos movemos hacia la derecha esta figura, nos daremos cuenta de que el error de validación cruzada no disminuye mucho sino que más bien permanece plano. Por lo tanto, si los algoritmos de aprendizaje tienen una oscilación realmente alta, el hecho de obtener más datos de entrenamiento no ayudará mucho.

Un ejemplo concreto de esto es, en la figura

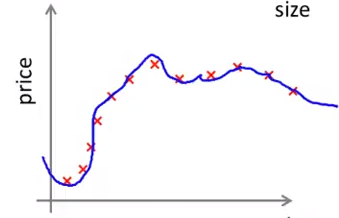


Empezamos sólo cinco ejemplos de entrenamiento y ajustamos cierta línea recta y después tuvimos muchos más datos de entrenamiento y aún así terminamos con la misma línea recta. Si un algoritmo de aprendizaje tiene un alta oscilación , darle más datos de entrenamiento no ayudará a tener un error de validación cruzada o un error del conjunto de prueba más bajo. Saber si tu algoritmo de aprendizaje sufre de un sesgo alto es útil porque puede evitar que pases recolectando más ejemplos de entrenamiento datos de entrenamiento cuando quizá no sean útiles.

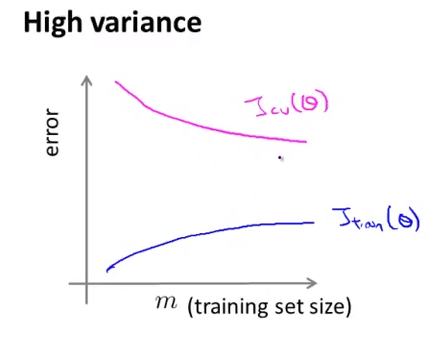
A continuación veremos la situación de un algoritmo de aprendizaje con una varianza alta.   
  
Veamos, primero, el error de entrenamiento. Si tienes un conjunto de entrenamiento muy pequeño con, digamos, cinco ejemplos, como se muestra en la figura, y estas ajustando un polinomio de alto grado ,aquí escribí un polinomio del 100mo grado que nadie utiliza realmente, sólo como ejemplo, con un valor de «lambda» relativamente pequeño que quizá no llegue a cero, pero sí un valor pequeño para «lambda», entonces acabaremos ajustando mejor estos datos que con una función que causa sobreajuste.



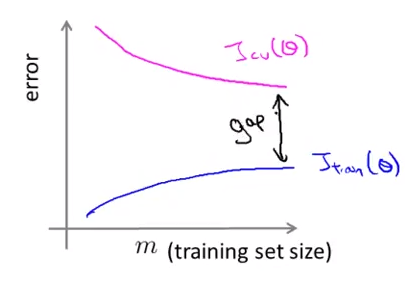
Entonces, si el conjunto de entrenamiento es pequeño, nuestro error de entrenamiento, es decir, “J entrenamiento” de «theta» también será pequeño.   
  
A medida que este conjunto de aprendizaje aumenta, quizá sigamos sobreajustando estos datos un poco y se volverá cada vez más difícil ajustar este conjunto de datos perfectamente y a medida que crece el conjunto de entrenamiento, encontraremos que “j entrenamiento” también se incrementa y será un poco más difícil ajustar perfectamente el conjunto de entrenamiento cuando tenemos más ejemplos, aunque el error del conjunto de entrenamiento será aún muy bajo.



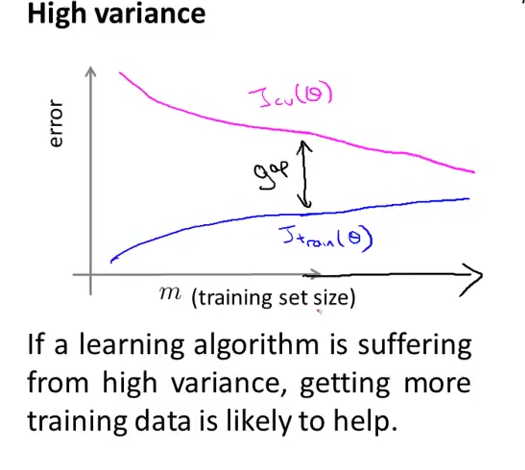
Pero ¿Qué pasa con el error de validación cruzada? En una situación de alta varianza, la hipótesis se sobreajusta, por lo que nuestro error de validación cruzada se mantendrá alto, aún si tenemos un número moderado de ejemplos de entrenamiento. El error de validación cruzada, entonces, se vería así.



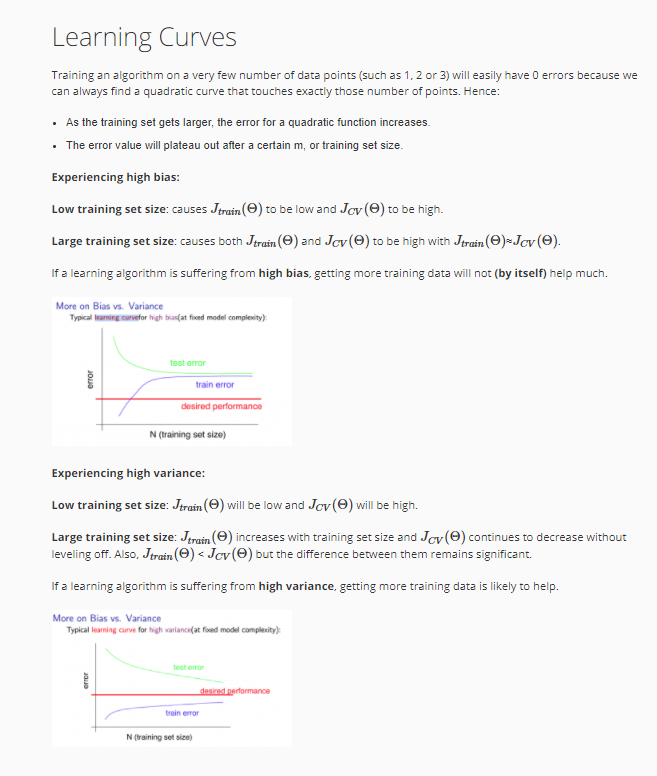
Un diagnóstico indicativo de que tenemos un problema de alta varianza es el hecho de tener este gran espacio entre el error de entrenamiento y el error de validación cruzada.



Al mirar esta figura, si pensamos en añadir más datos de entrenamiento, es decir, si tomamos esta figura y la extrapolamos a la derecha, podemos predecir que las dos curvas, la curva azul y la curva magenta, convergerá una con la otra y si extrapolamos esta figura a la derecha, parece probable que el error de entrenamiento seguirá subiendo y que el error de validación cruzada seguirá bajando.



Lo que realmente nos importa es el error de validación cruzada o el error del conjunto de aprendizaje ¿cierto? En este tipo de figuras, podemos predecir que si seguimos añadiendo ejemplos de entrenamiento y extrapolándolos a la derecha, nuestro error de validación cruzada seguirá disminuyendo. Por lo tanto, es probable que en una situación de varianza alta, obtener más datos de entrenamiento sea, efectivamente, de ayuda. De nuevo, saber si tu algoritmo tiene un problema de varianza alta es útil porque indica que puede valer la pena averiguar si puedes obtener más datos de entrenamiento.   
  
Ahora, en la diapositiva anterior y en esta dibujé curvas muy idealizadas y limpias. Si trazas estas curvas en un algoritmo de aprendizaje real, lo que verás en son curvas, como las que dibujé aquí, pero algunas veces serán más irregulares y con más ruido que estas. Trazar las curvas de aprendizaje, como estas, puede decirte o puede ayudarte a saber si tu algoritmo de aprendizaje sufre de un alta oscilación o una alta varianza o un poco de los dos. Algo que siempre hago cuando intento mejorar el desempeño de un algoritmo de aprendizaje, es trazar estas líneas de aprendizaje. Esto, generalmente, te dará una mejor idea de si hay un problema de oscilación o de varianza.   
  
En el siguiente video veremos cómo esto puede sugerir acciones específicas que se deben realizar o no para mejorar el desempeño de un algoritmo de aprendizaje.



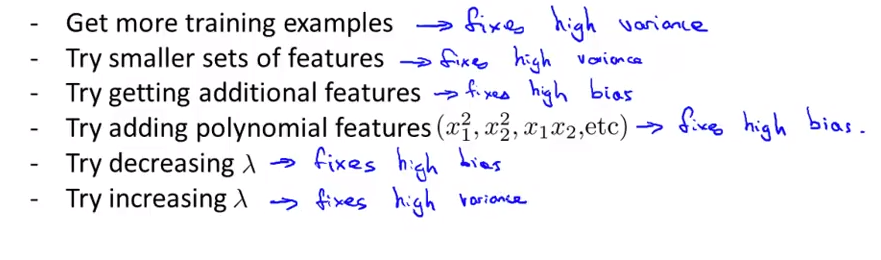
Decidiendo el siguiente paso a realizar

Ya hemos hablado de cómo evaluar los algoritmos de aprendizaje, de la selección de modelos y hemos hablado mucho acerca de oscilación y varianza. Ahora, ¿cómo nos ayuda esto a entender cuáles son las vías más fructíferas que podemos seguir para mejorar el desempeño de un algoritmo de aprendizaje?

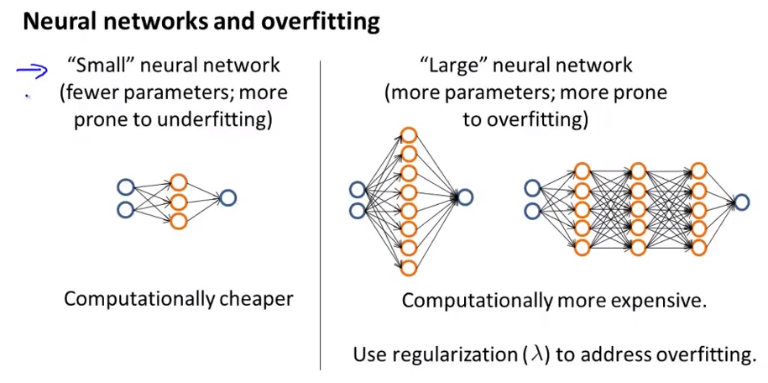


Tomemos de nuevo nuestro ejemplo original y busquemos el resultado. Aquí tenemos nuestro ejemplo anterior en el que ajustamos una regresión lineal regularizada y nos encontramos con que no funciona tan bien como esperábamos. Dijimos que teníamos un menú de opciones. ¿Hay alguna manera de saber cuál de estas opciones será efectiva?

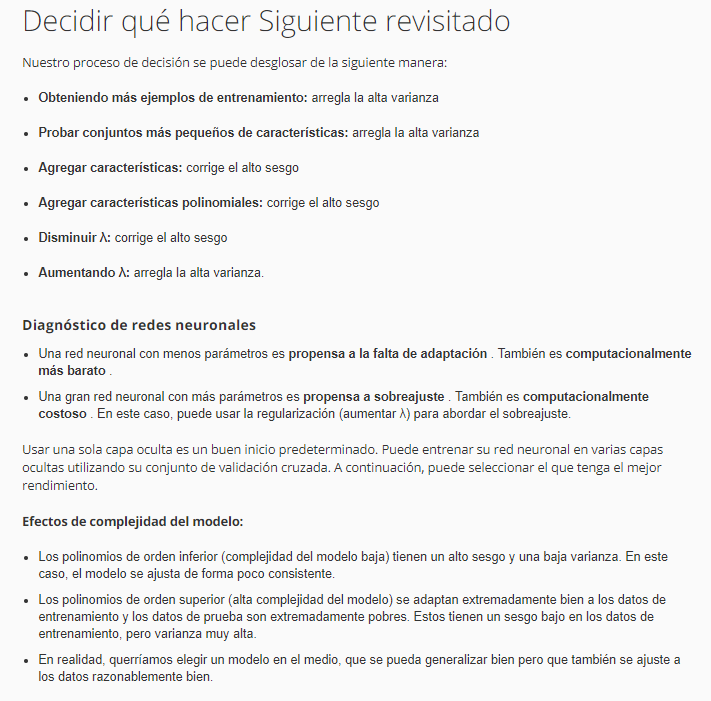
* Lo primero que intentamos fue obtener más ejemplos de entrenamiento. Esto resultó bien para corregir la varianza alta. Pero, por el contrario, si tienes un problema de alta oscilación y no de varianza alta, entonces, como vimos en el video anterior, obtener más ejemplos de entrenamiento no será de mucha ayuda. Esta primera opción, entonces, sólo es útil si trazas las curvas de aprendizaje y te das cuenta de que tienes un problema, aunque sea pequeño, de varianza alta; es decir, que el error de validación cruzada es mucho más grande que el error del conjunto de entrenamiento.
* Y ¿qué pasa si intentamos con un conjunto de características más pequeño? Cuando intentamos con un conjunto de características más pequeño, de nuevo, podemos corregir la varianza alta. En otras palabras, si te das cuenta, al ver las curvas de aprendizaje o algún otro indicador que hayas usado, que tienes un problema de oscilación alto, por favor, no pierdas tu tiempo intentando seleccionar cuidadosamente un conjunto de variables más pequeño, porque si tienes un problema de alta oscilación, utilizar menos funciones no ayudará. Por el contrario, si ves las curvas de aprendizaje o algún otro indicador y te das cuenta que tienes un problema de varianza alta, entonces, intenta seleccionar un conjunto de funciones más pequeño. En este caso sí valdría la pena invertir tu tiempo.
* ¿Qué pasa si añadimos variables adicionales? Pensamos que añadir variables es una solución para corregir problemas de alta oscilación . Si añades variables adicionales es, generalmente, porque tu hipótesis actual es muy simple. Por lo tanto, intentamos añadir funciones para que nuestra hipótesis sea más capaz de ajustar el conjunto de entrenamiento.
* Igualmente, añadir funciones de polinomios es otra manera de añadir variables y otra manera de corregir el problema de alta oscilación . Por el contrario, si las curvas de aprendizaje te muestran que tienes un problema de varianza alta, entonces, quizá esto no sea un buen uso para tu tiempo.
* Y finalmente, la disminución y el aumento de «lambda». Intentar esto es fácil y rápido. Es menos probable que esto represente tiempo perdido. Unos meses de tu vida. Disminuir «lambda», como ya lo sabes, corrige el alta oscilación . Si todavía no tienes claro esto, te recomiendo que pongas pausa al video y pienses y te convenzas de que disminuir «lambda» corrige el alta oscilación , mientras que incrementar «lambda» corrige la varianza alta. Si no estás seguro de por qué pasa esto, pon pausa al video y asegúrate de convencerte de que así es o mira las curvas que trazamos al final del video anterior y asegúrate de entender por qué resultan así.



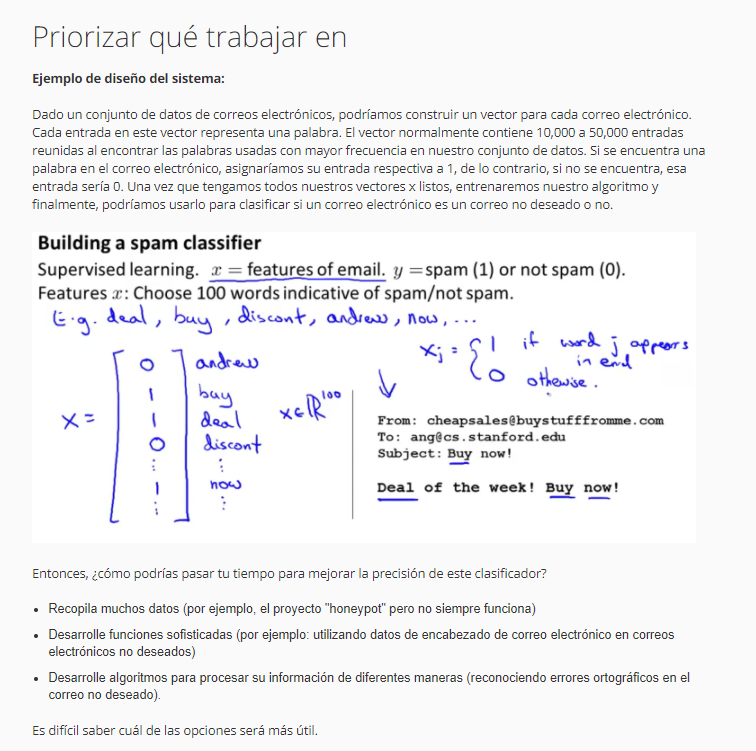
Finalmente, veamos todo lo que hemos aprendido y relacionemos con las redes neuronales. Aquí tenemos evidencias prácticas de cómo elegir generalmente la arquitectura o el patrón de conectividad de las redes neuronales que utilizamos.

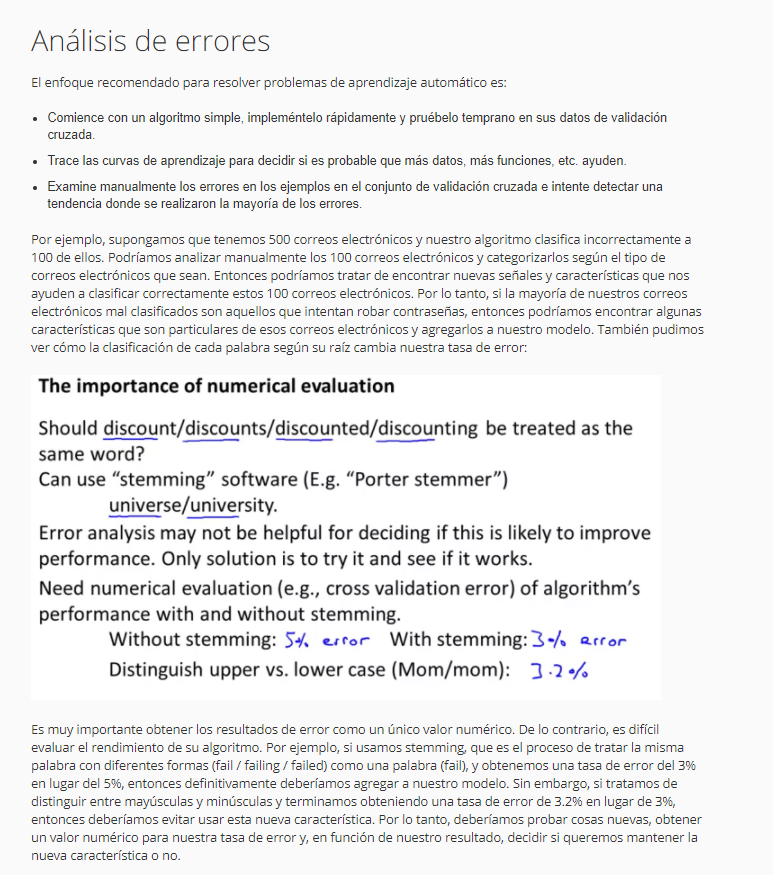


Si estás ajustando una red neuronal, una opción sería ajustar, digamos, una red neuronal pequeña con relativamente pocas unidades ocultas. Quizá sólo una unidad oculta. Si estás ajustando una red neuronal, una opción sería ajustar una red neuronal relativamente pequeña con pocas o quizá sólo una capa oculta y tal vez un número relativamente bajo de unidades ocultas. Una red como la de la izquierda tendría relativamente pocos parámetros y sería menos propensa al subajuste. La ventaja principal de estas pequeñas redes neuronales es que su cálculo computacional es más barato.   
  
Una alternativa sería ajustar una red neuronal relativamente grande con, ya sea, más unidades ocultas, o más capas ocultas.   
Estas redes neuronales tienden a tener más parámetros y, por lo tanto, son más propensas al sobreajuste.   
Una desventaja, que generalmente no es grave, pero es algo en lo que debemos pensar, es que si tienes un número alto de neuronas en tu red, puede resultar más caro calcularlas. Aunque esto no es un gran problema, dentro de lo razonable. El problema potencial más importante de las redes neuronales muy grandes es que pueden ser más propensas al sobreajuste y resulta que si aplicas una red neuronal, usar una red neuronal grande es mejor. Generalmente, entre más grande, mejor, Pero si se sobreajusta, puedes utilizar la regularización para corregir el sobreajuste. Con redes neuronales más grandes, utilizar la regularización para corregir el sobreajuste es, a veces, más efectivo que utilizar una red neuronal más pequeña. La desventaja principal es que puede ser más caro calcularlas.   
  
Finalmente, una de las otras decisiones es, digamos el número de capas ocultas que quieres tener ¿Cierto? Es decir, si quieres tener una capa oculta o quieres tener tres capas ocultas como mostramos aquí, o dos capas ocultas.   
Usualmente, como creo que mencioné en el video anterior, utilizar una sola capa oculta es un valor por defecto razonable, pero si quieres elegir el número de capas ocultas, otra cosa que puedes intentar es conseguir una división de conjuntos de entrenamiento, validación cruzada y de prueba e intentar entrenar redes neuronales con una capa oculta o dos capas ocultas o tres capas ocultas y ver cuál de esas redes neuronales se desempeña mejor en el conjunto de validación cruzada. Tomas tus tres redes neuronales con una, dos y tres capas ocultas y calcula el error de validación cruzada o el “Jcv” en todas ellas y utiliza esto para seleccionar la red neuronal que crees que sea la mejor.   
  
Eso es todo sobre la oscilación y la varianza, sobre los métodos, como las curvas de aprendizaje, para diagnosticar esos problemas, y sobre el impacto de esto en la determinación de si un método es útil o no para mejorar el desempeño de un algoritmo de aprendizaje.   
  
Si has entendido el contenido de estos últimos videos y si lo aplicas, eres, ahora, más efectivo para hacer que tus algoritmos de aprendizaje funcionen en problemas. Una gran parte o quizá la mayoría de los desarrolladores de aprendizaje automático aquí en Silicon Valley realizan estas actividades como su trabajo de tiempo completo.   
  
Espero que estos consejos sobre la oscilación, la varianza, las curvas de aprendizaje y el diagnóstico te ayuden a aplicar algoritmos de aprendizaje más efectivamente y a hacerlos funcionar mucho mejor.



# Extra

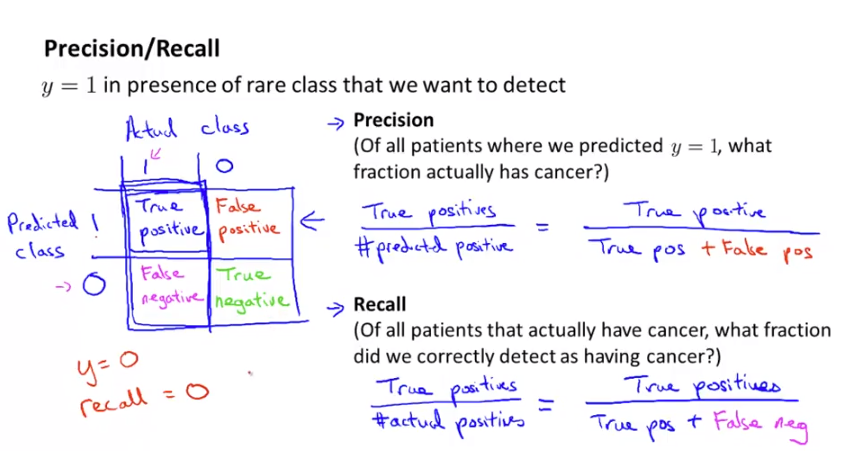
****

****

# Precision/Recall

En el video anterior hablé del análisis de errores y de la importancia de tener una métrica de errores; es decir, tener una métrica de evaluación con un número real simple para saber cómo se desempeña un algoritmo de aprendizaje.   
  
En el contexto de la evaluación y de la métrica de errores, hay un caso importante en el que es especialmente difícil obtener una métrica de errores o de evaluación apropiada para el algoritmo de aprendizaje. A este caso se le llama clases sesgadas. A continuación les diré que significa.   
  
Consideremos el problema de clasificación de cáncer, donde tenemos variables de pacientes médicos y queremos decidir si tienen cáncer o no. Este es como el ejemplo de clasificación de tumores malignos y benignos que vimos antes. Digamos que “y” es igual a 1 si el paciente tiene cáncer y “y” es igual a 0 si no tiene cáncer. Entrenaremos el clasificador de regresión logística y lo probaremos en el conjunto de pruebas y encontraremos un error del 1 por ciento. Por lo tanto, tenemos un 99% de diagnósticos correctos. Parece ser un resultado impresionante, ¿cierto? Estamos en lo correcto el 99% de las veces.   
  
Ahora, supongamos que sólo el 0.5% de los pacientes en nuestro conjunto de entrenamiento y de prueba tienen cáncer realmente; es decir, sólo medio punto porcentual de los pacientes en el proceso de selección tiene cáncer. En este caso, el 1% de error ya no parece tan impresionante.   
Aquí hay unas líneas de código, un código que no es de aprendizaje que toma las variables de entrada “x” y las ignora. Sólo ajusta “y” igual a 0 y siempre predice que nadie tiene cáncer. Este algoritmo arrojó un error del 0.5% que es mucho mejor que el 1% de error que estábamos obteniendo y este es un algoritmo que no es de aprendizaje y que predice que “y” es igual a 0 todo el tiempo. Este parámetro en el que la proporción de ejemplos positivos y negativos está muy cerca a alguno de los extremos, como en este caso, que el número de ejemplos positivos es mucho más pequeño que el número de ejemplos negativos porque “y” es igual a 1, esto es a lo que llamamos caso de clases sesgadas. Tenemos muchos más ejemplos para una clase que para la otra. Al predecir que “y” es igual a 0 todo el tiempo, o que “y” es igual a 1 todo el tiempo, el algoritmo puede desempeñarse muy bien. El problema cuando utilizamos el error de clasificación o la precisión de clasificación como nuestra métrica de evaluación es el siguiente:   
  
Digamos que tenemos un algoritmo que arroja una precisión del 99.2%. En otras palabras, un error de 0.8%. Digamos que haces algún cambio a tu algoritmo y ahora obtienes una precisión del 99.5%. Es decir, un error de 0.5%. ¿Esto sería una mejora para el algoritmo? Una de las ventajas de tener una métrica de evaluación con un número real simple es que nos ayuda a decidir rápidamente si necesitamos realizar cambios en el algoritmo o no. Al aumentar la precisión del algoritmo de 99.2% a 99.5% ¿hicimos algo útil, o simplemente reemplazamos nuestro código con algo que predice que “y” es igual a 0 más seguido? Si tienes clases muy sesgadas se vuelve mucho más difícil utilizar la precisión de la clasificación, porque puedes obtener una precisión de clasificación alta y un error bajo pero no siempre queda claro si hacer esto mejora la calidad de tu clasificador, porque predecir que “y” es igual a 0 todo el tiempo no es un buen clasificador. Cuando predecimos que “y” es igual a 0 más seguido, podemos reducir el error tanto como hasta un 0.5%. Cuando nos encontramos con estas clases sesgadas queremos encontrar una métrica de errores o una métrica de evaluación diferente. Una de esas métricas de evaluación es lo que llamamos precisión/recuperación.

A continuación explicaré qué es. Imaginemos que estamos evaluando un clasificador en el conjunto de prueba. Para este ejemplo, la clase real para el conjunto de prueba será ya sea cero o uno si es un problema de clasificación binaria. Lo que hará nuestro algoritmo de aprendizaje es predecir un valor para la clase. Nuestro algoritmo de aprendizaje predirá el valor de cada ejemplo en mi conjunto de prueba y el valor predicho también será ya sea uno o cero.



Permítanme dibujar una tabla de dos por dos, como esta, dependiendo de estas entradas, o de la clase real y la clase predicha. Si tenemos un ejemplo en el que la clase real es uno y la clase predicha es uno, entonces le llamaremos un ejemplo positivo verdadero. Esto quiere decir que nuestro algoritmo predijo que es positivo y de hecho el ejemplo es positivo. Si nuestro algoritmo predijo que algo es negativo, clase cero, y la clase real es cero, entonces a esto le llamamos negativo verdadero. Predijimos que sería cero y es realmente cero.   
En las otras dos celdas, si nuestro algoritmo predice que la clase es uno, pero la clase real es cero, entonces lo llamamos un falso positivo. Esto quiere decir que el algoritmo pensó que el paciente tiene cáncer cuando en realidad no lo padece. Y finalmente, la última casilla es cero-uno. A esto se le llama falso negativo porque nuestro algoritmo predijo un cero, pero la clase real fue de uno.   
  
Entonces, tenemos esta tabla de dos por dos basada en la clase real y la clase predicha.   
Esta es una manera distinta de evaluar el desempeño de nuestro algoritmo. Calcularemos dos números. Al primero se le llama precisión. Lo que nos indica es, de   
todos los pacientes a los que les diagnosticamos cáncer ¿qué fracción tienen cáncer realmente?   
  
Voy a escribir esto: la precisión de un clasificador es el número de positivos verdaderos dividido entre el número que predijimos como positivo. ¿sí?

De todos los pacientes a los que les dijimos “Creemos que usted puede tener cáncer”, ¿qué fracción realmente tiene cáncer? A esto se le llama precisión. Otra manera de escribir esto sería positivos verdaderos en el cociente el número de positivos predichos en el denominador. Esta sería la suma de las entradas en esta primera fila de la tabla. Entonces, tenemos los positivos verdaderos divididos entre los positivos, abreviaré positivo como "pos" más los falsos positivos, abreviando de nuevo positivo como "pos".   
  
A esto se le llama precisión. Como puedes ver, una precisión alta es deseable. Una alta precisión quiere decir que, de los pacientes a quienes les dijimos “creemos que tiene cáncer, lo sentimos mucho” la mayoría realmente tienen cáncer; por lo tanto, que hicimos las predicciones precisas acerca de ellos.   
  
El segundo número que calcularemos se llama recuperación. La recuperación nos dice, de todos los pacientes del conjunto de prueba o de validación cruzada, es decir, de todos los pacientes en el conjunto de datos que realmente tienen cáncer ¿en qué fracción de ellos detectamos el cáncer correctamente? Si todos los pacientes tienen cáncer, ¿a cuántos de ellos les dijimos correctamente que necesitan tratamiento?   
  
Ahora, escribamos esto: la recuperación se define como el número de positivos o el número de positivos verdaderos; es decir, el número de personas que tienen cáncer y que predijimos correctamente que tienen cáncer. Tomaremos esto y lo dividiremos entre el número de positivos verdaderos. Este será el número correcto de positivos verdaderos de todos los que sí sufren cáncer; en otras palabras, la fracción que marcamos y mandamos a tratamiento. Para escribir esto de manera diferente, el número real de positivos verdaderos estaría en el denominador y es la suma de las entradas de esta columna. Por lo tanto, este será el número de positivos verdaderos dividido entre el número de positivos verdaderos más el número de falsos negativos. Una vez más, tener una recuperación alta será bueno.   
Calcular la precisión y la recuperación nos dará un mejor entendimiento del desempeño de nuestro clasificador. De manera particular, si tenemos un algoritmo que predice que “y” es igual a 0 en todo momento; es decir, si predice que nadie tiene cáncer, este clasificador tendrá una recuperación igual a cero, porque no habrá ningún positivo verdadero. Esta es una manera rápida para reconocer que un clasificador que predice que “y” es igual a 0 todo el tiempo, no es un buen clasificador.

De manera más general, aún para las situaciones en las que tenemos clases muy sesgadas, no es posible posible que un algoritmo “haga trampa” y obtenga una precisión o una recuperación muy alta haciendo algo tan simple como predecir que “y” es igual a 0 o que “y” es igual a 1 todo el tiempo. Estamos mucho más seguros de que un clasificador con una precisión alta o una recuperación alta es un buen clasificador. Esto nos provee una métrica de evaluación mucho más útil y directa para entender si nuestro algoritmo se está desempeñando bien.   
  
Un último comentario: En la definición de precisión y recuperación, utilizaríamos la convención, es decir, “y” igual a 1, en presencia de la clase más rara. De manera que si intentamos detectar condiciones raras, como el cáncer, la precisión y la recuperación se definen con “y” igual a 1 en vez de igual a 0 para representar la presencia de esta clase rara que intentamos detectar. Utilizando la precisión y la recuperación nos encontramos con que lo que pasa, aún si tenemos clases muy sesgadas, es que no es posible que un algoritmo “haga trampa” y prediga que “y” es igual a 1 todo el tiempo o prediga que “y” es igual a 0 todo el tiempo, y que obtengamos una precisión y una recuperación altas. Particularmente, si un clasificador arroja una precisión y una recuperación alta, entonces tendremos la certeza de que nuestro algoritmo se está desempeñando bien, aún si tenemos clases muy sesgadas.   
  
Para el problema de clases sesgadas, la precisión y la recuperación nos dan un entendimiento más directo sobre cómo se desempeña el algoritmo de aprendizaje y, a veces, es una manera mucho más efectiva de evaluar nuestros algoritmos de aprendizaje que el análisis del error de clasificación o la precisión de clasificación, cuando las clases están muy sesgadas.

Trading off precision and recall

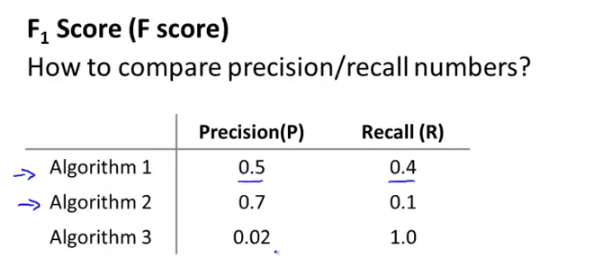
En el último vídeo hablamos hablamos de la precisión y la recuperación como métricas de evaluación para los problemas de clasificación con clases sesgadas. Para algunas aplicaciones, queremos controlar, de alguna manera, la compensación entre la precisión y la recuperación. Ahora les diré cómo hacerlo y les mostraré maneras aún más efectivas de utilizar la precisión y la recuperación como una métrica de evaluación para algoritmos de aprendizaje.   
  
Como recordatorio, aquí tenemos las definiciones de precisión y recuperación del video anterior. Continuemos con nuestro ejemplo de clasificación del cáncer en el que “y” es igual a 1 si el paciente tiene cáncer y “y” es igual a 0 si el paciente está sano, y digamos que entrenamos un clasificador con regresión logística que nos arroja posibilidades de entre cero y uno. Como de costumbre, supondremos que “y” es igual a 1 si “H” de “X” es mayor o igual a 0.5 y prediremos que “y” es igual a 0 si la hipótesis arroja un valor menor a 0.5. Este clasificador puede darnos un valor para la precisión y un valor para la recuperación.

Por ahora, supongamos que queremos predecir que un paciente tiene cáncer sólo si estamos seguros de que en realidad lo padece, porque si vas con un paciente y le dices que tiene cáncer, tendrá un impacto enorme Realmente son muy malas noticias para el paciente y puede acabar soportando un proceso y un tratamiento muy dolor. Por lo tanto, sólo le queremos decir a un paciente que creemos que tiene cáncer cuando estamos muy seguros.   
  
Una manera de hacer esto es modificar el algoritmo para que, en vez de fijar el umbral en 0.5, supondremos que “y” es igual a 1 sólo si “H” de “X” es igual o mayor que 0.7. Entonces, sólo le diremos a alguien que tiene cáncer si hay una probabilidad igual o mayor al 70% de que lo padece. Si haces esto, puedes predecir el cáncer sólo cuando estás seguro y, por lo tanto, obtendrás un clasificador con una mayor precisión. Por esto, los pacientes a quienes les des la noticia de que tienen cáncer estarán seguros de que realmente lo padecen. En otras palabras, una fracción más alta de los pacientes que predijiste que tendrían cáncer tendrán cáncer realmente, porque cuando hiciste estas predicciones dejaste un margen de seguridad muy cerrado. Por el contrario, este clasificador tendrá una recuperación baja, porque ahora haremos una predicción de que “y” es igual a 1 en un número menor de pacientes. Podemos llevar esto más lejos. En vez de fijar el umbral en 0.7, podemos subirlo a 0.9 y predecir que “y” es igual a 1 sólo si tenemos más del 90% de seguridad de que el paciente tiene cáncer. Una fracción mayor de pacientes tendrá cáncer; por lo tanto, este será un clasificador con precisión alta y con una recuperación baja porque detectaremos correctamente sólo a los pacientes que tengan cáncer. Ahora, consideremos un ejemplo distinto.

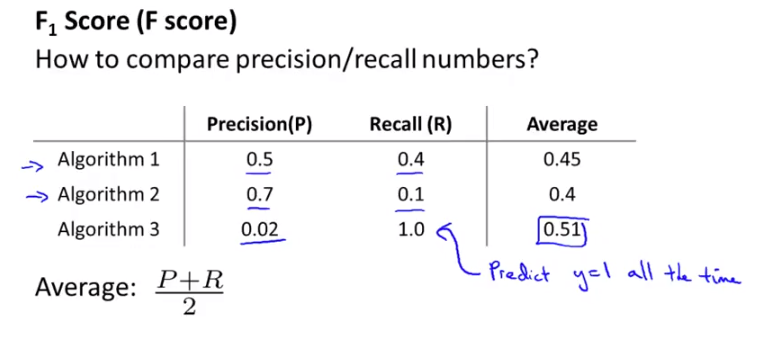
Supongamos que queremos evitar obviar muchos casos de cáncer. Queremos evitar falsos negativos. Especialmente, si un paciente realmente tiene cáncer pero no le dijimos que lo padece, puede tener consecuencias graves. Ya que, si no le dijimos al paciente que no tiene cáncer, no obtendrán tratamiento; es decir, si resulta que sí tiene cáncer pero no le dijimos, no recibirá su tratamiento. Esto puede ser un resultado trágico. El paciente podría morir porque le dijimos que no tenía cáncer y no tomó un tratamiento, pero resulta que sí tenía cáncer. Cuando haya dudas, lo que queremos predecir es que “y” es igual a 1. Así que, cuando tenemos dudas, queremos predecir que tienen cáncer para que, por lo menos, puedan investigar más y tratarse en caso de que realmente lo padezcan. En este caso, en vez de poner un umbral de probabilidad alto, tomaremos este valor y lo bajaremos hasta, digamos, 0.3 así. Al hacer esto, estamos decidiendo que si hay más del 30% de probabilidad de que el paciente tenga cáncer, seremos conservadores y le diremos que puede tener cáncer para que busque el tratamiento necesario. En este caso, lo que haremos es poner un clasificador con una recuperación más alta para marcar efectivamente una fracción mayor de los pacientes con cáncer, pero acabaremos con una precisión menor. Porque entre más grande sea la fracción de pacientes a los que les diagnostiquen cáncer, mayor será la fracción de ellos que, finalmente, no padecen cáncer.   
  
Como paréntesis, cuando hablo de esto con otros estudiantes les resulta muy impresionante y algunos me preguntan cómo puedo contar la historia desde ambos puntos de vista. Ya sea si queremos tener una precisión o una recuperación más alta. Pero esto funciona en ambos casos. Espero que los detalles de mi algoritmo sean verdaderos. El principio más general es que, dependiendo de si quieres una precisión alta y una recuperación baja o una precisión baja y una recuperación alta, puedes terminar prediciendo que “y” es igual a 1 cuando “h(x)” es mayor que el umbral.   
  
En general, para la mayoría de los clasificadores, habrá una compensación entre la precisión y la recuperación. A medida que varía el valor del este umbral, el valor del umbral que tracé aquí, se puede trazar una curva que compensa entre la precisión y la recuperación en donde el valor de aquí arriba corresponde a un umbral con un valor muy alto, quizá mayor a 0.99. Digamos que predijimos que “y” es igual a 1 sólo cuando tenemos el 99 por ciento de probabilidad o por lo menos el 99 por ciento de certeza de que “y” es igual a 1. Esto nos daría una precisión alta y una recuperación relativamente baja. Por el contrario, el punto de aquí abajo corresponde a un valor de umbral mucho más bajo, tal vez 0.01. Cuando tengas duda, supón que “y” es igual a 1. Haciendo esto tendremos con una precisión más baja y una recuperación más alta.   
A medida que varíes el umbral, puedes trazar la curva de tu clasificador para ver la gama de valores que puedes obtener para la precisión y la recuperación.   
Por cierto, la curva de precisión y recuperación puede tener muchas formas distintas. A veces se verá como esta, y otras como esta otra. Hay muchas formas posibles para la curva de precisión y recuperación que dependen de los detalles del clasificador.



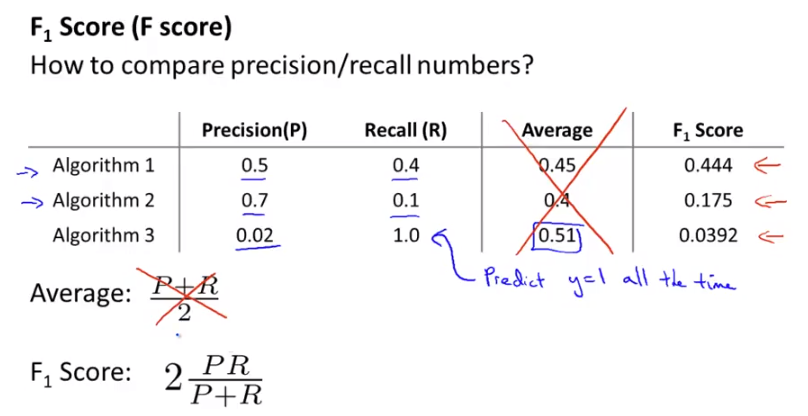
Esto nos lleva a otra pregunta interesante: ¿hay alguna manera de elegir el umbral automáticamente? o, si tenemos algoritmos diferentes o ideas diferentes para algoritmos ¿cómo comparamos los diferentes números de precisión y recuperación? Supongamos que tenemos tres algoritmos de aprendizaje distintos o tal vez son un sólo algoritmo de aprendizaje con umbrales de tres valores distintos. ¿Cómo decidimos cuál de estos algoritmos es el mejor?   
  
Antes, hablamos de la importancia de tener una métrica de evaluación con un número real simple. La idea de esto es tener un sólo número que te indique cómo se está desempeñando tu clasificador. Cuando, en vez de esto, utilizamos la métrica de precisión y recuperación, perdemos esta ventaja. Ahora tenemos dos números reales, por lo que a veces nos enfrentamos con situaciones en las que, si intentamos comparar el algoritmo 1 con el algoritmo 2, terminamos preguntándonos si tener una precisión de 0.5 y una recuperación de 0.4 es mejor o peor que tener una precisión de 0.7 y una recuperación de 0.1. Cada vez que vez que evalúes un algoritmo terminarás preguntándote si 0.5 y 0.4 son mejores valores que 0.7 y 0.1. No sé. Sentarte a pensar sobre estas decisiones puede retrasar tu proceso de toma de decisiones de los cambios útiles que puedes incorporar a tu algoritmo.



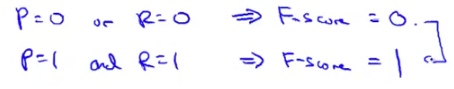
Por el contrario, si tenemos una métrica de evaluación con un valor único; es decir, un número que nos diga si el algoritmo 1 es mejor que el 2 o al revés, nos será de mucha ayuda para decidir más rápidamente qué algoritmo debemos utilizar. También nos ayuda a evaluar más rápidamente los cambios que estamos contemplando para un algoritmo. Entonces ¿cómo podemos obtener una métrica de evaluación con un sólo número real?   
  
Una cosa que puedes intentar es ver el promedio entre la precisión y la recuperación. Utilizaremos P y R para denotar precisión y recuperación, respectivamente. Lo que puedes hacer es calcular el promedio y ver cuál clasificador tiene el valor promedio más alto. Esta no es una solución tan buena porque, al igual que en el ejemplo anterior, resulta que si tenemos un clasificador que predice que “y” es igual a 1 todo el tiempo, encontraremos una recuperación muy alta y terminaremos con un valor muy bajo para la precisión. Por el contrario, si tenemos un clasificador que predice que “y” es igual a 0 casi todo el tiempo, es decir, si la predicción de que “y” es igual a 1 aparece rara vez, indica que tenemos un umbral muy alto utilizando la notación de la línea anterior. Por lo tanto, podemos acabar con una precisión muy alta y con una recuperación baja.   
  
Los dos extremos son tener ya sea un umbral muy alto o un umbral bajo y ninguno de ellos nos dará un clasificador particularmente bueno. Podemos reconocer esto viendo si tenemos una precisión muy baja o una recuperación muy baja. Si sólo tomas el promedio de “p” más “r” sobre 2, en este ejemplo, el promedio más alto será el del algoritmo 3. Aunque puedes obtener este desempeño prediciendo “y” igual a 1 todo el tiempo, no tendrás un clasificador de buena calidad ¿cierto? Si predices que “y” es igual a 1 todo el tiempo, el clasificador no será muy útil si tu resultado siempre es “y” es igual a 1.   
  
Entonces, el algoritmo uno o el algoritmo dos serían más útiles que el algoritmo tres, pero en este ejemplo, el algoritmo número tres tiene un valor más alto de precisión y recuperación que los algoritmos uno o dos. Generalmente pensamos en el promedio de precisión y recuperación como una manera no muy efectiva de evaluar un algoritmo de aprendizaje.



10:38  
En contraste, hay una manera distinta de combinar la precisión y la recuperación. Se llama Valor F y utiliza esta fórmula.   
10:46  
Aquí tenemos los valores F para este ejemplo. Con base en estos valores F podemos decir que el algoritmo 1 tiene el valor F más grande. El algoritmo 2 tiene el segundo más grande y el algoritmo 3 tiene el más bajo. Si nos guiamos por el valor F probablemente elegiremos el algoritmo 1 en vez de los otros.   
  
El valor F también se le llama valor F1 y que se escribe como valor F1 como lo tengo aquí, pero muchos le llaman solamente valor F. Se utiliza cualquiera de los dos nombres. El valor F es como la precisión y la recuperación promedio, pero usa el valor más bajo de precisión o de recuperación - el que sea más bajo - y arroja un peso más alto. En el numerador puedes ver que el valor F es un producto de la precisión y la recuperación. Si la precisión o la recuperación son iguales a cero, el valor también será igual a 0. En ese sentido, combina la precisión y la recuperación. para que el valor F sea mayor, entonces, la precisión y la recuperación también tendrán que ser mayores.



Debo decir que hay muchas fórmulas posibles para combinar la precisión y la recuperación. La fórmula del valor F es una de muchas posibilidades pero, históricamente o tradicionalmente es la que se utiliza en el aprendizaje automático.   
  
El término valor F realmente no significa nada, así que no te preocupes por el nombre, ya sea valor F o valor F1. Este valor generalmente nos da el efecto que queremos porque si la posición o la recuperación son iguales a 0, obtendremos un valor F bajo. Si queremos un valor de F mayor, necesitará que la precisión o la recuperación sean iguales a 1. De manera específica, si “p” es igual a 0 o “r” es igual a 0 entonces indica que el valor F también es igual a 0. Un valor perfecto de F, es decir, una precisión igual a 1 o una recuperación igual a 1 nos dará un valor de F igual a 1 por 1 entre 2 por 2. El valor F será igual a 1 si tienes una precisión y una recuperación perfecta. Los valores intermedios entre 0 y 1 nos dan un orden por rangos de los diferentes clasificadores.



En este video hablamos de la noción de compensación entre la precisión y la recuperación y de cómo podemos variar el umbral para decidir si debemos predecir “y” es igual a 1 o “y” es igual a 0. Este umbral nos indica que debemos tener una certeza del 70% o del 90% antes de predecir que “y” es igual a 1. Puedes controlar la compensación entre la precisión y la recuperación variando el umbral. También hablamos acerca del valor F, que toma la precisión y la recuperación y arroja el resultado como una métrica de evaluación con un número real simple. Por supuesto, si tu meta es ajustar ese umbral automáticamente para decidir si “y” es igual a 1 o “y” es igual a 0, una manera muy razonable de hacerlo es intentar una gama de valores para el umbral. Intenta aplicar un rango de valores para el umbral y evaluar cada valor, digamos, en el conjunto de validación cruzada. Luego elige el umbral que te de el valor de F más alto en el conjunto de validación cruzada. Esta sería una manera razonable de elegir automáticamente el umbral para tu clasificador.