week 6

Advice for Applying Machine Learning

Modelinizi eğittikten sonra hata değeri (cost function) olarak olması gerekenden daha yüksek bir değer alıyorsunuz. Bunu düzeltmek için ne yapmalısınız?

- Daha fazla veri toplayabilirsiniz. (Bazen işe yaramıyor.)
- Daha az feature ile çalışmayı deneyebilirsiniz.
- Belki elinizdeki feature'lar probleminizi tam açıklamadığından yeni feature'lar ekleyebilirsiniz.
- Polinom feature'ler ekleyebilirsiniz. (x_1^2, x_2^2, x_1x_2)
- λ'yı azaltmak veya artırmak.

Hipotez Değerlendirmesi

Çok fazla feature'ımız varsa bunu grafiksel olarak incelememiz pek mümkün değildir. Hipotez değerlendirmesi için veri kümemizi genellikle %70'e %30 olarak ikiye ayırırız. %70'lik kısım ile modelimizi eğitiriz. Bu veri setine train set denir. Geri kalan %30 ile ise test set'ten elde ettiğimiz ağırlık değerlerini (θ veya w) kullanarak cost function değerini hesaplarız ardındaki farkı gözlemleriz. Bu yöntem regresyon modelleri için iyidir.

Sınıflandırma modelleri ise yine regresyon modelindeki gibi verimizi böleriz. train set ile modelimizi eğitiriz. Geri kalan %30'luk test set'imizi ise forward propogation ile ne doğrulukta verimizi tahmin ettiğini buluruz. Aynı tahmini train set verileri içinde yaparız ve elde ettiğimiz accuracy değerlerini yorumlarız.

$$\text{accuracy}(h_\theta, x) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \text{ ve } y = 1 \text{ veya } h_\theta(x) \leq 0.5 \text{ ve } y = 0 \\ 0 & \text{otherwise (diğerleri)} \end{cases}$$

$$\text{accuracy value} = \frac{1}{m_{\substack{\text{Test} \\ \text{Train}}}} \sum_{i=1}^{m_{\substack{\text{Test} \\ \text{Train}}}} \text{accuracy}\left(h_\theta(x^{(i)}_{\substack{\text{test} \\ \text{train}}}), y^{(i)}_{\substack{\text{test} \\ \text{Train}}}\right)$$

Model Seçimi ve Train / Validation / Test Sets

Modelinizi belirlemek istetken kaçıncı dereceden bir polinomun daha iyi bir sonuç vereceğini bilemezsiniz bunun için örneğin 10. dereceye kadar ayrı ayrı hipotez düşünün ve bunların $J(\theta)$ değerlerini optimum noktaya getirin

$d=1$ $h = \theta_0 + \theta_1 x_1$

$d=2$ $h = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$

$d=3$ $h = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

$\vdots$

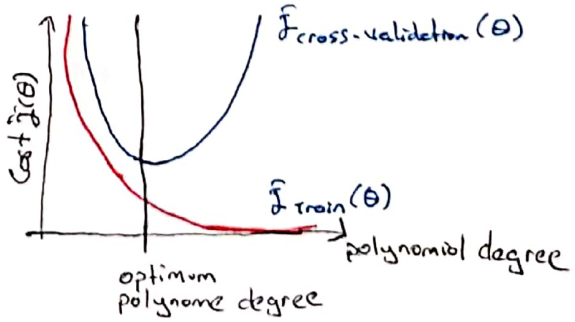
$d=10$ $h = \theta_0 + \theta_1 x_1 + \dots + \theta_{10} x_{10}^{10}$

daha sonra cross validation değişimiz veri setinde teker teker deneyin hangisinde daha düşük hata değeri verirse o modeli seçin ordinden seçtiğiniz modelin tüm verileri genelleyip genellemediğini görebilmeniz için test veri setinde θ değerlerini kullanarak hata değerini hesaplayın. En başta 10 tane polinomial modeli hesaplamak için train veri setini kullanıyoruz. Buradan alınan θ değerlerini cross validation (dev) ve test setlerinde hata değerini hesaplamak için kullanıyoruz.

Genelde train / validation / test veri setleri sırasıyla %60, %20, %20 olacak ayrılıyor. Ancak günümüzde veri setleri çok geniş olduğu için bu kısıtlamaya uymak zorunda değilsiniz. Dikkat etmeniz gereken en önemli nokta veri setlerini ayırlarken homojen bir veri dağılımı gerçekleştirmeniz gerekir. Tüm olumsuz örnekleri test setinde bulundurmamanız gerekiyor. Bunun gibi hatalardan uzak durun.

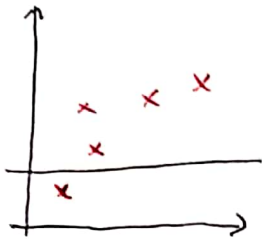
Diagnosing (Teşhis) Bias vs. Variance

Train setinizin ve validation setinizin hata değerleri bize modelimiz ile ilgili teşhisler yapmamızı sağlar.

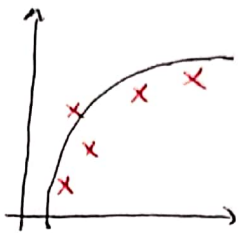


Eğer train setinizin cost function değeri büyükse ve validation setinizinde hata değeri büyükse biz bu duruma high bias (underfiting) diyoruz.

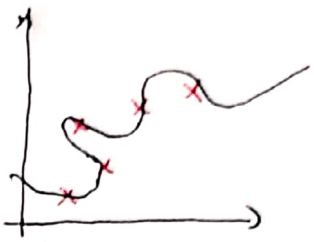
Eğer train setinizin hata değeri düşük ve validation setinizinde hata değeri büyükse biz bu duruma high variance (overfitting) denir.



λ large
example $\cong$ 10000
high bias

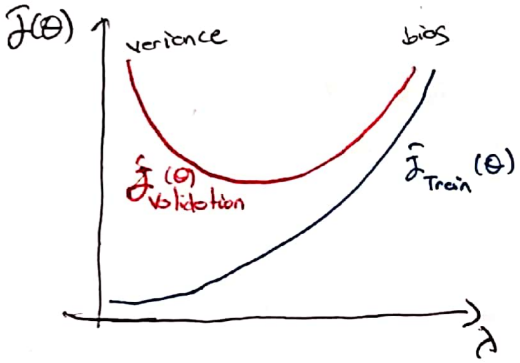


λ intermediate
"just right"



λ small
$\lambda \cong 0$
high variance

Regularization overfitting'i engellemek amacıyla kullandığımızı belirtebiliriz. Peki λ parametresi nasıl seçilmeli?



$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

$$J_{Train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

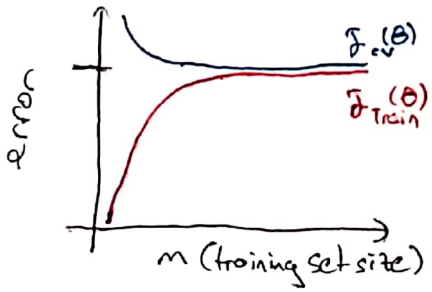
Bir hipotezin lambda değeri düşük olursa hipotez overfitting olur ve train sette başarılı gözükürken validation set'te başarısız gözükür. Eğer lambda değerini çok yüksek ayarlarsak bu sefer train ve validation set'lerin hata değerleri çok yüksek olur ve underfitting gerçekleşir.

Not Lambda parametresini 0.01'in 2'şerli katlarını deneyerek bulabilirsiniz. Train setinde eğittikten sonra validation set'inde deneyin. (Validation cost function'ı regularize edilmemeli yahut $\lambda = 0$ olmalı)

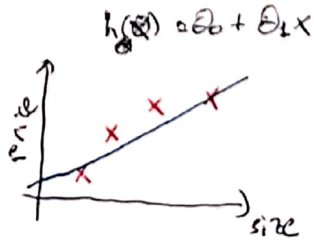
Learning Curves

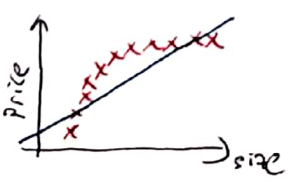
Modelinizin performansını arttırma high bias veya high variance olup olmadığını teşhis etmede yardımcı olur.

High Bias

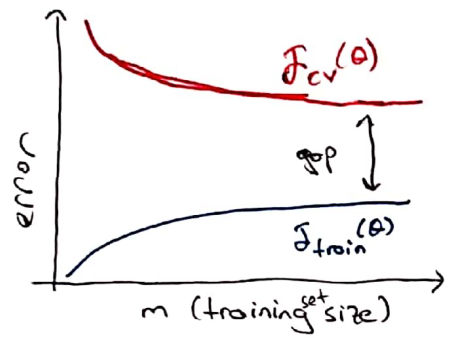


Yüksek bias'li validation setinizin hata değerinin çok az düşmesinden anlayabilirsiniz. ve train/dev hata değerleri bir noktada birbirlerine yaklaşırlar. Fazla veri problemi çözmez.



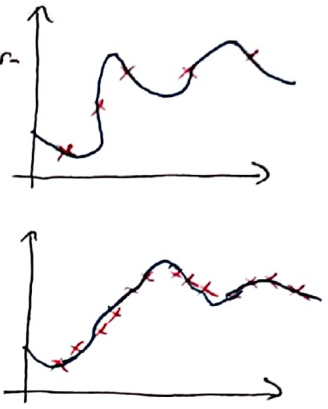


High Variance



Test ve dev setlerinin hata değerleri arasında büyük bir açıklık oluşur. Ancak eğitim verisi arttıkça validation veri setindeki hata değeri düşerek aradaki açıklık kapanır. (J_{train}'de veri arttıkça hatası da artar)

$h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_{100} x^{100}$



High Bias Solving	High Variance Solving
• Ek feature'lar bulmak	• Daha fazla eğitim örneği
• Polinom feature'lar eklemek	• Feature'ları azaltmak.
• λ azaltmak	• λ artırmak

Diagnostic Neural Network

Az parametreye sahip bir sinir ağı underfitting'e eğilimlidir. Hesaplaması ucuzdur. Fazla parametreye sahip bir sinir ağı overfitting'e eğilimlidir. Hesaplaması pahalıdır. Regularization kullanabilirsiniz.

Tek bir gizli katman ile başlamak iyi bir başlangıçtır. Belli katman sayılarında sinir ağınızı eğiterek validation set'te hangisi daha iyi sonuç veriyorsa onu kullanırsınız.

Not Learning Curve hesaplanırken; train set örnek adedine göre tekrar tekrar cost function'ı hesaplanır ancak validation setin tüm örnekleri için train setin ~~theta~~'ı adedine göre hesaplanan thetalsı ile cost function'ı hesaplanır.

```
for i = 1:m
    theta = trainLinearReg(X(1:i,:), y(1:i), lambda);
    error_train(i) = linearRegCostFunction(X(1:i), y(1:i), theta, lambda);
    error_val(i) = linearRegCostFunction(Xval, yval, theta, lambda);
```

Machine Learning System Design

Bir ~~email~~ spam sınıflandırıcı için belirlediğiniz her bir kelime bir feature olabilir. Bu sınıflandırıcıyı nasıl daha iyi bir hale getirebilirsiniz?

- Veri toplayarak.
- Daha iyi özellikler bularak (Örneğin; posta header'ları)
- Metinlerdeki hataları düzeltecek algoritmalar.

ML Tavsiye Edilen Yaklaşım

- Hızlıca basit bir algoritma oluşturun ve validation set üzerinde test edin.
- Elde edilen modellerin learning curve'lerini çizin. Size yardımcı olacaktır.
- Error analysis: Modelinizin doğru ve yanlış tahmin ettiği örnekleri manuel olarak kontrol edin. Bu size modelinizin hangi yapıdaki örnekleri tahmin edebileceğini ve yeni feature'lar çıkarma açısından size yol göstericektir.

Not: Porter stemmer algoritması aynı anlama gelen kelimeleri bulmanıza yardımcı olur.

Bir model eğitiyorsunuz ve %99 accuracy alıyorsunuz. Ancak probleminizdeki hasta sayısı sadece %0.5 bu accuracy ilk bakıldığında çok iyi gibi gözükse de pozitif hastalara baktığımızda bu fikrimiz değişir. Modelinizin durumunu ölçecek iki yöntem daha vardır: (Skewed - Çarpık veri)

Predicted class \ Actual Class	1	0
1	True positive	False positive
0	False Negative	True Negative

Precision: Hassasiyetlik. Eğitim kümenizdeki positive değerler ile (eşleşen positive değerler) modelinizin tahmin ettiği positive değerlerin birbirine olan oranıdır. 1 en iyi sonuçtur. Hassasiyetlik %100'dür.

$$\frac{\text{True positives}}{\#\text{Predicted pos}} = \frac{\text{True positive}}{\text{True pos + False pos}}$$

Recall: Veri kümenizdeki tüm pozitif değerlerin sizin doğru tahmin ettiğiniz positive değerlere olan oranıdır.

$$\frac{\text{True positives}}{\#\text{actual positives}} = \frac{\text{True positive}}{\text{True pos + false neg}}$$

$$\text{accuracy} = \frac{\text{true pos + true neg}}{\text{total examples}}$$

Precision ve recall değeri arasında ters bir orantı vardır. Eğer siz threshdd'unuzu yükseltirseniz precision değeriniz artarken recall değeriniz düşer. Threshold'unuzu düşürürseniz " " azalırken " " artar.

Biz birden fazla threshold'a göre model sonucu elde edersek hangisini neye göre seçeceğiz? Eğer precision ve recall değerlerinden tek bir parametreye indirgersek threshold değeri seçmeniz daha kolay olacaktır.

	Precision (P)	Recall (R)	Average	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

Average : $\frac{P+R}{2}$ iyi bir yöntem değil

F_1 Score : $2\frac{PR}{P+R}$

$P=0$ and $R=0$ $\Rightarrow$ $F=0$

$P=1$ and $R=1$ $\Rightarrow$ $F=1$

Büyük veri kümelerinin model eğitiminize yardım edebilmesi için gerekli özellik (feature) sayısına sahip olması gerekir.