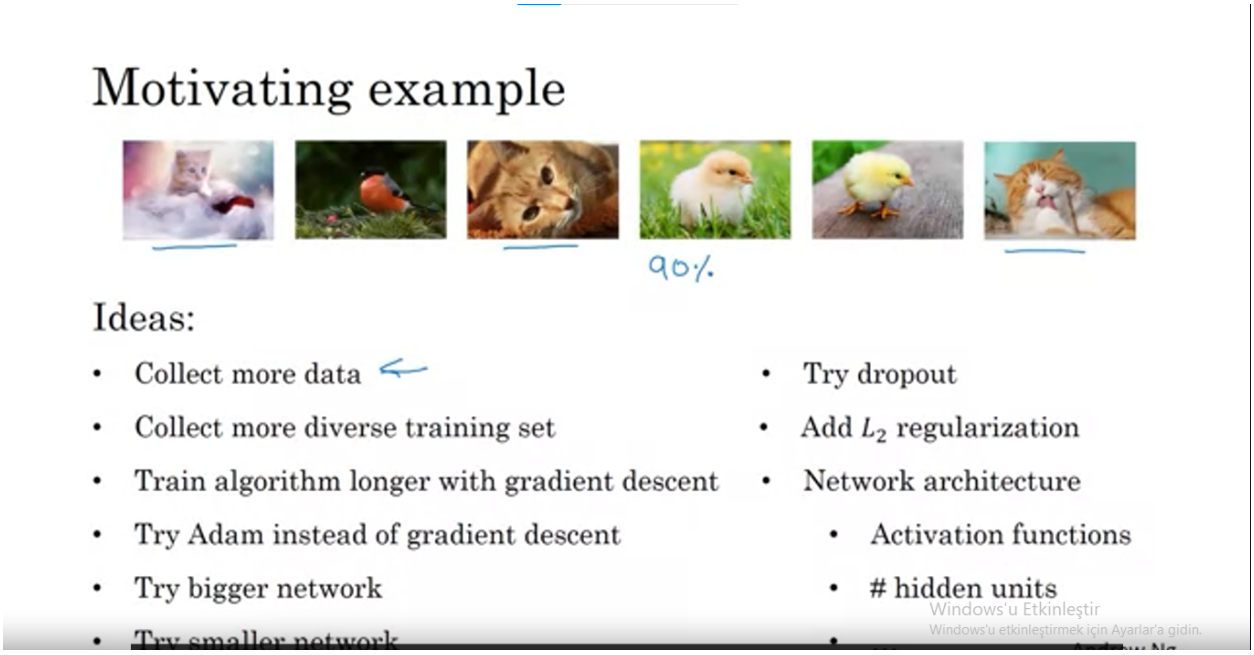
Week-1

Introduction to ML Strategy

Why ML Strategy

Diyelim ki kediler üzerinde çalışıyorsunuz 90 doğrulama oranı yakaladınız fakat bu uygulamanız için yeterli bir oran değil. Bu durumda sisteminizi geliştirmek için birçok yol bulabilirsiniz. Daha fazla eğitim verisi toplayabilirsiniz, veya train setiniz yetersiz kalabilir ya daha çeşitli negatif örnekler toplayabilirsiniz. Farklı bir optimazsyon algoritması deneyebilir(like Adam), algoritmanızı daha uzun gradient descent uygulayabilirsiniz. Büyük,küçük bir ağ deneyebilirsiniz veya Dropout veya L2 regularization deneyebilirsiniz. Ağınızın mimarisini değiştirebilirsiniz. Bunu yapmak için aktivasyon fonksiyonlarını veya gizli katmanların sayısını değiştirirsiniz. Boşa geçen bir zaman olmaması için hangi yöntemleri deneyebileceğimizi anlayabilmek için bir yöntem olması oldukça iyi olur. Bu kursta problemi analiz etmek için en doğru yolları bulacağız.

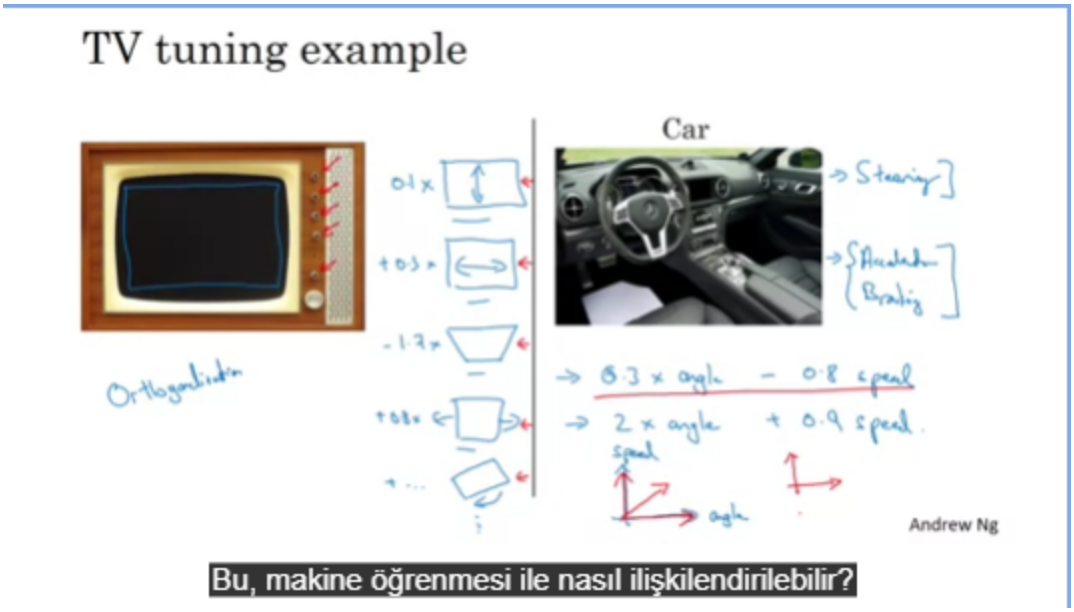


Orthogonalization

Makine öğrenmesi modeli uygularken karşılacağımız en temel sorunlardan biri yukarıda görüldüğü üzere deneyebileceğiniz ve değiştirebileceğiniz çok şey olmasıdır. Örneğin, ayarlayabileceğiniz bir çok aşırı değişken olması gibi. En etkin makine öğrenmesi insanları hakkında farkettiğim şeylerden biri onların neyi ayarlanması gerektiği konusunda keskin görüşlü olmalarıdır. Bu dikgenleştirme(orthogonalization) dediğimiz bir süreçtir.

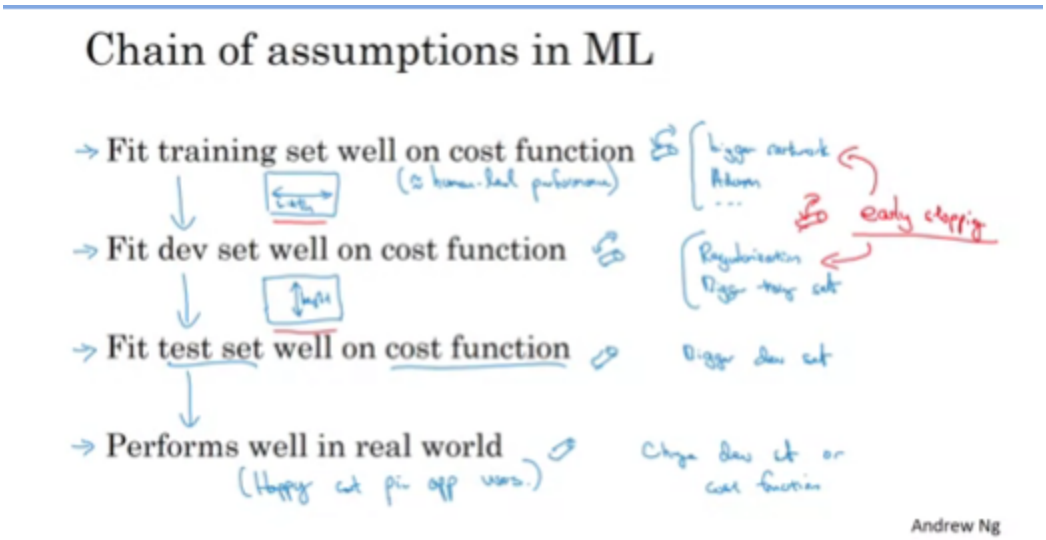
Bu eski TV'nin üstünde bulunan tuşlar farklı ayarlamalar yapmak için kullanılırdı. Her bir düğmenin görece tahmin edilebilir bir fonksiyona sahipti. Bu numerik değerleri tek bir tuşla ayarlasaydık görüntünün ekranın ortasına gelecek şekilde ayarlamak mümkün olmazdı. Orthogonalization, kavramı düğmeleri, her bir düğme sadece tek bir şey yapacak şekilde tasalarmalarını ifade eder.

Eğer bir araba kullanmayı öğrenmeyi düşürseniz, bir arabanın üç ana kontrolü vardır, bunlar direksiyon, gaz ve frendir. Yön için bir, hızınız için iki kontrolünüz vardır. Aşağıda görülen numerik değerlerini iki düğmede arabayı kontrol ettiğini ayarladığınızı düşünerek arabanızı istediğiniz açıya ve istediğiniz hıza getirebilirsiniz. Ama bu, direksyon açısını ayarlamak için tek bir kontrol ve hızı ayarlamak için ayrı bir grup kontrole sahip olduğnuz durumdan daha zordur.



Bu, makine öğrenmesi ile nasıl ilişkilendirilebilir?

Eğer erken durdurma yaparsanız eğitim veri setinizden daha az verim alırsınız!! Dev set'in doğruluğunu olumlu yönde etkiler. Fakat train setini negatif yönde etkileyebilir.



Setting Up your Goal

Single Number Evaluation Metric

Hiperparametre ayarı, farklı model gibi denemeler yaptığınızda , eğer son denediğiniz fikrin işe yaradığını ya da kötüye mi götürdüğünü hızlıca söyleyebilecek tek gerçek sayı değerlendirme ölçeğiniz(single number evaluation metric) varsa ilerlemeniz çok daha hızlı olacaktır. Dolayısıyla takımlar makine öğrenimi projesine başlarken genellikle problem için bu metric'in kurulmasını tavsiye ediyoruz. Bir örneğe bakalım.

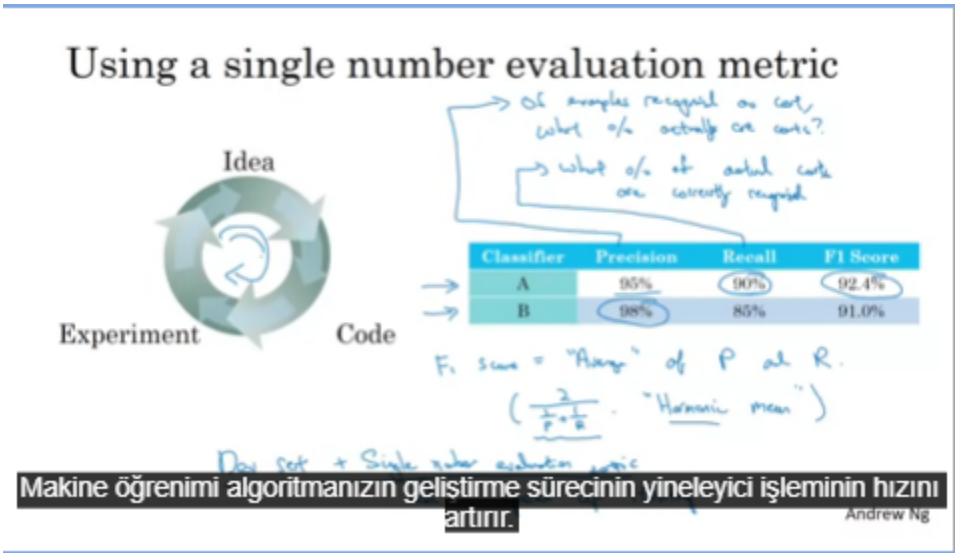
Sürekli bu döngünün içinde döndüğümüzü biliyoruz. Diyelim ki, daha önce bir A sınıflandırıcısı inşa ettiniz ve üst parametreleri, eğitim setlerini veya başka şeyleri değiştirerek şu anda yeni B sınıflandırıcısında eğitiyorsunuz. Burada sınıflandırıcınızın performasını mantıklı bir şekilde değerlendirmenin bir yolu Kesinlik(Precision) ve Recall(Duyarlılık) kriterlerine bakmaktır.

Sınıflandırıcınızın kedi olarak tanıdığı örneklerden yüzde kaç gerçekten kedidir? (Kesinlik) %95 olasılıkla kedidir.(A)

Kedilerin yüzde kaçı sınıflandırıcınız tarafından doğru tanımlanmaktadır? %90 olasılıkla kedi görür. Yani dev setin içindeki verilerden %90'ını kedi olarak görür.

İkisinden birinin öneminden feragat etme söz konusudur ve ikisini de önemseriz. Sınıflandırıcınız bir şeye kedi dediği zaman bunun büyük oranda gerçekten kedi olmasını istersiniz.Fakat kedilerin tüm görüntüleri arasından aynı zamanda büyük bir oranını kedi olarak belirlemesini de istersiniz. Sonuçlar A ve B gibi bir durum oluşuyorsa hangi sınıflandırıcınzın daha iyi olduğundan emin olmamanız problemdir. Daha fazla sınıflandırıcı eğiterek sonuca daha geniş perspektifte bakabilirsiniz.

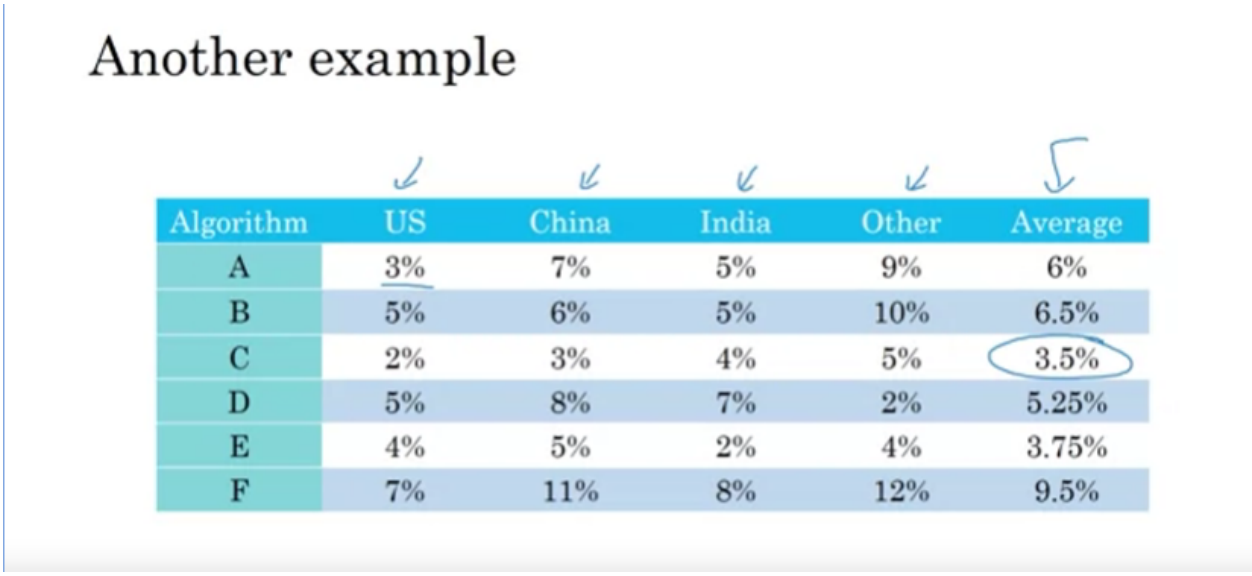
Bundan dolayı sınıflandırıcı seçmek için önerdiğimiz şey, iki ölçek(kesinlik ve duyarlılık) kullanmaktansa kesinlik ve duyarlılığın birleştirildiği tek bir değerlendirme ölçeği bulmanız olacaktır. Makine öğrenimi literatüründe, kesinlik ve duyarlılığı anmayı birleştirmenin standart yolu F1 skoru diye andlandırılan bir skordur. Detaydan ziyade F1 skorunu kesinlik ve anmanın harmonik ortalaması şeklinde düşünebilrsiniz. Şimdilik sınıflandırıcı A'nın, daha iyi F skoruna sahip olduğunu görebilirsiniz ve F1 skorunun kesinlik ve duyarlılığı kombine etmenin mantıklı bir yolu olduğunu varsayarsak, sınıflandırıcı A'yı sınıflandırıcı B üzerine kolaylıkla seçebilirsiniz.



Makine öğrenimi algoritmanızın geliştirme sürecinin yineleyici işleminin hızını artırır.

Diyelim ki 4 büyük coğrafyadaki kedi severler için bir kedi uygulaması geliştiriyorsunuz: USA, China,Indıa ve diğerleri yani düyanın geriye kalan kısmı. Diyelim ki iki sınıflandırıcınız bu 4 farklı coğrafyalardan alınmış verisetlerinde farklı hatalara ulaşıyorlar. Algoritma A, USA'da kişiler tarafından yüklenmiş fotoğraflarda %3 hataya ulaşıyor vs. Dolayısıyla sınıflandırıcınızın bu farklı pazarlarda veya coğrafyalarda ne

kadar iyi sonuç verdiğini takip etmek mantıklı olacaktır. 4 farklı coğrafya için hata analizi zor olacağından performansı takip etmek için ortalamayı hesaplamak mantıklı olabilir. Sonuç olarak algoritma C'nin en düşük ortalama skora sahip olduğunu görebiliriz.



Satisficing and Optimizing Metric

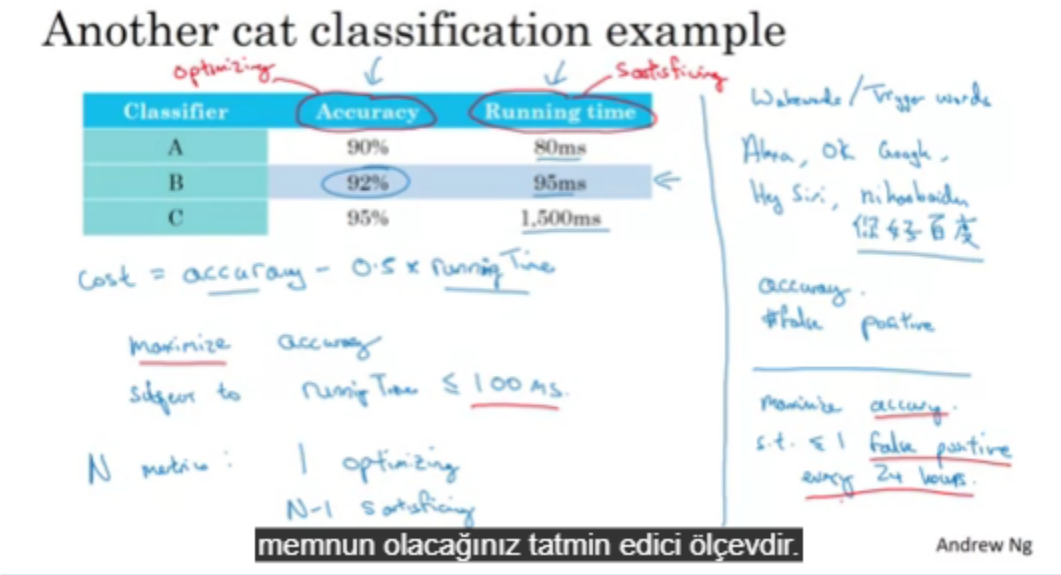
Kedi sınıflandırıcınızın doğruluğunu önemsemeye karar verdiğinizi varsayalım, bu F1 skoru veya başka bir doğruluk ölçüsü olabilirdi. Ama diyelim ki, doğruluğun yanı sıra, çalışma süresini de önemsersiniz. Yapacağınız şe, doğruluk ve çalışma süresini genel bir değerlendirme ölçeviyle birleştirmektir. Yapabileceğimiz ilk yol fakat bu yol biraz yapay görünebilir. cost = acc - 0.5*runningTime

Bunun yerine yapabileceğiniz başka bir şey daha var;

maximize acc ve 100msden küçük sınıflandrıcıyı seçmek olabilir. Bu durumda dorğuluğun bir optimazsyon ölçevi olduğunu söyleyebiliriz. Çünkü genel amacımız doğruluğu maks yapabilmek. 100msden aşağısı kullanıcının hissedebileceği düzeyde değildir.

Başka bir örnek için ise, tetikleyici kelimeler olarakta adlandırılan uyadırma kelimelerini algılamak için bir sistem kurduğunuzu varsayalım. Bu sistemde tetikleyici sözcüğü algılama sistemlerinizin doğruluğunu önemseyebilirsiniz aynı zamanda False pozitive(yani yanlışa doğru diyen) alakasız uyanmalarıda önemsersiniz. Bu durumda, bu iki değerlendirme matrisini bir araya getirmenin belki de tek makul yolu, doğruluğu en üst düzeye çıkarmak olabilir, bu yüzden birisi cihazınızınn uyanma şansını en üst

düzeye çıkarmak için tetikleyici kelimelerden birini söylerken, buna tabi olarak, her 24 saatlik çalışmada en fazla bir yanlış pozitifiniz vardır. Böylece cihazınız, hiç kimse aslında konuşmuyorken, ortalama günde sadece bir kez rastgele uyanır. Bu durumda, doğruluk opmizasyon ölçevidir ve her 24 saate bir dizi yanlış pozitif satisficing(tatmin edici max) olur.

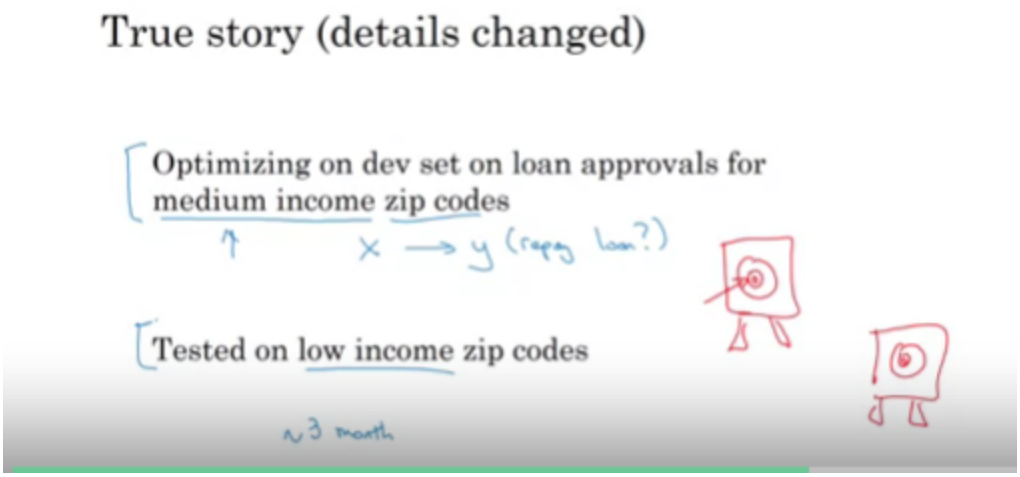


Train/Dev/Test Distributions

Eğitim dev setlerinizi ve test setlerinizi nasıl kurduğunuz, sizin ya da ekibinizin makine öğrenme uygulaması inşa ederken ne kadar seri bir şekilde ilerleme kaydedbileceği üzerinde büyük etkisi olabilir. Bu veri kümelerini takımızın verimini maksimuma çıkaracak şekilde nasıl kurabilirsiniz bakalım. Dev set, developer set veya holdoput cross validation olarak adlandırılır. Sonunda sizi memnun eden bir durum elde edinceye kadar dev setinin performansını iyileştirmek için yeniliklere devam edin ve sonra bu durumu test kümenizde değerlendirin. Örnek olarak diyelim ki kedi sınıflandırıcı kuruyorsunuz ve siz aşağıdaki bölgelerde çalışıyorsunuz; Peki siz dev ve test kümelerinizi nasıl kurarsınız? Bunu yapmanını bir yolu, bu bölgelerden dördünü seçmek. Rastgele seçilmiş dört bölge olabilir ve diyelim ki, bu dört bölgeden gelen veriler dev sete gidecek ve diğer dört bölge rastgelede seçebilir, test kümesine gider. Bu kötü bir örnektir. Çünkü bu örnekte dev ve test setiniz farklı dağılımlardan geliyor. Ben bunun yerine, dev ve test kümelerinizin aynı dağılımdan gelmesini sağlamanın bir yolunu bulmayı öneririm. Sıkıntı olarak seçilen bu dev setinde gerekli iyileştirmeler yaparsanız, test setinde denediğinizde farkına varırsınız. Çünkü dev setindeki veriden

çok farklı olacaktır. Bu durumda dev setinde yapılan tüm iyileştirmeler boşa gidecektir. Dolayısıyla, farklı dağıtımlardan gelen dev ve test kümeleri, bir hedef belirleyip ekibinizin dart'ın gözüne daha da yaklaşmayı hedefleyerek aylar harcamasından sonra farkına varmasıdır. Bundan kaçınmamız için, tüm bu rastgele karıştırılmış verileri dev ve test kümenize almanız gerekmektedir. Böylece, dev ve test kümelerinin sekiz bölgeden veriye sahip olması ve böylece, dev ve test kümelerinin her sekiz bölgeden veriye sahip olması ve dev ve test kümelerinin gerçekten aynı dağılımdadn gelmesi, yani tüm verilerinizin birbirne karıştırılmasıdır.

Bir kredi başvurusu hakkında verilen bir X girdisiyle krediyi neden geri ödeyip ödemeyeceklerini tahmin edebilir misin? Yani, bu krediyi onaylayıp onaylamamaya karar vermenize yardımcı olur ve böylece dev setine kredi başvurularıdndan geldi ve zip codları ve sonuucunda orta seviyeli gelir ve düşük seviyeli gelir arasında zip bağlantısı bulundu. Yani zip kodlarının dağılım verisi çok farklıydı. İlk sınıflandırıcı için ayırdığınız iyileştirme işlemi ikinci sınıflandırıcı için işe yaramayacaktır.

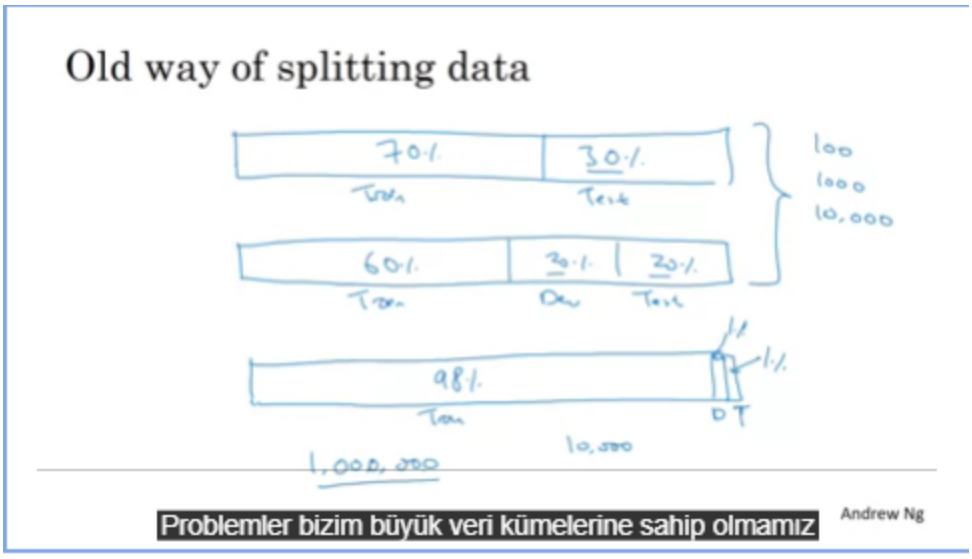


Yani, bir dev seti ve test kümesi kurmak için önerdiğim şey, ileride almayı beklediğiniz verileri yanıstacak şekilde bir dev ve test kümesi seçin, iyi yapılmasının önemli olduğunu göz önünde bulundurun ve özellikle, dev ve test seti aynı dağılımdan gelmedilir. Dolayısıyla, ileride iyi performans vermesini istediğiniz ne tarz bir veri bekliyor olursanız olun, buna benzeyen şekilde veri elde etmeye çalışın, ve bu veri her ne ise, buu hem dev ve hem test setinize koyun.

Size of the Dev and Test Sets

Dev ve test setlerinin dağılımları ve boyutları Deep Learningte biraz değişmekte. Makine öğremesinde belki temel kural olarak duymuşsunuzdur, (70-30). (60 20 20) şeklinde klasik bölünmedir.

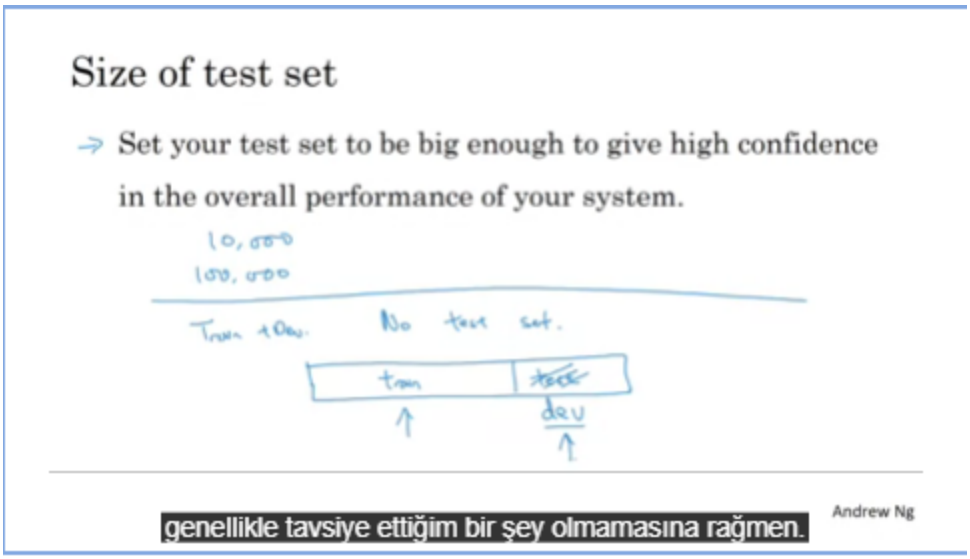
Diyelim ki 1 milyon eğitim örneğiniz var, verinizin %98'ini train set, %1 dev, %1 test olarak ayrılabilir. Dev ve test seti için 10bin'er örnek gayet yeterli kalır. Dolayısıyla, bazen daha da büyük veri kümelerine sahip olduğumuz modern Derin Öğrenme döneminde, mantıklı olan verilerinizin %20 veya %30undan daha küçük bir miktarını dev ve test seti için kullanmak olabilir.



Peki ya test setinin boyutu?

Test setinin amacını hatırlarsak, bir sistemi geliştirdikten sonra test kümesi, son sisteminizin ne kadar iyi olduğunu değerlendirmenize yardımcı olur. Temel ilke, test setini yeterli büyüklükte ayarlamak sistemin genel performasını en yüksek güvenilrilik değerini verecek şekilde. Yani, sizin son sisteminizin ne kadar iyi performans gösterdiğine dair çok doğru bir ölçüme sahip olmanız gerekmedikçe, belki de bir test kümesinde milyonlarca örneğe ihtiyacınız yoktur.

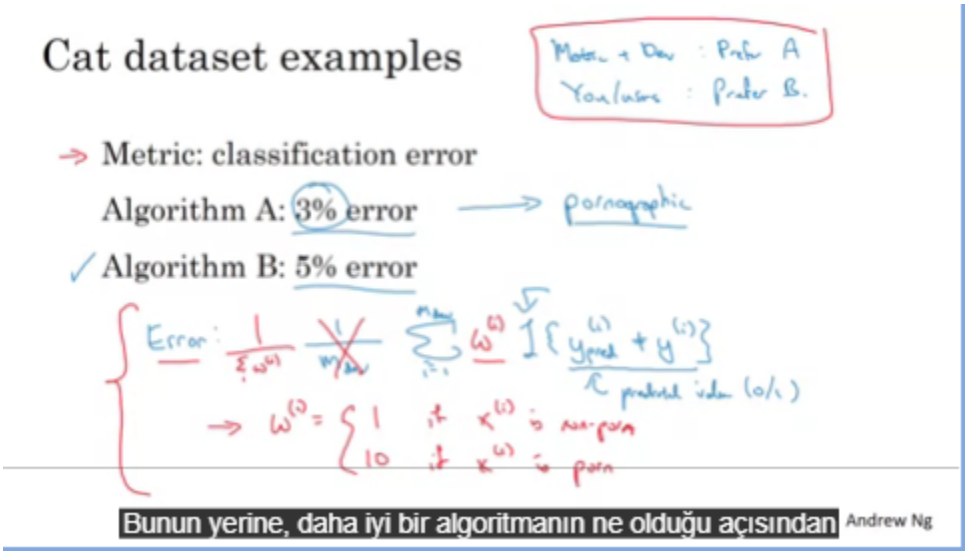
Bazı uygulamalar için belki de son sisteminizin genel performansında yüksek bir güvene ihtiyacınız yoktur. Belkide tek ihtiyacın olan bir train ve dev settir ve bir test kümesinin olmaması daha iyi olabilir diye düşünüyorum. Gerçekte, bazen eğitim ve test böler ama onlar aslında test kümesinde yenileme yapıyorlardı. Yani o değişikliklerin yapıldığı test seti aslında dev setti.



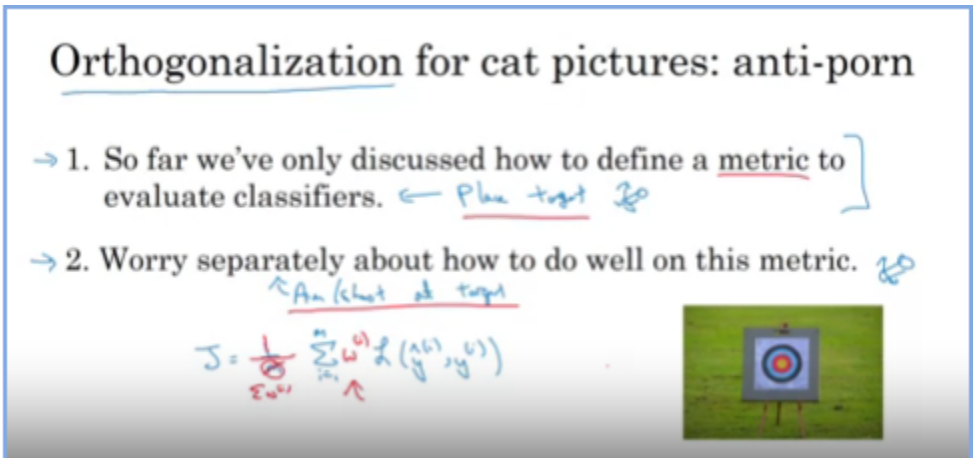
When to Change Dev/Test Sets and Metrics?

Kedi seven kullanıcılara gösterilmek üzere çok sayıda kedi fotoğrafı bulmayı denemek için bir kedi sınıflandırıcısı hazırladığınızı ve kullanmaya karar verdiğiniz ölçevin sınıflandırma hatası olduğunu varsayalım. Algoritma A daha iyi çalışıyor gözüküyor fakat bir şekilde A'nın bir çok pornografik görüntüye izin verdiğini görüyorsunuz. Fakat, algoritma B yüzde 5 hataya sahiptir, bu yüzden daha az görüntü sınıflandırır. Ancak pornografik görüntüleri içermemektedir. Bu sebeple algoritma B seçilecek olan olur. Yani, bu örnekten anladığımız şey, algoritma A'nın değerlendirme ölçevi hususunda daha iyi bir performans gösterdiğidir. Fakat daha kötü bir algoritma olduğu gerçeği söz konusudr. Değerlendirme ölçevi ve dev set Algoritma A'yı tercih ediyorlar fakat siz pornografik görüntülere izin vermediği için algoritma B'yi tercih ediyorsunuz. Yani böyle bir durumda, değerlendirme ölçeviniz algoritmalar arasındaki sıralanmış tercihleri artık doğru bir şekilde dahil etmiyorsa -ki bu durumda Algoritma A'nın daha iyi bir algoritma olduğu hakkında yanlış tahminde bulunuyor, o zaman bu değerlendirme öleçvini ve ya dev setinizi değiştirmeniz gerektiğine dair bir işarettir. Bu durumda kullandığınız yanlış sınıflandırma hatası metriği şu şekilde yazılabilir: 1/m_dev toplam i1'den m_dev'e kadar. Yani bu formül sadece yanlış sınıflandırılmış örneklerin sayısını hesaplar. Bu değerlendirme ölçevindeki sorun, pornografik ve pornografik olmayan görüntüleri eşit şekilde ele almalarıdır. Ancak siz, sınıflandırıcınızn pornografik görüntüleri yanlış etiketlemesini istemiyorsunuz. Örneğin, kedi görüntüsündeki pornografik bir görüntünün kuşkulanılmayan kullanıcıya gösterilmesi durumunda, beklenmedik bir şekilde porno gösterilmesinden dolayı rahatsızlık duyarsınız. Bu değerlendirme ölçevinin değiştirmenin bir yolu şu olabilir. Eğer buraya ağırlık terimini eklerseniz ve bu formül ile

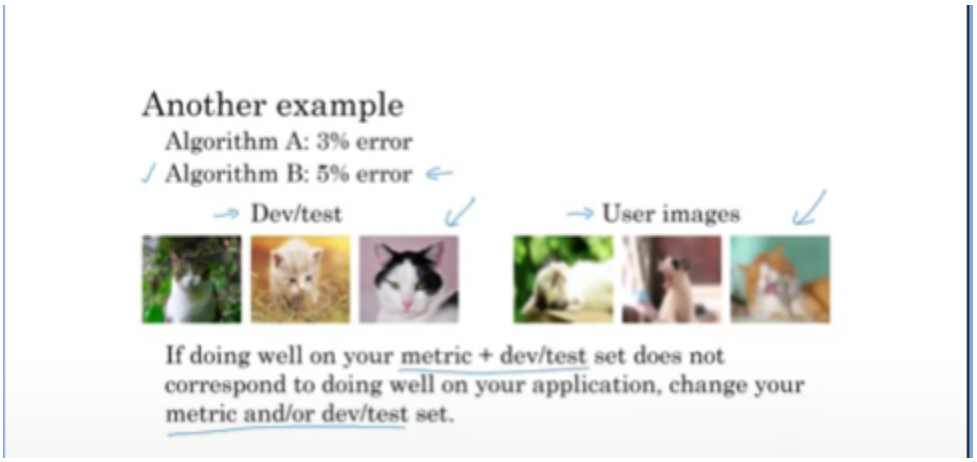
çok pornografik içeriklere çok daha büyük bir ağırlık değer atamış oluyorsunuz. Böylece algoritma bir pornografik görüntüyü bir kedi görüntüsü olarka sınıflandırırken bir hata yaparsa hata terimi çok daha fazla artar. Fakat tüm veri kümesini incelemeniz ve pornografik görüntüleri etikletmeniz gerekir. Böylece bu ağırlık atmaa fonksiyonunu uygulamaya koyabilirsiniz. Altı çizilmesi gereken mesele şu ki; değerlendirme ölçevinin, aslında daha iyi bir algoritma için doğru sıralama tercihlerini vermediğini tespit ederseniz, o zaman yeni bir değerlendirme ölçevi tanımlananın zamanı gelmiştir diye düşünmek gerekir. Değerlendirme ölçütünün amacı, iki sınıfladırıcı verildiğinde, uygulamanız için hangisinin daha iyi olduğunu size doğru bir şekilde bildirmektir.



Fark edeceğiniz üzere, şimdiye kadar yalnızca sınıflandırıcıları değerlendirmek için bir ölçevi nasıl tanımlayacağımız hakkında konuştuk. Bu aslında bir makine öğrenimi problemini ele alırken problemi önce farklı adımlara bölmenizi düşündüğüm bir ortogonalizasyon örneğidir. Birinci adım, ne yapmak istediğinizi gösteren bir ölçevi tanımlamak olacaktır ve bu ölçev üzerinde nasıl daha iyi çalışacağına dair olan meseleyi ayrı olarak ele alacağız. Bu nedenle, makine öğrenimi görevini ik ifakrlı adım olarak düşünün. Hedefi belirlemenin bir adım, hedefe ulaşmanın ve ona nişan almanın ayrı adımlar olması gerektiğini bilmeniz gerekmektedir.



Kedi sınıflandırıclarınız olan A ve ‘Bnin dev setinizde değerlendirildiği üzere algoritma A’nın sadece yüksek çözünürlükle veriler üzerinden daha iyi çalıştığını tespit ettiğinizde algoritma B’nin daha iyi performans gösterdiğini gözlemleyebilirsiniz. Dolayısıyla bu, ölçevinizin ve dev ve test setlerinizin işe yaramadığını gösteren başka bir örnek. Mesele şu ki, siz dev ve test kümeleriniz üzerinde çok güzel, yüksek çözünürlüklü, iyi çerçeveli görüntüleri değerlendiriyorsunuz. Ancak, kullanıcılarınızın gerçetken önemsedikleri şey, yükledikleri pek profesyonel olmayan, bulanık ve daha az çerçeveli resimlerde dahi iyi performans sergileyebiliyor olmanız. Dolayısıyla kuralınız, ölçevinizde ve dev setinizde veya dev ve test kümelerinizin dağılımında iyi performans göstermesine rağmen sizin içina sıl önemli olan uygulamada iy iperformans göstermiyorsa, ölçevinizi ve dev-test kümelerinizi değiştirmek olmalıdır. Başka bir deyişle, dev-set kümenizin bu çok yüksek kaliteli görüntülere sahip olduğunu, ancak bu dev-set kümesinde yapılan değerlendirmenin uygulamanızın gerçekte ne kadar iyi performans gösterdiğine dair öngörüde bulunmadığını fark ederseniz, çünkü uygulamanızın daha düşük kaliteli resimlerle uğraşması gerekiyor. O zaman dev-set kümenizi değiştirmenin tam zamanıdır. Böylece verileriniz iyi bir şekilde çalışmasını stediğiniz veri trünü daha iy iyansıtmış olacaktır.



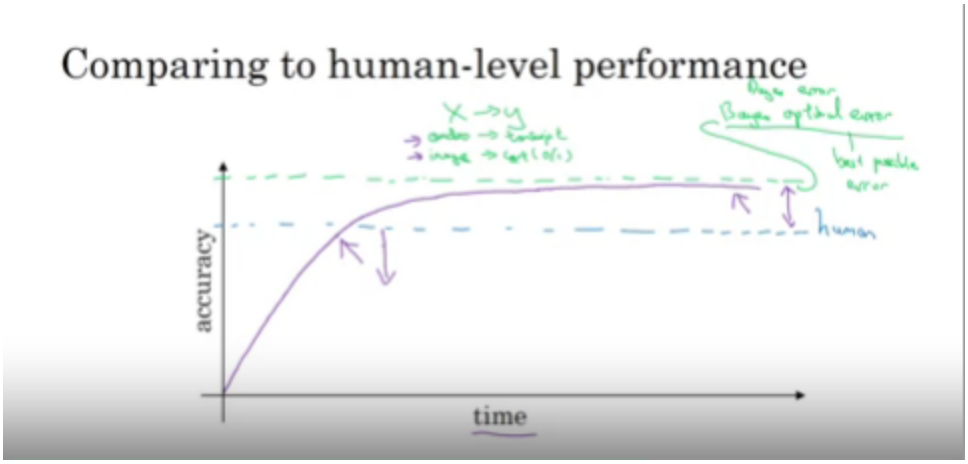
Comparing to Human-level Performance

Why Human-level Performance

Makine öğrenmesi modelleri o kadar çok gelişti ki insan düzeyindeki performansı taklit etmek oldukça doğal bir hal aldı.

Bir çok makine öğrenmesi çalışmasında görüldüğü üzere zamanla bir problem üzerinde çalıştıkça, x eksenini zaman olarak ele alırsak, bu eksen bazı takımlar veya bazı araştırma ekiplerinin üzerinde harcadığı aylara ve hatta yıllara eşit olabiliyor. İnsan düzeyindeki performansa ulaşırken ilerleyiş nispeten daha hızlı oluyor, ancak bir süre sonra algoritma insan düzeyindeki eprformansı aşıyor ve sonra ilerleyiş ve tutuarlılık gerçekten yavaşlıyor. Accurary zaman içerisinde durma noktasına gelebilir ve teorik olarak optimum düzeyde bir performans seviyesine ulaşır ve zamanla siz bu algoritmayı daha büyük modeller ve daha fazla veri ile eğittikçe performans teorik olarak limite yaklaşıyor ancak asla üzerine çıkamıyor. Bu noktaya ise Bayes optimal error adı verilir. Yani Bayes optimal error'u mümkün olan en olası hata olarak düşünebiliriz ve bu X'den Y'ye belrili bir accurary seviyesini aşmanın, herhangi bir işlev eşleşmesiyle gidilen yoludur. Yani konuşma algılama için örnek verirsek, eğer X ses kaydı ise, bazı sesler o kadar gürültülüdür ki ses kaydında ne olduğunu anlamak imkansızdır. Yani mükemmel hata seviyesi %100 olmayabilir. Ya da görüntülerdeki kedileri algılamak için bazı görüntüler o kadar bulanık olabilir ki, kedi olup olmadığını görebilmek tam anlamıyla imkansız hale gelebilir. Bayes hatası, X'den Y'ye haritalandırma yapmak için en iyi teorik işlevdir ve bu değer asla aşılamaz. İnsan düzeyini geçtikten sonra accurary

yavaşlamasının sebebi geliştiilecek pek bir şey kalmamasıdır. İnsan seviyesini geçmek için uyguladığınız optimazassyon ve iyileştirme metodları insan düzeyini geçtikçen sonra bir hayli zorlaşıyor.



İnsanların gerçekten iyi olduğu konulrda verileri etiketlemek için insanları kullanabilirsiniz. Alttaki resimde görülen taktikleri insan üstü bir model oluşturduğunuzda uygulamak oldukça zor olacaktır.

Avoidable Bias

Öğrenme algoritmalarınızın train setinde ne kadar iyi performans sergilemesi gerektiğini konuşmuştuk fakat bazen çok fazla iyi olmasını istemezsiniz ve insan seviyesi performansını bilmek size train setinde tam olarka ne kadar iyi bir performans sergilemeniz gerektiği konusunda yardımcı olur.

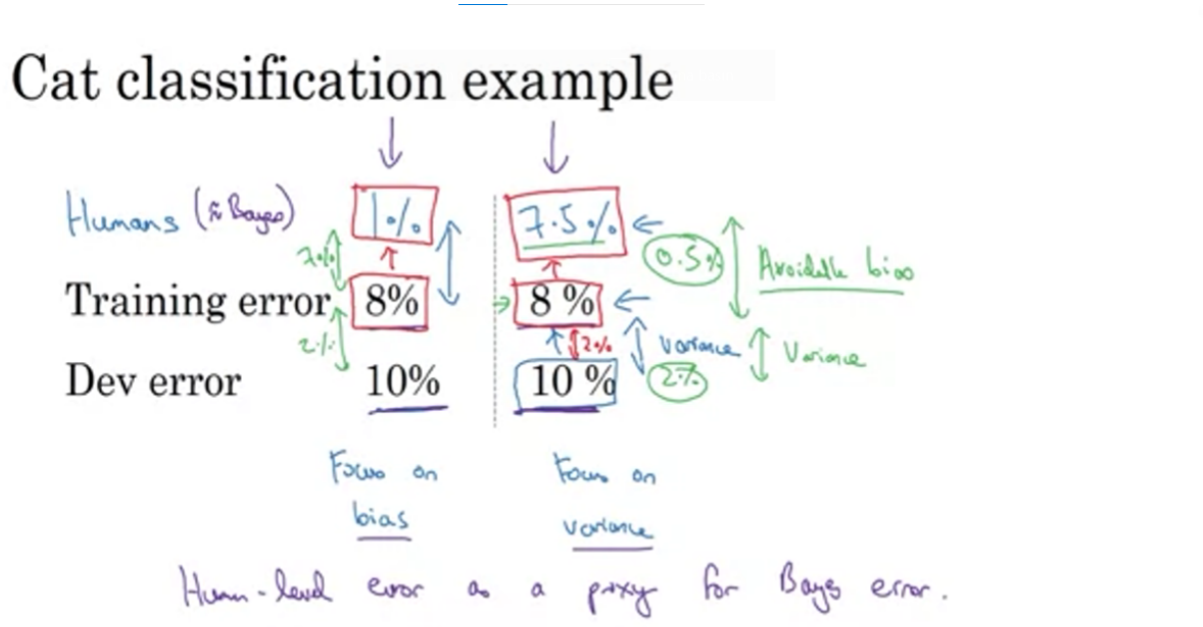
Bir kedi resmini ele alalım, ve aşağıdaki değerlere bakarak train setinin insan düzeyinde iyi performans göstermediğini yani iyi fit edilemediğini görürüz. Dolayısıyla, bias ve varyans azaltmayı sağlayan araçlar bakımından bu durumda, odağınızı bias azaltmaya yönlendirmelisiniz. Bunun için daha büyük bir ağ eğitebilirsiniz veya daha fazla eğitim örneği toplayabilirsiniz.

Farklı bir örnekte aynı train dev errorlarıyla insan hatasını 7.5 olduğunu düşünürsek, belki verilerinizdeki görüntüler fazla miktarda bulanık ve insanlar bile bunun kedi mi değil mi olduğunu ayırt etmekte güçlük çekiyor.

Bu durumda eğitim setinizin iyi bir performans gösterdiğini söyleyebilirsinzi. Burada insan seviyesinden briaz daha kötü bir performans sergiliyor. Bu algoritma için odak noktanız varyansı azaltmaya yönelik olabilir. Dolayısıyla dev setinizdeki hata oranını

yaklaştırmak için regularization deneyebilirsiniz. Örneğin, görüntü algılama konularında insanın aldığı sonuç Bayes hatasından çokta uzak değildir.

Yani istenilen train seti hatassını bayes hatasına yaklaşırıncaya kadar geliştirmeye devam ettirmenizdir.

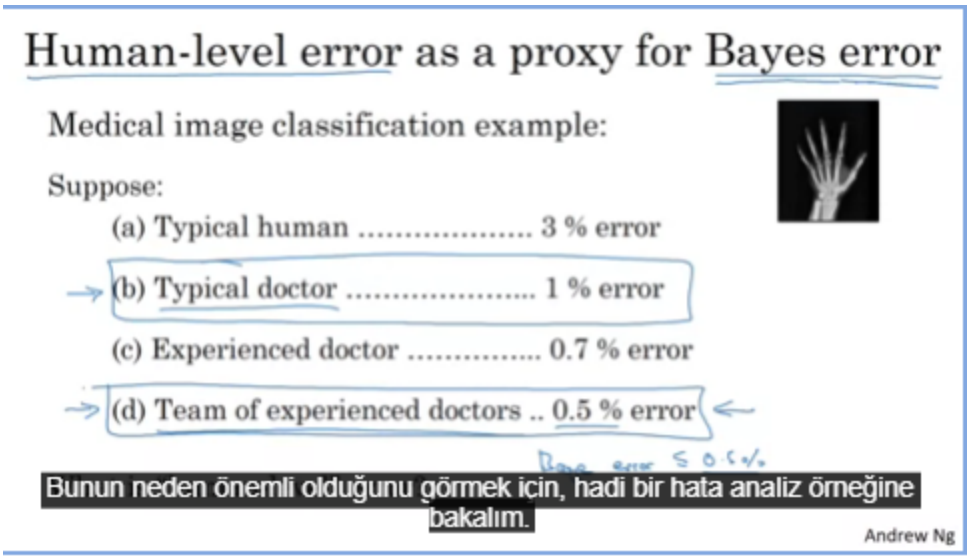


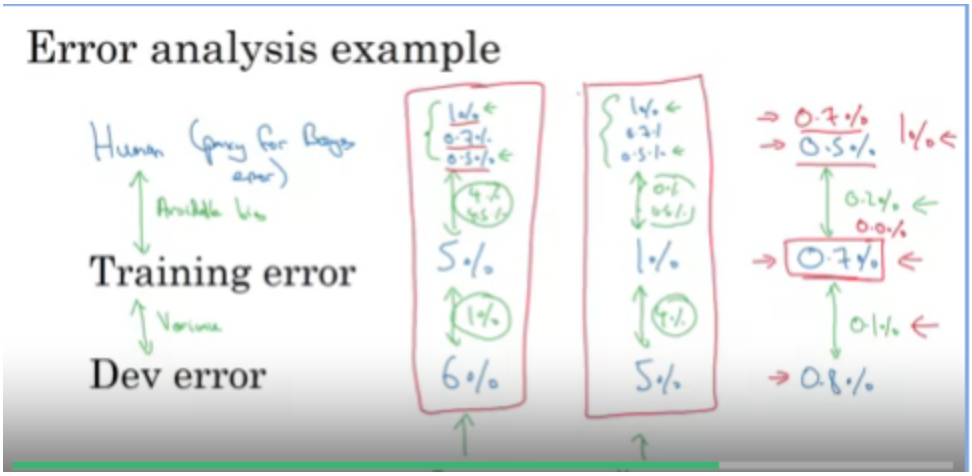
Understanding Human-level Performance

Bayes error ulaşılacak minimum hatayı göz önünde bulundurarak, medikal bir görüntünün sınıflandırma örneğine bakalım. Diyelim ki, bu tarz bir radyoloji görüntüsü örneğine bakmak istiyorsunuz ve teşhisi sınıflandırmak için bir karar vermek istiyorsunuz. Farz edelim ki sıradan, eğitimsiz insan bu görevi %3 hata ile tamamlıyor. Tipik bir doktor, belki de tipik bir radyoloji doktoru, %1 hata ile tamamlıyor. Tecrübeli bir doktor ise çok daha iyi bir sonuç alıyor. %0.7 hata. Eğer tecrübeli bir doktor ekibine ulaşırsanız, %0.5 gibi bir hata payına sahip oluyor.

Yani sorulacak soru, buradaki insan düzeyindeki hatayı nasıl tanımlamalıyız?

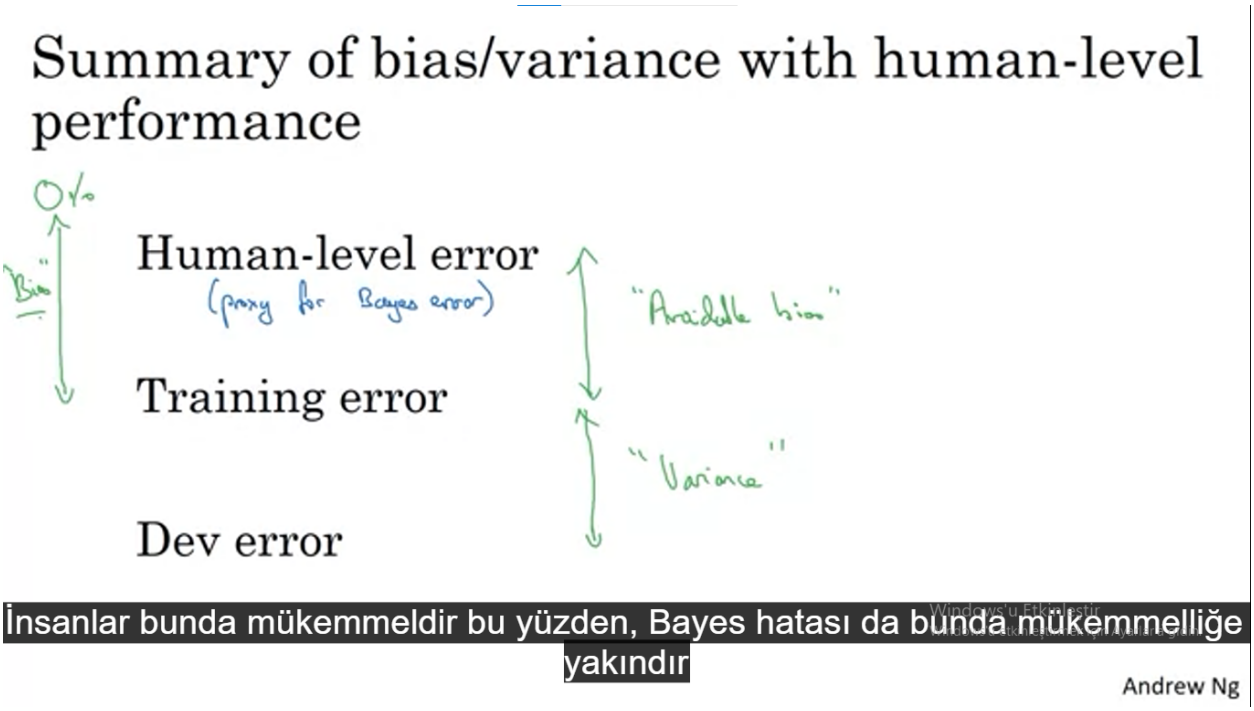
En düşük hata %0.5 olduğundan Bayes teoreminin %0.5ten daha az bir değere eşit olduğunu biliyoruz. Standart olarak bayes değerini %0.5 olarak belirleyebiliriz.





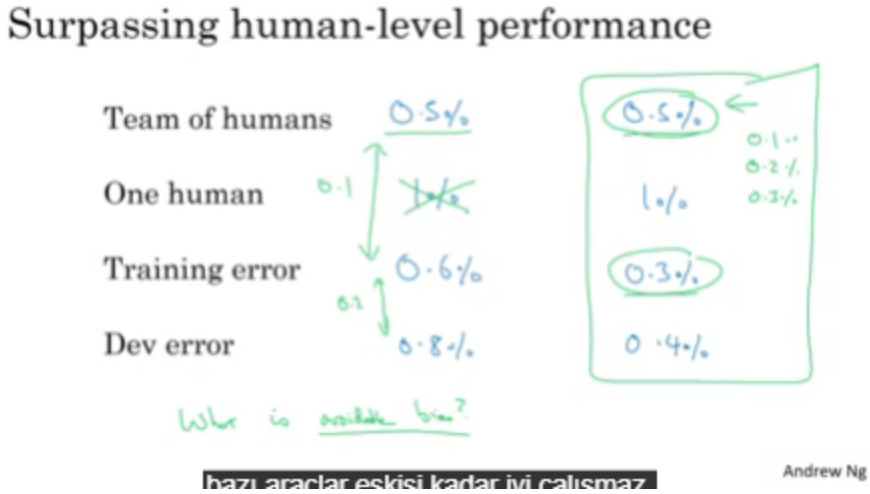
Ne hakkında konuştuğumuzu özetlemek gerekirse;

Eğer, insanların gayet iyi yapabileceği bir çalışma için insan düzeyi hatanın tahminini yaptığınız yerdeki bias,varyansı anlamaya çalışıyorsunuz, insan-düzeyi hatasını Bayes hatası için birer vekil veya tahmin olarak kullanabilirsiniz ve böylece, Bayes hatası tahmininizle olan fark size önlenebilir bias'ın ne kadar büyük bir problem olduğunu gösterecektir. Eğitim hatanız ve dev set hatanız arasındaki fark, varyansın ne kadar büyük bir problem olduğunu, algoritmanızın train setinden dev setine genelleyip genellemediğini anlatacaktır.



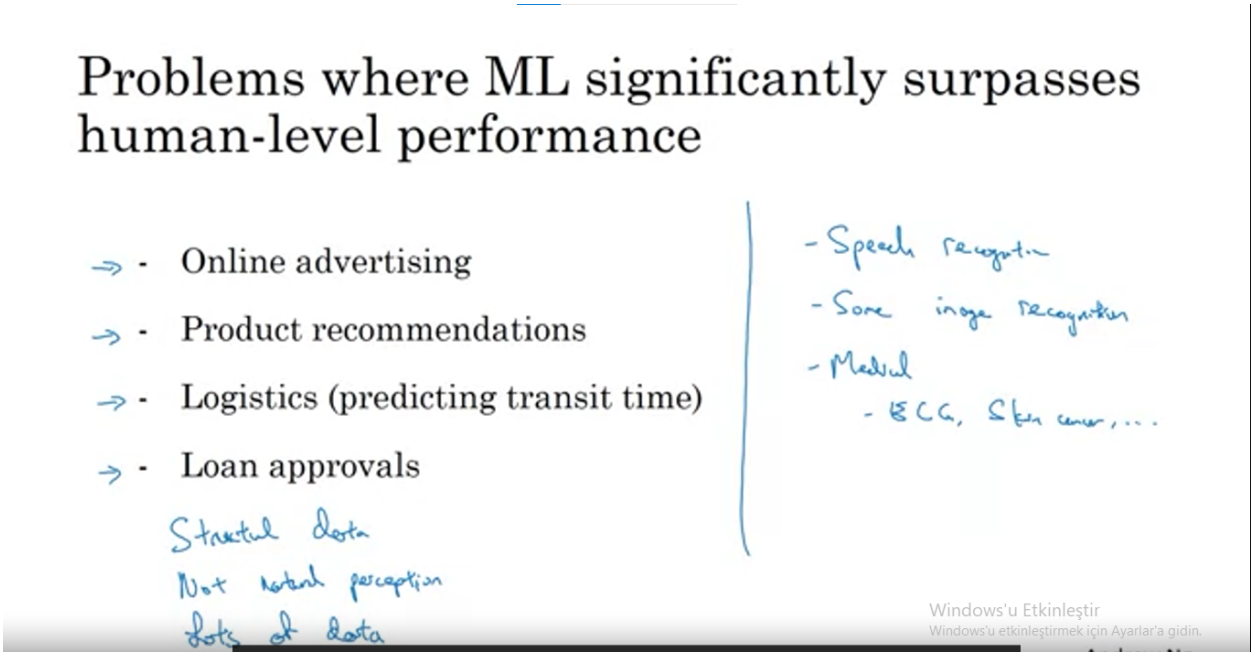
Surpassing Human-level Performance

İnsan seviyesini aşma hakkında bahsedelim. Soldaki örnekte 0.1 bias, 0.2 varyans mevcuttur sonuç olarak varyans ile ilgilenmek daha mantıklıdır. Sağdaki örnek 0.2 oranında overfitting olduğu anlamına mı gelir ya da taban hatanızın %0.1 olduğu anlamına mı gelir? Fakat bu örnekte verilen bilgilere dayanarak söylemek gerekirse, algoritmanızdaki bias mı varyansı mı azaltmanız gerektiğine odaklanmanızı söyleyecek yeterli bilgiye sahip olmadığınızı söylemek mümkün. Train setinde insandan iyi performans gösteriyorsanız, insandan yardım almak daha zor hal alabilir.



İnsandan daha iyi performans gösterecek model örnekleri;

4 verideki ortak nokta yapılı verilerle kullanıldığı satın alıdığı tıkladığı vs gibi önceki veirlerden çıkarımlı veriler mevcuttur.



Improving your Model Performance

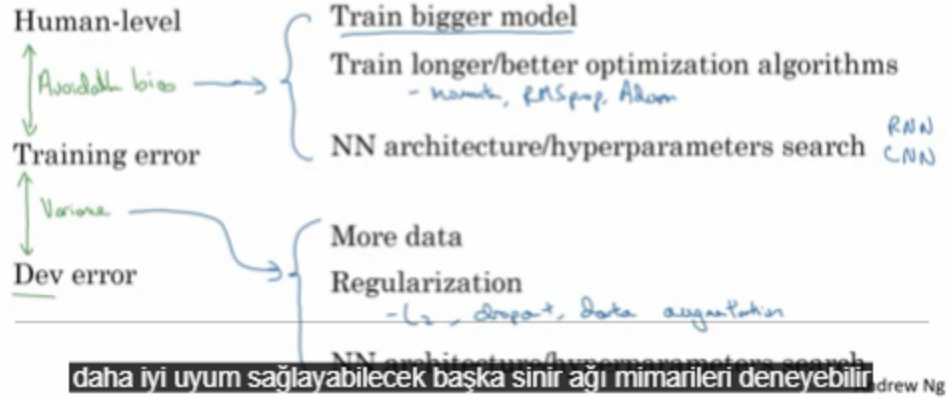
Tüm bu öğrendiklerimize bakarak bir çıkarımda bulunalım. Bir denetimli öğrenme algoritması seçmek doğru olacaktır. İyi bir train seti eğitip bias'ı düşük tutmaya çalışmak. Dev ve test settede iyi performans göstermesi gerekmektedir. dev ve test sette iyi performans gösteremeyen algoritma için daha büyük bir ağ eğitmek veya daha büyük bir train seti edinmek olabilir.

The two fundamental assumptions of supervised learning

1. You can fit the training set pretty well.
2. The training set performance generalizes pretty well to the dev/test set.

örneğin düzenlileştirme ya da daha fazla veri ile eğitme gibi. Andrew Ng

Reducing (avoidable) bias and variance



daha iyi uyum sağlayabilecek başka sinir ağı mimarileri deneyebilir drew Ng