-W6 - Spam Classifier-

- We will talk about ML System Design. These videoss will touch on the main issues that we may face when designing a complex machine learning system.
- We will see how to strategise putting together a complex ML system.
- ilerleyen konular alakasız görünebilir, bunun sebebi hepsinin complex ML tasarlarken karşılaşılabilecek farklı sorunlara değinecek olması.
- Çok matematiksel olgulardan bahsetmeyeceğiz ama yine de, bunlar bize büyük zaman kazandıracak.

- Let's begin with prioritizing how to spend our time on what to work on

---

-Let's say we wanna build a Spam Classifier-

- Diyelim ki elimle spam (1) ve non-spam (0) olarak etiketlenmiş mailler olsun How do we build a classifier using supervised learning?

1st Decision we must make: How do we want to represent x, that is the features of email.

- One way to choose: Choose 100 words indicative of spam/not spam.
  E.g: deal, buy, discount, Erdogan, urgent, ---

  Bunlardan bazıları spam olasılığını diğerleri ise spam olmama olasılığını temsilen seçiliyor.

- Bundan sonra her email'i encode edeceğim ve bundan bir input vektörü elde edeceğim. Eğer seçilen kelimelen ilgili mail-de var ise buna karşılık 1 ve yok ise 0 koyarız.

$$x = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \end{bmatrix} \begin{matrix} \text{deal} \\ \text{buy} \\ \\ \text{discount} \\ \\ \text{erdogan} \\ \text{urgent} \\ \\ \end{matrix} \Rightarrow x \in \mathbb{R}^{100} : \text{sanki 100 kelime seçtim}$$

$\Leftarrow$ Mail şu olsun: "Deal of the week! Buy now!"

$$\Rightarrow x_j = \begin{cases} 1 & \text{if word } j \text{ appears in email} \\ 0 & \text{otherwise} \end{cases}$$

- Bu bu örnekte manually 100 kelime seçiyoruz gibi göründük ama gerçekte what's most commonly done is look through a training set, and in the training set pick the most frequently occured n words (10000 - 50000).

W6 - Spam Classifier II -

Bu örnek için (high accuracy)

- How to spend our time to make the classifier have low error?

  - One natural inclination (meyil) is ~~good~~ collecting lots of data
    - ↳ Ama daha önceki videolarda gördük getting a lot of data will often help but not all the time.

  - Develop Sophisticated Features based on email routing information (from email header).
    - ↳ Genelde spamler farklı headerlara sahipler, mail farklı şekillerde kullanılır. Bunlardan features elde edersek performası artırabiliriz.

  - Develop sophisticated features for message body
    - ↳ E.g. should "discount" and "discounts" be treated as the same word? How about "deal" and "Dealer"? Features about punctuation?

  - Develop sophisticated algorithm to detect misspellings
    - ↳ E.g. m0rtgage, med1cine, w4tches.

Bir sonraki videoda Error Analysis konseptinden bahsedeceğiz. We will talk about the way where you can try to have a more systematic way and choose amongst the options of the many different things you might work on.

-W6- Error Analysis -

- Geçtiğimiz derste machine learning problems ile karşılaşıldığında bir çok farklı algoritma geliştirme yolu olduğundan bahsettik.
- Bu videoda ise Error Analysis kavramından bahsedeceğiz. Böylece problemlerle karşılaşıldığında daha sistematik bir yol izleyebileceğiz.

If you are starting work on a machine learning problem, or building a machine learning application there is a recommended approach:

- Instead of building a very complicated system with lots of complex features an so on, building a simple algorithm as a start that you can implement quickly is better choice. Implement something simple and quick and test it on your cross-validation data.

- Then we can plot learning curves to decide if more data, more features, etc. are likely to help. Algoritmanın high bias, high variance vb. ne problemi var gördüğü, bunu kullanarak ne gerektiğine karar veriyor.

  - The reason that this is a good approach is often, when you are just starting on a learning problem, there is really no way to tell in advance whether you need more complex features or you need more data or something else.
  - we should let evidence guide our decision on where to spend our time rather than use gut feeling, which is often wrong,

- In addition to plotting learning curves one other very useful thing is Error Analysis: Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.
  - ↳ Often by doing that is the process that inspires you to design new features.

Let's look at a specific example:

-W6 - Error Analysis II -

Error Analysis:

- mcv = 500 examples in cross validation set.
  Algorithm misclassifies 100 emails.
  Manually examine the 100 errors, and categorize them based on:
  (i) What type of email it is (might be about pharmacies, replicas, steal password)
  (ii) What cues (features) you think would have helped the algorithm classify them correctly.

- 1. işlem ile diyelim ki e-mail leri ayırdık (halalı olanları) ve geriye bu 100 hatanın 12'si pharmacy, 4'ü fake hesap satan, 53'ü şifre çalmaya çalışan kalan 31'i de başka şeyler. O zaman demek ki şifre çalmacı maillerinde algoritma kötü çalışıyor. Bunları daha iyi ayırabilecek features'lar ekleyebilirim.

- Ya da bakalım misspellings (medicine gibi) 4 tane, unusual email routing 16 tane, unusual punctuation 32 tane o zaman misspellingleri düzeltmek veya algılamak için advanced algorithmasına gerek yok. Bunun yerine punctuation'lara odaklanmak, buna zamanı harcamak daha makul görünüyor

Sonuç olarak bu şekilde bir Error Analysis ile manual olarak hata yapılan datalar incelenerek buradan sonuçlar çıkartılmaya çalışılır

Numerical Evaluation

Lastly when developing learning algorithms, one often useful tip is to make sure that you have a numerical evaluation of your learning algorithm.

- Yani algoritmayı evaluate eden ve sonucunda tek bir real number return eden bir yapı çok yararlı olur. Belki accuracy döndüren belki error. Single real number that tells you how well your learning algorithm is doing

Importance of Numerical Evaluation: Let's say we are not sure whether or not to treat same to words: discount / discounts / discounted / discounting

- "Stemming" software ile başlahi kelimelere bakılarak benzer ise aynı kabul edilir. Bunu yapmak yararlı olabilir ama zarar da verebilir.
- universe / university karışabilir.
- Thus, Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try and see if it works.
- Jcv ile karar verilebilir.
- Aynı şekilde uppercase - lowercase ayırmalımıyım aynı mı kabul etmeliyim? Yine Jcv ile veya başka bir evaluation ile karar verebilirim.

-W6- Error Metrics for Skewed Classes

- For error metrics there is 1 important case, where it's particularly tricky to come up with an error/evaluation metric for the training algorithm. This case is the case called skewed (eğik) classes. Let's talk about it:

Consider a Cancer Classification Example: Elimizde medical patients feature olsun, bunlara göre hasta cancer mı değil mi onu söyleriz. Malignant - benign tumor örneği gibi.

- Train a logistic regression model $h(x)$. ($y=1$ if cancer, $y=0$ o/w)
- Let's say we've test our classifier on a test set and find that we get 1% error on test (99% correct diagnoses)
- ★ Only 0.5% of patients training and test sets actually have cancer.
- Artık 1% error iyi görünmüyor çünkü benim hipotesim sürekli 0 tahmin etse yani her gelene kanser yok dese 0.5% error oluyor daha iyi performans sergiliyormuş gibi görünüyor.
- O zaman benim bu error metriğim bu tip durumlarda çok anlamlı sonuçlar vermiyor.
- # of positive examples is much much smaller than the # negative ex.s Bu yüzden bu durumlar case of skewed classes diyoruz. (vice versa da olu

---

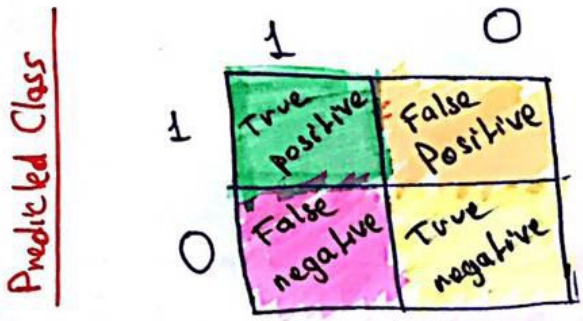
- Let's say we have a learning algorithm has 99.2% accuracy meaning 0.8% error. Then you made a change and the accuracy became 99.5%.
- Is this an improvement or not? Did we just made an useful change or did we just replace our code with something predicts "0" more often?
- So if you have skewed classes it becomes much harder to use just classification accuracy, because you can get very high classification accuricies or very low errors but it's not always clear that if you've improved your classifier performance. Sadece daha fazla 0 tahmin ederek accuracy artırabilirim.

Sonuç olarak skewed classes için ayrı bir error metric kullanmalıyız.

One such evaluation metric → Precision/Recall

Precision / Recall

- When defining Precision/Recall usually we use convention: $y=1$ for the rare class! (cancer / key)
- If we are making a binary classification there are 2 possible options for Actual Class: 0 or 1. Predicted class için de aynı.



Precision: Of all the patients where we predicted $y=1$, what fraction actually has cancer?

$$\frac{\text{True Positives}}{\#\text{ of predicted positives}} = \frac{\text{True Positives}}{\text{True Pos.s} + \text{False Pos.s}}$$

Recall: Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?

$$\frac{\text{True Pos.s}}{\#\text{ of actual pos.s}} = \frac{\text{True Pos.s}}{\text{True Pos.s} + \text{False Negs}}$$

If a predictor always predict $y=0$ all the time and says no one has this rare condition (cancer) than the recall will be zero!

Trade-Off b/w Precision and Recall

- Precision ile Recall arasında bir tradeoff vardır. Bunu kontrol etmek isterz.
- Aynı example ile devam edelim logistic regression ile cancer classification
  - $0 < h(x) < 1$
  - Predict 1 if $h(x) \geq 0.5$
  - Predict 0 if $h(x) < 0.5$

Suppose we want to predict $y=1$ (cancer) only if very confident → avoids false positives

- Adama durduk yere kanser oldun demek ve hata yapmak istemiyoruz.
- Bunu sağlamanın bir yolu $h(x) \geq 0.9$ için $y=1$ $h(x) < 0.9$ için $y=0$ predict etmektir. Böylece sadece kendime güvenirsem $y=1$ diyeyim emin olunca positive diyorum ama false negative olan için 0.9 sağlasa da sadece değil diyorum.
- Higher precision, lower recall.

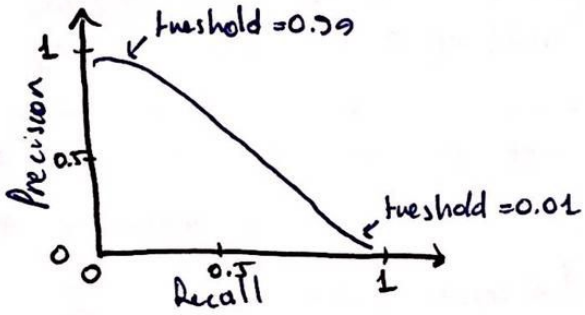
Suppose we want to avoid missing too many cases of cancer → avoids false negatives

- Kanseri olabilecek adama üzgün demek de cannot mal olun. İstemiyoruz.
- $h(x) \geq 0.1$ için kanser ($y=1$), $h(x) < 0.1$ için $y=0$ diyebiliriz.
- Higher recall, lower precision.

- W6 - Trade-off btw Precision / Recall -

Sonuçta $h(x) \geq$ threshold için threshold değerine göre $y=1$ ve $y=0$ tahmini yapıyor idik. Bu threshold değeri Precision ile Recall trade-off'unu belirliyor.

- Precision ile Recall arasında aşağıdakine benzer bir ilişki vardır (yalın)



- How to choose this Threshold?
- Farklı algoritmaların farklı Precision ve Recall değerleri olsun:

	P	R
Alg 1 :	0.5	0.4
Alg 2 :	0.7	0.1
Alg 3 :	0.02	1.0

Which one is better?

Farklı threshold değerleri farklı Precision (P) ve Recall (R) değerleri verir hangisinin daha iyi olduğuna karar vermek zor. Bize tek bir numerical evaluation lazım ki direkt karar verebilelim. Nasıl?

- $\frac{P+R}{2}$ alsak nasıl olur? Olmaz. Çünkü sanki $y=1$ tahmin ederse recall = 1 olur ortalama alınca da sonuncu mesela 0.51 gelir en iyisi ama aslında ise yaramaz bir algoritma. Ya da her zaman $y=0$ derse high precision low recall.

F_1 Score: $2\frac{PR}{P+R}$ if $P=0$ or $R \approx 0$ $F_1 \approx 0$ yani iyi bir F_1 score için ikisi de iyi olmalı!

$F_1 = 0$ for $P=0$ and $R=0$

$F_1 = 1$ for $P=1$ and $R=1$

Böylece evaluation metrics'den bahsetmiş olduk.

W6 - When to Use Large Data Sets -

- We will focus on the issue of how much data to train on?
- Önceki videolarda durduk yere gidip daha fazla training data toplamanın her zaman işe yaramayacağını görmüştük.

Under certain conditions, getting a lot of data and training of certain type of learning algorithm can be a very effective way to get a learning algorithm to do very good performance.

- "It's not who has the best algorithm that wins. It's who has the most data."
- So when is this true and when is not?

2 conditions önemli:

1) Feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately.

- Yani kullanılan x ile gerçekten sonuç tahmin edilebilir olmalı. Bunu anlamak için sorulabilecek sorulardan biri şu: Kullanılan input ile bir human expert güvenli bir tahmin yapabilir miydi?
  - Bu condition sağlanmıyorsa data toplamak useless.

---

2) Use a learning algorithm with many parameters (e.g. logistic regression / linear regression with many features; neural network with many hidden units.)

- Böylece amaç low bias, overfitting'e yatkın bir algoritma elde etmek. Overfitting de yüksek training set büyüklüğü ile elimine edilecek böylece low bias, low variance hipotesimiz olacak. • Low bias algorithms
  - $J_{train}(\Theta)$ will be small (çünkü training setle iyi olmaca)

Use a very large training set (unlikely to overfit) - provides low variance

- Training set çok büyük overfitti genecel.
- $J_{train}(\Theta) \approx J_{test}(\Theta)$ and $J_{test}(\Theta)$ will be small!