PROTEİN ETKİLEŞİMLERİNİN TAHMİNİNDE POZİTİF ETİKETLENMEMİŞ ÖĞRENME

CUMHUR KILIÇ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

TEMMUZ 2012

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Ünver KAYNAK
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU
Anabilim Dalı Başkanı

CUMHUR KILIÇ tarafından hazırlanan PROTEİN ETKİLEŞİMLERİNİN TAHMİNİNDE POZİTİF ETİKETLENMEMİŞ ÖĞRENME adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Yrd. Doç. Dr. Mehmet TAN
Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Tansel ÖZYER _____

Üye : Yrd. Doç. Dr. Mehmet TAN _____

Üye : Doç. Dr. Bülent TAVLI _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Cumhur KILIÇ

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Yrd. Doç. Dr. Mehmet TAN**
Tez Türü ve Tarihi : **Yüksek Lisans – Temmuz 2012**

Cumhur KILIÇ

PROTEİN ETKİLEŞİMLERİNİN TAHMİNİNDE POZİTİF ETİKETLENMEMİŞ ÖĞRENME

ÖZET

Bir veri kümesindeki örneklerin belli bir özelliğe sahip olup olmayışlarına göre etiketlendirilmeleri işlemine ikili sınıflandırma adı verilir. Bir ikili sınıflandırıcı eğitebilmek için, genel yaklaşımda, hem pozitif hem de negatif örnekler içeren bir eğitim verisine ihtiyaç duyulur. Ancak bazı çalışma alanlarında negatif örneklerin elde edilmesi zor, hatta imkansız olabilir. Bu durumlarda veri kümesi sadece pozitif örnekler ve üye oldukları sınıfların belirlenmesi hedeflenen etiketlenmemiş örneklerden oluşur. Bu tür problemlere bir örnek protein-protein etkileşim ağlarının tahminidir.

Bir canlı vücudunda hayati işlemlerin devamlılığı proteinlerin çalışmasına bağlıdır ve proteinler bu işlemler sırasında birbirleriyle etkileşime girerler. Hangi proteinlerin birbirleriyle etkileştiğinin bilinmesi tıbbi açıdan önemli bir bilgidir. Proteinlerin etkileştiği laboratuar deneyleri ile tespit edilebilirken, aksi durum kesin bir şekilde belirlenemez. Deneyler sırasında bir protein çiftinin etkileştiğine şahit olunmaması, bu çiftin başka bir zaman ve durumda etkileşmeyeceğinin kanıtı olamaz.

Bu çalışmamızda negatif eğitim verisinin mevcut olmadığı bu durumlarda kullanılabilinecek olan algoritmaları özetledik ve bu algoritmaların bir kısmını protein-protein etkileşimlerinin tahmininde kullanarak test edip karşılaştırdık. Böylece protein-protein etkileşim ağlarının tahmininde kullanılabilecek veya bu işlem için ümit vadeden algoritmaları belirledik.

Anahtar Kelimeler: veri kümesi, ikili sınıflandırıcı, pozitif örnek, negatif örnek, protein-protein etkileşim ağı.

University	**:**	**TOBB University of Economics and Technology**
Institute	**:**	**Institute of Natural and Applied Sciences**
Science Programme	**:**	**Computer Engineering**
Supervisor	**:**	**Asst. Prof. Mehmet TAN**
Degree Awarded and Date	**:**	**M.Sc. – JULY 2012**

Cumhur KILIÇ

POSITIVE UNLABELED LEARNING FOR DERIVING PROTEIN INTERACTION NETWORKS

ABSTRACT

Binary classification is the process of labeling the members of a given data set on the basis of whether they have some property or not. To train a binary classifier, normally one needs two sets of examples from each group, usually named as positive and negative examples. However, in some domains, negative examples are either hard to obtain or even not available at all. In these problems, data consist of positive and unlabeled examples. An example to this kind of problems is derivation of protein-protein networks.

Biological processes in a living organism depend on proteins and mostly interactions of proteins. It is important to determine which proteins interact to understand how an organism survives. While it is possible to derive by experiments that two proteins interact with each other, it is much harder to conclude that they do not. Even if we do not observe the interaction of two proteins during an experiment, they may interact in a different time or condition.

In this thesis we first present a survey of algorithms which can handle such problems, and then provide a comparison of some of these algorithms on the protein-protein interaction derivation problem by using the available (positive) interaction information. Thus we identify which algorithms can be used or have potential to be used for deriving protein-protein interaction networks.

Keywords: data set, binary classifier, positive example, negative example, protein-protein interaction networks.

TEŞEKKÜR

Yüksek lisans eğitimim boyunca beni yönlendiren, bana sevdiğim bir alanda araştırma yapma imkanı sağlayan danışmanım Mehmet Tan'a sonsuz teşekkürlerimi sunarım. Kendisiyle çalışmış olmak benim için büyük bir ayrıcalık olmuştur.

Başta tez jürimde yer alan Tansel Özyer ve Bülent Tavlı olmak üzere, bu iki sene içinde derslerine katılmış olduğum ya da asistanlıklarını yaparak birlikte çalışma şansı yakaladığım tüm hocalarıma minnettarım.

Benden hiçbir yardımı esirgemeyen asistan arkadaşlarıma, özellikle de TOBB ETU'yu benim için bir okul ve işyerinden çok daha fazlası haline getiren oda arkadaşlarıma hayatımı renklendirdikleri için teşekkür ederim.

Son ve en önemli olarak da, hayatımın her döneminde beni destekleyen, bana her aşamada yol gösteren ve her zaman yanımda olan aileme gönülden teşekkürlerimi sunarım.

İçindekiler

Tablo Listesi

1. GİRİŞ

İkili sınıflandırma problemleri 2 grup örnek içerirler. İlk grup belli bir özelliğe sahip olan örneklerden oluşur ve pozitif sınıf olarak adlandırılır. İkinci grup, yani negatif sınıf, ise örneklem uzayında bulunan diğer tüm örnekleri içerir. Bir örneğin pozitif ya da negatif olduğu bilinmiyorsa o örneğe etiketlenmemiş örnek denir. Bir ikili sınıflandırıcının hedefi de etiketleri hali hazırda bilinen pozitif ve negatif örneklerden elde edeceği bilgiler ışığında etiketlenmemiş örnekleri doğru şekilde sınıflandırmaktır.

Gözetimli öğrenme (Supervised learning) algoritmaları sınıflandırıcı eğitiminde genellikle pozitif ve negatif örnek kümeleri kullanırlar. Fakat çoğu çalışma alanında negatif örneklerin elde edilme maliyeti pozitif örneklerle karşılaştırıldığında çok daha yüksektir. Hatta bazı durumlarda negatif örneklerin elde edilmesi imkansız bile olabilir. Bu gibi durumlarda sadece pozitif ve etiketlenmemiş örnekler kullanarak çalışabilen algoritmalara ihtiyaç duyulur. Pozitif Etiketlenmemiş (Positive Unlabeled – PU) Öğrenme Algoritmaları [4] denilen metot grubu da negatif örneklerin yokluğunda sınıflandırma işlemini yapabilmeyi amaçlamaktadır.

İkili sınıflandırma; gen düzenleyici ağların türetilmesi, bulgu-hastalık ilişkileri, metin ve web sayfası sınıflandırmaları gibi farklı uygulamalarda kullanılmaktadır. Bu tezde PU öğrenme algoritmaları protein-protein etkileşimi (protein-protein interaction — PPI) ağlarının tahmini için kullanılmıştır. Bir PPI ağı; düğümlerin proteinleri, kenarların ise proteinler arasında olan ya da olmayan etkileşimi temsil ettiği bir çizge olarak gösterilebilir. Yapılan laboratuar deneyleri ile iki protein arasında var olan bir iletişimi gözlemlemek, göreceli olarak, kolaydır. Diğer yandan, iki protein arasında etkileşim olmadığını kanıtlamak ise çok daha zordur. Yapılan deneyler sırasında iki proteinin etkileşim içine girmemiş olması, onların

başka bir ortam veya zamanda etkileşmeyeceklerini kanıtlamaz.

Ele aldığımız problemin tabiatını aşağıdaki şekilde betimleyebiliriz;

1. Veri kümemizde sadece pozitif örnekler ve etiketlenmemiş örnekler bulunuyor. Eğitimde kullanılabilinecek negatif olduğu bilinen örneklere sahip değiliz. Diğer yandan, etiketlenmemiş örnekler aslen pozitif ya da negatif olabilirler.

2. Etiketlenmemiş küme içindeki aslen negatif olan örneklerin sayısının, aslen pozitif olan örneklerin sayısından daha fazla olması beklenir. Bilinen tüm olası protein çiftleri içinde sadece küçük bir yüzdelik protein çifti etkileşmektedir. Aynı şekilde pozitif kümenin boyutu da etiketlenmemiş kümeye oranla oldukça küçüktür.

Ele aldığımız konuyla ilişkili bir diğer problem ailesi yarı gözetimli öğrenmedir (semi supervised learning — SSL). SSL, etiketlenmiş pozitif ve negatif örnekleri elde etmenin zor olduğu problemleri içerir. Bu tür problemlerde mevcut örneklerin büyük kısmı etiketlenmemiş örneklerden oluşur ve bu etiketsiz örneklerin yanında her iki sınıftan da az sayıda örnek bulunur. PU öğrenme SSL'nin bir alt kategorisi olarak görülebilir. Fakat PU öğrenme problemlerinde eğitim verisinde negatif örnek bulunmadığı için PU öğrenme ve SSL algoritmaları farklılıklar göstermektedir. Negatif örneklerin eksikliği problemi daha zor hale getirerek algoritmaların bu eksiği telafi edecek şekilde çalışmalarını zorunlu kılar. SSL algoritmaları bizim bu çalışmamızın kapsamı dışındadır. Bu konuyla ilgilenen okurlar detaylı bilgiyi [27, 29, 30, 31]'da bulabilirler.

Bu çalışmada mevcut PU öğrenme algoritmalarını özetleyerek sınıflandırdık. Ele aldığımız bu algoritmalardan bazıları PPI ağları üzerinde kullanılmaya hali hazırda uygun algoritmalardır. Çoğu metin sınıflandırma için tasarlanmış olan diğer algoritmalar ise, kullanılmak için geliştirildikleri alanlara özel işlemler içermeleri sebebiyle PPI ağları ile doğrudan kullanılamamaktadır. Algoritma ailesinin bütünlüğünü sağlamak için bu algoritmaları da Bölüm 2'de ele aldığımız algoritmalar listemize dahil ettik. Zhang et al. Tarafından yazılmış olan [28] PU öğrenme algoritmalarını özetleyen bir başka kaynaktır. Fakat bizim çalışmamız

çok daha geniş kapsamlıdır ve onların makalesi karşılaştırmalı bir değerlendirmeyi içermemektedir.

Ele aldığımız algoritmaları detaylı şekilde gözden geçirip, nasıl çalıştıklarını, birbirlerinden farklılıklarını ve hangi özel durumlar için tasarlandıklarını açıkladık. Daha sonra bu algoritmalardan sekizini PPI ağlarının türetilmesindeki başarılarına göre karşılaştırdık. Bildiğimiz kadarıyla bu çalışmamızla biyolojik veriler temel alınarak PU algoritmaları protein etkileşimlerinin tahmininde ilk kez kullanılmıştır.

2. PU ÖĞRENME

PU öğrenme algoritmaları eğitim verisinde negatif örneklerin bulunmadığı durumlar için tasarlanmışlardır. Fakat bir algoritmanın etiketlenmemiş örnekleri sınıflandırabilmesi için pozitif ve/veya negatif örneklerin özelliklerini bilmesi gerekir. PU algoritmalarını diğer sınıflandırma algoritmalarından ayıran fark, sınıfların karakteristiklerini öğrenmek için izledikleri yollardır. Dolayısıyla bu çalışmada algoritmaları negatif örnekler olmadan sınıfların özelliklerini öğrenme stratejilerine göre sınıflandırdık.

Ele aldığımız algoritmaların neredeyse tümü sınıflandırma işlemlerinin çeşitli basamaklarında destekçi vektör makinası (support vector machine — SVM) ya da lojistik regresyon (logistic regression) gibi klasik gözetimli sınıflandırma yöntemlerini kullanmaktadırlar. Tek başlarına PU öğrenme problemleri için başarılı olamayacak olan bu yöntemlerin algoritmalar tarafından ne için ve nasıl kullanıldığı, algoritmaların klasik yöntemler dışında ne gibi metotlar ile sonuca ulaştıklarını açıklayacağız.

Bu bölümde iki ana yaklaşımı kullanan algoritmalar incelenmiştir: 1. Etiketlenmemiş örnekler arasından bir takım güvenilir negatif örnek seçen, daha sonra bu negatif küme ve başlangıçta sahip olduğumuz pozitif kümeyi kullanarak sınıflandırma yapan iki-basamaklı stratejiler. 2. Pozitif ve etiketlenmemiş örnekleri doğrudan yeni örnekleri sınıflandırmak için kullanan bir-basamaklı stratejiler.

2.1 İki-Basamaklı Algoritmalar

Etiketlenmemiş kümeden bir takım güvenilir (güçlü) negatif örnekler seçerek çalışmaya başlayan algoritmalara iki-basamaklı algoritmalar denir. Bu iki basamak şunlardır:

1. Etiketlenmemiş küme içinden güçlü (negatif olma olasılığı yüksek olan) bir takım negatif örneğin seçimi.
2. Pozitif küme ve hazırlanan yeni negatif küme ile bir ya da bir seri sınıflandırıcı eğiterek etiketlenmemiş örneklerin sınıflandırılması.

PU öğrenme algoritmaları arasında çok sayıda iki-basamaklı algoritma vardır. Ele alınabilinecek en ilkel yöntem, etiketlenmemiş kümenin tümünü negatif olarak kabul ederek eğitim verisi olarak kullanmaktır. Bu negatif örnekler ve baştan bilinen pozitif örnekler kullanılarak eğitilen bir sınıflandırıcı, etiketlenmemiş örneklerin büyük kısmı aslen negatif olduğu için bir takım doğru sınıflandırmalar yapacaktır. Diğer yandan, etiketlenmemiş kümenin içinde pozitif örnekler de bulunduğu için sınıflandırıcı negatif sınıfın özelliklerini yanlış öğrenir ve bunun sonucunda da sınıflandırmada yanlış etiketlemeler yapılabilir. İlkel olarak tanımladığımız bu metodu SVM_{only} adıyla kodlayarak sonuçlarını bölüm 3'te sunduk. Bu bölümde ele aldığımız algoritmalar, güvenilir negatifleri seçmek için daha sistematik yöntemler izlemektedirler.

2.1.1 Carter et al. Algoritması

[1]'deki algoritma, ilk olarak tüm U'yu negatif olarak etiketler. Daha sonra bu negatif küme ile P'yi kullanarak bir sınıflandırıcı eğitir. Önceki bölümde de açıkladığımız üzere, problemimizde U'nun boyutu P'ye göre çok daha büyüktür. Dengeli boyutlarda eğitim verisi kullanılarak yaratılan sınıflandırıcılar göreceli olarak daha başarılı olurlar. Dolayısıyla negatif kümesi ve P'nin tümünü eğitim için kullanan bir algoritma ile zayıf bir sınıflandırıcı elde edilecektir.

Bu sorunu çözmek için Carter et al. U'yu alt kümelere bölmüştür. Bu parçalama işleminde U, alt kümelerinin boyutu P'nin boyutuna yaklaşık olacak şekilde n

adet alt kümeye bölünür (makalede E. Coli veri kümesi için n 5 alınmıştır). Algoritma daha sonra her alt kümeyi teker teker ve birbirinden bağımsız olarak orijinal pozitif kümemizle birlikte eğitim ve sonrasında da sınıflandırma için kullanır.

Bu rastgele alt küme oluşturma stratejisi Bagging [16] algoritmasıyla benzerlik taşımaktadır. Diğer yandan, bu işlem algoritma tarafından yapılmamıştır. Yazarlar veri kümesini n alt kümeye bölmüş, programlarını her alt küme ve P için tekrar çalıştırmışlardır. Dolayısıyla algoritmanın asıl yaptığı, kendisine verilen U'nun tamamını herhangi bir ölçü ile seçim yapmadan N olarak kullanmaktır.

P ve oluşturduğu N'yi eğitim verisi olarak kullanacak olan algoritma, yine N üzerinde sınıflandırma yapacaktır. Bir sınıflandırıcının eğitiminde kullanılan verinin aynı zamanda test aşamasında da kullanılması sağlıklı bir sonuç vermeyeceği için, algoritma birini-dışarıda-bırak (leave-one-out) çapraz-doğrulama (cross-validation) (LOOCV) uygulamaktadır. LOOCV işleminde her seferin bir etiketlenmemiş örnek test verisi olarak kullanılırken, kümedeki diğer tüm örnekler eğitim için kullanılırlar.

Algoritma bu ilk sınıflandırmada pozitif olarak etiketlenen örnekleri negatif kümeden çıkararak negatif kümesini arındırır. Bu arındırma işleminden sonra LOOCV tekrar uygulanır ve örneklerin son etiketleri belirlenmiş olur.

U'nun büyük bir kısmının gerçekten de negatif olduğu göz önüne alınırsa, negatif örneklerin U'dan rastgele seçilmesi etkili bir yöntem olarak görülebilir. Rastgele seçim ile tamamen negatiflerden oluşan bir N yaratılma şansı yüksektir (alt kümelerin bir kısmında). Fakat yaratılan N içinde pozitiflerin bulunma ihtimali de vardır ki bu durum sınıf sınırlarını ve örneklerin etiketlerini hatalı şekilde belirleyecek olan bir sınıflandırıcı oluşturulmasına sebep olabilir. Bu sorunu aşmak için takip eden algoritmalarda negatif örnekler rastgele değil, bazı veriye-bağlı ölçümlerle seçilmektedir.

2.1.2 Positive Sample only Learning (PSoL)

PSoL [2] etiketlenmemiş kümeden negatif örnekleri Öklid Uzaklığı, Maksimum Uzaklık Minimum Fazlalık (Maximum Distance Minimum Redundancy –

MDMR) [24] teknikleri ve bir seri SVM sınıflandırıcısını kullanarak seçer. Seçilen örnekler güçlü negatif olarak değerlendirilir ve negatif kümesini oluştururlar. Daha sonra bu yeni negatif kümesi pozitif kümeyle birlikte eğitim için kullanılarak kalan etiketlenmemiş örnekler sınıflandırılır.

PSoL 3 adımdan oluşur: Başlangıç negatiflerinin seçilmesi, negatif kümesinin genişletilmesi, pozitif ve negatif kümeler kullanılarak sınıflandırmanın yapılması.

Algoritma ilk olarak U içindeki örneklerin P'deki örneklere uzaklıklarını hesaplar. Daha sonra bu uzaklıkları kullanarak P'deki örneklere toplam uzaklığı en fazla olan etiketlenmemiş örneği bulur. Bu örnek seçilen ilk güçlü negatiftir ve U'dan alınarak N'ye konulur. Algoritma bu örnekten başlayarak yinelemeli şekilde yeni negatif örnekler seçer. Her yinelemede denklem 2.1'i karşılayan örnek N'ye aktarılır.

$$\max_{x_i \in U} [\min_{x_j \in P} d(x_i, x_j) * \sum_{x_k \in N} d(x_i, x_k)] \tag{2.1}$$

Eğer U içinde negatif örnekler varsa, bu örneklerin öznitelik uzayında pozitif örneklerden uzakta bulunacaklarını varsayabiliriz. Denklem 2.1'de bunu temel alarak etiketi bilinen pozitiflerden uzak örnekleri negatif olarak seçmeyi amaçlar. Denklemin sağlamaya çalıştığı diğer sonuç ise; seçilecek olan yeni negatif örneklerin, hali hazırda N içinde bulunan örneklerden maksimum uzaklıkta olmalarıdır. Böylece veri kümesindeki tüm negatifleri temsil edebilecek çeşitliliğe sahip bir negatif kümesi oluşturmak hedeflenir.

Başlangıç negatifleri seçildikten sonra (algoritmanın ikinci basamağında) yinelemeli şekilde yeni negatifler seçilir. Her yinelemede P ve N'nin son hali kullanılarak yeni bir sınıflandırıcı eğitilir. Sınıflandırıcı kullanılarak U'daki örnekler sınıflandırılır. Sınıflandırma sonucunda negatif olma olasılığı belli bir sınırdan daha yüksek olan örnekler N'ye aktarılır. Böylece U ve N güncellenmiş olur. Sonraki yinelemede bu yeni U ve N kullanılacaktır. Sınıflandırıcıların örnekler için elde ettikleri karar fonksiyonu sonucu [-1,1] aralığındadır. Bir örneğe atanacak olan etiket, aslen örneğin sonucunun 0'dan küçük ya da büyük olduğuna göre belirlenir. Örneğin, -0,01 sonucu alınan bir örnek negatif olarak etiketlenecektir. PSoL'un bu aşamadaki hedefi tüm U'yu etiketlemek değil, sadece bazı

güçlü negatifler bulmak olduğu için, orijinal olarak 0 olan bu sınır yerine -0,2 kullanılmıştır. Bu sınırdan daha küçük değer alan örnekler negatif olarak seçilmiştir. Böylece gerçekten negatif olan örneklerin seçilme olasılığı arttırılmaya çalışılmıştır.

Bir diğer kısıt ise her yinelemede negatif olarak seçilerek N'ye aktarılan örnek sayısındadır (K). Yinelemelerde belli sayıda en güçlü örneği seçmek ve sonraki sınıflandırma işlemlerini bunların ışığında yapmak, belirlenmiş olan sınırdan düşük sonuç alan tüm örnekleri bir anda negatif olarak seçerek eğitimde kullanmaktan daha güvenli bir yaklaşımdır. Bu sebepten dolayı algoritma her yinelemede en fazla $K = |P| * r$ adet örneği N'ye aktarmaktadır. Algoritmada r için 3 değeri kullanılmıştır.

Daha fazla örnek negatif olarak seçilemediğinde yinelemeler sonlandırılır. Bu noktada PSoL'un elinde P, U ve U'dan çıkarttığı elemanlarla oluşturduğu N vardır. Algoritma P ve N'yi kullanarak son bir sınıflandırıcı eğitir ve bu sınıflandırıcıyı kullanarak U'da kalan örnekleri test edip sınıflandırır. Böylece tüm etiketlenmemiş örnekler sınıflandırılmış olur.

PSoL'da olduğu gibi Rocchio [5] tekniğinde de güçlü negatifler seçilirken bir benzerlik ölçüm yöntemi kullanılır. Aradaki fark şudur ki, PSoL veri kümesindeki tüm örnekleri ikililer halinde karşılaştırırken, Rocchio metodu pozitif ve negatif sınıfları temsil edecek birer prototip yaratır ve veri kümesindeki örnekleri bu prototiplerle karşılaştırır.

2.1.3 The Rocchio Technique ve SVM (RocSVM)

[5]'teki algoritma güçlü negatifleri seçmek için bir Rocchio sınıflandırıcısı kullanır. İkinci basamakta ise P ve seçtiğimiz negatifleri kullanarak bir SVM sınıflandırıcısı eğitir.

Rocchio metodu ilk basamağında kendi özel sınıflandırıcısını üretir. Rocchio sınıflandırıcısı temel olarak pozitif ve negatif sınıfları için birer prototip öznitelik vektörü (feature vector) tanımlanmaya dayanır. Bu prototipler P ve U kullanarak yaratılır ve prototipi oldukları sınıfların karakteristiklerini taşıyan birer örnektirler. Pozitif ve negatif prototip vektörleri sırasıyla $\vec{c}^{+}$ ve $\vec{c}^{-}$ ile gösterilir

ve aşağıdaki şekilde denklemler kullanılarak tanımlanırlar. Algoritmada $\alpha = 16$, $\beta = 4$ değerleri kullanılmıştır.

$$\vec{c^{+}} = \alpha \frac{1}{|P|} \sum_{\vec{d} \in P} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|U|} \sum_{\vec{d} \in U} \frac{\vec{d}}{||\vec{d}||} \tag{2.2}$$

$$\vec{c^{-}} = \alpha \frac{1}{|U|} \sum_{\vec{d} \in U} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|P|} \sum_{\vec{d} \in P} \frac{\vec{d}}{||\vec{d}||} \tag{2.3}$$

Prototipler yaratıldıktan sonra etiketlenmemiş örneklerin prototiplere benzerlikleri kosinüs benzerliği ile hesaplanır. $\vec{c^{-}}$'ye $\vec{c^{+}}$'dan daha çok benzeyen tüm örnekler güçlü negatif olarak seçilir ve RN adlı kümeye aktarılır. Bu basamakta pozitif kümeye aktarım yapılmaz.

İkinci basamakta algoritmanın elinde P ve seçilmiş negatiflerden oluşan RN vardır. Bu iki kümeyle başlayarak yinelemeli şekilde U'dan yeni negatifler seçilir. Her yinelemede algoritma P ve RN'nin o anki haliyle bir SVM sınıflandırıcısı eğitir ve U'daki örnekleri bu sınıflandırıcı ile test eder. Sınıflandırıcının negatif olarak etiketlediği örnekler RN'ye aktarılır. Yinelemeler daha fazla örneğin negatif olarak seçilemediği noktada sonlanır.

Yinelemeler sonlandığında U'da kalan örnekler vardır ve bu örneklerin pozitif mi yoksa negatif mi olduklarına karar verilmesi gerekir. Dolayısıyla son bir sınıflandırma işlemi yapılmalıdır. Bu işlem için son yinelemede üretilmiş olan sınıflandırıcı (C_{last}) kullanılabilir. Fakat algoritma C_{last}'u bu iş için kullanmadan önce test eder. C_{last} elemanlarının pozitif olduğundan emin olduğumuz P'yi sınıflandırmak için kullanılır. Bu sınıflandırma sonucunda örneklerin %5'inden daha fazlası negatif olarak etiketlenmişse, algoritma C_{last}'ın başarısız bir sınıflandırıcı olduğuna, dolayısıyla da yinelemeli negatif seçiminde hatalı seçimler yapıldığına kanaat getirir. Bu durumda ilk yinelemede üretilen, yani ilk basamaktan gelen P ve RN (genişletilmemiş hali) kümeleriyle eğitilmiş olan ilk sınıflandırıcı (C_{first}) son sınıflandırma için kullanılarak U'daki örnekler sınıflandırılır. Eğer Cson %5 veya daha az örneği negatif olarak seçerse, yani hata oranı %5 veya altındaysa, algoritma C_{last}'u başarılı olarak kabul eder ve son sınıflandırmayı bu sınıflandırıcı ile yapar.

Bu algoritma Rocchio sınıflandırıcısını elinde sadece etiketlenmemiş ve pozitif örnekler olduğunda, SVM sınıflandırıcısını ise pozitif, negatif ve etiketlenmemiş örnekler olduğunda kullanmaktadır. [5]'in yazarları bu durumu her iki sınıflandırıcının da birlikte kullanıldıkları veri türünde daha başarılı oluşuyla açıklamışlardır.

Bu algoritmanın opsiyonel bir işlemi daha vardır. Güçlü negatifler seçilerek RN kümesi oluşturulduğunda bazı örnekler hatalı şekilde negatif olarak etiketlenmiş olabilir. Bu hataları yakalamak için algoritma RN üzerinde öbekleme (clustering) yapar. Sınıflandırıcı eğitiminde tüm RN'yi kullanmak yerine işlemi RN'nin alt kümelerine indirgeyerek RN içindeki aykırı örnekleri yakalamayı hedefler. Seçilecek olan alt küme sayısı bu işlemin başarısı açısından önemlidir.

PSoL [2] ve Rocchio [5] gibi algoritmalar örneklerin öznitelik vektörlerini uzaklık/benzerlik ölçüm yöntemleriyle kullanarak çalışırken, PN-SVM [6] ve M-C [3] gibi algoritmalar özniteliklerin örneklerdeki sıklıklarını kullanırlar.

2.1.4 Positive-Negative Document Enlarged Classifier (PN-SVM)

[6], ilk olarak, veri kümesindeki örneklerin öznitelik değerlerini normalize eder. Daha sonra özniteliklerin P'de bulunan örneklerde görülme sıklıklarını hesaplar. Elde edilen sıklık değerleriyle, pozitif örneklerde görülme sıklığı (kuvveti) belli bir değerin üzerinde olan öznitelikleri içeren ve çekirdek sözlük (core dictionary) denilen bir sözlük oluşturur.

PN-SVM pozitif çekirdek sözlüğü yarattıktan sonra bu sözlükteki özniteliklerden en azına sahip olan etiketlenmemiş örnekleri bulur. Bu örnekler güçlü negatifler olarak seçilir ve N'ye aktarılır. Bu aşamada U'daki tüm negatif örnekler güçlü negatif olarak seçilmeyebilir. Bunun sebebi bazı negatiflerin pozitif çekirdek sözlüğündeki bazı özniteliklere sahip olma ihtimalinin olmasıdır. Algoritmanın bu aşamadaki hedefi de zaten negatiflerin tümünü yakalamak değil, bazı güçlü negatifler bulmaktır.

Güçlü negatiflerin seçilmesinden sonra algoritma etiketlenmemiş kümeden yinelemeli şekilde pozitif ve negatif örnekler seçerek P ve N'yi genişletir. Bu algoritmanın diğerlerinden önemli bir farkı, çoğu PU algoritması son sınıflandırma işlemi öncesinde U'dan sadece negatif örnekler çıkartırken, PN-SVM algoritmasının pozitif örnekler de seçerek P'yi de genişletmesidir. Bu özelliği PN-SVM'i az sayıda pozitif örnek bulunan veri kümeleri için elverişli kılar.

PN-SVM, negatif sınıfın bir takım alt sınıflardan oluştuğu durumlarda kullanılmak için geliştirilmiş bir algoritmadır. Dolayısıyla N, içerdiği örneklerin öznitelik değerlerine göre alt kümelere ayrılabilir. Algoritma N üzerinde k-ortalama (k-means) algoritmasıyla öbekleme işlemi uygular ve P için de bir merkez örnek seçer. Rocchio [5] metodunun pozitif ve negatif sınıfları temsil etmesi için prototip yaratması gibi, PN-SVM de negatif alt kümelerin ve P'nin merkezlerini kullanır. U'daki elemanlar bu merkezlerle karşılaştırılarak hangi örneklerin hangi sınıfa aktarılacağına karar verilir.

Bir örneğin (d) pozitif olarak etiketlenmesi için aşağıdaki iki koşul sağlanmalıdır.

$$S(d, C_P) > \frac{1}{|P|} \sum_{d_i \in P} S(d_i, C_P) \tag{2.4}$$

$$S(d, C_P) - \max_{j=1,\ldots,k} S(d, C_{Nj}) > \frac{1}{|P|} \sum_{d_i \in P} \left(S(d_i, C_P) - \max_{j=1,\ldots,k} S(d_i, C_{N_j}) \right) \tag{2.5}$$

k değeri N'nin alt küme sayısı olup, S benzerlik fonksiyonudur. Benzer şekilde, d'nin negatif olarak etiketlenmesi için aşağıdaki iki koşul sağlanmalıdır.

$$S(d, C_N) > \frac{1}{k} \sum_{i=0}^{k} \left(\frac{1}{|N_i|} \sum_{d_j \in N_i} S(d_j, C_{N_i}) \right) \tag{2.6}$$

$$\max_{i=1,\ldots,k} S(d, C_{Ni}) - S(d, C_P) < \frac{1}{k} \sum_{i=0}^{k} \left(\frac{1}{|N_i|} \sum_{d_j \in N_i} (S(d_j, C_{N_i}) - S(d_j, C_P)) \right) \tag{2.7}$$

Bu koşulları sağlamayan örnekler bu basamakta etiketlenmezler ve U'da etiketlenmemiş olarak bırakılırlar.

N'nin kümelerine ayrılması algoritma için önemli bir işlemdir. Negatif kümenin tümünü öbekleme yapmadan kullanmak hatalı etiketlemelere sebep olabilir. Örneğin, N'nin alt kümelerinden birine (C_{N_i}) çok benzeyen fakat diğer alt kümelere hiç benzemeyen bir e örneğini ele alalım. e her alt kümenin merkeziyle ayrı ayrı karşılaştırıldığında, negatiflerin geneline benzemiyor olsa bile negatif olarak etiketlenmesi gerekti fark edilebilir. Oysa e tüm N'nin merkeziyle (C_N) karşılaştırılıyor olsaydı, negatiflerin ciddi bir kısmına benzemeyen bu örnek P'nin merkezine (C_P) daha benzer çıkabilir ve hatalı etiketlenebilirdi. PN-SVM N'yi kümelerine ayırarak her alt kümeyi ayrı ayrı ele alır. Bu şekilde belli bir alt kümeye önemli derecede benzer olan (e gibi) örnekler negatif olarak seçilebilir.

Algoritmanın ilk basamağının ilk ve ikinci aşamasında seçilen pozitif ve negatif örnekler başta elimizde olan P ile birlikte kullanılarak bir SVM sınıflandırıcısı eğitilir. Algoritma bu sınıflandırıcıyı kullanarak U'da kalan örnekleri etiketler.

Algoritma yayınlandıktan sonra yazarları tarafından gözden geçirilerek yenilenmiş ve Positive Examples and Negative Examples Labeling Heuristic (PNLH)[7] algoritması geliştirilmiştir.

2.1.5 Positive examples and Negative examples Labeling Heuristic (PNLH)

PNLH algoritması [7] PN-SVM'in [6] geliştirilerek yenilenmiş versiyonudur. Pozitif ve negatif örnekleri etiketlenmemiş kümeden çekirdek sözlük ve öbekleme teknikleriyle PN-SVM'de olduğu gibi seçer. Bunun yanında iki algoritma arasında önemli farklılıklar vardır.

PN-SVM algoritmasında çekirdek sözlük oluşturulurken bir özniteliğin sözlüğe konulması için bu özniteliğin P'deki örneklerde bulunma sıklığı (kuvveti) önceden belirlenmiş sabit bir sınırdan yüksek olmalıdır. Fakat veri kümesi işlenmeden seçilen, yani veri kümesine özel olmayan bir sınırın bu işlem için kullanılması mantıksızdır. PNLH algoritmasında bu sınır algoritma tarafından çalışma zamanında belirlenir. Algoritma özniteliklerin kuvvetlerinin ortalamasını hesaplayarak sınır olarak kullanır.

Bir diğer farklılık, oluşturulan alt küme sayısının (k) belirlenmesinde izlenilen yöntemdedir. PN-SVM'de k değeri de önceden belirlenmiş bir sayı iken, PNLH'de algoritma tarafından çalışma zamanında hesaplanarak seçilir. Algoritma bir veri kümesi üzerinde çalışırken, kullanılan k'nın bu veri için optimum değer olması algoritmanın başarısı açısından önemlidir. Gereğinden büyük ya da küçük k değerleri hatalı etiketlemelere yol açabilir. PNLH'de k, P ve N kümelerinin eleman sayılarına bağlı olarak aşağıdaki gibi seçilir.

$$k = \sqrt{\frac{|N|}{|P|}} \tag{2.8}$$

Üçüncü farklılık, U'dan yinelemeli şekilde pozitif ve negatif örnekler seçilirken veri kümesindeki öznitelik sayısının ele alınışındadır. Bu makalede örneklerin öznitelik sayılarını azaltmanın hatalı pozitif ve negatif etiketlemeleri azalttığı öne sürülmüştür. Dolayısıyla PNLH öbekleme işlemini yapmadan önce öznitelik seçimi (feature selection) yaparak bazı öznitelikleri eler. Çekirdek sözlüğü yaratırken de algoritma en yüksek kuvvete sahip olan n adet özniteliği seçer (n çalışma zamanından önce belirlenen bir değerdir).

PNLH algoritmasında güçlü negatifler seçilirken özniteliklerin kuvvetleri de göz önüne alınır. Örneğin, a ve b örnekleri çekirdek sözlükte bulunan aynı sayıda özniteliğe sahip olsalar bile pozitif olma ihtimalleri farklı olabilir. Bu ayrımı yakalayabilmek için PNLH, hangi örneğin ilgili öznitelikerinin daha kuvvetli olduğuna bakar. Eğer a örneğinin çekirdek sözlükte bulunan öznitelikleri b'ninkilerden daha kuvvetli ise, a'nın pozitif olma ihtimali b'den daha yüksektir. Dolayısıyla da a güçlü negatif olarak seçilmeyecektir.

Son olarak, PN-SVM son basamağında bir SVM sınıflandırıcısı kullanırken, PNLH algoritması ikinci basamağında kullanılacak olan sınıflandırıcının türünden bağımsızdır. Algoritmanın amacı güçlü negatif ve pozitifleri bulmak olarak belirlenmiş, sonrasındaki klasik sınıflandırma işlemi algoritmanın içinde gösterilmemiştir.

Öznitelik sıklıklarını kullanan diğer bir algoritma da Mapping-Convergence (M-C) [3] algoritmasıdır. PN-SVM ve PNLH algoritmalarında olduğu gibi M-C algoritmasında da U'ya göre P'de daha fazla görünen öznitelikler saptanır ve

örneklerin etiketleri bu özniteliklere sahip olup olmamaları göz önüne alınarak belirlenir.

2.1.6 Mapping-Convergence (M-C) Algoritması

[3]'te Positive Example Based Learning (PEBL) çatısı geliştirilmiştir. PEBL şu iki basamaktan oluşan Mapping-Convergence (M-C) algoritmasını kullanır: Haritalama basamağı (güçlü negatiflerin seçilmesi) ve yakınsama basamağı (yinelemeli sınıflandırma).

Haritalama basamağında algoritma etiketlenmemiş kümeyi iki alt kümeye ayırır: Güçlü negatifler kümesi (N_1) ve diğer örnekler kümesi (P_1). Bu kümeler ikinci basamaktaki yinelemelerle oluşturulacak olan küme serileri N_i ve P_i'nin ilk elemanıdırlar. Algoritma güçlü negatifleri bulmak için her özniteliğin pozitif kümede görülme sıklığı (f_p) ve etiketlenmemiş kümede görülme sıklığını (f_u) hesaplar. f_p/f_u değeri belli bir sınırdan düşük olan tüm özniteliklerin negatif örnekleri temsil ettiği düşünülür ve bu öznitelikler kullanılarak bir sınıflandırıcı eğitilir. Algoritma eğittiği bu sınıflandırıcıyla etiketlenmemiş kümeyi test eder. Bu sınıflandırıcı tarafından negatif olarak etiketlenen örnekler güçlü negatif olarak seçilir ve N_1'i oluştururlar. [3] 'te öznitelik seçme sınırının sınıflandırıcının hatalı negatif seçmemesini sağlanacak şekilde belirlenmesi gerektiği belirtilmiştir.

Yakınsama basamağında algoritma yinelemeli şekilde P_i.'den negatifler seçerek çıkartır. Her yinelemenin başında algoritma N_i'deki örnekleri güçlü negatifler kümesi olan NEG'e aktarır. Bu noktada algoritmanın elinde bir pozitif kümesi (POS), NEG ve etiketlenmemiş örneklerin bulunduğu küme (P_i) vardır. M-C, POS ve NEG'i SVM ile kullanarak öznitelik uzayında pozitif ve negatif örnekler arasında bir sınır belirler. Bu sınır pozitif ve negatif örneklerle arasında maksimum aralık sağlanacak şekilde belirlenir. Algoritma sınırı P_i üzerinde uygular ve P_i'yi iki kısma ayırır: Negatif olarak etiketlenecek olan örneklerin bulunduğu kısım ve henüz etiketlenemeyecek olan örneklerin bulunduğu kısım. İlk kısım N_{i+1} kümesine taşınırken, ikinci kısım da P_{i+1} kümesine taşınır.

Yinelemeli sınıflandırma hiçbir örnek negatif olarak seçilemeyinceye kadar devam eder. Sonlandığında P_{last} pozitif örnekleri NEG ise negatif örnekleri içerir.

2.1.7 Augmented Expectation Maximization (A-EM)

A-EM'de [8] kullanılmış olan fikir bu bölümde kapsadığımız diğer algoritmalardan oldukça farklıdır. Bu algoritma, pozitif kümede bulunan örneklerin etiketlenmemiş kümede bulunan pozitiflerle karakteristik olarak özdeş olmadığı veri kümelerinde kullanılabilinecek şekilde tasarlanmıştır. Çoğu PU algoritması bu pozitiflerin özdeş olduğunu varsayarken, bu durumun tersine de rastlanabilir. A-EM algoritması ise her iki durumda da kullanılabilir.

Bu algoritmanın bir diğer farkı da, pozitif ve etiketlenmemiş kümelerin yanında üçüncü bir küme olarak bir alakasız örnekler kümesi (O) kullanmasıdır. Bu küme pozitif sınıfla alakası olmayan örneklerden oluşur. Dolayısıyla O içinde neredeyse hiç pozitif örnek bulunmamasını bekleriz. Bu küme algoritma tarafından U ile birleştirilerek U'daki negatif örnek yoğunluğu arttırılır ve böylece küme içindeki gürültü (noise) azaltılır. Örneğin, [8]'te kullanılmış olan veri şu 3 kümeyi içerir: P (belli bir ticaret sitesindeki belli bir tür cihazların internet sayfaları), U (Diğer ticaret sitelerindeki tüm ürünlerin internet sayfaları) ve O (20 haber grubu ve haber sitesinin sayfaları — 20 Newsgroup and Reuters).

Algoritma U ve O'yu birleştirerek bütünleşik bir negatif küme (N) oluşturur. Daha sonra bir Sade Bayes Sınıflandırıcısı (Naive Bayes Classifier – NBC) eğitir. A-EM, Beklenti-Maksimizasyon (Expectation-Maximization -– EM) algoritmasıyla çalışır. EM'in her yinelemesinde yeni bir sınıflandırıcı oluşur. Bu sınıflandırıcılar N kümesinden pozitif elemanların çıkarılmasında kullanılır ve yinelemeler tekrarlandıkça pozitif kümesi zenginleşmiş, negatif kümesi ise arındırılmış olur. Dolayısıyla oluşturulacak olan sınıflandırıcıların giderek daha başarılı olmaları beklenir. Fakat tabi ki bu durum ilk yinelemede P ve N 'nin ilk haliyle eğitilmiş olan sınıflandırıcının performansına bağlıdır. A-EM'in U'yu doğrudan kullanmayarak O'yu U'ya eklemesinin sebebi de budur. Alakasız kümeyle U'daki gürültü azaltılarak bu kümelerle eğitilecek olan sınıflandırıcının başarısını arttırmak amaçlanmaktadır.

EM bir seri sınıflandırıcı üretir ve algoritma da bunların arasından nihai bir sınıflandırıcı seçer. Aşağıdaki F değeri her sınıflandırıcı için hesaplanarak sınıflandırıcıları karşılaştırmada kullanılır:

$$F = \frac{2 * TP}{(TP + FP) + (TP + FN)} = \frac{2/TP}{|CP| + PD} \tag{2.9}$$

$TP + FP$ pozitif olarak etiketlenmiş olan örneklerin sayısı (CP) , $TP + FN$ ise etiketlenmemiş kümenin boyutudur (PD).

Algoritma nihai sınıflandırıcıyı seçerken sınıflandırıcıların F değerlerindeki değişimi göz önüne alır. Aşağıdaki denklem i. yinelemede F değerindeki değişimi hesaplamak için kullanılır.

$$\Delta_i = \frac{F_i}{F_{i-1}} = \frac{TP_i}{TP_{i-1}} * \frac{|CP_{i-1}| + PD}{|CP_i| + PD} \tag{2.10}$$

F değerini arttıran sınıflandırıcıların en sonuncusu nihai sınıflandırıcı olarak seçilir. Örneğin, eğer n. yinelemede eğitilmiş olan sınıflandırıcı serideki Δ değeri 1'in üzerinde olan son sınıflandırıcı ise, n. sınıflandırıcı nihai sınıflandırıcı olarak seçilir.

CP ve PD bilinen değerlerken, algoritma hangi etiketlenmemiş örneklerin aslında pozitif olduğunu bilmediği için TP_i/TP_{i-1} değeri bilinmemektedir ve tahmin edilmelidir. Algoritma bu tahmini yapabilmek için öncelikle pozitif sınıfı temsil edebilecek olan öznitelikleri (anahtar öznitelikler) seçer ve bu özniteliklerden bir K kümesi oluşturur. Daha sonra TP_i/TP_{i-1} değerinin tahmini aşağıdaki denklem ile gerçekleştirilir:

$$\sum_{t}^{|K|} N(f_t, d_i), d_i \in CP_i \Big/ \sum_{t}^{|K|} N(f_t, d_i), d_i \in CP_{i-1} \tag{2.11}$$

$\sum_{t}^{|K|} N(f_t, d_i), d_i \in CP_i$ CP_i içindeki örneklerde geçen anahtar öznitelik sayısıdır. İki CP karşılaştırıldığında, biri diğerinden daha çok anahtar özniteliğe sahip ise bu CP'de daha çok pozitif örnek bulunduğu, dolayısıyla da daha başarılı bir küme olduğu varsayılır.

O'nun hiçbir pozitif içermediğini varsayarak hareket ettiğimize göre A-EM'in neden P ve O'yu doğrudan bir sınıflandırıcı eğitmekte kullanmadığı sorulabilir. Bu fikirdeki sorun şudur ki; O içindeki alakasız örnekler U'daki aslen negatif olan

örneklere de çok uzak örnekler olabilirler. Bu durumda da P ve O kullanılarak oluşturulan sınıflandırıcı U'daki negatifleri (dolayısıyla da pozitifleri) doğru şekilde bulamayabilir.

A-EM gibi Sade Bayes Sınıflandırıcısı kullanan bir diğer algoritma da PU Learning by Generating Negative Examples (LGN) [9] algoritmasıdır.

2.1.8 PU Learning by Generating Negative Examples (LGN)

LGN [9] algoritması entropi hesaplaması yaparak yapay bir negatif örnek (A_n) yaratır ve bu örnek ile P'yi kullanarak da bir Sade Bayes Sınıflandırıcısı (NBC) eğitir.

NBC, bir örneğin pozitif ve negatif olma olasılıklarını hesaplamada kullanılır. Bu olasılıklar $Pr(d|c)$ ile gösterilir; d bir örnek, c ise bir sınıftır (+ veya -). Eğer $Pr(d|+) > Pr(d|-)$ ise d örneği pozitif olarak, aksi durumda da negatif olarak etiketlenir.

NBC'in bir örneğin $Pr(d|c)$'i değerini hesaplayabilmesi için 2 farklı tür değeri biliyor olması gerekir. Bu değerlerden ilki $Pr(f|+)$ ve $Pr(f|-)$ koşullu olasılıklarıdır (f bir özniteliği temsil eder). Her özniteliğin pozitif ya da negatif örneklerde görülme olasılığı vardır ($Pr(f|c)$). Bir örneğin sahip olduğu özniteliklerin pozitif örneklerde rastlanma olasılıkları yüksek ise, bu örneğin pozitif olma ihtimali de yüksektir. Gerekli olan ikinci tür değer $Pr(+)$ ve $Pr(-)$ ilk olasılıklarıdır (prior probability). Bu olasılıkları hesaplayabilmek için algoritmanın her iki sınıftan da örneklere ihtiyacı vardır. Algoritmanın elinde negatif örnek bulunmadığı için bu değerler tahmin edilmelidir.

Algoritma elindeki negatif örnek eksikliğini gidermek için yapay bir negatif örnek (A_n) oluşturmayı amaçlar. Bunun için öncelikle özniteliklerin $Pr(f|-)$ değerlerini tahmin eder. Eğer özniteliklerin negatif örneklerde bulunma olasılıkları bilinirse, yapay bir negatif örnek bu olasılıklara bağlı kalınarak yaratılabilir. LGN $Pr(f|-)$ değerlerinin tahmini için özniteliklerin sıklık değerlerine entropi uygular.

$$entropy(f_i) = -\sum_{x \in \{+,-\}} Pr(f_i|c) * log(Pr(f_i|c)) \tag{2.12}$$

Daha çok pozitif örneklerde görülen öznitelikleri pozitif öznitelik (f_+), negatif örneklerde görülen öznitelikleri ise negatif öznitelik (f_-) olarak adlandırırız. Bir pozitif örnek hem P hem de U kümelerinde görünebilir, çünkü her iki kümede de pozitif örnekler bulunmaktadır. Diğer yandan, negatif bir özniteliğe sadece U'da rastlanabilir. LGN özniteliklerin sıklıklarının entropisini hesaplayarak hangi özniteliklerin iki kümede de bulunduğunu hangilerinin ise bulunmadığını öğrenir. Örneğin, bir öznitelik yüksek entropiye sahipse bu öznitelik hem P hem de U'da görülmektedir ve dolayısıyla pozitif olma ihtimali yüksektir. Algoritma son olarak özniteliklerin entropilerinin ağırlıklarını hesaplar. Aşağıdaki denklem ile hesaplanan bu ağırlıklar, öznitekilerin $Pr(f|-)$ değerlerini temsil ederler.

$$weight(f_i) = \frac{entropy(f_i)}{\max_{j=1,2,...,|V|} entropy(f_j)} \tag{2.13}$$

LGN elde ettiği ağırlıklara uygun şekilde bir A_n oluşturur. Eğer bir özniteliğin yüksek bir ağırlığı varsa, A_n'ye diğer özniteliklerden daha çok kez yerleştirilecektir. Böylece negatif öznitelikler A_n içinde çok defa geçerken, diğer özniteliklere A_n içinde az defa ya da hiç yer verilmeyecektir.

A_n'nin yaratılmasından sonra, elinde artık pozitif ve negatif örnekler bulunan algoritma P ve A_n'i kullanarak $Pr(f|+)$ ve $Pr(f|-)$ değerlerini hesaplar. Böylece başta bahsettiğimiz ilk tür veriyi elde etmiş oluruz. Fakat sorun şu ki, tek bir negatif örnek kullanılarak ihtiyacımız olan ikinci tür veri, yani $Pr(+)$ ve $Pr(-)$ değerleri, hesaplanamaz. [9]'de yapılan deneylerde farklı ilk olasılık değerleri test edilmiş ve birbirine çok yaklaşık sonuçlar elde edilmiştir. Bu sebepten dolayı Pr(+) ve Pr(-) için $Pr(+) = Pr(-) = 0.5$ kullanılmıştır.

NBC'nin yaratılması için gereken olasılık değerlerini elde eden algoritma bir NBC eğitir. Son olarak da bu sınıflandırıcıyı veri kümesindeki etiketlendirilmemiş örnekleri sınıflandırmak için kullanır.

Bu bölümde entropi tekniğini kullanan 2 adet iki-basamak algoritması kapsanmıştır. Bunlardan ilki LGN, ikincisi ise Entropy-Based Semi-Supervised

Learning (SLE)'dir [10].

2.1.9 Entropy-Based Semi-Supervised Learning (SLE)

SLE [10], pozitif sınıfın alt sınıflardan oluştuğu durumlar için tasarlanmış bir algoritmadır. Entropi hesabı ile U'dan pozitif ve negatif örnekler seçer. U'dan çıkarttığı bu örnekler ve başlangıçta bilinen pozitif örnekleri kullanarak bir lojistik regresyon sınıflandırıcısı eğitir ve bu sınıflandırıcı ile kalan etiketlenmemiş örnekleri sınıflandırır.

SLE üç temel işlem içerir. Bunların ilki öznitelik çıkarmadır (feature extraction). Özniteliklerin örneklerdeki sıklıkları ölçülerek ağırlıkları hesaplanır. Sınıflardan herhangi birini temsil etmediği görülen öznitelikler gerekli görülmeyerek öznitelik vektöründen çıkartılır. İkinci işlem bir tekrar örnekleme (resampling) türü olan aşırı örnekleme (oversampling). Bir veri kümesinde iki ya da daha fazla sınıf varsa ve bu sınıflardan birinin örnek sayısı diğer(ler)inin örnek sayısından daha az ise, aşırı örnekleme bu sınıfın örnek kümesine uygulanır. Bu şekilde veri kümeleri arasındaki denge sağlanmış olur. Bizim problemimizde de pozitif kümenin boyutu etiketlenmemiş kümeyle karşılaştırıldığında oldukça küçüktür. Üçüncü işlem LGN [9] algoritmasında da kullanılmış olan entropiye dayanır. Bu algoritmada etiketlenmemiş örneklerin ilk olasılıklarının entropisi hesaplanır. Entropi sonuçları örneklerin pozitif ve negatif sınıflara ait olma olasılıklarını gösterir. Dolayısıyla bu sonuçlar SLE'de sınıflandırma yapmak için kullanılır. Etiketlenmemiş örneklerin ilk olasılıklarının entropisi aşağıdaki denklem ile hesaplanır:

$$H(d_i) = -\sum_{j=1}^{|C|} p(c_j|d_i) \lg p(c_j|d_i) \tag{2.14}$$

$p(c|d)$ d örneğinin c sınıfına ait olma ilk olasılığını (prior probability) gösterir. $|C|$ ise eğitim kümesindeki bilinen sınıf sayısıdır.

İlk basamakta, öncelikle, pozitif sınıfın her alt sınıfı için birer boş küme yaratılır. Daha sonra U içindeki örnekler ait olma olasılıkları en yüksek olan alt sınıfın kümesine aktarılır. Tüm örnekler aktarıldığında kümelerin boyutları birbirinden farklı olabilir. Bu durumda algoritma tüm kümelerin boyutunu en küçük kümeyle

aynı olacak şekilde azaltır ve dengeyi sağlar. Boyut azaltma işlemi alt kümelerden en yüksek entropiye sahip olan gerekli sayıda örnek çıkartılarak yapılır. Son olarak da alt kümeler birleştirilerek pozitif örnekler için tek bir küme oluşturulur (S_p).

İkinci basamakta U'da kalmış olan etiketlenmemiş örneklerden en yüksek entropiye sahip olanlar alınarak negatif örnek kümesi oluşturulur (S_n).

Son sınıflandırma yapılmadan önce algoritma eğitim kümeleri üzerinde aşırı örnekleme (oversampling) uygular. Bundan sonra SLE, pozitif ($P \cup S_p$) ve negatif (S_n) örnekleri kullanarak bir lojistik sınıflandırıcı eğitir. Bu sınıflandırıcıyı kullanılarak etiketlenmemiş örnekleri sınıflandırır.

2.1.10 Annotating Genes with Positive Samples (AGPS)

AGPS algoritması [11] PU öğrenme ile gen fonksiyonu tahmini yapmak için tasarlanmıştır. Algoritma, kullanacağı veri kümesini ilk adımında 3 farklı veri türünü birleştirerek hazırlar: protein-protein etkileşimleri, protein kompleks verisi (protein complex data) ve gen sentezlenme verisi (gene expression data). Fakat bu makalede biz AGPS'nin PU problemini ele alış tarzını kullanılan veri türünden bağımsız olarak inceledik.

AGPS 3 adımdan oluşur; güçlü negatiflerin seçilmesi, negatif kümenin genişletilmesi ve sınıflandırmanın yapılması. Tüm bu basamaklardan önce, algoritma P'yi iki parçaya böler: P_1 ve P_2. P_2 kümesi U'ya eklenir ve bu iki kümenin birleşimi olan U_{new} yaratılır.

Algoritma P_1'i ikinci basamakta yaratılacak olan sınıflandırıcıların eğitilmesinde kullanır. U_{new}'in içindeki P_2, yani etiketlenmemiş örneklerin arasına gizlenmiş olan pozitif olduğu bilinen örnekler ise, ikinci basamakta eğitilecek olan sınıflandırıcılarının başarısını test etmek için kullanılacaktır.

P'de bulunan tüm örneklerin hem eğitim hem de test amaçlı kullanılmış olması için, algoritma 10-tekrarlı (10-fold) çapraz-doğrulama uygular. P'yi 10 alt kümeye bölen algoritma her yinelemede bu alt kümelerden bir tanesini P_2 olarak, diğerlerinin birleşimini de P_1 olarak kullanır. Böylece algoritmanın her üç basamağı da 10 defa farklı P_1 ve P_2 kümeleriyle tekrarlanır.

İlk basamakta AGPS P_1'i kullanarak bir 1-sınıf SVM sınıflandırıcısı eğitir. Bu sınıflandırıcı kullanılarak Unew içindeki örnekler sınıflandırılır. Sınıflandırma sonucunda negatif olarak etiketlenmiş olan örnekler güçlü negatifler olarak seçilir ve (başlangıçta boş olan) N kümesine aktarılır.

İkinci basamakta algoritma U'dan yinelemeli şekilde negatifler seçerek N'yi genişletir. Bu sefer her yinelemede o ana kadar seçilmiş olan negatifleri ve P_1'i kullanarak bir 2-sınıf SVM sınıflandırıcısı eğitir. Algoritma bu sınıflandırıcıyı U'daki kalan örnekleri sınıflandırmada kullanır. Negatif olarak etiketlendirilen örnekler N'ye atılırken, pozitif olarak etiketlendirilenler U'da bırakılır. Yinelemeler U'nun boyutu $|P|$'ye ulaşana kadar devam eder.

Yapılan yinelemelerde eğitilen her bir sınıflandırıcı, eğitim kümesi ve sonuçlarıyla birlikte kaydedilerek saklanır. Algoritma 3. basamakta bu sınıflandırıcılar arasından P_2'nin örneklerinden en çoğunu doğru şekilde etiketlemiş olan sınıflandırıcıyı bulur. Seçilen bu sınıflandırıcı U_{new}'in yinelemeler başlamadan önceki ilk halini sınıflandırmak için kullanılır. Böylece tüm U sınıflandırılmış olur.

10-tekrarlı çapraz-doğrulamanın her tekrarında algoritma tüm U'yu sınıflandırır. Dolayısıyla her tekrarda bir negatif olarak etiketlendirilmiş elemanlar listesi oluşur. 10 tekrar da bittikten sonra, U'daki örnekler tekrarlarda kaç kez negatif olarak etiketlenmiş olduklarına göre sıralanır. Örneğin, 9 tekrarda negatif olarak etiketlenmiş bir örneğin gerçekten negatif olma ihtimalinin sadece 5 tekrarda negatif olarak seçilmiş bir örnekten fazla olduğu varsayılmaktadır. Negatif olarak etiketlenme sayılarına göre sıralanmış olan örneklerden ilk $|P|$ adeti en güçlü negatifler olarak seçilir ve FN adlı son bir negatif küme yaratılmak için kullanılırlar. U'dan alınarak FN'ye aktarılan bu negatifler ve P kümesi ile eğitilen son bir 2-sınıf SVM sınıflandırıcısı ile U'da kalmış olan örnekler sınıflandırılır.

2.2 Tek-Basamaklı Algoritmalar

Bu algoritma ailesine üye metotlarda negatif örnek seçilerek problem klasik pozitif negatif öğrenmeye dönüştürülmeye çalışılmaz. Pozitif ve etiketlenmemiş örneklerden elde edilen bilgi doğrudan sınıflandırma için kullanılır. Bu algoritma kümesi, negatif örneklerin etiketlenmemiş kümedeki yüksek yoğunluğu sebebiyle

U ve N arasında oluşan benzerliğe dayalı işlemler kullanır.

2.2.1 Positive Naive Bayesian (PNB)

PNB algoritması [12], pozitif ve etiketlenmemiş örneklerle kullanılabilir hale getirdiği bir Sade Bayes Sınıflandırıcısı kullanır. Bu algoritma aslen doküman sınıflandırmak için tasarlanmıştır. Fakat biz PNB'yi bu alanda ele almak yerine algoritmadaki dokümanları örnekler, kelimeleri ise öznitelikler olarak inceleyeceğiz.

PNB diğer PU algoritmaları gibi pozitif ve etiketlenmemiş kümeleri girdi olarak almasının yanında bir de pozitif sınıf olasılığı değerine ($\hat{P}(1)$) ihtiyaç duyar. $\hat{P}(1)$, pozitif örneklerin veri kümesindeki tahmini yüzdesidir. Aynı şekilde negatif örneklerin yüzdesi $\hat{P}(0) = 1 - \hat{P}(1)$ denklemiyle hesaplanabilir.

Algoritma, özniteliklerin pozitif ya da negatif olma, yani pozitif ya da negatif sınıfları temsil ediyor olma olasılıklarını hesaplar. Her özniteliğin pozitif örneklerde görülme sayısını temel alan algoritma, bu özniteliklerin pozitif sınıf için önemini, yani bir özniteliğin pozitif sınıfı temsil edebilme derecesini hesaplar. Özniteliklerin pozitif sınıf için önemi elde edildikten sonra, aynı işlem negatif sınıf için de yapılabilir.

PNB, aşağıdaki denklemi kullanarak bir örneğin (d) etiketine karar verir. Bu denklemde d'nin sınıfı, d'nin özniteliklerinin pozitif ve negatif sınıflara ait olma olasılıklarıyla belirlenir.

$$PNB(d) = \arg\max_{c \in \{0,1\}} \hat{P}(c) \prod_{i=1}^{i=n} \hat{P}r(w_i|c) \tag{2.15}$$

PNB algoritması daha sonra tekrar gözden geçirilerek yenilenmiş olup algoritmanın yenilenmiş halinin adı PNNB'dir [13].

2.2.2 PNNB Algoritması

PNNB algoritması [13] PNB'nin [12] yenilenmiş halidir. PNB'nin aksine PNNB algoritması veri kümesinde negatif örnekler varken de kullanılabilir. Bu algoritma

pozitif öznitelik olasılıklarını (özniteliklerin pozitif sınıfa ait olma olasılıkları, $\hat{Pr}(w_i|1)$ PNB algoritmasıyla aynı şekilde hesaplar. Fakat negatif öznitelik olasılıklarının ($\hat{Pr}(w_i|0)$) hesaplanmasında PNB'den farklı bir yol izler.

Bir öznitelik için $\hat{Pr}(w_i|0)$'yi hesaplamanın iki yolu vardır; doğrudan hesaplama ve dolaylı yoldan hesaplama. Dolaylı yoldan hesaplama tekniği PNB algoritmasında kullanılmıştır. Bu teknikte, negatif örnekten yoksun olan algoritma negatif öznitelik olasılıklarını da pozitif ve etiketlenmemiş örneklerden edindiği bilgilerle hesaplar.

PNNB algoritması ise doğrudan hesaplama yöntemini kullanır. Bu yaklaşımda pozitif ve etiketlenmemiş kümelerin yanı sıra, negatif kümeden de yararlanılır. Bu çalışmada ele aldığımız problemde algoritmanın elinde başlangıçta hiçbir negatif olmasa bile, yinelemeli sınıflandırmalar yaparak çalışan algoritma her yinelemede U'dan negatif örnekler seçerek N kümesindeki eleman sayısını giderek arttırır.

$$\hat{Pr}(w_i|0) = (1-\alpha)\hat{Pr}(w_i|0,P,U) + \alpha\hat{Pr}(w_i|0,N) \tag{2.16}$$

Öznitelik olasılıkları hesaplanırken (2.16), pozitif ve etiketlenmemiş kümelerle negatif kümenin hesaplamadaki ağırlıkları farklıdır. Bu ağırlık α değerine bağlıdır ve bu değer aşağıdaki gibi hesaplanır.

$$\alpha = \frac{1}{2} \times \frac{|N|}{|P|} \times \frac{\hat{Pr}(1)}{1-\hat{Pr}(1)} \tag{2.17}$$

Denklemde görüldüğü üzere α özellikle negatif kümenin eleman sayısına bağlıdır. Eğer negatif kümede çok sayıda eleman varsa, negatif kümeden elde edilen olasılık değerlerinin hesaplamadaki etkisi daha çok olacaktır. Fakat, örneğin, negatif kümede hiçbir eleman yoksa α değeri de 0 olacak, dolayısıyla negatif kümenin hesaplamaya etkisi de 0 olacaktır.

PNNB'nin yanında aynı makalede [13] PNCT adında ikinci bir algoritma daha yayınlanmıştır. PNCT algoritması öznitelik vektörünün ikiye bölünebileceği durumlarda bu iki farklı vektör ile iki farklı PNNB sınıflandırıcısı eğiterek çalışır.

2.2.3 PNCT Algoritması

PNCT algoritması [13] veri kümesinde çok az sayıda pozitif örnek olan durumlarda verimli çalışabilecek bir algoritma olarak tasarlanmıştır. PNCT bu eksikliğin üstesinden gelebilmek için pozitif örneklerin çeşitliliği yerine özniteliklerin çeşitliliğine dayanır. Blum ve Mitchell tarafından geliştirilmiş olan [19] iki sınıflandırıcıyı birlikte kullanma tekniği PNCT'de kullanılmaktadır.

Bu algoritma veri kümesindeki öznitelik vektörünün birbirinden bağımsız iki parçaya ayrılabildiği durumlarda kullanılabilir. Bu iki parçanın da kendi başlarına geçerli bir sınıflandırıcı eğitmek için yeterli olmaları gerekmektedir.

Algoritma ilk anda pozitif, etiketlenmemiş ve negatif kümelere sahiptir (Bizim problemimizde negatif küme başlangıçta boştur). Ayrıca veri kümemizdeki her örneğin birbirinden bağımsız iki öznitelik vektörü (fv_a ve fv_b) mevcuttur.

PNCT algoritması PNNB'de olduğu gibi yinelemeli şekilde çalışır. Her yinelemede iki PNNB sınıflandırıcısı üretilir: fv_a kullanılarak $PNNB_a$ ve fv_b kullanılarak da $PNNB_b$ (PNNB sınıflandırıcılarının nasıl eğitildiği önceki alt bölümde anlatılmıştır). Her yinelemede, $PNNB_a$ ve $PNNB_b$ eğitildiğinde, algoritma $|P|/\hat{P}(1)$ adet etiketlenmiş örneği seçer bu örnekleri ait oldukları sınıfın kümesine aktarır. Aktarılacak olan örneklerin seçimi örneklerin sınıflandırma sonucunda belirlenmiş olan olasılıksal sonuçlarının derecelerine göre yapılır. Dolayısıyla pozitif ya da negatif sınıflara ait olma olasılığı diğerlerine göre daha yüksek olan belli sayıda örnek her yinelemede ilgili oldukları sınıfa aktarılmaktadır. Her yinelemede pozitif ve/veya negatif küme genişleyecek, sonraki yinelemede yaratılacak olan sınıflandırıcı, bu genişlemiş olan kümeler kullanılarak eğitilecektir.

Etiketlenmemiş kümedeki tüm örnekleri etiketlenerek P veya N'ye aktardığında algoritma sonlanır.

PNB [12], PNNB ve PNCT [13] aynı algoritma ailesinde olup benzer özelliklere sahiptirler. Bu algoritma ailesinde olduğu gibi özniteliklerin sıklıklarını kullanarak çalışan bir diğer algoritma da Biased-PrTFIDF [14] algoritmasıdır.

2.2.4 Biased-PrTFIDF Algoritması

Biased-PrTFIDF algoritması [14] makalesinde yayınlanmıştır. Eğer bir pozitif örneğin etiketlenmemiş olma olasılığını p, pozitif sınıfı C_+ ve negatif sınıfı C_- ile gösterirsek aşağıdaki denklem yazılabilir:

$$Pr[P|x] = Pr[C_+|x](1-p) \tag{2.18}$$

$$Pr[U|x] = p * Pr[C_+|x] + Pr[C_-|x] \tag{2.19}$$

Bu denklemde $Pr[P|x]$ bir örnek olan x'in P kümesinde olma olasılığı ve $Pr[U|x]$ de x'in U'da olma olasılığıdır.

Denklemler (2.18) ve (2.19) kullanılarak aşağıdaki eşitlik gösterilebilir:

$$Pr[C_+|x] - Pr[C_-|x] = ((1+p)/(1-p))Pr[P|x] - Pr[U|x] \tag{2.20}$$

Bu denklemdeki $(1+p)/(1-p)$ ilgili makalede b olarak gösterilmiştir. Dolayısıyla sınıflandırma metodu şu şekilde gösterilebilir:

$$f(x) = sgn(Pr[C_+|x] - Pr[C_-|x]) \tag{2.21}$$

Denklemler (2.20) ve (2.21)'i birleştirerek sınıflandırma metodunu şu şekilde gösterebiliriz:

$$f(x) = sgn(b * Pr[P|x] - Pr[U|x]) \tag{2.22}$$

Bu sınıflandırıcıyı üretebilmek için, denklemden de görüldüğü üzere, sınıf olasılıklarına ($Pr[P|x]$ ve $Pr[U|x]$) ve b değerine ihtiyaç duyulur.

Bu algoritmada bir örneğin pozitif ve etiketlenmemiş sınıf olasılıklarını hesaplayabilmek için PrTFIDF metodu [20] kullanılmıştır. PrTFIDF, bir kümeyle bu

kümenin bir alt kümesini parametre olarak alır ve kümede bulunan örneklerin alt kümenin elemanı olma olasılıklarını hesaplar. Bu hesaplama, örneklerin sahip olduğu özniteliklerin kümelerde bulunan örneklerde geçme sıklığına göre yapılır. Bizim problemimizde algoritma PrTFIDF'yi iki kere çalıştırır: $P \cup U$'yu küme ve P'yi alt küme olarak kullanarak pozitif sınıfın olasılıklarını öğrenmek için, $P \cup U$'yu küme ve U'yu alt küme olarak kullanarak etiketlenmemiş sınıfın olasılıklarını öğrenmek için.

Algoritma ikinci kısmında b değerini bulur. p değeri bilinmediği için $b$doğrudan hesaplanamaz. Dolayısıyla da tahmin edilmesi gerekir. Üzerinde çalışılan veri kümesinin ideal p değerini bulabilmek için algoritma sınıflandırıcının en iyi sonucu almasını sağlayan değeri seçer. Bu seçim işlemi de sınıflandırıcıların başarısının ölçülebiliyor olmasını gerektirir. Etiketlenmemiş örneklerin gerçek sınıfları algoritma tarafından bilinmediği için ölçüm doğrudan yapılamaz. Bu ölçüm için sonraki bir bölümde anlatacağımız Ağırlıklı Lojistik Regresyon (Weighted Logistic Regression) algoritmasıyla birlikte geliştirilmiş olan performans ölçüsü denklemi (Equation 2.26) bu algoritmada da kullanılmıştır.

p değeri belirlendikten ve sınıf olasılıkları hesaplandıktan sonra Biased-PrTFIDF sınıflandırma metodunu oluşturur. Algoritma daha sonra bu sınıflandırıcıyı kullanarak etiketlenmemiş kümedeki tüm elemanları test ederek etiketler.

2.2.5 Spy Technique and The Expectation-Maximization (S-EM)

[4]'te yayınlanmış olan S-EM, Ajan Tekniği'ni (Spy Technique) ve Beklenti-Maksimizasyon (Expectation-Maximization) algoritmasını (Dempster et al. 1977) kullanır.

EM algoritması pozitif ve etiketlenmemiş verileri kullanarak bir Sade Bayes Sınıflandırıcısı eğitir. Bu algoritmanın ilk basamağında (Beklenti) beklenen tahmini değerlerle eksik veri tamamlanır. Beklenen tahmini değerler hali hazırda algoritmanın elinde olan verilerden yola çıkılarak belirlenir. S-EM'in yazarları EM algoritmasının bu özelliği sebebiyle bizim problemimiz için uygun olduğu öne sürmüştür. Maksimizasyon basamağında ise gerekli parametreler belirlenir.

Algoritma yinelemeli şekilde Sade Bayes Sınıflandırıcıları eğitir. Her yinelemede eğitilen sınıflandırıcı etiketlenmemiş örnekler üzerinde olasılıksal sınıflandırma yapmak için kullanılır. Bir örnek için etiketlendirme yapıldıktan sonra bu örnek ilgili kümeye aktarılır ve sonraki yinelemeye geçilir. Kümelerin yeni haliyle yeni bir sınıflandırıcı eğitilir ve bu yeni sınıflandırıcı diğer bir örneğin etiketlenmesinde kullanılır.

Yinelemeler sona erdiğinde her örnek için bir olasılıksal sonuç elde edilmiş olur. Hangi örneğin hangi sınıfa ait olduğuna karar verilmesi için bu sonuçların karşılaştırılacağı bir sınır belirlenmelidir. Bir örnek için elde edilmiş olan pozitif sınıfa ait olma olasılığı belirlenen sınır ile karşılaştırılacak, eğer olasılık bu sınırı geçiyorsa pozitif, sınırın altında kalıyorsa negatif olarak etiketlenecektir. Bu sınırın belirlenmesi işlemi Ajan Tekniğinin kullanıldığı noktadır.

Algoritma, anlatmış olduğumuz olasılıksal sonuç üretme işlemine başlamadan önce, seçtiği bir grup pozitif örneği eğitim verisinden çıkartarak aslen sadece etiketlenmemiş örnekleri içeren test kümesine aktarır. Test verisine saklanan bu gizli pozitiflere Ajan Pozitifleri denir. Artık ajan pozitiflerin de içinde bulunduğu test kümesindeki örneklerin tümünün sınıflandırılmasından sonra, algoritma ajan pozitifler için elde edilmiş olan sonuçlara bakar. Bu sonuçlar test kümesinde bulunan bir pozitifin nasıl sonuç alması gerektiğini gösterir. Bu sonuçları kullanarak olasılık sınırını belirleyen algoritma, etiketlenmemiş örneklerin sonuçlarını bu sınır ile karşılaştırır ve karşılaştırmanın sonucuna göre örnekleri etiketler.

Benzer bir yaklaşım PosOnly [17] algoritmasında uygulanmıştır. Hem S-EM hem de PosOnly pozitif kümenin bir alt kümesini sınıflandırıcı ile test eder ve bu örneklerin olasılıksal sonuçlarına göre ihtiyacı olan sınır ve katsayıları belirler.

2.2.6 PosOnly Algoritması

PosOnly [17] makalesinde yayınlanmış ve [18]'de gen etkileşim ağlarının tahmininde kullanılmıştır. Bu algoritma 2 basamaktan oluşur fakat ilk basamağında negatif örnek seçimi yapılmadığı için iki-basamaklı algoritma olarak sınıflandırılmamıştır. İlk basamakta olasılıksal sınıflandırıcı ile örneklerin etiketli olma

olasılığı hesaplanır. İkinci basamakta ise bu olasılıklar kullanılarak örneklerin pozitif olma olasılıkları bulunur.

$y \in \{0,1\}$ ve $s \in \{0,1\}$ sırasıyla bir örneğin pozitif ve etiketlenmiş olup olmadığını gösteren iki rastgele değişkendir. Rastgele bir örnek için $s = 1$ ise bu örnek etiketlenmiş, $s = 0$ ise etiketlenmemiştir. Pozitif kümedeki örnekler veri kümesindeki tek etiketlenmiş örnekler olduğu için, eğer bir örnek için $s = 1$ ise $y = 1$ olduğu da kesindir. Diğer yandan, etiketlenmemiş örneklerin ($s = 0$) gerçek etiketleri (y) pozitif ($y = 1$) ya da negatif ($y = 0$) olabilir. Bu metotta öncelikle örneklerin (x) etiketlenmiş olma olasılığı ($Pr(s = 1|x)$) hesaplanır. Bu hesaplama olasılıksal bir sınıflandırıcı kullanılarak pozitif ve etiketlenmemiş kümelerle yapılır.

Asıl hedefimiz, $Pr(y = 1|x)$, yani bir x örneğinin pozitif olma olasılığının hesaplanabilmesidir. Pozitif örneklerin rastgele şekilde etiketlenmiş ya da etiketlenmemiş olduklarını varsayarak aşağıdaki denklem yazılabilir:

$$Pr(y = 1|x) = Pr(s = 1|x)/Pr(s = 1|y = 1) \tag{2.23}$$

Bu denkleme nasıl ulaşıldığı ve denklemin detaylı açıklaması [17] ve [18]'te görülebilir. Denklemdeki $Pr(s = 1|y = 1)$, yani pozitif olan bir örneğin etiketlenmiş olma olasılığı, üzerinde çalışılacak olan veri kümesi kullanılarak hesaplanması gereken bir katsayıdır. Bu katsayıyı hesaplamak için ilgili makalelerde birden çok yöntem öne sürülerek denenmiştir. Fakat katsayının asıl değeri rastgele seçilmiş örneklerden oluşan bir V kümesi içindeki pozitif örneklerin $Pr(s = 1|x)$ değerlerinin ortalamasıdır. V içindeki pozitif örnekler alt kümesini V_P ile gösterirsek, bu eşitlik şu şekilde gösterilebilir:

$$Pr(s = 1|y = 1) = \frac{1}{n} \times \sum_{x \in V_P} Pr(s = 1|x) \tag{2.24}$$

$Pr(s = 1|y = 1)$'in belirlenmesi için [18]'de kullanılmış olan yöntem de şu şekildedir:

$$Pr(s = 1|y = 1) = \frac{\sum_{x \in V_P} Pr(s = 1|x)}{\sum_{x \in V} Pr(s = 1|x)} \tag{2.25}$$

Algoritma, örneklerin etiketlenmiş olma olasılıklarını ($Pr(s = 1|x)$) ve $Pr(s = 1|y = 1)$ katsayısını hesapladıktan sonra, örneklerin gerçek sınıf olasılıklarını denklem 2.23'e göre belirler. PosOnly son olarak, elde etmiş olduğu $p(y = 1|x)$ değerlerini bir sınırla karşılaştırarak örneklerin pozitif ya da negatif olduğuna karar verir. Bu katsayı [18]'te 0.5 olarak belirlenmiştir. Bir e örneği için $Pr(y = 1|e) > 0.5$ ise algoritma e'yu pozitif olarak, aksi durumda ise negatif olarak etiketler.

2.2.7 Bagging SVM

PU öğrenme algoritmaları temel olarak 2 kaynak kullanır: Pozitif küme ve etiketlenmemiş küme. Pozitif sınıf hakkındaki bilgiler doğrudan pozitif kümeden elde edilebilirken, negatif kümenin karakteristiği dolaylı şekilde pozitif ve etiketlenmemiş kümeler elde edilir. Etiketlenmemiş küme algoritma tarafından negatif sınıf için bir referans olarak kullanıldığı için, U'daki pozitif örnek sayısı bu kümeler ile eğitilecek bir sınıflandırıcının başarısı için belirleyicidir. Eğer etiketlenmemiş kümede çok sayıda pozitif örnek yoksa, U'yu negatif küme gibi kullanan bir algoritma için U daha az gürültülü bir veri olacaktır. Dolayısıyla da daha iyi bir sınıflandırıcı eğitilecektir. Fakat tabi ki U içinde ne kadar pozitif bulunduğunu bilmemiz mümkün değildir.

Bu soruna bir çözüm olarak Bagging SVM [16] algoritması geliştirilmiştir. Bu algoritmada tüm U'yu tek bir sınıflandırıcı eğitmek için kullanmaktansa, bu küme rastgele alt kümelere bölünmüş ve bu alt kümelerin her biri P ile birlikte kullanılarak birer sınıflandırıcı üretilmiş, daha sonra da bu sınıflandırıcılar birleştirilmiştir. Bu şekilde sınıflandırıcılarda çeşitlilik sağlanmış ve bu çeşitli sınıflandırıcılarla nihai bir sınıflandırıcı üretilmiş olur. Etiketlenmemiş alt kümeler rastgele seçildikleri için içinde hiç pozitif bulunmayan ya da tamamen pozitif örneklerden oluşan alt kümeler oluşturma ihtimali vardır. Bu çeşitliliğin sonucu olarak oluşturulacak bazı başarılı sınıflandırıcılar nihai sınıflandırıcının kalitesini attırabilirler.

Alt küme boyutunun (K) algoritmanın başarısına etkisi [16]'te incelenmiştir. Farklı K değerleriyle yapılan deneyler sonucunda K'nın nihai sınıflandırıcı üzerinde çok az bir etkisi olduğu görülmüş ve $K = NP$ 'nin güvenli bir değer olduğu açıklanmıştır.

Bagging SVM algoritmasının indüktif (inductive) ve transdüktif (transductive) öğrenme için iki versiyonu vardır. İndüktif bagging'de üretilen sınıflandırıcıların tümü tek bir nihai sınıflandırıcı oluşturmak için kullanılır. Oluşacak olan bu nihai sınıflandırıcı herhangi bir etiketlenmemiş örneği test etmek için kullanılabilir. Fakat tabi ki nihai sınıflandırıcıyı eğitimde kullandığımız olan örnekleri test etmek için kullanmak mantıksızdır. Transdüktif bagging'de de birden fazla sınıflandırıcı üretilir. Fakat bu versiyon, eğitim verisinin etiketlenmesi gereken durumlarda kullanılabilinecek şekilde tasarlanmıştır. Transdüktif bagging'de bir sınıflandırıcı eğitildiğinde bu sınıflandırıcı kendi eğitim verisindeki alt kümenin elemanları üzerinde test için kullanılmaz. Bu sınıflandırıcı diğer etiketlenmemiş örneklerin (U/bu sınıflandırıcının eğitiminde kullanılmış olan alt küme) etiketlenmesinde kullanılacak olan sınıflandırıcılar listesine eklenir.

2.2.8 Weighted Logistic Regression (W-LR)

PU öğrenme algoritmalarının eğitim basamaklarında pozitif ve etiketlenmemiş kümelerin doğrudan kullanılmasının zayıf sınıflandırıcıların üretilmesine sebep olacağını önceki bölümlerde belirtmiştik. Bunun sorunun sebebi de etiketlenmemiş küme içinde bulunan pozitif örneklerdir. Dolayısıyla negatif küme yerine etiketlenmemiş kümenin kullanıldığı bir algoritmada etiketlenmemiş küme, gürültü içeren bir negatif küme gibi görülebilir.

W-LR algoritması [15] da işte bu mantık temel alınarak tasarlanmıştır. Algoritma U'yu, içindeki pozitif örneklerin gürültü yarattığı bir negatif küme olarak ele alır. Bir doğrusal fonksiyon kullanarak bu gürültülü veri kümesi üzerinde sınıflandırma yapar.

Algoritma ilk olarak yüksek gürültüyü ele almak için örnekleri ağırlıklandırır. Daha sonra bu veri kümesiyle kullanılması gereken doğrusal fonksiyonu elde etmek için lojistik regresyon uygular. Son olarak da bu doğrusal fonksiyon kullanılarak

veri kümesindeki etiketlenmemiş örnekler sınıflandırılır.

Aşırı uyumdan (Overfitting) kaçınmak için doğrusal fonksiyonda optimum regularization parametresine ihtiyaç duyulur. Ayrıca bu makalede sadece pozitif ve etiketlenmemiş örneklerin mevcut olduğu durumlarda kullanılabilinecek bir performans ölçüm denklemi yayınlanmıştır:

$$PerformanceMeasure = \frac{r^2}{Pr[f(X) = 1]} \tag{2.26}$$

Bu denklemde f sınıflandırıcı fonksiyon, r ise pozitif örneklerin etiketlenme sonuçlarıyla hesaplanabilinecek olan hassasiyet (recall) değeridir 2.27:

$$r = Pr[f(X) = 1|Y = 1] \tag{2.27}$$

3. DENEYSEL SONUÇLAR

Bu bölümde 8 PU öğrenme algoritmasını PPI ağı tahmin etme sonuçlarına göre karşılaştırdık. PSoL, Bagging SVM, AGPS, PosOnly, Roc-SVM, SVM_{only} adını verdiğimiz algoritma ve Carter et al.'ın algoritmasını (bu algoritmadan bu bölümde Carter adıyla bahsettik) kodlayarak test edilebilir hale getirdik. Ayrıca PSoL algoritmasının alternatif bir versiyonunu da kodladık. Bu versiyonun ayrıntıları Deney Ayarları bölümünde açıklanmıştır. SVM_{only}'yi kodlamamızın sebebi etiketlenmemiş kümenin tamamını 10-tekrarlı çapraz-doğrulama ile doğrudan negatif küme gibi kullanan, yani tasarlanabilinecek en basit algoritmanın sonuçlarını çalışmamızda göstermektir. S-EM algoritması için ise yazarlarının yayınlamış olduğu programı kullandık [1].

Önceki bölümlerde açıklamış olduğumuz 19 algoritmanın 8'i tarafımızdan kodlanarak test edildi. Bu sekiz algoritma, tasarımlarında bir değişikliğe ihtiyaç duyulmadan gen sentezlenme verisiyle kullanılabilinecek olan algoritmalardır. Diğer algoritmalar ise bu çalışmada test edilmemiş, bu algoritmalardaki uyumsuzluk sorununu gidermek için onları modifiye etmek isteyebilecek kişilere kaynak olması ve yeni algoritmalar geliştirecek olan kişilere fikir verebilecek olmaları sebebiyle çalışmamızda yer almış ve önceki bölümde detaylı şekilde açıklanmıştır.

PN-SVM, PNLH, PNB, PNNB, PNCT, Biased-PrTFIDF, M-C ve LGN algoritmaları veri kümelerinde bulunan dokümanları (örnekleri) içerdikleri kelimelerin (özniteliklerin) sıklıklarını kullanarak etiketlerler. A-EM algoritması bir seri sınıflandırıcı üretir ve bu sınıflandırıcılar arasından en iyisini bulmak için sınıflandırıcıların sonuçlarını birbiriyle karşılaştırır. Bu karşılaştırma işleminde A-EM, örneklerin içindeki öznitelik sıklıklarını kullanır. Bizim problemimizdeki

[1] `http://www.cs.uic.edu/~liub/LPU/LPU-download.html`

öznitelik vektörü sayısal verilerden oluştuğu için bu tür algoritmalar problemimize uygun değildir. SLE algoritması pozitif kümenin birden çok pozitif alt kümeden oluştuğu durumlarda kullanılmak üzere tasarlanmıştır. Fakat bizim pozitif kümemiz alt kümelerden oluşmamaktadır. Son olarak, WLR algoritması, makalesinde değinilmemiş ya da açık şekilde anlatılmamış noktalara sahip olduğundan tarafımızca kodlanamamıştır. Yazarları tarafından kodlanmış çalıştırılabilir bir versiyonu da mevcut olmadığı için da test edilmemiştir.

3.1 Veri Kümeleri

Çalışmamızda Feith et al. [21] tarafından da kullanılmış olan *Escherichia coli*'nin gen sentezlenme verisi kullanılmıştır. Bu veri kümesi 4345 adet gen için 445 microarray dışavurum profili (microarray expression profiles) içerir. Etkileşimde olan proteinler listesi için ise IntAct veri tabanı [2] [25] kullanılmıştır.

Eğer iki proteinin etkileştiği biliniyorsa, bu proteinin çiftinin sentezlenme miktarını içeren örnek pozitif bir örnek olarak kabul edilir. Bir örneğin öznitelik vektörü örnekte etkileşimi temsil edilen iki proteinin sentezlenme miktarları serilerinin birleşiminden oluşur. Sentezlenme verisi her protein için 445 numuneyi tutan sayısal bir dizidir. Dolayısıyla bir örneğin öznitelik vektörünün uzunluğu 890 olur.

Etkileşim içinde olduğu bilinen protein çiftlerinin örnekleri pozitif örneklerdir. Diğer bütün olası protein çiftleri ise negatif olarak kabul edilmiştir.

Veri kümelerimizdeki pozitif küme (P) ve etiketlenmemiş küme (U) [18]'te de olduğu şekilde hazırlanmıştır: Pozitif kümenin içinde sadece pozitif olduğu bilinilen örnekler olacakken, U kümesinin içine hem pozitif hem de negatif örnekler saklanmalıdır. Böylece algoritmalar U üzerinde test edilerek etiketi gizlenmiş olan bu örneklerin gerçek etiketlerini ne kadar başarılı şekilde tahmin ettiklerine bakılacaktır. U'nun içinde saklı olan pozitif örnekler kümesine Q, yine U'da saklı olan negatif örnekler kümesine ise N adı verilir.

İlk olarak, pozitif olduğu bilinen örnekler kullanılarak istenilen boyutta bir P

[2] `http://www.ebi.ac.uk/intact/main.xhtml`

kümesi yaratılır. P'den belli sayıda bir grup örnek rastgele seçilerek alınır ve bu örneklerle Q kümesi oluşturulur. P, içinde kalan örnekler ile veri kümesinin pozitif kümesidir. N kümesi ise pozitif olmayan belli sayıda rastgele örnek seçilerek yaratılır. Q ve N 'de bulunan örneklerin etiketleri gizlenir. Dolayısıyla, P kümesi etiketi bilinen pozitif örnekler, Q etiketlenmemiş pozitif örnekler, N ise etiketlenmemiş negatif örneklerden oluşmaktadır. Son olarak Q ve N birleştirilerek U yani etiketlenmemiş küme oluşturulur; $U = Q \cup N$. Test edilecek algoritmaların amaçları U içindeki örneklerin pozitif (Q üyesi) ya da negatif (N üyesi) olduğunu bulmaktır.

Bu yöntemi kullanarak farklı küme boyutu oranlarına sahip 27 veri kümesi oluşturduk. Gerçek hayatta negatif örnekler pozitiflerden çok daha fazla olduğu için hazırladığımız veri kümelerinde negatif örneklerin sayısı pozitif örneklerin sayısının 5 katı olacak şekilde ayarlandı: $|N| = (|P| + |Q|) * 5$. Bu oran tüm 27 veri kümesi için aynıdır. Fakat r olarak adlandırdığımız oran değişkendir: $r = |P|/(|P| + |Q|) = \{\%10, \%20, ..., \%90\}$. Kümeden de görüleceği üzere 9 farklı olası r değeri vardır. Her r değeri için 3 rastgele veri kümesi üreterek toplamda 27 veri kümesi oluşturduk. Örneğin, r=%40 olduğu durum için algoritmaların test edileceği 3 veri kümesi vardır ve algoritmaların bu oran için elde edecekleri sonuç bu 3 farklı veri kümesinin sonucunun ortalaması olacaktır. Veri kümelerimizde küme boyutlarını $|P| + |Q| = 250$ ve $|N| = 1250$ olacak şekilde yarattık. Bu boyutlar güvenli testler yapmamızı sağlayacak derecede büyük, algoritmaların makul çalışma sürelerinde sonuç elde edebileceği kadar küçük oldukları için seçilmiştir.

3.2 Deneysel Ayarlar

Bu bölüm, test ettiğimiz algoritmaların kodlanmasındaki bazı detaylar ve testler sırasında kullanılan parametreler hakkında bilgiler içerir.

Bagging SVM algoritmasının iki versiyonu vardır; indüktif ve transdüktif (bkz: bölüm 2.2.7). Problemimizde, eğitim için kullandığımız etiketlenmemiş küme elemanlarını sınıflandırmak istediğimiz için bagging SVM'i transdüktif versiyonuyla kodladık. Algoritmadaki K değeri için [16]'de de tavsiye edildiği gibi pozitif küme

boyutunu ($|P|$), farklı alt kümelerle oluşturulacak sınıflandırıcı sayısı (T) için 20 kullandık. [16]'te alt küme boyutunun 30'dan büyük olduğu durumlarda $T = 10$ kullanılmıştı. Fakat biz, daha fazla sınıflandırıcı çeşitliliği arttırarak daha yansız sonuçlar üreteceği için T değeri için 20'yi seçtik.

PSoL'un önceki bölümde anlattığımız orijinal versiyonunun yanında bir de alternatif versiyonunu kodlayarak test ettik. PSoL, negatif kümeyi genişlettiği aşamada, etiketlenmemiş kümeden bir grup örneği negatif örnek olarak seçer ve bu örnekleri negatif kümeye aktarır. Bu işlem yinelemeli şekilde tekrarlanır. Orijinal versiyonda her yinelemede aktarılabilinecek örnek sayısı bir sınır değeri (T_1) ile sınırlanmış olup $T_1 = |P| * 3$'tür. Fakat daha çok örnek negatif kümeye aktarıldıkça negatif küme kontrolsüz bir şekilde büyümeye başlayabilir ve bu eğitim verisindeki kümelerin dengesizleşmesine sebep olur. Bu durum örneklerin neredeyse tümünün negatif olarak sınıflandırılmasına sebep olabilir. Bizim versiyonumuzda her yinelemede aktarılacak olan örnek sayısını sınırlamak yerine, negatif olarak seçilebilinecek toplam örnek sayısını sınırlandırdık. Bu sınırı da (T_2): $T_2 = |P| * 4$ olarak belirledik. Bu değişiklik ile nihai sınıflandırıcının dengesiz bir eğitim kümesiyle eğitilmesini engellemek amaçlandı. Algoritmanın modifiye edilmiş versiyonu $PSoL_m$, orijinal versiyonu ise $PSoL_o$ olarak adlandırıldı.

Rocchio algoritması [5] etiketlenmemiş yeni örnekleri sınıflandırabilecek olan nihai bir sınıflandırıcı üreterek sonlanır. Bizim problemimizde eğitim ve test verisi birbirinden ayrı olmadığı ve eğitim verisindeki örneklerin sınıflandırılması gerektiği için algoritmayı 10-tekrarlı çapraz-doğrulama ile kodladık. Bu durumda etiketlenmemiş küme 10'a bölündü ve her yinelemede bu alt kümelerden biri test için kullanılırken diğer dokuzu eğitim verisi olarak kullanıldı.

[1]'te Carter algoritmasının ikinci basamağı için hem sinir ağı (neural network) hem de SVM teknikleri test edilmiş, bu iki teknik için farklı sonuçlar elde edilmiştir. Bu çalışmada karşılaştırdığımız çoğu algoritmada SVM kullanıldığı için adil bir karşılaştırma yapabilmek amacıyla Carter'ın ikinci basamağında da SVM kullandık.

S-EM algoritmasını test ederken LPU programı "-s1 spy -s2 em -f demo" parametreleriyle çalıştırıldı.

Algoritmaları kodlarken SVM uygulaması olarak LIBSVM[3][22] kullandık. 2-sınıf SVM için PosOnly algoritmasında [18] kullanılmış olan çekirdek türü (kernel type) ve eğitim parametrelerinin aynıları kullanıldı. Dolayısıyla çekirdek türü için radyal temelli fonksiyon (radial basis function), $\gamma = 1.0/128$ ve maliyet parametresi (cost parameter) $C = 1000$ seçildi. AGPS algoritmasındaki 1-sınıf SVM'in çekirdek türü için algoritmanın makalesinde [11] olduğu gibi radyal temelli fonksiyon seçilirken diğer parametrelerde varsayılan (default) değerleri kullandık. Ayrıca [26]'te SVM'in biased formülasyonu yayınlanmıştır. Bagging SVM bu formülasyonu kullanıyor olsa bile, C-SVM'i kullanan diğer algoritmalarla adil bir şekilde karşılaştırma yapabilmek için Bagging de dahil SVM sınıflandırıcısı kullanan tüm algoritmalar C-SVM ile gerçekleştirilmiştir.

Algoritmaların veri kümeleri üzerindeki başarısını karşılaştırmak için kesinlik (Precision – PR), hassasiyet (Recall – RC), F-ölçümü (F-measure – FM) ve Matthews correlation coefficient (MCC) ölçüm yöntemleri kullanılmıştır. Her r değeri (P-Q boyut oranı) için 3 farklı veri kümesi hazırladığımızdan, tabloya her üç veri kümesinin sonuçlarının ortalaması yazıldı. Ayrıca algoritmaların birebir karşılaştırmak için FM ve MCC değerleri üzerinde tek-yönlü Wilcoxon işaretlimertebe testi (one-sided Wilcoxon signed-rank test) uygulandı.

$$PR = \frac{TP}{TP + FP} \tag{3.1}$$

$$RC = \frac{TP}{TP + FN} \tag{3.2}$$

$$FM = \frac{2 * PR * RC}{PR + RC} \tag{3.3}$$

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.4}$$

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

3.3 Sonuçlar

Kesinlik, hassasiyet, F-ölçümü ve matthews correlation coefficient (MCC) değerleri sırasıyla tablo 3.1, 3.2, 3.3 ve 3.5'te, F-ölçümü ve MCC'nin tek-yönlü Wilcoxon işaretli-mertebe testlerinin p-değerleri ise tablo 3.4 ve 3.6'da görülebilir.

r	SVM_{only}	$Carter$	$PSoL_o$	$PSoL_m$	$Roc–SVM$	$AGPS$	$S-EM$	$PosOnly$	$Bagging$
10%	0.000 ± 0.000	0.155 ± 0.002	0.107 ± 0.105	0.267 ± 0.208	0.581 ± 0.569	0.251 ± 0.039	0.258 ± 0.047	0.144 ± 0.009	$\mathbf{0.583} \pm 0.589$
20%	0.333 ± 0.653	0.150 ± 0.008	0.409 ± 0.421	0.484 ± 0.395	$\mathbf{0.689} \pm 0.330$	0.203 ± 0.050	0.217 ± 0.008	0.141 ± 0.005	0.512 ± 0.512
30%	0.883 ± 0.142	0.141 ± 0.005	$\mathbf{0.932} \pm 0.076$	0.691 ± 0.161	0.854 ± 0.050	0.248 ± 0.023	0.204 ± 0.006	0.126 ± 0.008	0.916 ± 0.027
40%	$\mathbf{0.889} \pm 0.218$	0.127 ± 0.011	0.805 ± 0.041	0.376 ± 0.049	0.704 ± 0.037	0.197 ± 0.030	0.156 ± 0.010	0.125 ± 0.019	0.704 ± 0.090
50%	$\mathbf{0.939} \pm 0.119$	0.110 ± 0.006	0.830 ± 0.137	0.482 ± 0.129	0.708 ± 0.026	0.136 ± 0.018	0.145 ± 0.003	0.112 ± 0.004	0.767 ± 0.049
60%	$\mathbf{0.605} \pm 0.107$	0.101 ± 0.011	0.575 ± 0.173	0.300 ± 0.106	0.456 ± 0.098	0.130 ± 0.037	0.105 ± 0.008	0.169 ± 0.080	0.494 ± 0.104
70%	0.548 ± 0.065	0.070 ± 0.006	$\mathbf{0.668} \pm 0.176$	0.202 ± 0.045	0.449 ± 0.105	0.115 ± 0.012	0.095 ± 0.018	0.116 ± 0.044	0.513 ± 0.160
80%	$\mathbf{0.510} \pm 0.261$	0.051 ± 0.002	0.504 ± 0.120	0.181 ± 0.071	0.315 ± 0.050	0.079 ± 0.010	0.060 ± 0.001	0.089 ± 0.014	0.484 ± 0.269
90%	0.196 ± 0.060	0.024 ± 0.002	$\mathbf{0.267} \pm 0.088$	0.089 ± 0.033	0.147 ± 0.058	0.033 ± 0.008	0.029 ± 0.013	0.038 ± 0.009	0.248 ± 0.104
avg.	0.545	0.103	0.566	0.341	0.545	0.154	0.141	0.118	**0.580**

Tablo 3.1: **PU Öğrenme algoritmalarının elde ettikleri kesinlik değerleriyle karşılaştırılması.** Satırlar r oranlarını ($r = |P|/(|P| + |Q|)$), sütunlar ise algoritmaları temsil etmektedir. Tablodaki her değer bir algoritmanın belli bir r oranında elde ettiği ortalama kesinlik değeridir.

r	SVM_{only}	$Carter$	$PSoL_o$	$PSoL_m$	$Roc\text{–}SVM$	$AGPS$	$S-EM$	$PosOnly$	$Bagging$
10%	0.000 ± 0.000	$\mathbf{0.993} \pm 0.006$	0.498 ± 0.549	0.514 ± 0.521	0.030 ± 0.033	0.576 ± 0.239	0.604 ± 0.087	0.948 ± 0.080	0.014 ± 0.015
20%	0.005 ± 0.010	0.948 ± 0.028	0.172 ± 0.307	0.210 ± 0.272	0.083 ± 0.057	0.497 ± 0.149	0.592 ± 0.062	$\mathbf{0.963} \pm 0.040$	0.059 ± 0.058
30%	0.027 ± 0.024	0.931 ± 0.053	0.059 ± 0.030	0.170 ± 0.047	0.147 ± 0.049	0.714 ± 0.123	0.606 ± 0.028	$\mathbf{0.941} \pm 0.083$	0.118 ± 0.030
40%	0.022 ± 0.012	$\mathbf{0.931} \pm 0.016$	0.071 ± 0.016	0.180 ± 0.026	0.147 ± 0.033	0.549 ± 0.016	0.544 ± 0.049	0.820 ± 0.170	0.105 ± 0.028
50%	0.067 ± 0.019	$\mathbf{0.939} \pm 0.014$	0.117 ± 0.032	0.205 ± 0.014	0.208 ± 0.031	0.688 ± 0.196	0.621 ± 0.014	0.850 ± 0.130	0.128 ± 0.027
60%	0.087 ± 0.026	$\mathbf{0.953} \pm 0.043$	0.083 ± 0.046	0.167 ± 0.023	0.200 ± 0.074	0.557 ± 0.101	0.520 ± 0.030	0.553 ± 0.173	0.113 ± 0.046
70%	0.125 ± 0.023	$\mathbf{0.916} \pm 0.046$	0.142 ± 0.009	0.204 ± 0.031	0.267 ± 0.026	0.671 ± 0.071	0.604 ± 0.077	0.648 ± 0.075	0.173 ± 0.054
80%	0.187 ± 0.065	$\mathbf{0.940} \pm 0.023$	0.180 ± 0.060	0.240 ± 0.078	0.247 ± 0.069	0.707 ± 0.026	0.580 ± 0.000	0.660 ± 0.185	0.193 ± 0.052
90%	0.160 ± 0.045	$\mathbf{0.920} \pm 0.091$	0.147 ± 0.069	0.187 ± 0.069	0.240 ± 0.091	0.627 ± 0.114	0.547 ± 0.232	0.683 ± 0.118	0.160 ± 0.045
avg.	0.075	**0.941**	0.163	0.231	0.174	0.621	0.580	0.785	0.118

Tablo 3.2: **PU Öğrenme algoritmalarının elde ettikleri hassasiyet değerleriyle karşılaştırılması.** Satırlar r oranlarını ($r = |P|/(|P| + |Q|)$), sütunlar ise algoritmaları temsil etmektedir. Tablodaki her değer bir algoritmanın belli bir r oranında elde ettiği ortalama hassasiyet değeridir.

r	SVM_{only}	*Carter*	$PSoL_o$	$PSoL_m$	*Roc–SVM*	*AGPS*	$S-EM$	*PosOnly*	*Bagging*
10%	0.000 ± 0.000	0.268 ± 0.003	0.173 ± 0.170	0.203 ± 0.112	0.056 ± 0.061	0.338 ± 0.022	$\mathbf{0.361} \pm 0.060$	0.250 ± 0.016	0.026 ± 0.029
20%	0.010 ± 0.019	0.259 ± 0.011	0.093 ± 0.127	0.160 ± 0.087	0.139 ± 0.090	0.287 ± 0.074	$\mathbf{0.317} \pm 0.018$	0.246 ± 0.006	0.105 ± 0.104
30%	0.051 ± 0.046	0.245 ± 0.005	0.110 ± 0.053	0.266 ± 0.041	0.248 ± 0.070	$\mathbf{0.366} \pm 0.014$	0.305 ± 0.010	0.221 ± 0.009	0.209 ± 0.046
40%	0.043 ± 0.022	0.223 ± 0.018	0.130 ± 0.026	0.241 ± 0.012	0.242 ± 0.044	$\mathbf{0.289} \pm 0.032$	0.243 ± 0.015	0.216 ± 0.023	0.181 ± 0.043
50%	0.124 ± 0.033	0.197 ± 0.009	0.203 ± 0.045	0.285 ± 0.029	$\mathbf{0.321} \pm 0.040$	0.225 ± 0.020	0.235 ± 0.004	0.198 ± 0.004	0.218 ± 0.039
60%	0.151 ± 0.042	0.182 ± 0.017	0.144 ± 0.075	0.212 ± 0.043	$\mathbf{0.276} \pm 0.088$	0.210 ± 0.056	0.175 ± 0.012	0.244 ± 0.065	0.183 ± 0.068
70%	0.203 ± 0.034	0.130 ± 0.011	0.233 ± 0.006	0.201 ± 0.028	$\mathbf{0.333} \pm 0.045$	0.197 ± 0.020	0.164 ± 0.030	0.194 ± 0.057	0.259 ± 0.081
80%	0.272 ± 0.104	0.096 ± 0.003	0.263 ± 0.074	0.206 ± 0.076	$\mathbf{0.275} \pm 0.057$	0.142 ± 0.016	0.109 ± 0.001	0.155 ± 0.014	0.273 ± 0.092
90%	0.175 ± 0.048	0.047 ± 0.004	0.189 ± 0.080	0.120 ± 0.045	0.183 ± 0.071	0.062 ± 0.015	0.055 ± 0.024	0.072 ± 0.017	$\mathbf{0.193} \pm 0.063$
avg.	0.114	0.183	0.171	0.210	0.230	**0.235**	0.218	0.200	0.183

Tablo 3.3: **PU Öğrenme algoritmalarının elde ettikleri F-ölçümü değerleriyle karşılaştırılması.** Satırlar r oranlarını ($r = |P|/(|P| + |Q|)$), sütunlar ise algoritmaları temsil etmektedir. Tablodaki her değer bir algoritmanın belli bir r oranında elde ettiği ortalama F-ölçümü değeridir.

	SVM_{only}	*Carter*	$PSoL_o$	$PSoL_m$	*Roc–SVM*	*AGPS*	$S-EM$	*PosOnly*	*Bagging*
SVM_{only}		0.91	0.99	0.97	0.99	0.93	0.91	0.93	1
Carter	0.11		0.5	0.89	0.87	0.99	0.99	0.84	0.5
$PSoL_o$	0.01	0.54		0.91	0.97	0.91	0.85	0.75	0.95
$PSol_m$	0.04	0.12	0.10		0.84	0.71	0.5	0.24	0.37
Roc–SVM	0.004	0.15	0.04	0.18		0.5	0.41	0.15	0.005
AGPS	0.08	0.004	0.10	0.32	0.54		0.06	0.08	0.17
S–EM	0.10	0.004	0.17	0.54	0.63	0.95		0.21	0.32
PosOnly	0.08	0.18	0.28	0.78	0.87	0.93	0.82		0.41
Bagging	0.002	0.54	0.06	0.67	0.99	0.85	0.71	0.63	

Tablo 3.4: Algoritma çiftlerinin F-ölçümü değerleri üzerinde uygulanan tek-yönlü Wilcoxon işaretli-mertebe testinin p-değerleri.

r	SVM_{only}	$Carter$	$PSoL_o$	$PSoL_m$	$Roc–SVM$	$AGPS$	$S-EM$	$PosOnly$	$Bagging$
10%	0.000 ± 0.000	0.041 ± 0.028	0.019 ± 0.023	0.056 ± 0.059	0.117 ± 0.118	0.196 ± 0.042	$\mathbf{0.217} \pm 0.088$	-0.101 ± 0.110	0.078 ± 0.076
20%	0.034 ± 0.078	0.090 ± 0.038	0.053 ± 0.075	0.129 ± 0.147	$\mathbf{0.184} \pm 0.100$	0.135 ± 0.103	0.178 ± 0.028	0.030 ± 0.032	0.154 ± 0.154
30%	0.134 ± 0.064	0.115 ± 0.009	0.212 ± 0.053	0.298 ± 0.015	$\mathbf{0.324} \pm 0.047$	0.281 ± 0.020	0.187 ± 0.017	0.042 ± 0.043	0.305 ± 0.034
40%	0.130 ± 0.050	0.118 ± 0.050	0.220 ± 0.018	0.200 ± 0.010	$\mathbf{0.290} \pm 0.032$	0.187 ± 0.044	0.123 ± 0.025	0.077 ± 0.052	0.244 ± 0.043
50%	0.236 ± 0.035	0.124 ± 0.018	0.289 ± 0.022	0.270 ± 0.056	$\mathbf{0.355} \pm 0.035$	0.144 ± 0.033	0.150 ± 0.006	0.125 ± 0.017	0.290 ± 0.026
60%	0.209 ± 0.049	0.154 ± 0.027	0.197 ± 0.090	0.176 ± 0.062	$\mathbf{0.266} \pm 0.086$	0.142 ± 0.085	0.090 ± 0.021	0.184 ± 0.071	0.210 ± 0.071
70%	0.243 ± 0.038	0.097 ± 0.037	0.290 ± 0.037	0.153 ± 0.032	$\mathbf{0.316} \pm 0.058$	0.178 ± 0.034	0.124 ± 0.053	0.174 ± 0.060	0.276 ± 0.097
80%	$\mathbf{0.291} \pm 0.134$	0.101 ± 0.009	0.284 ± 0.074	0.171 ± 0.081	0.253 ± 0.055	0.151 ± 0.021	0.086 ± 0.002	0.164 ± 0.010	0.286 ± 0.127
90%	0.162 ± 0.050	0.057 ± 0.026	$\mathbf{0.186} \pm 0.080$	0.103 ± 0.049	0.167 ± 0.075	0.071 ± 0.042	0.051 ± 0.069	0.111 ± 0.020	0.185 ± 0.070
avg.	0.160	0.100	0.194	0.173	**0.252**	0.165	0.134	0.089	0.225

Tablo 3.5: **PU Öğrenme algoritmalarının elde ettikleri Matthews correlation coefficient değerleriyle karşılaştırılması.** Satırlar r oranlarını ($r = |P|/(|P| + |Q|)$), sütunlar ise algoritmaları temsil etmektedir. Tablodaki her değer bir algoritmanın belli bir r oranında elde ettiği ortalama MCC değeridir.

	SVM_{only}	$Carter$	$PSoL_o$	$PSoL_m$	$Roc–SVM$	$AGPS$	$S-EM$	$PosOnly$	$Bagging$
SVM_{only}		0.05	0.99	0.67	0.99	0.59	0.32	0.002	0.99
$Carter$	0.96		0.99	1	1	0.99	0.92	0.45	1
$PSoL_o$	0.01	0.01		0.24	0.99	0.24	0.15	0.002	0.97
$PSol_m$	0.37	0.002	0.78		1	0.21	0.10	0.06	1
$Roc–SVM$	0.006	0.002	0.01	0.002		0.01	0.006	0.002	0.02
$AGPS$	0.45	0.004	0.78	0.82	0.99		0.05	0.10	0.97
$S–EM$	0.72	0.10	0.87	0.91	0.99	0.96		0.32	0.97
$PosOnly$	1	0.58	1	0.95	1	0.91	0.71		1
$Bagging$	0.006	0.002	0.03	0.002	0.98	0.04	0.04	0.002	

Tablo 3.6: Algoritma çiftlerinin MCC değerleri üzerinde uygulanan tek-yönlü Wilcoxon işaretli-mertebe testinin p-değerleri.

SVM_{only}, beklediğimiz gibi, genel olarak diğer algoritmalardan daha kötü bir performans gösterdi. Algoritma etiketlenmemiş kümedeki neredeyse tüm örnekleri negatif olarak etiketledi. Bu durum her ne kadar hatalı pozitif sayısını çok az sayıda tutsa da (yüksek kesinlik) çok sayıda hatalı negatife yol açar (düşük hassasiyet). SVM_{only} 3 diğer algoritmadan daha yüksek MMC sonuçlarına sahip olsa bile en düşük FM sonucuna sahiptir ve dolayısıyla da bizim problemimiz için uygun olmayan bir algoritma olarak görülmektedir.

Test sonuçlarında Carter'ın yüksek hassasiyet değerlerine sahip olduğu görülmüştür. Fakat hatalı pozitif sayısı yüksek olduğu için çok düşük kesinliğe sahiptir. Bu algoritma U'nun büyük kısmını pozitif olarak etiketlemiştir. Problemimizdeki etiketlenmemiş kümenin büyük kısmı negatif olduğu için bu algoritmanın hatalı pozitif sayısı diğer algoritmaların ortalamasından daha yüksektir.

$PSoL_o$ beklediğimizden daha kötü F-ölçümü sonuçları aldı. Algoritmalar arasındaki en düşük ikinci F-ölçümü sonucunu alan $PSoL_o$, U'nun büyük kısmını negatif olarak etiketledi. Alternatif versiyonumuzda ($PSoL_m$) hatalı negatifleri azaltıp doğru negatif sayısını arttırarak hassasiyet ve F-ölçümü sonuçlarının artmasını sağladık. Versiyonların sonuçları arasındaki fark göze alınırsa, algoritmadaki yinelemeler sırasında negatif kümenin kontrolsüz ve aşırı büyümesi $PSoL_o$'nun başarısız oluşunun sebeplerinden biri olabileceği düşünülebilir.

PosOnly, testlerimizden yetersiz sonuçlar alan bir diğer algoritmadır. Ortalama bir F-ölçümü sonucu olsa bile, algoritmalar arasındaki en düşük MCC'ye sahiptir. Yaptığımız deneylerde [17]'de bahsedilen 3 denklemi de denememize rağmen, optimum $Pr(s = 1|y = 1)$ katsayısının algoritma tarafından doğru şekilde hesaplanamadığını gözlemledik. Örneklerin olasılıksal sonuçlarıyla karşılaştırılmakta kullanılan bu katsayının hatalı bir değeri olmasının hatalı sınıflandırmalara sebep olacağı da açıktır.

Bagging SVM ve Carter benzer F-ölçümü sonuçları elde ettiler. İki algoritma da etiketlenmemiş kümenin rastgele alt kümelerini negatif küme olarak kullandığı için benzer sonuçlar elde etmeleri de mantıklıdır. Fakat diğer yandan, ürettiği sınıflandırıcılarda çok daha fazla çeşitlilik sağlayan Bagging algoritmasının Carter'dan bir miktar daha başarılı sonuçlar elde etmesi beklenebilir. Beklendiği üzere de Bagging, Carter'ınkilerden çok daha yüksek MCC sonuçları

elde etmiştir. RocSVM'den sonra en yüksek ikinci MCC sonucunu alan Bagging, etiketlenmemiş kümenin büyük kısmını negatif olarak etiketleyen bir diğer algoritmadır. Negatif örneklerini U'dan rastgele seçmesi, dolayısıyla da aslında pozitif olan bazı örnekleri negatif kümesinde kullanması bunun sebeplerinden biri olabilir.

Algoritmalar arasındaki en yüksek F-ölçümü değeri ortalamasına AGPS'nin sahip olmasına rağmen bu algoritmanın MCC sonuçları diğer algoritmaların ortalamasına yakındır. Diğer çoğu algoritma örneklerin büyük kısmını negatif ya da pozitif olarak etiketlemek gibi dengesiz (yanlı) sınıflandırmalar yaparken, AGPS daha dengeli tahminler yapmıştır. Dolayısıyla da AGPS'nin F-ölçümü sonuçları diğer algoritmalardan daha yüksek olmuştur.

Düşük r değerleri olan veri kümelerinde en iyi sonuçları S-EM ile elde ettik. S-EM'in bunu sağlayan özelliği etiketlenmemiş kümeye yerleştirdiği ajan pozitifleri olabilir. Bu örnekler S-EM'e pozitif örneklerin etiketlenmemiş kümede alması gereken sonuçları gösterdiğinden, pozitif kümenin eleman sayısı az olduğu için pozitif sınıfın karakteristiklerini öğrenmenin zor olduğu bu durumlarda S-EM'i diğer algoritmaların bir adım önüne taşıyor olabilir. P'den örnek alarak spy pozitif kullanma işlemi eğitim kümesindeki pozitif örnek sayısını düşürmekte olup riskli bir yaklaşımdır. Fakat bu yaklaşımın en azından düşük r değerine sahip veri kümelerinde işe yaradığı görülmektedir.

Orta ve yüksek r değerli veri kümeleri için ise RocSVM hem en yüksek F-ölçümü hem de en yüksek MCC sonuçlarını elde etti. Deneylerimiz sırasında bu algoritmanın prototip tekniğinin güçlü negatif seçmede yüksek bir başarı oranına (%94) ulaştığını gördük. Yinelemeli sınıflandırmasına bu başarılı negatif kümeyle başlayan RocSVM'in yüksek sonuçlar alması tabi ki beklenen bir durumdur. Sonuç olarak prototip tekniği ve kosinüs benzerliğinin bizim bu problemimiz ve veri türümüz için uygun olduğu söylenebilir.

Tek-yönlü Wilcoxon işaretli-mertebe testi (one sided Wilcoxon signed rank test) p-değerleri Tablo 3.4 ve 3.6'da verilmiştir. Bu değerler algoritmaların sonuçlarının çiftler halinde karşılaştırılmalarının sonucudur. F-ölçümünün p-değerlerine baktığımızda RocSVM ve AGPS'nin diğer algoritmalardan üstün

olduğunu görüyoruz. MCC'nin p-değerlerinde ise RocSVM'in diğer tüm algoritmalardan daha iyi olduğu görünüyor. Dolayısıyla mertebe testi sonuçlarına bakarak iki-basamaklı algoritmalarda AGPS ve RocSVM'in bizim problemimiz için en iyi algoritmalar olduğunu söyleyebiliriz. Bir-basamaklı algoritmalarda ise hiçbir algoritma diğerlerini önemli derecede geride bırakamadığı için bir sonuca ulaşmak zordur.

4. SONUÇ

Bu çalışmada PU öğrenme algoritmaları adı verilen ve negatif örneklerin bulunmadığı veri kümelerinde kullanılabilen sınıflandırma algoritmalarını özetleyerek açıkladık. PU öğrenme algoritmalarını iki ana sınıf altında grupladık: İki-basamaklı algoritmalar bir takım etiketlenmemiş örnekleri negatif olarak seçerek problemi klasik ikili sınıflandırma problemine dönüştürürken, Bir-basamaklı algoritmalar pozitif ve etiketlenmemiş örnek kümelerini doğrudan kullanırlar. Ele aldığımız tüm algoritmaların nasıl çalıştıklarını ve birbirleriyle benzer/farklı yanlarını açıkladık.

Son olarak da PU öğrenmenin sekiz önemli algoritmasını PPI ağı veri kümeleri üzerinde test ederek karşılaştırdık. Algoritmaların ürettiğimiz veri kümelerinde elde ettiği sonuçları gösterdik.

Bildiğimiz kadarıyla bu PU öğrenme algoritmalarının PPI ağlarının tahmin edilmesinde ilk kez kullanımıdır. Genelde dokümanların sınıflandırılmasında kullanılan bu algoritmaların biyolojik veri kümesiyle ağ tahmini yapılmasında ne kadar başarılı olduğunu gösterdik. Algoritmaların bazıları (özellikle RocSVM ve AGPS) ümit vadeden sonuçlar elde etmiştir ve protein etkileşim tahminlerinde kullanıma daha da adapte edilebilir görünmektedirler.

Deneylerimizde öznitelik vektörü için microarray verisi kullandık. Proteinlerin karakteristiklerini temsil edebilecek olan diğer veri türleri de bulunmaktadır: GO (Gene Ontology) terimleri, aminoasit dizileri, 3 boyutlu yapı bilgisi, vb. Bu çalışmamızı, bu gibi farklı veri türlerini de öznitelik vektörüne ekleyerek kullanılabilecek şekilde genişletmeyi amaçlıyoruz. Farklı veri türlerinin bir arada kullanımı biyoenformatikte sıkça kullanılan bir yöntem olup bu çalışmadaki algoritmaların daha iyi performans göstermelerini sağlayabilir.

Kodlayarak hazır hale getirdiğimiz bu algoritmaları; farklı benzerlik ölçümleri kullanmak, kullanıcı tarafından belirlenen katsayıların çalışma zamanında hesaplanmasını sağlamak gibi değişikliklerle problemimize daha uygun hale getirmek ileri çalışmalarımızda gerçekleştirmek istediğimiz bir diğer hedefimizdir. Böylece bu algoritmaları protein-protein etkileşim ağlarının tahmininde kullanılmaya daha uygun hale getirebilir, daha iyi sonuçlar alabiliriz.

Kaynakça

[1] Richard J. Carter, Inna Dubchak and Stephen R. Holbrook (2001) A computational approach to identify genes for functional RNAs in genomic sequences. Oxford Univ Press, Nucleic Acids Research, Vol.29, No.19, pp. 3928–3938

[2] Chunlin Wang, Chris Ding, Richard F. Meraz, Stephen R. Holbrook (2006) PSoL: a positive sample only learning algorithm for finding non-coding RNA genes. Bioinformatics, Vol. 22 no. 21 pages 2590–2596

[3] Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang (2004) PEBL: Web Page Classification without Negative Examples, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, NO. 1, January, Pages 70–81

[4] Bing Liu, Wee Sun Lee, Philip S. Yu, Xiaoli Li (2002) Partially Supervised Classification of Text Documents, Proceedings of the Nineteenth International Conference on Machine Learning (ICML).

[5] Xiaoli Li, Bing Liu (2003) Learning to classify texts using positive and unlabeled data, IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence (2003), pp. 587–592

[6] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Hongjun Lu, Philip S. Yu (2005) Text Classification without Labeled Negative Documents, ICDE '05 Proceedings of the 21st International Conference on Data Engineering, Pages 594–605

[7] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Hongjun Lu, Philip S. Yu (2006) Text Classification without Negative Examples Revisit. IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 1, January, Pages 6–20

[8] Xiaoli Li, Bing Liu (2004) Dealing with Different Distributions in Learning from Positive and Unlabeled Web Data, WWW Alt. '04 Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, Pages 440–441

[9] Xiao-Li Li, Bing Liu, See-Kiong Ng (2007) Learning to Identify Unexpected Instances in the Test Set. Proceedings of the IJCAI'07 Proceedings of the 20th international joint conference on Artificial intelligence, Pages 2802–2807

[10] Xiaoling Wang, Zhen Xu, Chaofeng Sha, Martin Ester, and Aoying Zhou (2010) Semi-supervised Learning from only Positive and Unlabeled Data using Entropy. WAIM'10 Proceedings of the 11th international conference on Web-age information management, Pages 668–679

[11] Xing-Ming Zhao, Yong Wang, Luonan Chen, Kazuyuki Aihara (2008) Gene function prediction using labeled and unlabeled data. BMC Bioinformatics. Jan 28, Volume 9:57

[12] François Denis, Rémi Gilleron, Marc Tommasi (2002) Text classification from positive and unlabeled examples. IPMU'02, 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems.

[13] François Denis, Anne Laurent, Rémi Gilleron, Marc Tommasi (2003) Text Classification and Co-training from Positive and Unlabeled Examples. ICML Workshop on the Continuum from Labeled to Unlabeled Data, Pages 80–87

[14] Dell Zhang, Wee Sun Lee (2005) A Simple Probabilistic Approach to Learning from Positive and Unlabeled Examples. In Proceedings of the 5th Annual UK Workshop on Computational Intelligence (UKCI), London, UK, Sep

[15] Wee Sun Lee, Bing Liu (2003) Learning with positive and unlabeled examples using weighted logistic regression. Proceedings of the Twentieth International Conference on Machine Learning (ICML).

[16] Fantine Mordelet, Jean-Philippe Vert (2010) A bagging SVM to learn from positive and unlabeled examples.

[17] Charles Elkan, Keith Noto (2008) Learning classifiers from only positive and unlabeled data. KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA: ACM 2008:213-220.

[18] Luigi Cerulo, Charles Elkan, Michele Ceccarelli (2010) Learning gene regulatory networks from only positive and unlabeled data, BMC Bioinformatics, May, Volume 11, Number 1, p.228

[19] Avrim Blum, Tom Mitchell (1998). Combining labeled and unlabeled data with co-training. Proc. 11th Annu. Conf. on Comput. Learning Theory (pp. 92-100). ACM Press, New York, NY.

[20] Joachims, T. (1997). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. Proceedings of the 14th International Conference on Machine Learning (ICML). pp. 143-151. Nashville, Tennessee, USA.

[21] Faith et al. (2008) Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. Nucleic Acids Research 36(Database issue):D866D870, Jan 2008.

[22] Chih-Chung Chang, Chih-Jen Lin (2011) LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27

[23] Thorsten Joachims (1999) Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, Pages 169–184

[24] Chris Ding, Hanchuan Peng (2005) Minimum redundancy feature selection from microarray gene expression data. CSB '03 Proceedings of the IEEE Computer Society Conference on Bioinformatics, Page 523

[25] Samuel Kerrien, Bruno Aranda, Lionel Breuza, Alan Bridge, Fiona Broackes-Carter, Carol Chen, Margaret Duesbury, Marine Dumousseau, Marc Feuermann, Ursula Hinz, Christine Jandrasits, Rafael C. Jimenez, Jyoti Khadake, Usha Mahadevan, Patrick Masson, Ivo Pedruzzi, Eric Pfeiffenberger, Pablo Porras, Arathi Raghunath, Bernd Roechert, Sandra Orchard1 and

Henning Hermjakob (2011) The IntAct molecular interaction database in 2012, Nucleic Acids Research doi: 10.1093/nar/gkr1088, Volume40, IssueD1Pp. D841-D846

[26] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, Philip S. Yu (2003) Building Text Classifiers Using Positive and Unlabeled Examples. ICDM '03 Proceedings of the Third IEEE International Conference on Data Mining, Page 179

[27] X. Zhu (2005) Semi-supervised learning literature survey. Computer Sciences Technical Report 1530, University of Wisconsin-Madison.

[28] Bangzuo Zhang, Wanli Zuo (2008) Learning from Positive and Unlabeled Examples: A Survey. Proceedings of the 2008 International Symposiums on Information Processing.

[29] Kristin P. Bennett, and Ayhan Demiriz (1998). Semi-supervised Support Vector Machines. In Advances in Neural Information Processing Systems (NIPS),12, 368-374.

[30] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), pages 200-209, Bled, Slovenia, 1999. Morgan Kaufmann.

[31] Thorsten Joachims. Transductive Support Vector Machines. In Semi-supervised Learning by Editors Olivier Chapelle,Bernhard Scholkopf, Alexander Zien, MIT press, pages 105-117, 2006.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: KILIÇ, Cumhur
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 24.04.1987 Ordu
Medeni hali	: Bekar
Telefon	: 05557140977
e-mail	: cumhurkilic01@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Y. Lisans	TOBB Ekonomi ve Teknoloji Üniversitesi	2012
Lisans	Hacettepe Üniversitesi	2010

İş Deneyimi

Yıl	Yer	Görev
2010-2012	TOBB Ekonomi ve Teknoloji Üniversitesi	Ders Asistanı

Yabancı Dil

İngilizce (Çok iyi)

Yayınlar

Cumhur Kılıç ve Mehmet Tan, Positive unlabeled learning for deriving protein interaction networks, Network Modeling And Analysis In Health Informatics And Bioinformatics, 2012, DOI: 10.1007/s13721-012-0012-8