Chapter 13 학습기의 결합

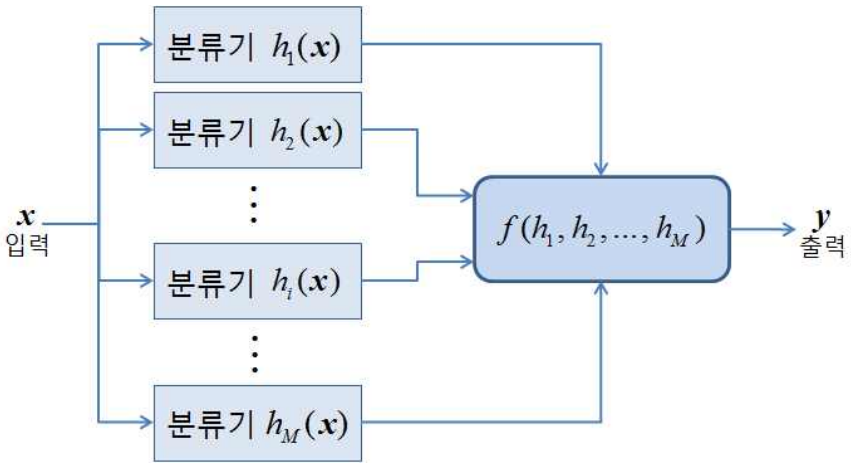
[학습목표]
지금까지 패턴인식을 위한 여러 가지 학습 방법에 대하여 알아보았다. 이 장에서 살펴볼 앙상블 학습이란, 여러 개의 인식기를 각각 학습하여 이들을 결합함으로써 보다 좋은 성능을 가진 시스템을 만드는 방법을 말한다. 결합에 사용되는 인식기의 종류와 결합 방법에 따라 다양한 방법이 존재하지만, 여기서는 가장 기본적인 결합 방법인 보팅법, 배깅에 의한 방법, 그리고 가장 널리 사용되는 부스팅 방법에 대하여 알아보겠다. 또한 결합 방법을 중심으로 하는 캐스케이딩 방법과 전문가 혼합법에 대해서도 간단히 살펴보겠다.

13. 학습기의 결합

13.1. 학습기 결합의 개요

13.1.1 학습기 결합의 필요성

지금까지 우리는 다양한 패턴인식기에 대하여 살펴보았다. 책의 전반부에서 살펴본 기본적인 선형 분류기에서부터 후반부에서는 보다 심화된 형태의 분류기들에 대해서도 알아보았다. 특히 후반부에서 배운 신경망이나 SVM과 같은 인식기들은 최근 들어 다양해지고 복잡해진 패턴인식 문제들을 풀기위하여 개발된 정교화되고 대규모화된 인식기들로, 어려운 응용문제에서도 성공적인 인식 결과를 보여주고 있다.
그러나 이러한 복잡하고 대규모화된 인식기들은 학습의 대상이 되는 파라미터의 수가 많아 사실상 학습하는데 많은 시간이 걸릴 뿐 아니라 학습을 통해 최적해를 찾기도 힘들다는 문제를 안고 있다. 일반적으로 학습에 걸리는 시간은 파라미터의 수에 비례하여 급격히 증가할 뿐 아니라 1장에서 설명한 과다학습의 문제도 동반된다. 과다학습의 문제는 12장에서도 살펴본 바와 같이 지나치게 복잡한 모델을 학습기로 사용하는 경우 불완전하고 불충분한 수의 학습 데이터에 과다적합되므로 인해 일반화 오차가 증가하는 현상을 의미한다. 12장에서는 이 문제를 해결하기 위하여 선형 분류기를 도입하되, 선형성의 제약을 해결하기 위하여 입력 공간의 차원을 높이는 접근 방법을 택하였다. 이 장에서는 또 다른 해결책으로, 선형 분류기와 같은 간단한 인식기로 학습을 수행하되, 복수개의 선형 분류기의 학습 결과를 결합함으로써 결과적으로 보다 좋은 성능을 가진 인식기를 만들고자 하는 방법이다.



[그림 13-1] 앙상블 학습의 개념

[그림 13-1]에 분류기의 결합에 대한 개념을 그림으로 나타내었다. 주어진 문제에 대하여 각각 서로 다른 접근 방법에 의해 M개의 서로 다른 가설(인식 결과)들을 만들어 내고, 최종적으로는 이들의 결과를 적절히 결합하여 인식 결과를 도출한다. 이러한 접근 방법은 지금까지 여러 분야에서 서로 다양한 형태로 시도되어 그에 대한 호칭도 다양하다. 이 책에서는

먼저 다양한 접근 방법들을 몇 가지 기준으로 나누어 분류해 보고, 최근 공학적 응용문제에 성공적으로 적용되고 있으면서 이론적 연구도 활발히 이루어지고 있는 대표적인 방법들에 대하여 알아보겠다.
학습기 결합에 의해 어떤 문제를 해결하고자 할 때에는 다음 두 가지 사항에 대해 먼저 결정해 주어야 할 것이다.

► 어떤 학습기들을 사용할 것인가?
비교적 간단하면서 서로 차별성이 존재하는 분류기를 선택함으로써 결합을 통한 효과를 높일 수 있어야 한다.
► 어떻게 결합할 것인가?
학습이 완료된 학습기들로부터 얻어지는 인식 결과를 각 학습기의 특성을 고려하여 효과적으로 결합하여야 한다.

이 두 가지 사항에 대하여서는 매우 다양한 선택이 존재한다. 먼저 서로 다른 복수개의 분류기를 학습하는 방법에 대하여 생각해보면 다음과 같은 여러 수준의 차별화 방법이 존재할 수 있다.

► 학습 알고리즘의 차별화: 베이즈 분류기와 K-NN 분류기를 결합하거나, 신경망 분류기와 SVM 분류기를 결합하는 등과 같이 접근 방법 자체를 다른 것으로 선택한다.
► 모델 선택과 관련된 파라미터의 차별화: 같은 K-NN 분류 알고리즘을 적용하되 K값을 달리하면서 서로 다른 분류기를 복수개 만들어 사용하거나, 다층 퍼셉트론의 경우 은닉층의 뉴런 수를 달리하면서 여러 가지 모델을 만들어 사용한다.
► 학습 데이터의 차별화: 같은 모델을 사용하되, 학습에 사용되는 데이터 집합을 달리하여 복수개의 분류기를 만든다. 예를 들어, 같은 신경망 모델을 사용하되, 전체 학습 데이터를 적절히 조합하여 서로 다른 학습 데이터 집합들을 만들어 이들을 학습에 이용한다.

또한 학습된 분류기를 결합하는 방법도 다음 두 가지 경우로 크게 나눌 수 있으며, 각 경우에 대해서 다양한 접근법이 존재한다.

► 병렬적 결합 방법: 각각의 분류기로부터 얻어진 결과를 한 번에 모두 함께 고려하여 하나의 최종 결과를 얻는다.
► 순차적 결합 방법: 각 분류기의 결과를 단계별로 나누어, 앞 단계에 배치된 분류기의 결과가 뒤에 배치된 분류기의 학습과 분류에 영향을 미친다.

이러한 여러 가지 변형을 고려하면 하나의 문제를 해결하기 위해 시도해 볼 수 있는 방법은 매우 다양하다. 그러나 중요한 것은 단순히 직관에 의존하거나 주먹구구식 접근법으로 학습기와 결합 방법을 선택하는 것은 좋은 결과를 기대하기 힘들며, 설사 해당 문제에 대하여 좋은 결과를 보인다고 하더라도 그 일반적 성능이나 학술적 가치에 있어서 좋은 평가를 내리기 힘들 것이다. 사실 특정 응용문제에 대하여 공학적으로 최적의 해법을 찾는다는 측면에서는 주어진 응용문제의 특성을 심도 있게 고려하여 그에 특화된 학습기와 결합 방법을 선택하는 것이 가장 적절한 방법일 것이며, 이에 대한 사례들도 많이 발표되고 있다. 그러나

이러한 문제 의존적 접근법은 이론적 연구의 대상으로 거론되기는 힘들다. 이 책에서는 특정 문제에 의존하지 않고 일반적인 경우에 대하여 학습기들을 선택하고 결합함으로써 어떠한 효과를 얻을 수 있는지에 대한 연구를 중심으로 알아본다.
이러한 취지에서 이 책에서는 학습기의 선택 방법에 있어서는 학습 데이터를 달리함으로써 서로 다른 분류기들을 얻는 방법을 중심으로 대표적인 방법들을 소개할 것이다. 이러한 방법들을 총칭하여 <앙상블 학습(Ensemble Learning)>이라고 부른다. 학습 데이터를 달리하는 방법은 학습기에 변화를 주는 가장 간단한 방법으로 볼 수 있으나, 이를 통해서 하나의 학습기를 사용함으로써 얻을 수 없는 성능 향상의 효과를 충분히 얻어낼 수 있을 뿐 아니라, 그 효과에 대한 이론적 연구도 충실히 수행되어 있다. 또한 결합 방법을 다양화함으로써 단순히 학습 데이터만 달리한 분류기들만으로도 충분히 다양한 특성을 가진 시스템을 만들어 낼 수 있다.

13.1.2 앙상블 학습의 개요

구체적인 방법들을 소개하기에 앞서, 앞으로 논의의 대상이 되는 학습기의 결합 문제를 수학적으로 정의하자. 먼저 학습에 사용되는 간단한 학습기들을 $h(\boldsymbol{x},\boldsymbol{\theta})$로 나타내면, 앞서 언급한 바와 같이 각 학습기들은 그 기본 모델은 같으나 데이터를 달리하여 학습을 수행하게 되므로, 결국 얻어지는 파라미터의 값들이 달라진다. i번째 학습기가 학습을 통해 얻게 된 파라미터를 $\boldsymbol{\theta_i}$라고 하면, 입력 $\boldsymbol{x}$가 주어졌을 때, 각 학습기의 출력은 다음과 같이 쓸 수 있다.

$$h_i(\boldsymbol{x}) = h(\boldsymbol{x},\boldsymbol{\theta_i}) \qquad \text{[식 13-1]}$$

모두 M개의 학습기를 결합하여 최종 인식 결과를 얻는다고 하고, 사용된 결합 방법을 함수 f로 나타내면, 학습기의 결합에 의해 얻어지는 최종 인식 결과는 다음과 같이 쓸 수 있다([그림 13-1] 참조).

$$f_M(\boldsymbol{x}) = f(h_1(\boldsymbol{x}),h_2(\boldsymbol{x}),...,h_M(\boldsymbol{x})) \qquad \text{[식 13-2]}$$

서로 다른 M개의 분류기 $h_i(\boldsymbol{x})$를 얻기 위하여 학습 데이터 집합들을 생성하는 구체적인 방법과, 얻어진 분류기들의 결합 방법을 정의하는 함수 f의 구체적인 형태에 대해서는 다음 절부터 대표적인 방법들을 소개할 것이다. 여기서는 다음 절에서 소개될 방법들을 학습 데이터를 생성하는 방법을 중심으로 크게 세 가지로 나누어 간략히 설명하겠다.

► 필터링(filtering)에 의한 방법: 하나의 분류기를 학습할 때마다 새로운 데이터를 생성하되, 이를 바로 학습에 적용하기에 앞서 이전에 학습이 완료된 분류기들을 이용하여 필터링함으로써, 미리 학습된 분류기에 의해 제대로 분류되지 못하는 데이터들을 학습하도록 한다. 초기에 개발된 부스팅 방법(13.3절)에서 이러한 전략을 사용하였으며, 학습의 특성을 결합 방법에도 적용하면 캐스케이딩(13.4절)에 의한 학습기 결합이 이에 적합하다.
► 리샘플링(resampling)에 의한 방법: 학습 데이터를 매번 새로 생성하는 대신, 주어진 전체 학습 데이터로부터 일부 집합을 추출하여 각 분류기를 학습한다. 가장 단순한 샘플링 기법을 사용하는 방법으로 배깅 방법(13.2절)이 있으며, 이후 분류가 어려운 샘플들이 보다 자

주 선택될 수 있도록 하는 샘플링 기법을 적용한 방법으로 MadaBoost 방법이 개발되었다.
► 가중치조정(reweighting)에 의한 방법: 모든 분류기에 대해 같은 학습 데이터를 사용하되, 각 데이터에 가중치를 주어 학습에 대한 영향도를 달리한다. 가장 대표적인 앙상블 학습 방법인 AdaBoost 방법(13.2절)이 이에 해당한다.

이상에서 설명한 학습기의 결합 방법들은, 각각의 개별적인 학습기들은 그 분류 성능이 그다지 좋지 못한 간단한 학습기(weak learner)들로, 이들을 결합함으로써 그 성능을 증폭시켜서 결과적으로 복잡한 학습기로부터 기대할 수 있는 인식 결과를 얻고자 하는 목적을 가지고 있다. 위에서 소개한 대표적인 결합 방법들이 "Boost"라는 단어를 사용하여 명명된 것도 이에 기인한 것이다. 여기서 논의를 발전시키기 위해서는 결합을 통해 개선하고자 하는 분류기의 "성능" 혹은 "오차"에 대한 정의가 필요하다. 앞에서 언급한 배깅이나 AdaBoost와 같은 방법들은 하나의 분류기를 사용함에 비해 오차가 어떻게 개선될 수 있는지에 대한 분석 연구가 존재하며, 다음 절에서 이에 대해 간단히 알아볼 것이다.
여기서는 그 준비 단계로 1장에서 소개한 오차의 정의에 대해 간략히 되돌아보겠다. 주어진 입력 $\boldsymbol{x}$에 대해 우리가 학습을 통하여 얻은 분류기의 출력을 $f(\boldsymbol{x})$라 하고, 원하는 목표 출력을 t라고 하면, 이 데이터에 대한 오차는 두 값 $f(\boldsymbol{x})$와 t에 의해 정의되는 함수 $e(\boldsymbol{x}) = e(f(\boldsymbol{x}),t)$ 로 나타낼 수 있다. 또한 목표 출력을 제공하는 실제 시스템을 함수 $f^*(\boldsymbol{x})$로 표현하면 $e(\boldsymbol{x}) = e(f(\boldsymbol{x}),f^*(\boldsymbol{x}))$ 로도 나타낼 수 있다. 가장 대표적인 오차함수로는 [식 13-3]과 같이 정의되는 제곱오차를 생각할 수 있으나, 이 외에도 여러 가지 다양한 형태의 오차함수를 정의할 수 있다. 13.3절에서는 그 예 중 하나인 지수오차를 소개할 것이다.

$$e^2(\boldsymbol{x},\boldsymbol{\theta}) = \{f(\boldsymbol{x};\boldsymbol{\theta}) - t\}^2 \qquad \text{[식 13-3]}$$

하나의 데이터에 대해 오차함수가 정의되면, 가능한 모든 입력 공간에 대해 정의되는 일반화 오차는 [식 13-3]을 확률분포 $p(\boldsymbol{x},t)$와에 대한 적분, 즉 기대치로 정의되어 [식 13-4]와 같이 쓸 수 있다.

$$E_{gen} = \mathrm{E}_{\boldsymbol{x}}[e(f(\boldsymbol{x}),t)] = \int e(f(\boldsymbol{x}),f^*(\boldsymbol{x}))p(\boldsymbol{x})d\boldsymbol{x} \qquad \text{[식 13-4]}$$

다음 절에서는 간단한 학습기의 결합에 의하여 오차가 어떻게 개선될 수 있는지에 대하여 알아보겠다.

13.2 배깅과 보팅

13.2.1 배깅에 의한 학습과 보팅에 의한 결합

이 절에서 소개하는 것은 가장 기본적인 앙상블 학습법으로, <배깅(bagging)>은 학습기의 선택과 관련된 방법이며, <보팅(voting)>은 학습기의 결합과 관련된 방법이다. 주어진 제한된 크기의 데이터 집합을 이용하여 여러 개의 분류기를 학습시키는 가장 간단한 방법은 리샘플링 기법을 사용하는 것이다. 리샘플링 기법은 원래 제한된 데이터 집합을 이용하여 시

스템을 학습시키면서 동시에 평가하기 위하여 제안된 것으로, <부트스트랩(Bootstrap)>이라고 알려져 있다. 배깅은 부트스트랩 방법을 학습기 결합에 적용한 것으로, Bootstrap aggregating의 약자로 이름이 지어졌다. 다음은 배깅에 의해 M개의 서로 다른 분류기를 학습하는 과정을 단계별로 나타낸 것이다.

[배깅에 의한 분류기 학습]

① N개의 데이터로 이루어진 학습 데이터 집합 X을 준비하고, 학습을 위한 분류기 모델을 정의한다. 각 분류기의 학습에 사용될 데이터 집합의 크기 $\widetilde{N}$을 정한다. ($\widetilde{N} \leq N$)

② i번째 분류기를 학습하기 위해 분류기 모델의 파라미터를 초기화 하고, 학습 데이터 집합 X로부터 $\widetilde{N}$개의 데이터를 랜덤하게 선출하여 데이터 집합 X_i를 만든다. 이때 같은 데이터가 중복해서 선출되는 것도 허락한다.(복원 추출)

③ 데이터 집합 X_i를 이용하여 각 분류기를 학습하여 i번째 분류기를 위한 판별함수 $h_i(\boldsymbol{x})$를 얻는다.

④ ②~③과정을 M번 반복하여 서로 다른 M개의 분류기를 생성하고, 이들을 결합하여 최종 판별함수 $f(h_1, h_2, ..., h_M)$ 을 찾는다.

배깅법에 의해 분류기를 생성할 때 고려할 사항으로는 데이터 집합의 크기 N'와, 학습에 사용되는 분류기의 모델이다. 일반적으로 주어진 전체 학습 데이터 집합 X의 크기 N이 충분히 크지 않은 경우에는 N'과 N은 같은 값으로 둔다. 이렇게 함에도 불구하고, 복원 추출에 의해 데이터 집합이 생성되므로, 매 단계마다 생성되는 데이터의 집합은 달라질 것이다. 그러나 분류기 간의 변화를 보다 높이기 위해서는 이와 함께 분류기의 선택에 있어서도, 찾아지는 판별함수가 데이터 집합의 변화에 민감한 분류기를 선택하는 것이 바람직 할 것이다. 다층 퍼셉트론이나 최근접이웃 분류기 등을 예로 들 수 있겠다.
배깅법에 의해 분류기가 학습되면, 이를 이용하여 최종 결과를 얻기 위한 결합함수 f를 정의해 주어야 한다. 가장 간단한 방법으로는 M개의 분류기 결과를 모두 동일한 정도로 반영하여 평균한 결과를 얻는 방법을 생각해 볼 수 있다. 이는 각 분류기가 일종의 위원회(committee) 역할을 하여, 최종 결과에 각각 한 표씩 투표를 하는 것으로 볼 수 있어서, 이러한 방법을 <보팅법(voting)> 혹은 <커미티머신(committee machine)>이라고 한다. 보팅법에 의한 결합함수를 수식으로 나타내면 [식 13-5]와 같다.

$$f(\boldsymbol{x}) = f(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), ..., h_M(\boldsymbol{x})) = \frac{1}{M}\sum_{i=1}^{M} h_i(\boldsymbol{x}) \qquad \text{[식 13-5]}$$

엄밀히 말하면 [식 13-5]는 각 함수 $h_i(\boldsymbol{x})$와 $f(\boldsymbol{x})$가 연속된 실수값을 내야하는 함수 근사 문제에 적합한 것으로, 분류 문제에서는 [식 13-5]의 결과를 이용하여 최종 분류 결과(클래

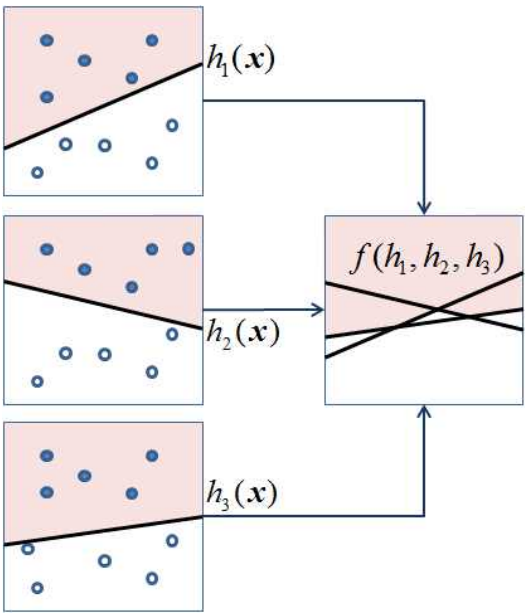
스 라벨)를 결정해 주는 처리를 수행해야 할 것이다.

13.2.2 배깅과 보팅의 효과

이상과 같은 단순한 랜덤 리샘플링에 의한 분류기의 학습과 단순 보팅법에 의한 결합을 통하여 얻을 수 있는 성능 향상의 효과에 대하여 생각해 보겠다. 이진 분류 문제의 경우에, 각 분류기가 간단한 선형 판별함수를 이용하여 다음과 같은 처리를 수행한다고 하자.

$$h_i(\boldsymbol{x}) = sign(\boldsymbol{w}_i^T \boldsymbol{x}) \qquad \text{[식 13-6]}$$

배깅 방법에 의해 세 개의 선형분류기를 [그림 13-2]와 같이 얻었다면, [식 13-5]와 같은 보팅 방법에 의해 분류를 수행하는 경우 두 클래스에 대한 결정영역은 [그림 13-2]의 오른쪽에 보이는 것과 같아져서, 단순한 하나의 선형 분류기를 사용하는 것보다 다양한 형태의 결정경계를 얻을 수 있음을 알 수 있다. 이와 같이 간단한 학습기(weak learner)들을 복수개 결합하여 얻어지는 함수는 개별적인 학습에 사용된 분류기 모델의 표현 능력을 능가하는 새로운 모델을 표현할 수 있게 됨을 알 수 있다.



[그림 13-2] 선형 분류기의 결합으로 얻어지는 결정경계

이와 함께 13.1절에서 잠깐 언급한 오차에 대해서도 생각해 보자. 각각의 분류기 $h_i(\boldsymbol{x})$에 대한 일반화 오차를 제곱오차를 이용하여 [식 13-7]과 같이 정의한다.

$$\mathrm{E}_{\boldsymbol{x}}\left[e_i^2(\boldsymbol{x})\right] = \mathrm{E}_{\boldsymbol{x}}\left[\{h_i(\boldsymbol{x}) - f^*(\boldsymbol{x})\}^2\right] \qquad \text{[식 13-7]}$$

보팅에 의한 결합으로 얻어진 분류기의 판별함수가 [식 13-5]와 같이 주어진다고 하면, 이에

대한 일반화 오차는 다음과 같이 쓸 수 있다.

$$\mathrm{E}_{\boldsymbol{x}}\left[\{f(\boldsymbol{x})-f^*(\boldsymbol{x})\}^2\right]=E_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{i=1}^{M}h_i(\boldsymbol{x})-f^*(\boldsymbol{x})\right\}^2\right] \quad \text{[식 13-8]}$$

$$=\mathrm{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{i=1}^{M}e_i(\boldsymbol{x})\right\}^2\right]$$

$$=\frac{1}{M^2}\sum_{i=1}^{M}\mathrm{E}_{\boldsymbol{x}}\left[e_i^2(\boldsymbol{x})\right]+\frac{1}{M^2}\sum_{i,j}\mathrm{E}_{\boldsymbol{x}}\left[e_i(\boldsymbol{x})e_j(\boldsymbol{x})\right]$$

이 때 만약 각 분류기에 대하여 [식 13-9]가 성립한다고 하면 [식 13-8]을 간단히 하여 [식 13-10]과 같은 관계식을 얻을 수 있다.

$$\mathrm{E}_{\boldsymbol{x}}\left[e_i(\boldsymbol{x})e_j(\boldsymbol{x})\right]=0 \quad \text{[식 13-9]}$$

$$\mathrm{E}_{\boldsymbol{x}}\left[\{f(\boldsymbol{x})-f^*(\boldsymbol{x})\}^2\right]=\frac{1}{M}\left\{\frac{1}{M}\sum_{i=1}^{M}E_{\boldsymbol{x}}\left[e_i^2(\boldsymbol{x})\right]\right\} \quad \text{[식 13-10]}$$

[식 13-10]은 결합된 분류기의 일반화 오차가 각각의 개별적인 분류기들의 평균적인 일반화 오차의 $1/M$배로 감소함을 의미한다. 즉, 학습기의 결합을 통해 단순히 하나의 학습기를 사용함에 비해 월등히 좋은 일반화 성능을 기대할 수 있으며, 이는 결합하는 학습기의 수 M에 비례한다.

그러나 이것은 어디까지나 [식 13-9]의 조건을 만족하는 경우, 즉 각각의 분류기들이 서로 독립적인 경우에 적용된다. 만약 각 분류기들이 양의 상관관계를 가지면 일반화 오차는 증가할 것이며, 반대로 각 분류기들이 음의 상관관계를 가지도록 학습된다면 일반화 오차는 더욱 감소할 것이다. 단순한 배깅에 의해 분류기를 학습하는 경우에는 각 분류기들이 서로 양의 상관관계를 가지고 있다고 볼 수 있으므로, 일반적으로는 [식 13-10]에서 보이는 바와 같은 일반화 오차의 감소가 클 것으로 기대하기는 어렵다. 그러나 만약 분류기를 학습함에 있어서 보다 정교한 방법을 적용한다면 [식 13-10]의 경우보다 더 큰 성능 향상을 기대할 수 도 있을 것이다. 다음 절에서 소개하는 부스팅 방법은 먼저 학습된 분류기의 분류 결과를 활용하여 다음 학습할 분류기의 데이터를 적절히 조정하는 가중치조정 방법을 사용함으로써 오차를 줄이는 보다 효율적인 결합 방법을 제공한다.

13.3 부스팅

13.3.1 AdaBoost 알고리즘

앞서 살펴본 바와 같이, 학습기들을 결합함으로써 성능을 향상시키기 위해서는 결합되는 각각의 학습기들이 상호보완적인 역할을 할 수 있도록 해야 한다. 그런데 앞 절에서 소개한 배깅은 단순히 데이터의 확률적 리샘플링에 의존하여 학습기 간의 차이가 주어지므로, 보다

전략적인 학습 방법이 필요할 것이다. 부스팅은 간단한 학습기들(Weak learners)이 상호보완적 역할을 할 수 있도록 단계적으로 학습을 수행하여 결합함으로써 그 성능을 증폭시키는 것을 기본 목적으로 하는 방법으로, 그 이름도 이러한 목적에서 유래하였다. 부스팅이 배깅과 다른 가장 큰 차이점은 분류기들을 순차적으로 학습하도록 하여, 먼저 학습된 분류기의 결과가 다음 분류기의 학습에 정보를 제공하여, 이전의 분류기의 결점을 보완하는 방향으로 학습이 이루어지도록 한다는 것이다.

가장 처음 제안된 부스팅 방법에서는, 각 학습기별로 서로 다른 데이터 집합을 사용한다. 먼저 첫 번째 데이터 집합 X_1을 이용하여 첫 번째 분류기 h_1을 학습한다. 이어서 두 번째 분류기를 학습할 때에는, 새로운 데이터 집합 X_2를 생성하여 먼저 첫 번째 분류기에 입력으로 주어 분류를 수행하게 한 후, 잘못된 분류 결과를 내는 데이터들을 찾는다. 이 오분류된 데이터와 함께 바르게 분류된 데이터들도 일부 추출하는데, 그 수가 오분류된 데이터와 같은 정도로만 추출한다. 이렇게 만들어진 학습 집합을 이용하여 두 번째 분류기를 학습한다. 이어서 세 번째 분류기의 경우에는, 새로운 데이터 집합 X_3를 먼저 분류기 h_1과 h_2에 입력으로 주어 두 분류기의 결과가 일치하지 않는 데이터들만 모아서 세 번째 분류기를 위한 학습 데이터 집합을 만든다. 이렇게 세 개의 분류기의 학습이 완료되면, 새로운 데이터가 주어졌을 때 인식을 위해서는 먼저 h_1과 h_2를 이용하여 분류를 수행한 후, 그 결과가 일치하면 그것을 최종 결과로 하고, 만약 일치하지 않으면 h_3의 결과를 최종 결과로 선택한다. 이러한 방법을 필터링에 의한 부스팅(Boosting by filtering)이라고 하며, 이 방법을 제안한 Schapire는 이렇게 구성된 시스템이 간단한 분류기의 분류 성능을 증폭시킬 수 있음을 보였다.

그러나 이 방법은 각 분류기를 학습할 때마다 새로운 데이터 집합을 생성해야 하며, 두 번째와 세 번째에 생성되는 데이터는 학습에 그 일부만 활용될 수 있다. 따라서 제대로 학습이 수행되기 위해 필요한 학습 데이터의 규모가 매우 크다는 문제점을 가지고 있다. 이를 해결하기 위하여 AdaBoost에서는 같은 데이터 집합을 반복해서 사용하되, 학습할 때마다 각 데이터에 대한 중요도(가중치)를 적절히 조정하여 학습에 변화를 준다. 데이터에 대한 중요도가 적응적으로 변한다는 특성을 따서 Adaptive Boost의 줄임말인 AdaBoost라는 이름이 붙여졌다.

AdaBoost는 배깅과 달리 분류기의 학습 방법 뿐 아니라 결합 방법도 함께 고려하고 있다는 점도 알아두어야 할 것이다. 각 분류기가 학습될 때 사용되는 데이터에 대한 가중치들은 분류기 학습의 목적이 되는 오분류율을 정의할 때도 사용된다. 이 오분류율은 다시 각 분류기가 전체 결합에 있어서 어느 정도 영향을 주게 될 것인지를 결정하는 결합계수를 정의하는 데 사용된다. 이렇게 얻어진 결합계수를 활용하여 AdaBoost에서는 가중치를 가진 보팅 방법으로 학습기들을 결합한다. AdaBoost의 전체적인 처리 과정을 단계별로 정리하면 다음과 같다.

[AdaBoost에 의한 분류기 학습과 병합]

① N개의 입출력 쌍으로 이루어진 학습 데이터 집합 $X=\{(\boldsymbol{x}_j,t_j)\}_{j=1\cdots N}$ 을 준비하고, 각 데이터에 대한 가중치 $w_j(j=1,...,N)$를 $w_j^{(1)}=1/N$로 초기화 한다. 이때 목표 출력 값은 $t_j\in\{-1,1\}$ $(j=1,...,N)$을 만족한다.

② $i=1,...,M$에 대해 다음과 같은 과정을 수행한다.

②-1. 각 학습 데이터에 가중치가 적용된 오분류율을 다음과 같이 정의한다.

$$\epsilon_i=\sum_{j=1}^{N}w_j^{(i)}I(h_i(\boldsymbol{x}_j)\neq t_j)$$

이때 $I(h_i(\boldsymbol{x}_j)\neq t_j)$는 $h_i(\boldsymbol{x}_i)\neq t_j$이 만족할 때만 1의 값을 가지고 그렇지 않은 경우는 0의 값을 가지는 함수이다.

②-2. 오분류율 ϵ_i를 최소화하는 분류기 $h_i(\boldsymbol{x})$를 학습을 통해 얻는다.

②-3. 오분류율 ϵ_i를 이용하여 각 분류기 $h_i(\boldsymbol{x})$의 중요도 값 α_i를 계산한다.

$$\alpha_i=\frac{1}{2}ln\left\{\frac{1-\epsilon_i}{\epsilon_i}\right\}$$

②-4. 각 데이터에 대한 가중치를 다음 식에 의해 수정한다.

$$w_j^{(i+1)}=\frac{w_j^{(i)}\exp\{-\alpha_i t_j h_i(\boldsymbol{x}_j)\}}{Z_i}$$

이때 Z_i는 가중치 값들의 합이 1이 되도록 정규화하기 위한 값으로, 다음과 같이 계산된다.

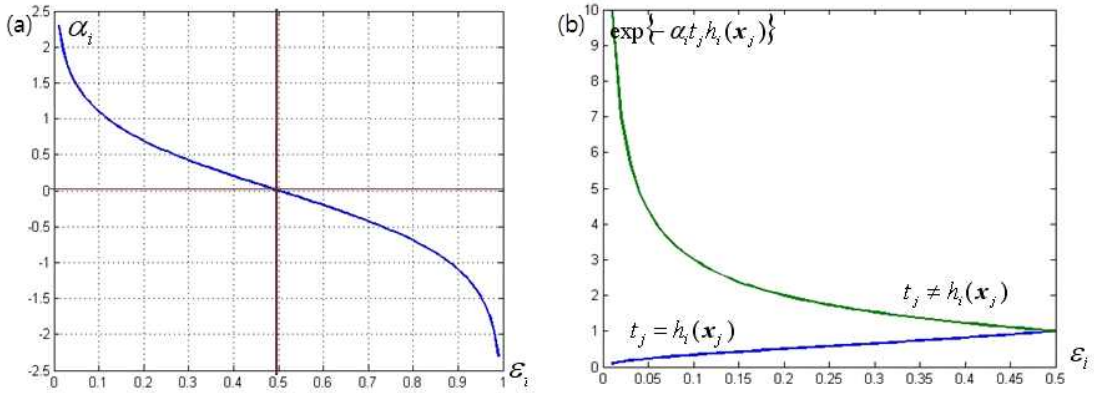
$$Z_i=\sum_{j=1}^{N}w_j^{(i)}\exp\{-\alpha_i t_j h_i(\boldsymbol{x}_j)\}$$

③ M개의 분류기가 모두 학습되면, 각 분류기의 중요도 값을 이용하여 M개의 분류기를 결합한 최종 판별함수를 만든다.

$$f_M(\boldsymbol{x})=\mathrm{sign}\left(\sum_{i=1}^{M}\alpha_i h_i(\boldsymbol{x})\right)$$

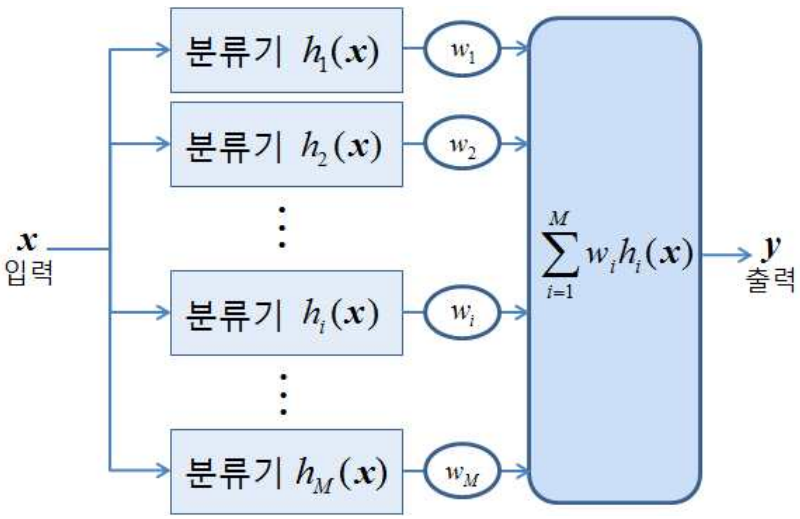
각 단계별 처리 과정에 대하여 좀 더 자세히 살펴보자. ②-1 단계에서 분류기를 학습하기 위한 오분류율을 정의한다. 첫 번째 분류기를 학습할 때에는 모든 데이터가 같은 가중치를 가지므로 일반적인 학습의 경우와 동일하나, 그 이후에는 이전 단계의 분류기의 결과에 따라 수정된 가중치가 사용될 수 있도록 정의되었다. ②-2 단계에서 분류기를 학습하는 구체적인 방법은 사용하는 분류기에 맞추어 선택해야 할 것이다. 13.6절에서는 간단한 퍼셉트론 분류기를 사용하여 구체적인 학습을 수행하는 예를 보일 것이다. 하나의 분류기에 대한 학습이 완료되면, 그 결과에 의존하여 다음 분류기의 학습을 위해 적용될 데이터의 가중치를 수정한다. 이를 위해 먼저 ②-3에서 계산되는 α_i의 값은 오분류율 ϵ_i에 의존하는 값으로, ϵ_i 값이 작을수록 큰 값을 가진다([그림 13-3a] 참조). 특히 오분류율이 0.5보다 작을 때 (즉, 랜

덤한 분류기 보다 분류성능이 높을 때) 양의 값을 가지고, 그렇지 못하면 음의 값을 가진다. 또한 ϵ_i이 작을수록 그 값이 커져서 오분류율이 작은 분류기가 더 큰 중요도를 갖도록 정의되어서, 이후 최종 판별함수 생성에서 분류기의 결합중요도로 사용된다.



[그림 13-3] 오분류율에 따른 결합중요도와 가중치수정에 사용되는 비례상수의 변화

α_i는 또한 다음 분류기의 학습을 위한 데이터의 가중치를 결정하는 과정에도 활용된다. ②-4 단계에서 가중치의 수정에 적용되는 값 $\exp\{-\alpha_i t_j h_i(\boldsymbol{x}_j)\}$ 는 [그림 13-3]과 같은 값을 가진다. [그림 13-3]에서 보이듯이 $h_i(\boldsymbol{x}_j) = t_j$인 경우에는 $\exp\{-\alpha_i t_j h_i(\boldsymbol{x}_j)\} = \exp(-\alpha_i)$ 가 되어 1보다 작은 값을 가지고, 결과적으로 가중치가 감소하는 효과를 얻는다. 반면 $h_i(\boldsymbol{x}_j) \neq t_j$인 경우에는 $\exp\{-\alpha_i t_j h_i(\boldsymbol{x}_j)\} = \exp(\alpha_i)$ 가 되어 1보다 큰 값을 가지게 되고, 가중치는 증가한다. 또한 오분류율이 0.5에 가까운 경우, 즉 랜덤하게 클래스를 할당하는 분류기와 그 성능이 크게 다르지 않은 경우에는 가중치의 변화폭이 크지 않은 반면, 전체 데이터에 대한 오분류율이 적어질수록 분류에 성공하지 못한 일부 데이터들에 대한 가중치 증가폭이 급격히 커지는 특성을 가진다.



[그림 13-4] 부스팅에 의한 결합 방법

이와 같이 조정된 가중치를 사용하여 각 분류기들을 단계적으로 학습함으로써 분류기들의 차별성을 높이고, 최종 결합단계에서는 학습에 사용된 가중치 α_i의 값을 적용하여 적은 오분류율을 가진 분류기가 판단에 더 중요한 역할을 할 수 있도록 한다. 이러한 결합 방법은 개념적으로 단순한 보팅법에 가중치를 적용한 것으로 [그림 13-4]와 같이 간단히 나타낼 수 있다.

13.3.2 AdaBoost 알고리즘과 오차변화

AdaBoost 알고리즘에서 순차적으로 분류기를 학습하고 가중치 파라미터를 수정하는 과정이 어떻게 결과적으로 오차를 감소시키는 방향으로 진행되는지 알아보겠다. m개의 분류기를 결합하여 얻어진 판별함수 $f_m(\boldsymbol{x})$에 대한 오차함수를 다음과 같이 지수오차함수 (exponential error function) 형태로 정의한다.

$$E = \sum_{j=1}^{N} \exp\{-t_j f_m(\boldsymbol{x_j})\} \qquad \text{[식 13-11]}$$

판별함수 $f_m(\boldsymbol{x})$는 m개의 간단한 분류기 $h_1(\boldsymbol{x}),...,h_m(\boldsymbol{x})$ 의 결합으로, 여기서는 마지막 m번째 분류기의 학습에 대해 생각하므로 [식 13-12]와 같이 분리하여 나타내었다.

$$f_m(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i h_i(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \alpha_m h_m(\boldsymbol{x}) \qquad \text{[식 13-12]}$$

이를 오차함수 식에 대입하면 다음과 같은 관계식을 얻을 수 있다.

$$\begin{aligned} E &= \sum_{j=1}^{N} \exp\{-t_j f_{m-1}(\boldsymbol{x_j}) - t_j \alpha_m h_m(\boldsymbol{x_j})\} \qquad \text{[식 13-13]} \\ &= \sum_{j=1}^{N} \exp\{-t_j f_{m-1}(\boldsymbol{x_j})\} \exp\{-t_j \alpha_m h_m(\boldsymbol{x_j})\} \\ &\propto \sum_{j=1}^{N} w_j^{(m)} \exp\{-t_j \alpha_m h_m(\boldsymbol{x_j})\} \end{aligned}$$

이 식에서 마지막 부분의 비례관계는 AdaBoost 처리 과정의 ②-4 단계에서 사용한 가중치에 대한 관계식을 재귀적으로 적용함으로써 얻게 되는 다음의 [식 13-14]를 사용하여 계산되었다.

$$\begin{aligned} w_j^{(m)} &= \frac{1}{Z_{m-1}} w_j^{(m-1)} \exp\{-\alpha_{m-1} t_j h_{m-1}(\boldsymbol{x_j})\} \qquad \text{[식 13-14]} \\ &= \frac{1}{Z_{m-1}} \frac{1}{Z_{m-2}} w_j^{(m-2)} \exp\{-\alpha_{m-2} t_j h_{m-2}(\boldsymbol{x_j})\} \exp\{-\alpha_{m-1} t_j h_{m-1}(\boldsymbol{x_j})\} \end{aligned}$$

$$= \left(\prod_{i=1}^{m-1}\frac{1}{Z_i}\right) w_j^{(1)} \exp\left\{-\sum_{i=1}^{m-1}\alpha_i t_j h_i(\boldsymbol{x_j})\right\}$$
$$= \frac{1}{N}\left(\prod_{i=1}^{m-1}\frac{1}{Z_i}\right) \exp\left\{-\sum_{i=1}^{m-1}\alpha_i t_j h_i(\boldsymbol{x_j})\right\}$$
$$= \frac{1}{N}\left(\prod_{i=1}^{m-1}\frac{1}{Z_i}\right) \exp\{-t_j f_{m-1}(\boldsymbol{x_j})\}$$

계속해서 [식 13-13]으로부터 ②-1에서 주어진 오분류율의 정의를 이용하면 다음과 같이 식의 변형이 가능하다.

$$\sum_{j=1}^{N} w_j^{(m)} \exp\{-t_j \alpha_m h_m(\boldsymbol{x_j})\} \qquad \text{[식 13-15]}$$
$$= \sum_{j=1}^{N} w_j^{(m)} \exp\{-t_j \alpha_m h_m(\boldsymbol{x_j})\}\left(I(t_j = h_m(\boldsymbol{x_j}) + I(t_j \neq h_m(\boldsymbol{x_j}))\right)$$

$$= \sum_{j=1}^{N} w_j^{(m)} \exp\{-t_j \alpha_m h_m(\boldsymbol{x_j})\} I(t_j \neq h_m(\boldsymbol{x_j})) + \sum_{j=1}^{N} w_j^{(m)} \exp\{-t_j \alpha_m h_m(\boldsymbol{x_j})\} I(t_j = h_m(\boldsymbol{x_j}))$$
$$= \sum_{j=1}^{N} w_j^{(m)} \exp(\alpha_m) I(t_j \neq h_m(\boldsymbol{x_j})) + \sum_{j=1}^{N} w_j^{(m)} \exp(-\alpha_m) I(t_j = h_m(\boldsymbol{x_j}))$$
$$= \exp(\alpha_m)\epsilon_m + \exp(-\alpha_m)\sum_{j=1}^{N} w_j^{(m)} I(t_j = h_m(\boldsymbol{x_j}))$$
$$= \exp(\alpha_m)\epsilon_m + \exp(-\alpha_m)\left(1 - \sum_{j=1}^{N} w_j^{(m)} I(t_j \neq h_m(\boldsymbol{x_j})\right)$$
$$= \exp(\alpha_m)\epsilon_m + \exp(-\alpha_m)(1-\epsilon_m) = \epsilon_m(\exp(\alpha_m) - \exp(-\alpha_m)) + \exp(-\alpha_m)$$

이 식을 우리가 추정하고자 하는 파라미터 α_m에 대하여 최소화하면 다음과 같은 식을 얻을 수 있다.

$$\alpha_m = \frac{1}{2} ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\} \qquad \text{[식 13-16]}$$

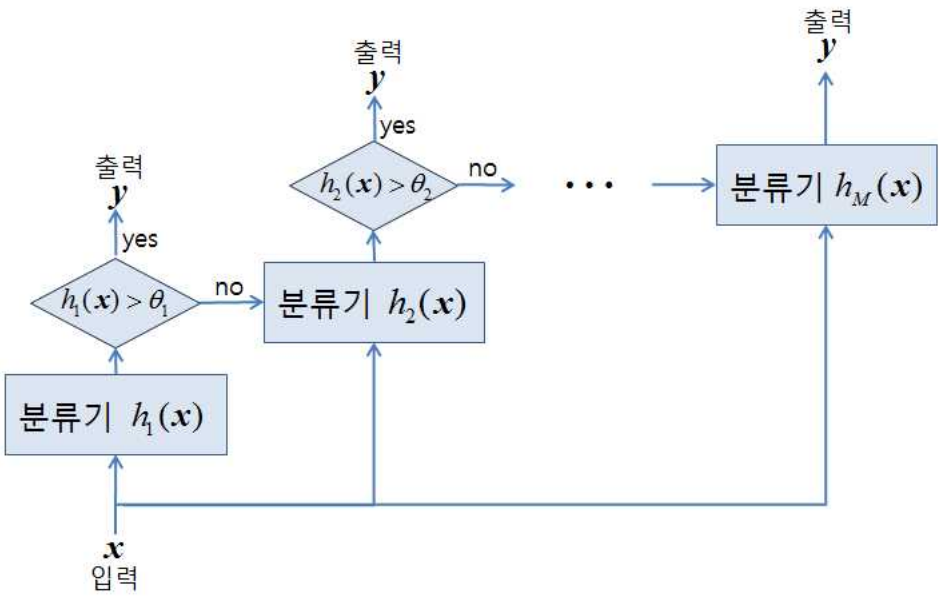
이는 AdaBoost 처리 과정의 ②-3 단계에서 사용한 값으로, 결국 AdaBoost에서 사용하는 분류기에 대한 가중치 파라미터는 지수오차함수에 대해 최적화된 값임을 알 수 있다. 또한 [식 13-15]의 마지막 항에서 α_m이 고정되면, 최소오차는 ϵ_m을 최소화하도록 분류기 h_m을 학습함으로써 얻게 된다. 이 또한 AdaBoost 처리 과정의 ②-2 단계에서 수행하고 있음을 확인할 수 있다.

이상에서 살펴본 바와 같이, AdaBoost 알고리즘은 이전 단계의 분류기의 학습 결과를 활용하여 다음 단계의 학습에 사용될 데이터에 가중치를 부여함으로써 분류기들 간의 차별성을 부여하고, 지수오차함수의 측면에서 최적화된 결합가중치를 찾아 분류기들을 결합한다. Fruend와 Schapire는 각각의 간단한 분류기의 오분류율이 0.5보다 작은 조건만 만족하면 분류기의 결합을 통해 학습 데이터에 대한 오차를 기하급수적으로 감소시킬 수 있음을 보였

다. 또한 일반화 성능에 대한 분석도 수행되었으나, 이론적 분석으로부터 얻어진 결과에 비해 실제 문제에서 월등히 좋은 결과를 보여주고 있다.
마지막으로, AdaBoost 알고리즘은 12장에서 소개한 SVM과 마찬가지로 두 개의 클래스에 대한 분류 문제에 적합한 방법이다. 이를 다중 클래스 문제에 적용하기 위해서는 6장에서 설명한 각 클래스별 판별함수를 만들어 사용하는 방법이나 클래스 쌍에 대한 판별함수를 만들어 사용하는 방법 등을 적용해야 할 것이다.

13.4 캐스케이딩

<캐스케이딩(cascading)>은 결합 방법에 중점을 둔 것으로, 인식에 많은 계산 비용이 요구되는 문제에서 계산 효율을 높이면서 안정적인 분류 성능을 얻기 위하여 전략적으로 여러 가지 복잡도를 가진 분류기를 결합하는 방법으로 제안되었다. [그림 13-5]에 분류기의 결합 구조를 나타내었다. 그림에서 보이는 바와 같이, 분류기들은 순차적으로 결합된다. 먼저 분류 성능은 떨어지지만 계산 비용이 적게 드는 간단한 분류기를 이용하여 데이터 집합을 학습한다. 학습이 완료되면, 별도의 데이터 집합에 대한 분류를 수행하여, 분류 결과가 좋지 못한 데이터들을 찾는다. 이 때 유의해야 할 것은, 단순히 오분류된 데이터만을 선택하는 것이 아니라, 분류기의 출력에 대한 신뢰도를 측정하여 신뢰도가 높지 않은 데이터들도 함께 선택한다. 이렇게 선택된 데이터들은 다음 단계의 분류기를 학습하는데 사용된다. 학습기의 선택에 있어서는, 단순 부스팅과 같이 같은 종류의 분류기를 사용하는 것도 가능하다. 하지만, 단계가 높아질수록 이전 단계에서 바르게 분류되기 힘든 데이터들이 학습에 사용된다는 점을 고려하면 높은 단계에서는 보다 복잡하면서 분류 능력이 좋은 분류기들을 사용하는 것이 효과적일 것이다. 예를 들어 다층 퍼셉트론을 사용하는 경우, 은닉 뉴런의 수나 은닉층의 수를 증가시킴으로써 단계가 높아짐에 따라 분류기의 복잡도를 증가시킬 수 있을 것이다.



[그림 13-5] 캐스케이딩에 의한 결합 방법

학습이 완료되고 나면, 인식 단계에서도 단계적 처리를 거친다. 새롭게 주어진 데이터에 대하여 먼저 첫 번째 분류기를 통과시켜 분류 결과와 함께 신뢰도를 계산한다. 만약 신뢰도 값이 충분히 높다면, 더 이상 다음 단계의 분류기에 대한 처리를 수행하지 않고 첫 번째 단계의 분류기의 결과를 출력으로 낸다. 만약 충분한 신뢰도가 얻어지지 않는다면, 다음 단계의 분류기에 다시 데이터를 제공하여 역시 분류 결과와 신뢰도를 계산한다. 얻어진 신뢰도에 따라 출력을 내고 분류 과정을 완료하거나 다음 단계의 분류기로 진행하는 과정을 순차적으로 진행하게 된다.

이러한 형태의 결합 방법이 가지는 가장 큰 장점은 인식에 필요한 계산 시간을 절약할 수 있다는 점이다. 특히, 인식의 대상이 되는 데이터 집합 중 많은 부분이 간단한 분류기로도 쉽게 분류될 수 있는 데이터로 구성되는 문제에서는 첫 분류 단계에서 대부분의 데이터가 처리될 수 있으므로 계산 시간이 줄어들면서 동시에 복잡한 데이터에 대해서는 다음 단계의 분류기들을 활용함으로써 분류 성능도 보장해 주는 효과를 가지게 된다. 특히 최근 들어 많이 연구되고 있는 영상 데이터에서 객체인식이나 객체탐지 등의 문제에 효과적으로 적용될 수 있다. 입력으로 주어지는 전체 영상에서 탐지의 대상이 되는 객체가 차지하는 부분은 극히 일부일 경우라도, 탐지를 위해서는 전체 영상을 스캔하면서 인식을 수행해야 한다. 이러한 경우에 배경 영상 등과 같이 영상의 대부분이 쉽게 분류될 수 있는 데이터들이므로, 이러한 부분들을 초기 단계 분류기를 활용하여 빠르게 걸러낼 수 있을 것이다. 이후 탐지 대상이 되는 객체와 유사한 부분이 입력으로 들어올 경우, 보다 복잡한 분류기로 넘겨줌으로써 정확한 탐지를 기대할 수 있을 것이다. [그림 13-6]은 자동차 영상에서 번호판 부분을 탐지하는 문제를 나타낸 것으로 캐스케이딩 방법이 효과적으로 적용될 수 있는 예이다.

[그림 13-6] 캐스케이딩에 의한 결합 방법의 응용 예 – 영상에서 차량번호판의 탐지
(저작권 고려하여 교체 필요)

13.5 전문가 혼합

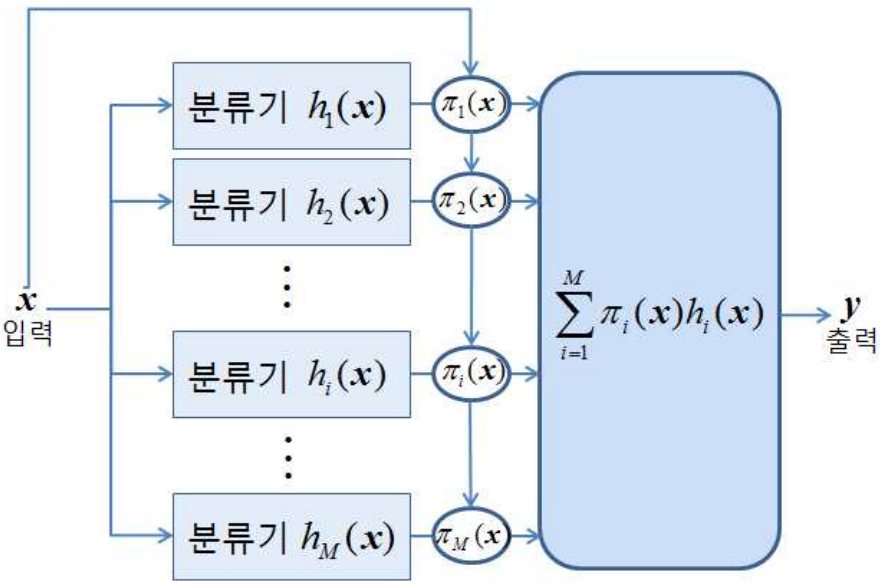
<전문가 혼합(Mixture of Experts)> 방법도 복수개의 분류기를 가중합하여 최종 분류기를

만드는 결합 방법에 대한 연구로, 결합에 의해 얻어지는 분류기는 다음과 같이 정의될 수 있다.

$$f(\boldsymbol{x}) = f(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), ..., h_M(\boldsymbol{x})) = \sum_{i=1}^{M} \pi_i(\boldsymbol{x}) h_i(\boldsymbol{x}) \qquad \text{[식 13-17]}$$

이 방법이 앞서 소개한 보팅 방법이나 가중치를 가진 결합 방법과 다른 점은 가중합에 사용되는 계수 $\pi_i(\boldsymbol{x})$가 입력에 대한 함수로 주어진다는 점이다.([그림 13-7] 참조) 단순한 보팅 뿐 아니라 AdaBoost에서 사용한 가중결합의 경우에 사용된 α_i도 분류 성능에 의존하여 각 분류기에 대하여 각각 하나로 정해지는 값이지만, 전문가 혼합법에서는 입력에 대한 함수형태인 $\pi_i(\boldsymbol{x})$를 사용함으로써 어떤 입력이 주어지느냐에 따라 어떤 분류기를 중요하게 사용할 것인가가 달라진다.

이러한 특성을 학습 단계에서도 충분히 활용한다면, 전문가 혼합에서는 입력 공간을 복수개의 영역으로 나누어 각각의 분류기가 특정 영역을 중점적으로 담당할 수 있도록 학습하고, 그에 맞추어 결합에 사용되는 가중치도 결정해 주는 전략을 취하는 것이 효과적일 것이다. 명칭에 사용된 "전문가"라는 용어는 입력 공간의 특정 영역을 담당한다는 의미에서 붙여진 것으로 볼 수 있다.



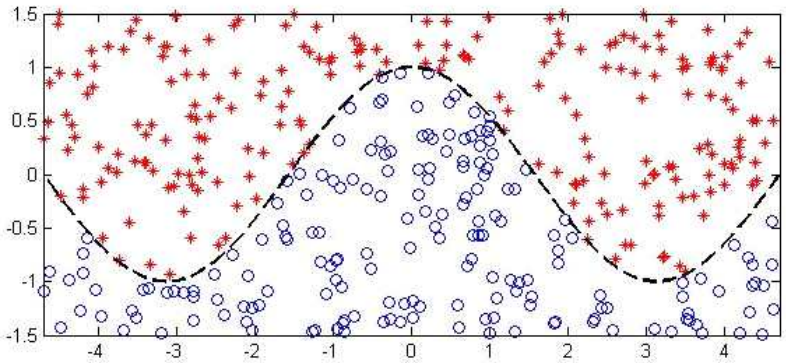
[그림 13-7] 전문가 혼합에 의한 결합 방법

각 분류기가 서로 다른 입력 공간의 영역을 담당하게 한다는 것은 학습을 함에 있어서도 각 분류기의 학습에 사용되는 데이터를 입력 공간에 따라 나누어 제공함을 의미한다. 이렇게 하면, 자연스럽게 각 분류기들 간의 상관관계가 줄어들거나 혹은 음의 상관관계를 갖게 되어 [식 13-8]에서의 상관관계에 의존하는 부분의 값이 줄어들고, 결과적으로 일반화 오차를 감소시키는 역할을 할 수 있다. 각 분류기가 담당하는 입력 영역이 명시적으로 정해지면, 이에 맞추어 가중치 함수 $\pi_i(\boldsymbol{x})$도 결정해 줄 수 있을 것이다.

그러나 인위적으로 입력 영역을 나누는 방법을 사용하는 대신 가우시안 혼합 모델에서와 같이 스스로 담당 영역을 학습하는 접근 방법을 취할 수 도 있을 것이다. 이 경우에는 10장에서 소개한 EM 알고리즘을 사용하여 분류기의 학습과 가중치 함수의 학습을 번갈아 수행하는 방법을 생각해 볼 수 있다.
또한 더 나아가서는 전문가 혼합에 의해 만들어지는 분류기들을 다시 하나의 간단한 분류기로 보고 이들을 혼합하여 보다 복잡한 규모의 분류기를 생성하는 계층적 전문가 혼합 방법에 대한 연구도 수행되고 있다. 이와 같이 다양한 방식의 결합 방법을 문제에 맞게 적용해 볼 수 있을 것이다.

13.6 매트랩을 이용한 실험

13.3절에서 소개한 AdaBoost 방법을 이용하여 퍼셉트론 분류기를 결합하는 실험을 수행해 보겠다. 분류 대상이 되는 데이터는 11장에서 다층 퍼셉트론을 이용한 분류 문제에 적용한 데이터를 사용한다. [그림 13-8]에 결정경계와 함께 학습 데이터를 다시 나타내었다. 비선형 결정경계를 가진 분류 문제를 해결하기 위해 선형 분류기인 퍼셉트론들을 결합한다. AdaBoost 방법을 사용하므로, 11장에서 사용한 400개의 학습 데이터 집합을 그대로 사용하되, 가중치를 조절해 가면서 퍼셉트론 분류기들을 학습할 것이다.



[그림 13-8] 학습 데이터와 결정경계

[프로그램 13-1]에 AdaBoost에 의한 분류기 학습과 결합을 수행하는 코드를 제시하였다. 먼저 데이터를 불러들이고 기본적인 변수 설정을 수행한다. 11장에서 사용한 데이터를 그대로 사용하지만, 11장에서는 2개의 출력 뉴런을 가진 다층 퍼셉트론을 학습하기 위해 목표 출력값도 2차원 벡터로 표현하였으나, 이 장에서는 1 또는 -1의 출력을 가지는 이진 분류기를 사용하므로 출력값을 1차원으로 변경하는 작업을 수행해 주었다. 이진 분류기로는 퍼셉트론을 사용하므로 퍼셉트론의 입력 뉴런 수와 출력뉴런 수도 설정해 주고, 결합하는 분류기의 수(M)도 설정해 주었다. 프로그램 코드에서 분류기의 개수는 10으로 설정하였으나, 1부터 200까지 변형해 가면서 찾아지는 결정경계의 변화를 살펴볼 것이다. 초기화의 마지막 단계로, 각 데이터에 대한 가중치를 $1/N$로 모두 동일하게 초기화하였다.

프로그램 13-1 AdaBoost 프로그램

2차원 데이터 분류를 위해 퍼셉트론을 결합한 분류기를 생성

```
001 load data12_20                    %학습 데이터 불러오기
002 T=T(:,1);                         %출력값을 1차원 벡터로 표현
003 M=10;                             %Weak Learner의 개수 설정
004 N=size(X,1);                      %데이터의 수
005 INP=size(X,2); OUT=1;             %퍼셉트론의 입출력 노드의 수 설정
006 W(:,1)=zeros(N,1)+(1/N);          %데이터에 대한 가중치 초기화
007 % Weak Learner의 학습 시작
008 for i=1:M
009     %train_perceptron함수를 호출하여 weak learner 학습
010     [V(:,i),H(:,i)]=train_perceptron(INP, OUT, X, T, W(:,i));
011     %V는 학습된 퍼셉트론의 가중치, H는 출력값
012     e(i,1)=(W(:,i)'*((-H(:,i).*T+1)/2));          %오분류율 계산
013     alpha(i,1)=log((1-e(i))/e(i))/2;              %분류기 결합 가중치 계산
014     Z(i,1)=(W(:,i)'*exp(-alpha(i)*(T.*H(:,i))));  %정규화 상수 계산
015     if (i<M)                                      %가중치 수정
016         W(:,i+1)=W(:,i).*exp(-alpha(i)*(T.*H(:,i)))/Z(i);
017     end
018 end
019 for m=1:M                             %분류기 결합 개수에 따른 분류오차 계산
020     F=sign(H(:,1:m)*alpha(1:m));
021     E=((-T.*F+1)/2);
022     Cerr(m)=mean(E);
023 end
```

이어서 간단한 분류기의 학습 과정이 수행되는데, 이를 위해서는 퍼셉트론을 학습하는 함수 train_perceptron을 별로로 정의하여 [프로그램 13-1-1]에 나타내었다. 이 함수는 퍼셉트론의 뉴런 수와 데이터 집합, 그리고 각 데이터에 대한 가중치 벡터를 입력으로 받아서 퍼셉트론을 학습하고, 그 결과 얻어진 퍼셉트론의 가중치와 학습 데이터에 대한 출력값을 반환한다. 퍼셉트론 학습은, 데이터에 대한 가중치를 고려하여 계산되는 오분류율이 0.45보다 작아지면 학습을 완료하도록 설정하였는데, 이는 부스팅 방법에서 결합을 통한 오차감소의 효과를 얻기 위해서는 간단한 분류기에 대한 오분류율이 0.5보다도 적어야 한다는 조건을 고려해서 결정하였다. 학습률과 퍼셉트론의 가중치를 초기화한 후, 학습을 수행하는데, 여기서는 6장에서 사용한 퍼셉트론 학습과는 달리, 전체 데이터 집합에 대한 오차함수를 감소시키는 방향으로 가중치가 조절되도록 하는 배치 모드 학습을 수행하도록 하였다. (6장과 11장의 다층 퍼셉트론 학습에서는 하나의 데이터가 주어질 때마다 가중치를 조정하는 온라인 모드 학습을 수행하였다.) 또한 원래의 퍼셉트론이 계단함수를 출력함수로 사용지만, 여기서는 다층

퍼셉트론과 같은 하이퍼탄젠트 함수를 적용하였다. 학습이 완료되면 얻어진 가중치 벡터와 출력값을 반환한다.

프로그램 13-1-1 퍼셉트론 분류기를 학습하는 함수 train_perceptron

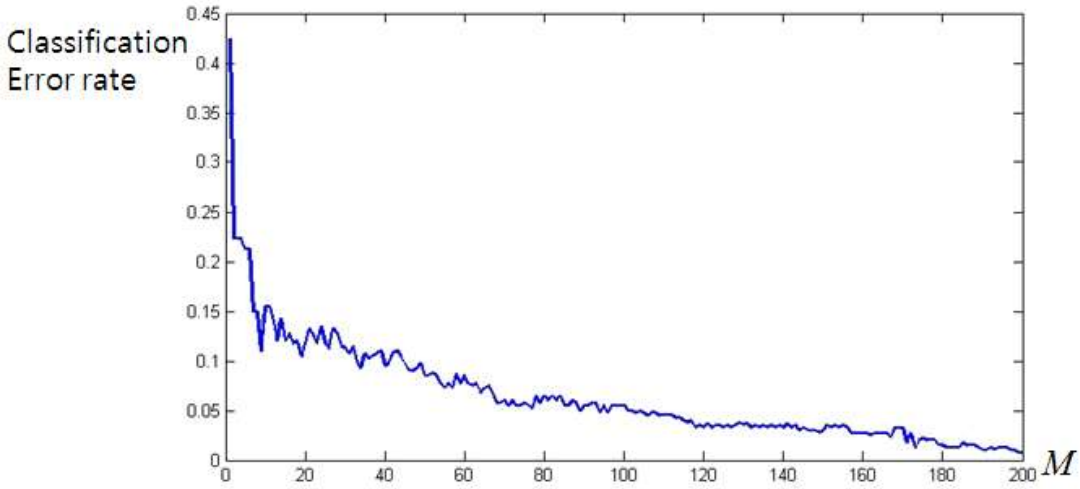
학습 데이터와 데이터에 대한 가중치, 그리고 신경망 크기를 입력받아 퍼셉트론을 학습한 후 가중치와 출력값을 결과값으로 돌려주는 함수

```
001 function [v,h]=train_perceptron(INP, OUT, X, T, w)
002 % 매개변수 INP: 입력뉴런 수, OUT: 출력뉴런 수
003 %          X : 학습 데이터입력     T: 학습 데이터 목표 출력
004 %          w : 데이터에 대한 가중치
005 MaxStep=100000;              %최대 학습횟수
006 Elimit=0.45;                 %오분류율이 0.45보다 작으면 학습 멈춤
007 N=size(X,1);                  %학습 데이터의 수
008 eta=0.1;                      %학습률
009 v=rand(INP+1, OUT)*2-1.0;    %신경망 가중치 초기화
010 B=zeros(size(X,1),1)+1;       %바이어스 입력을 위한 행렬
011 XB=[X, B];                     %입력과 바이어스 입력을 합쳐서 표현
012 for i=1:MaxStep               %전체 데이터에 대한 학습 시작
013     Y=tanh(XB*v);             %출력 노드의 출력값 계산
014     h=sign(Y);                %분류 결과(-1, 1의 값)
015     err=T-h;                  %목표 출력과의 차이
016     dv=XB'*((-w.*err).*(1-Y).*(1+Y));  %가중치 수정항 계산
017     v=v-eta*dv;                        %가중치 수정
018     E(i)=w'*((-T.*h+1)/2);             %오분류율 계산
019     if (E(i)<Elimit)
020         break                 %오분류율이 기준값보다 작으면 학습 완료
021     end
022 end
023 fprintf(1,'%d %.5f₩n', i,E(i));    %학습 결과 모니터에 출력
```

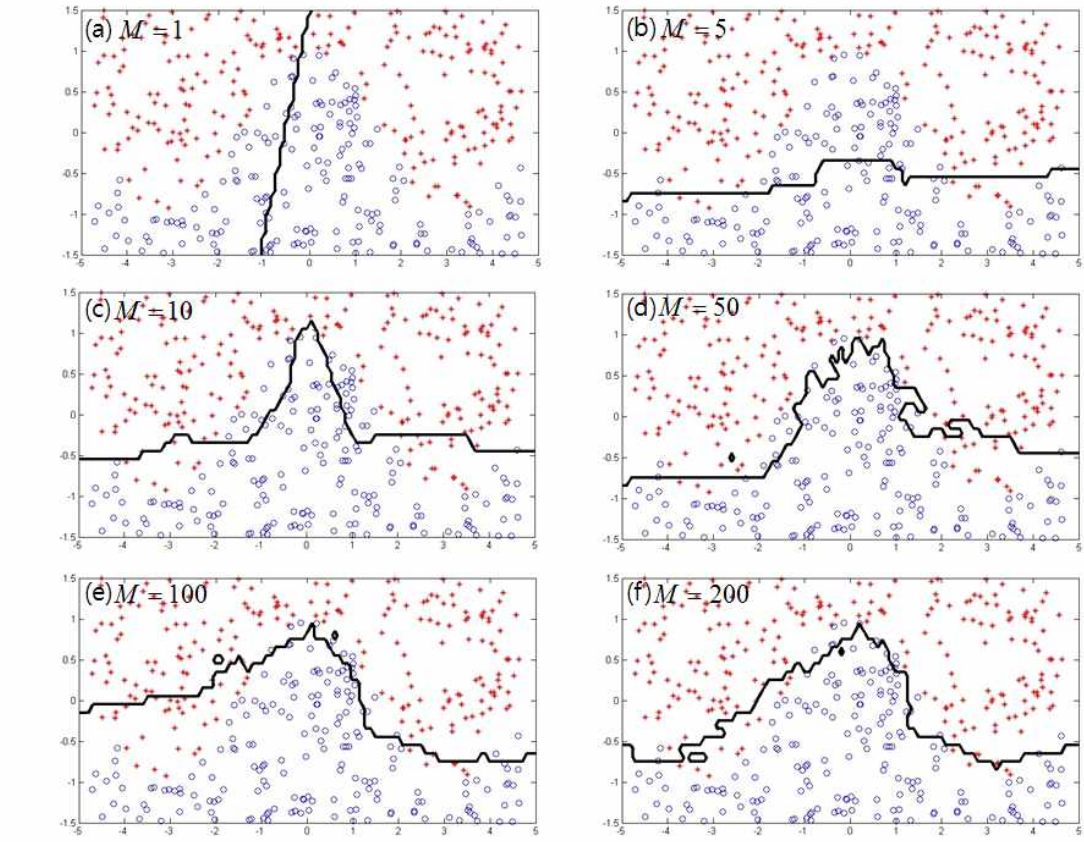
함수 train_perceptron으로부터 반환된 가중치와 출력값을 이용하여 [프로그램 13-1]에서는 계속해서 오분류율(ϵ_i)과 결합가중치(α_i)를 계산하고, 이어서 데이터에 대한 가중치 w_j를 수정한 다음 다시 새로운 분류기의 학습을 위하여 train_perceptron을 호출하는 과정을 반복한다. 이 과정을 통하여 얻어진 각 퍼셉트론 분류기의 가중치와 결합가중치는 별도로 저장해 두어 이후 인식과정에 쓸 수 있도록 준비해 두었다.

학습이 완료된 후, 학습 데이터에 대해 결합된 분류기의 출력값을 계산해 보고, 분류오차를 계산하였다. 분류오차는 결합하는 분류기의 수를 증가시키면서 그 변화를 저장하여 [그림 13-9]에 그래프로 나타내었다. 그림에서 200개까지 분류기가 하나씩 추가되는 동안 전체적으로 분류오차가 감소해 감을 확인할 수 있다. [그림 13-10]에는 결합한 분류기의 수에 따른

결정경계의 변화를 나타내었다. $M=1$ 인 경우는 선형 결정경계를 나타내었으며, 그 수가 늘어남에 따라 복잡한 비선형 결정경계를 형성하게 됨을 확인할 수 있다.



[그림 13-9] 결합하는 분류기의 개수 변화에 따른 분류오차의 변화



[그림 13-10] 결합하는 분류기의 개수 변화에 따른 결정경계의 변화

연습문제

1. 이 장에서 소개하고 있는 학습기의 결합 방법인 배깅과 부스팅, 그리고 전문가 혼합 방법의 공통점과 차이점을 비교하시오.

2. 캐스케이딩 방법이 적합한 응용문제의 예를 하나 들고, 그 설계 방법에 대하여 생각해 보시오.

3. 선형분류기의 보팅에 의한 결합방법에 의해 형성되는 결정경계와 다층퍼셉트론에서 얻어지는 결정경계의 특성을 비교해 보시오.

4. 이 장의 마지막 절에서 우리는 간단한 학습기로 퍼셉트론을 사용한 경우의 AdaBoost 알고리즘을 구현해 보았다. 퍼셉트론보다 더욱 간단한 분류기의 예로, n차원의 입력 벡터의 여러 요소 중 임의로 하나를 선택하여 그 값에 따라 분류를 수행하는 분류기를 생각해 볼 수 있다. 예를 들어 [그림 13-8]에 나타난 2차원 데이터의 경우, 가로축의 좌표값이 어떤 역치값보다 크면 C_1으로 할당하고 그렇지 않으면 C_2로 할당 하는 것과 같은 분류기를 생각해 볼 수 있다. 이와 같은 간단한 분류기를 사용한 AdaBoost 매트랩 코드를 구현하고, 13.6절에서 사용한 데이터에 대해 분류를 수행해 보시오.

5. 복수개의 학습기를 결합함에 있어서, 그 효과를 최대화하기 위해 가장 중요한 것은 결합하는 학습기의 출력특성을 차별화 하는 것이다. 이를 위한 전략으로 어떤 방법들이 있는지 나열하시오.

참고자료

간단한 학습기 (weak learner)들을 결합하여 복잡한 학습능력을 가진 학습기를 만드는 것에 관한 심도 깊은 논의의 시작은 [Schapire 90]에서 시작되었다. 이후 Schapire와 Fruend 등에 의해 부스팅과 관련된 다양한 연구들이 수행되었으며, 이 책에서 소개하고 있는 AdaBoost는 [Fruend & Schapire 95]에서 제안되었다. 부스팅과 관련된 연구들에 대한 간단한 소개는 [Fruend & Schapire 99]를 참조하기 바란다. 이밖에 이 책에서 소개하는 케스케이딩 방법에 대한 실제 문제에의 응용 사례가 [Viola & Jones 01]에 기술되어 있으며, 학습기의 결합에 대한 기본적인 내용은 [Alpaydin 04]를 참조하기 바란다.

[Schapire 90] R. E. Schapire. The strength of weak learnability. Machine Learning, 5(2):197-227, 1990.
[Fruend 95] Y. Fruend. Boosting a Weak Learning Algorithm by Majority. Information and Communication, 121(2):256-285, 1995.
[Fruend & Schapire 97] Y. Fruend and R. E. Schapire. A decision-theoretic generalization

of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119-139, 1997.
[Fruend & Schapire 99] Y. Fruend and R. E. Schapire. A Short Introduction to Boosting. Journal of Japanese Society for Artificial Intelligence, 14(5), 771-780, 1999.
[Viola & Jones 01] P. Viola and M. Jones, Rapid Object Detection using a Boost Cascade of Simple Features, Proc. of CVPR, 2001.
[Alpaydin 04] E. Alpaydin. Introduction to Machine Learning. MIT Press. 2004.