**Classification**: 주어진 데이터를 미리 정해진 class로 분류 하는 것

* spam filtering, weather prediction, face recognition

**Regression** : 주어진 데이터로부터 numeric value를 예측

* stock market, weather prediction

**Clustering** : data에서 구조를 발견하여 비슷한 데이터끼리 grouping

* class의 개수나 타입은 모름
* clustering image, websearch result

**Ranking**: Item을 비교하여 유사도를 판단

* web search, image

**Collaborative Filtering**: Recommendation system

**Embeding**: 차원 축소 및 data 시각화

**Structured prediction**: 구조적 특성을 가지고 정해진 class로 분류하는 것

* speech recognition

**Supervised Learning**

* 주어진 training set { (xi , yi)|i = 1,2…,N}을 통해 학습하여 f:X->Y의 good approxiation을 찾는 것

**Occam’s Razor**

* 어떤 것을 설명하기위한 entity의 수를 필요 이상으로 늘려서는 안된다.
* 가장 간단한 것을 골라야 한다.
* 가설 공간을 고를 때 사전 지식을 이용해 simple한 것을 고르고 이것이 곧 overfitting을 막아줌.

**Train set**: 학습시키는데 사용되는 데이터(정답 제공)

**Validation set**: train할 때 error rate 계산에 이용되며 overfitting을 막아줌(정답 제공)

**Test set**: 정답을 주지 않음

**Overfitting**

* trainset에 지나치게 학습되어 실제 test data에 대한 error가 증가하는 현상

**Loss Function**

**Perceptron**

* **->**
* binary classification
* linearly separable한 데이터로 제한
* two-layer neural network
* feature values가 input, 각 feature는 weight 가지고 있음
* error가 줄어드는 방향으로 w업데이트, weight vector는 hyperplane

**Regularization**

* ),
* ),
* losss function 에 penalty를 더해 overfitting 감소시켜줌
* w에서 (람다가 커질수록) 특정 wi값이 커지는 것을 막아줌.

**Cross validation**

* test set이 없을 때 train set로부터 validation set 만들어 값 구함.
* 나머지 data에서 여러 값으로 학습하고, validation set에서 test하여 best 구함.
* small validation set -> large error
* large validation set -> small training set -> bad

**k-fold cross-validation**

* data를 K blocks로 나눔(k=1…n)
* kth block을 제외하고 train시키고, kth block으로 테스트
* 이를 통해서 best 고름
* 주어진 train set 전체를 train set으로 사용하면서 또 test set으로도 사용하는 효과
* High computation cost

**Support Vector Machine**

* Linearly Separable 하지 않아도 됨
* Margin을 최대화하는 Hyperplane 고름
* Support Vectors: margin과 접촉하고 있는 vector들
* 이를 만족하는 unique global minimum value가 있음(convex optimization)
* minimum value를 제공하는 unique minimizer w, b 값이 있음
* linearly separable하지 않으면 풀지 못함
* Quadratic programming으로 solve 됨
* **Soft Margin**
* margin을 최대화 하면서 를 zero로 만들려고 노력해야함
* C가 무한대로 갈수록 hard margin에 가까워짐
* 항상 solution이 있음
* Outlier에 좋은 성능 보임
* Hard Margin에서는 C를 예측할 필요 없음
* **High Dimensional Mapping**
* W, C, P에 대한 학습 필요
* P(Polynomial function)에 대한 학습은 선형적인 시간에 학습할 수 있음
* 고차원으로 보낼수록 연산량이 많아짐.
* **Kernal Trick**: 고차원으로 mapping 하게되는 경우 maximum margin hyperplane을 구하기 위해 계산 비용이 많이 드는데 이를 해결하기 위해 사용, 고차원에서의 내적을 direct하게 구할 수 있음.

**Bayesian Classification**

* Bayes’s Theorem에 근거한 통계학적 분류기
* 각 train example은 가설의 정확도에 incrementally 영향을 미침.
* 다른 방법들을 측정할 수 있는 좋은 비교 기준 제공
* P(h): prior probability
* P(X|h): Likelihood
* P(X): marginal probability
* P(h|X): posterior probability
* **Naïve Bayes Classification**
* attribute들이 given class에서 조건부독립이라고 가정
* 이는 computation cost를 줄여줌
* 실제로는 독립이 아니지만 이러한 가정은 real world problem에 합리적으로 잘 적용됨
* 구현이 쉽다는 장점
* 대부분의 case에서 좋은 결과를 얻을 수 있다는 장점
* 조건부독립이라고 가정하기 때문에 정확도의 loss는 있음
* **Bayesian Belief Network**
* 는 immediate predecessors임.

**Regression**

* 입력변수로부터 출력 변수 또는 target 함수를 예측하는 모델
* **-> 최소화 해야함**
* **,**
* data set이 너무 크거나 역행렬을 구할 수 없는 경우 gradient descent
* **stochastic gradient descent:** training sample이 랜덤하게 골라진 경우
* divergent oscillation을 막기위해 learnoing rate을 잘 골라야 함.
* **Logistic Regression**
* binary 범주형 자료의 경우 유리
* Linear는 무한대의 범위를 가지므로

**Decision Tree**

* 사람이 판단의 과정을 이해하고 설명할 수 있음
* input attribute에 대한 어떤 형태의 함수도 나타낼 수 있음
* 지수적으로 node를 필요로 할 수 있음
* Uniform distribution은 좋지 않음
* H(Y): randomly drawn value의 y를 알아내는 데 필요한 expected number of bit(질문의 수)
* Entropy를 줄이는 방향으로 Tree를 Split해야함(Information Gain이 큰 쪽으로)
* Pruning: spliting이후 leaf node를 비교해서 accuracy를 감소시키는 가지를 제거하는 것
* random forests(ensembles): 여러 개의 tree를 만들어내어 이들을 종합해 최종 판단을 내림.
* High Entropy
* Y는 uniform like distribution
* Flat histogram
* sample링된 값을 예측하기 어려움