	장점
Stochastic gradient decent	전체 Input 데이터를 다 거치고 가중치가 업데이트 되는거라면, SGD는 전체 데이터가 아닌 batch단위로 가중치를 업데이트하는 방법 m개의 샘플 추출하므로 연산 이득 mini batch의 1회 학습이 빠른만큼 보다 더 빠르게 가중치 업데이트를 진행할 수 있음 neural network에서 각 layer nonlinear하므로 총 model도 nonliner. 즉 objective를 위한 loss도 스무스하지않기 때문에 Batch 방식보다 로컬 minimum에 빠지지 않음 데이터 전부를 하나의 batch로 넣으면 이 데이터 전체의 loss를 구해 주변의 local minimum에 빠질 위험이 있다. loss가 batch마다 다르므로 약간씩 엇나가며 왔다갔다 거리며 주변에 있는 local minimum을 지나쳐 global minimum에 도달할 확률이 높다
Stochastic gradient decent with momentum	앞선 방식은 gradient가 0일 때 Local Minima, Saddle points 에 빠질 가능성이 크지만 momentum 개념이 추가되면 이전 속도를 반영하기 때문에 이런 곳에 빠져 나올 수 있음
Stochastic gradient decent with Nesterov Momentum	momentum을 줘서 관성을 준 것은 좋은데, global minimum에 도달하려고 loss를 계속 계산하다가 global minimum 주변에서 w가 안착하지 못하고 왔다갔다 거릴 수가 있다. momentum step을 먼저 고려하여, momentum step을 먼저 이동한 후 그 자리에서의 gradient를 구해서 gradient step을 이동한다.
Adagrad	학습률이 너무 작으면 시간이 오래 걸리고, 너무 크면 최적값을 찾지 못하고 발산 >>가중치에 큰 변동이 있었으면 learning rate을 알아서 감소시키고, 가중치 변동이 작은 것에는 learning rate를 알아서 증가시키는 것 학습률 값은 초반에 큰 값을 주다가 점차 값을 줄이는데 이는 최적 값에 근접할 시 이동거리를 짧게 해 지나치지 않으려는 것이다
RMSProp	Adagrad는 학습을 진행하다보면 학습률이 너무 빨리 작아지며 0에 도달
Adam	momentum+RMSProp

Receptive field: 해당 노드에 영향을 미치는 노드의 개수들

CNN
- Spare interactions: 좁게 interaction한다
- Parameter sharing: The same kernel is used repeately
- **Equivariance** to transition: convolution(shift(input))=shift(convolution(input))

Pooling
- **Invariant** to small translations of the input: input의 값을 조금 변화시켜도 output 값은 거의 변하지 않음, 특히 max pooling에 대해서

머신러닝: X,Y간 관계 함수, 즉 파라미터를 찾는 것
딥러닝: 연결 가중치인 w의 최적 값을 찾는 것 / =representation learning
딥러닝의 성공 요인: 큰 데이터, GPU, Algorithm

AI>머신러닝-ai를 실현시키는 방법 중 하나 >딥러닝-머신러닝 중 각광받고 있는 방법

ai: 지적활동을 자동화하는 시스템을 만드는 것
symbolic ai> expert systems: 어떤 분야 전문가들의 지식을 규칙화하여 프로그래밍

머신러닝 - data+result >> rule

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its perfomance at tasks in T, as measured by P, improves with experience E >> T에 대한 성능이 향상된다면 (척도 P로 측정), 컴퓨터 프로그램이 learning 한다!

task: transcription(음성인식, OCR)/ Machine translation(input,output도메인이 다름)
StructedOutput(값들사이관계를가지고있을때,번역,transcription,parsing,segmentation 등)/anomaly detection
ex)선형회귀 task: x로부터 y값을 예측/P: MSE/E:MSE를 최소화하는 방향으로 데이터를 관찰함으로써

experience: unsupervised learning/ supervised learning
i부터n까지 곱 p(xi|x1~xn) p(y|x)

engineering oriented: 실험적인 관찰들로 가파르게 발전/ 즉 이전 이론들을 기반으로 자연스럽게 발전된 것이 아니라 이렇게 해봤더니 성능이 좋았고 왜 그런지 결과를 가직 살펴봄

차수: 예시로 회귀에서는 X3라면 M=3

M이 높을수록 모델 복잡도 커짐>overfitting /모델 너무 단순하면 >underfitting 가능성

어느정도 복잡도 큰 모델 중에서 최대한 overfit되지 않도록 하는 장치>> regularization!
(training error가 아닌 generalization error 줄이는 것)
Regularization은 overfitting 문제를 해결하기 위해 cost function에 weight의 합을 더해주는 것을 말합니다. Overfitting 문제를 해결하려면 model을 가능한 simple 하게 해야하는데 cost function에 weights의 합을 더해주고, 이를 미분해서 작아진 값으로 가중치를 업데이트 해 overfitting 문제를 개선

L2: 가중치제곱에 감마 곱한 항을 cost function에 추가
L1: 가중치 절댓값에 감마 곱한 항을 cost function에 추가

training set: to learn the parameters of model
validation set: to choose the hyperparameters of the model
test set: for final evaluation of the generalization error of the model

머신러닝에서 중요한 문제>> “data represetation” inputdata/expectedoutput/Performance measure
즉 representation 방식에 따라 task의 난이도가 완전히 달라지고 성능에 큰 영향

딥러닝: 좋은 representation를 스스로 학습
>>(layered representation들은 neural networks 모델을 통해 학습된다)
to lean all layers of representation jointly

loss 줄이는 방향으로 가중치 갱신>>optimizer의 job
큰 NN를 효과적으로 학습하는 방법 “BackPropagation” - 미분을 이용
kernal methods 중 SVM
- 1) 적당히고차원으로 매핑하여 hyperplane으로 표현 – useful representation>“feature engineering”
- 2) hyperplane & 이와 가장 가까운 데이터 사이 거리를 최대화하는 good decision boundary 찾기
- 장점: 이론 기반이라 해석 용이/ 단점: 큰 데이터 셋은 어려움

새로운 공간을 데이터들을 매핑하고 데이터들간 거리를 계산하는 것 “kernal trick”
kenal trick을 사용하는 기법이 kernel methods

앙상블: 여러 모델을 조합해 결과를 예측하는 방식, 정확도 낮은 모델 여러개
GBM: 다음모델로 넘겨주는 input을 gradient로 삼고 이에 가중치를 부여

XOR 반대 gate >> [1,1,-1]and [-1,-1,1]nand [1,1,0] or
XOR gate >> [1,1,-1]and [-1,-1,1]nand [-1,-1,1] and
[1,1,-1] and [1,1,0] or [1,-2,0] 애매함