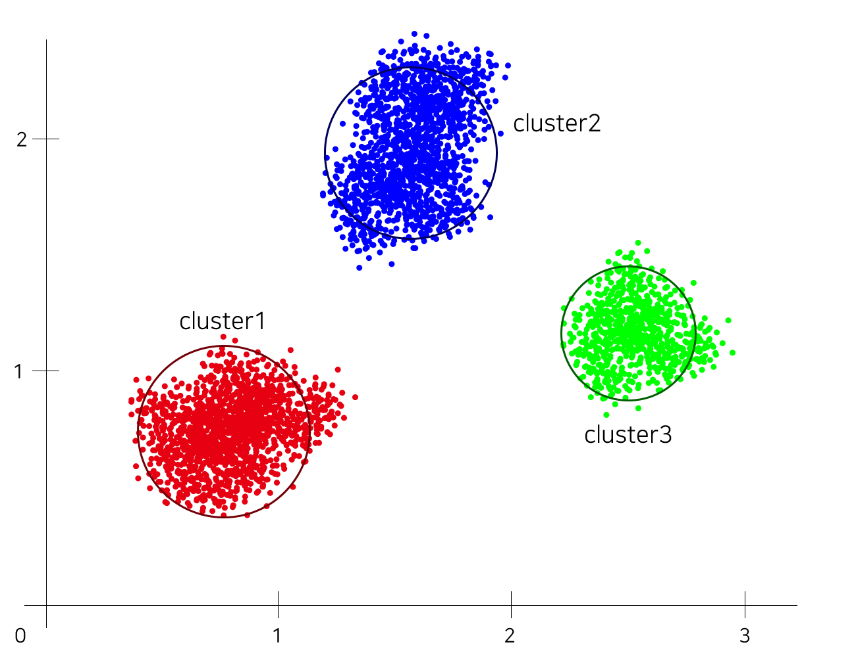
후보 샘플링 (candidate sampling)

* 학습 도중 소프트맥스 등을 사용해서 모든 긍정 레이블에 대한 확률을 계산하는 최적화
* 부정 레이블의 경우 무작위 샘플에 대해서만 계산
* Backpropagation을 위해서 모든 클래스의 softmax 확률 계산이 필요함
* 클래스가 많아질 경우에 softmax loss를 구하는데 많은 연산 비용이 필요해진다.
* 전체 클래스를 계산하는 것을 대신해 일부 카테고리만 랜덤하게 뽑아서 소프트맥스 확률값을 구하고 파라미터를 업데이트
* 방법에는
  + Noise Contrastive Estimation (NCE)
  + Negative Sampling
  + Sampled Logistic
  + Sampled Softmax
* 등이 있다.
* <https://imgur.com/gyeTKAn>
* <https://www.tensorflow.org/extras/candidate_sampling.pdf>

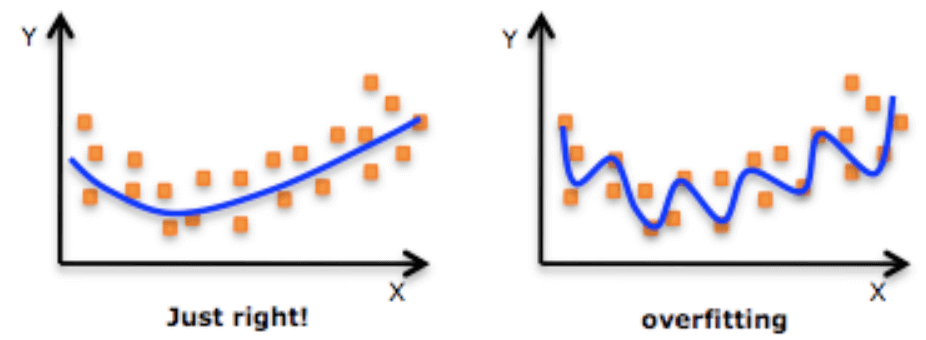
softmax

입력받은 값을 출력으로 0~1 사이의 값으로 모드 정규화해서 출력값들의 총합을 1로 만듬.

클러스터링

* 개체들이 주어졌을 때 이것을 몇 개의 그룹으로 나누는 과정
* 데이터 셋에서 서로 유사한 관찰치들을 그룹으로 묶어 분류하여 군집을 찾아내는 것
* 
* 정답이 없는 비지도 학습이다.
* 군집을 만든 결과가 얼마나 유용한지 군집타당성지표로 평가한다.
  + Davies-Bouldin index
  + Dunn index
  + Silhouette
* 군집간 거리, 군집의 지름, 군집의 분산 등
  + Hierarchical clustering
  + Centroid-based clustering
  + Distribution-based clustering
  + Density-based clustering
  + Grid-based clustering
* 등 여러가지가 있다.

드롭아웃 정규화 (dropout regularization)

* Overfitting을 막기 위한 방법 중 하나
* 
* 
* 훈련할 때 임의의 뉴턴을 골라 삭제하여 신호를 전달하지 않게 함. 테스트할 때에는 모든 뉴런들을 사용한다.
* 무작위로 다른 뉴런들을 생략하면서 반복적으로 학습을 수행한다.
* Voting 효과와 Co-adaptation을 피하는 효과를 가져온다.

Pytorch 사용법 정리

* Import torch 로 pytorch 모듈을 사용하기 위해 임포트로 불러온다.
* 행렬 생성 방법
  + X = torch.empty(a, b) : a\*b의 행렬을 생성한다.
  + X = torch.rand(a, b)
  + X = torch.zeros(a, b, dytpe = torch.long)
  + X = torch.tensor([5, 5, 5])
* 기존의 것을 이용해서 새로 만들기
  + X = X.new\_ones(a, b, dtype = torch.double)
  + X = torch.randn\_like(X, dtype = torch.float)
* 행렬 계산 행렬 X, 행렬 Y
  + X = X+Y
  + X = torch.add(X, Y)
  + torch.add(X, Y, out = X)
  + X.add\_(Y) : in-place 방법으로 값을 변경한다.
* a = X.view(a, b): 텐서의 크기를 변경한다.
* Numpy
  + a = torch.ones(5)
  + b = a.numpy()
  + b = torch.from\_numpy(a)
* Tensor
  + Device = torch.device(“cuda”)
  + X = torch.ones\_like(x, device = device)
  + X = X.to(device)