EBGAN(Energy-Based GAN)

Data manifold: $Y_2 = Y_1^2$

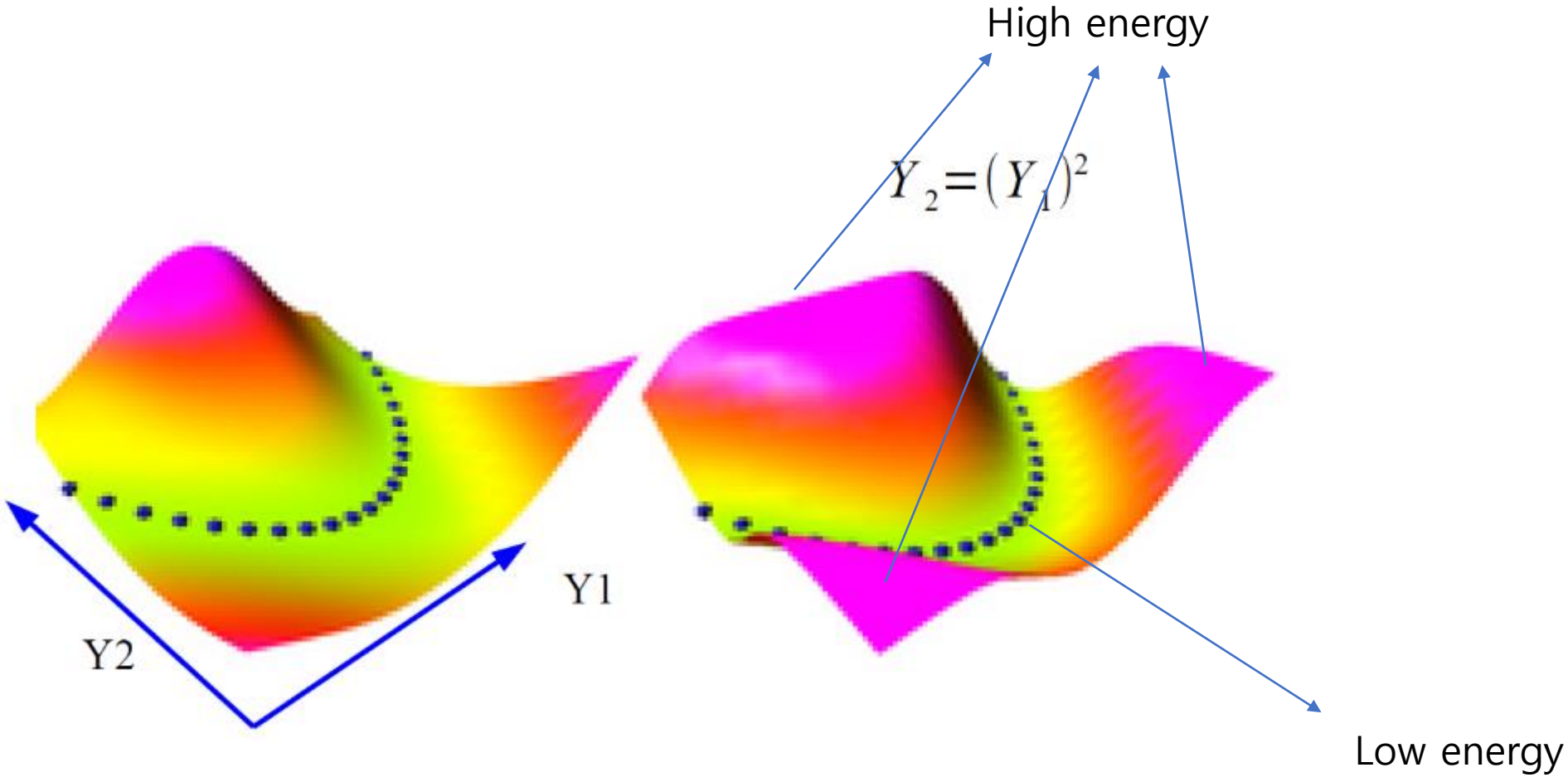
Data manifold 부분에만 낮은 에너지를 주도록 해야함

Supervised learning: (X,Y)가 주어질 때 Y가 정답이면 이 쌍의 energy에 낮은 값을 줌

Unsupervised learning: Data manifold 부분에만 낮은 Energy를 주도록 X를 잘 Clustering해야함

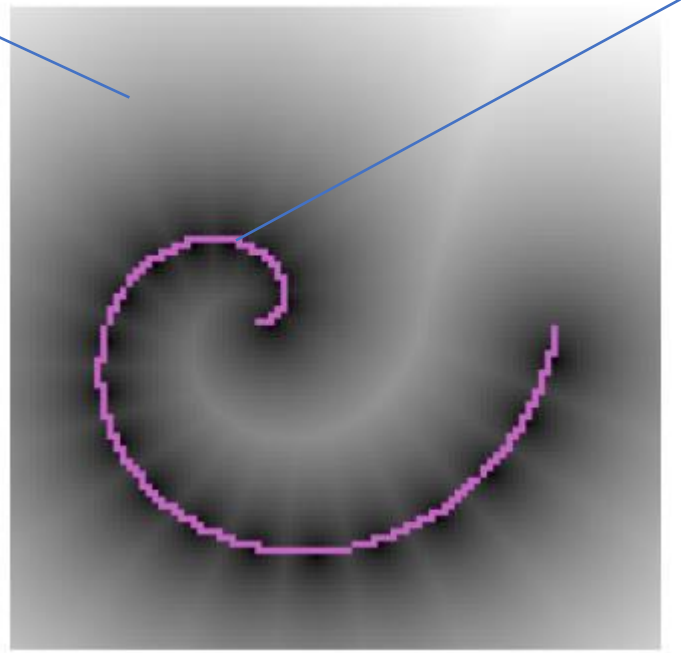
Unsupervised learning: Data manifold 부분에만 낮은 Energy를 주도록 X를 잘 Clustering해야함

→ How?



High energy
Contrative sample

Low energy
(more stable)



Spiral data

Seven Strategies to Shape the Energy Function

- 1. energy를 가질 수 있는 공간을 한정시킴 (k-means)
- 2. parametric 모델 사용하여 샘플에서 energy는 낮추고 나머지는 올린다.
- 3. 샘플의 energy 낮추고, 나머지 중 일부만 올린다.
- 4. 샘플 근방의 곡률은 maximize 시키고 gradient는 minimize
- 5. dynamical system을 학습시켜 data manifold로 수렴시킴
- 6. Regularizer를 사용해서 낮은 energy를 가질 수 있는 공간을 한정짓는다
- 7. ~~~

3. 샘플의 energy 낮추고, 나머지 중 일부만 올린다.

- -> GAN

- GAN은 이를 parametric model을 바탕으로 contrastive data를 생성하는 방법인 것이죠. 즉, GAN 학습과정에서 generator가 하는 역할을 **data manifold 밖에 속하는 데이터(contrastive data, e.g. 매우 이상한 사람 사진)**를 생성하는 것으로, 그리고 discriminator가 샘플에 energy를 할당하는 energy function 역할을 하는 것으로 생각할 수 있습니다.
- 학습이 진행됨에 따라 generator가 점차 data manifold에 가까운 샘플들을 생성하기 때문에 data manifold에서 멀리 있는 contrastive data들부터 생성해나가며 전체 energy surface를 구성할 수 있게 됩니다. 마찬가지로 discriminator 역시 data manifold에서 먼 곳의 data가 없어도 generator가 생성하는 sample로부터 이에 대한 정보를 자연스럽게 얻어내여 좋은 energy function을 학습할 수 있게 되구요.

margin loss

Int값 margin

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]^+$$

(m일 때 최소)

$$\mathcal{L}_G(z) = D(G(z))$$

$$[\cdot]^+ = max(0, \cdot).$$

```
def netD(input_images, batch_size, reuse=False):
    encoded = encoder(input_images, reuse=reuse)
    decoded = decoder(encoded, reuse=reuse)
    return mse(decoded, input_images, batch_size), encoded, decoded
```

Pullaway loss

```
pt_loss = pullaway_loss(embeddings_fake, BATCH_SIZE)
if PULLAWAY == 1:
    print 'Using pullaway'
    errG = errD_fake + 0.1*pt_loss
else:
    print 'Not using pullaway'
    errG = errD_fake
```

Net D의 encoded가 들어감(D_fake == 0에 가까움)

내생각: PULLAWAY == 1: 은 netD가 가짜 이미지를 진짜로 진짜 이미지를 가짜로 인식하는 경우를 의미 하는 듯, errG를 높여주어 더 많이 minimize하게 만듬(0에 가까운 D_fake에 margin을 주고 0.1을 곱해서 사용)

Collapse를 막기위한 reularizer

```
def pullaway_loss(embeddings, batch_size):
    norm = tf.sqrt(tf.reduce_sum(tf.square(embeddings), 1, keep_dims=True))
# 유클리디안 거리
    normalized_embeddings = embeddings / norm
    similarity = tf.matmul(normalized_embeddings, normalized_embeddings, transpose_b=True)
    pt_loss = (tf.reduce_sum(similarity) - batch_size) / (batch_size * (batch_size - 1))
    return pt_loss
```

유클리디안 거리의 + normalization + similarity