Learning deep representations by mutual information estimation and maximization

January 22, 2020

How to estimate mutual information (MI) with neural networks?

How to estimate mutual information (MI) with neural networks?

- Imagine we have two random variables X and $Y = E(X)$, where Y is obtained by encoding X via $E(.)$.

How to estimate mutual information (MI) with neural networks?

- Imagine we have two random variables X and $Y = E(X)$, where Y is obtained by encoding X via $E(.)$.
- By Donsker-Varadhan inequality for any function f we have:

$$I(X;Y) \triangleq \mathrm{D}_{\mathrm{KL}}[p(x,y) \parallel p(x)p(y)] \geq \mathop{\mathbb{E}}_{p(x,y)}[f(x,y)] - \log \mathop{\mathbb{E}}_{p(x)p(y)}\left[e^{f(x,y)}\right]$$

How to estimate mutual information (MI) with neural networks?

- Imagine we have two random variables X and $Y = E(X)$, where Y is obtained by encoding X via $E(.)$.
- By Donsker-Varadhan inequality for any function f we have:

$$I(X;Y) \triangleq \mathrm{D}_{\mathrm{KL}}[p(x,y) \parallel p(x)p(y)] \geq \underset{p(x,y)}{\mathbb{E}}[f(x,y)] - \log \underset{p(x)p(y)}{\mathbb{E}}\left[e^{f(x,y)}\right]$$

- This looks like a discriminator!

How to estimate mutual information (MI) with neural networks?

- Imagine we have two random variables X and $Y = E(X)$, where Y is obtained by encoding X via $E(.)$.
- By Donsker-Varadhan inequality for any function f we have:

$$I(X;Y) \triangleq \mathrm{D}_{\mathrm{KL}}[p(x,y) \parallel p(x)p(y)] \geq \underset{p(x,y)}{\mathbb{E}}[f(x,y)] - \log \underset{p(x)p(y)}{\mathbb{E}}\left[e^{f(x,y)}\right]$$

- This looks like a discriminator!
- Let's train a discriminator T_ω to maximize:

$$\underset{p(x,y)}{\mathbb{E}}[T_\omega(x,y)] - \log \underset{p(x)p(y)}{\mathbb{E}}\left[e^{T_\omega(x,y)}\right]$$

How to estimate mutual information (MI) with neural networks?

- Imagine we have two random variables X and $Y = E(X)$, where Y is obtained by encoding X via $E(.)$.
- By Donsker-Varadhan inequality for any function f we have:

$$I(X;Y) \triangleq \mathrm{D}_{\mathrm{KL}}[p(x,y) \parallel p(x)p(y)] \geq \underset{p(x,y)}{\mathbb{E}}[f(x,y)] - \log \underset{p(x)p(y)}{\mathbb{E}}\left[e^{f(x,y)}\right]$$

- This looks like a discriminator!
- Let's train a discriminator T_ω to maximize:

$$\underset{p(x,y)}{\mathbb{E}}[T_\omega(x,y)] - \log \underset{p(x)p(y)}{\mathbb{E}}\left[e^{T_\omega(x,y)}\right]$$

- I.e. T_ω is trained to distinguish between paired examples $(x, E(x))$ and unpaired examples (x, y') (i.e. $y' \neq E(x)$).

How to estimate mutual information (MI) with neural networks?

- Imagine we have two random variables X and $Y = E(X)$, where Y is obtained by encoding X via $E(.)$.
- By Donsker-Varadhan inequality for any function f we have:

$$I(X;Y) \triangleq \mathrm{D_{KL}}[p(x,y) \parallel p(x)p(y)] \geq \mathop{\mathbb{E}}_{p(x,y)}[f(x,y)] - \log \mathop{\mathbb{E}}_{p(x)p(y)}\left[e^{f(x,y)}\right]$$

- This looks like a discriminator!
- Let's train a discriminator T_ω to maximize:

$$\mathop{\mathbb{E}}_{p(x,y)}[T_\omega(x,y)] - \log \mathop{\mathbb{E}}_{p(x)p(y)}\left[e^{T_\omega(x,y)}\right]$$

- I.e. T_ω is trained to distinguish between paired examples $(x, E(x))$ and unpaired examples (x, y') (i.e. $y' \neq E(x)$).
- If we did a good job in training T_ω then we'll have a good MI estimate between X and Y.

How to *maximize* MI with neural networks?

How to *maximize* MI with neural networks?

- Just use your discrminator T_ω to train an encoder E_ψ!

How to *maximize* MI with neural networks?

- Just use your discrminator T_ω to train an encoder E_ψ!
- Updating an encoder will force use to retrain a discriminator T_ω

How to *maximize* MI with neural networks?

- Just use your discrminator T_ω to train an encoder E_ψ!
- Updating an encoder will force use to retrain a discriminator T_ω
- Then let's train them in parallel, just like a GAN model!

How to *maximize* MI with neural networks?

- Just use your discrminator T_ω to train an encoder E_ψ!
- Updating an encoder will force use to retrain a discriminator T_ω
- Then let's train them in parallel, just like a GAN model!
- It's exactly like a GAN model, but encoder E_ψ *helps* discriminator T_ω instead of spoofing it since both try hard to increase the lower bound:

How to *maximize* MI with neural networks?

- Just use your discrminator T_ω to train an encoder E_ψ!
- Updating an encoder will force use to retrain a discriminator T_ω
- Then let's train them in parallel, just like a GAN model!
- It's exactly like a GAN model, but encoder E_ψ *helps* discriminator T_ω instead of spoofing it since both try hard to increase the lower bound:
  - T_ω tries to increase LB to provide an accurate MI estimate.

How to *maximize* MI with neural networks?

- Just use your discrminator T_ω to train an encoder E_ψ!
- Updating an encoder will force use to retrain a discriminator T_ω
- Then let's train them in parallel, just like a GAN model!
- It's exactly like a GAN model, but encoder E_ψ *helps* discriminator T_ω instead of spoofing it since both try hard to increase the lower bound:
  - T_ω tries to increase LB to provide an accurate MI estimate.
  - E_ψ tries to increase it because it will increase the MI

Global MI vs Local MI

Global MI vs Local MI

- Imagine we have a dataset of images $x_1, ..., x_n$ and an encoder $y = E_\psi(x)$.

Global MI vs Local MI

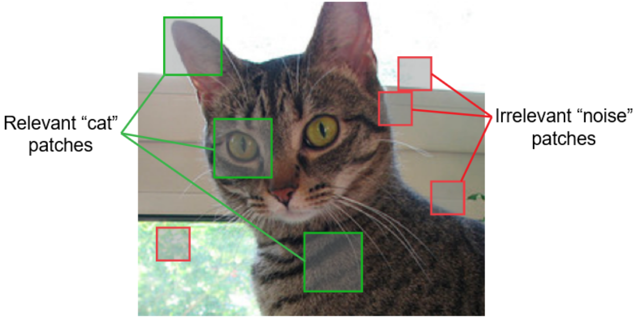
- Imagine we have a dataset of images $x_1, ..., x_n$ and an encoder $y = E_\psi(x)$.
- Is it a reasonable objective to maximize $I(X, Y)$, i.e. MI between *the whole* input and *the whole* output?

Global MI vs Local MI

- Imagine we have a dataset of images $x_1, ..., x_n$ and an encoder $y = E_\psi(x)$.
- Is it a reasonable objective to maximize $I(X, Y)$, i.e. MI between *the whole* input and *the whole* output?
- No! Because a lot of information is irrelevant (for classification):

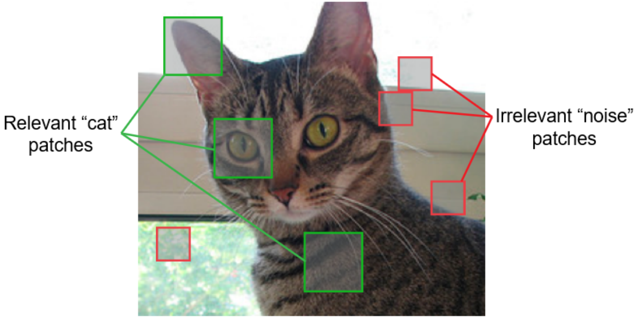
Global MI vs Local MI

- Imagine we have a dataset of images $x_1, ..., x_n$ and an encoder $y = E_\psi(x)$.
- Is it a reasonable objective to maximize $I(X, Y)$, i.e. MI between *the whole* input and *the whole* output?
- No! Because a lot of information is irrelevant (for classification):



Global MI vs Local MI

- Imagine we have a dataset of images $x_1, \ldots, x_n$ and an encoder $y = E_\psi(x)$.
- Is it a reasonable objective to maximize $I(X, Y)$, i.e. MI between *the whole* input and *the whole* output?
- No! Because a lot of information is irrelevant (for classification):



- Let's maximize MI between local patches and the output then!

Deep InfoMax with *global* objective

Deep InfoMax with *global* objective

- Pre-encode an image into 3d-tensor $c = C_\psi(x)$ of "local" feature vectors.

Deep InfoMax with *global* objective

- Pre-encode an image into 3d-tensor $c = C_\psi(x)$ of "local" feature vectors.
- Compute a "global" feature vector $y = E_\psi(c)$ (by avg-pooling followed by MLP, for example).

Deep InfoMax with *global* objective

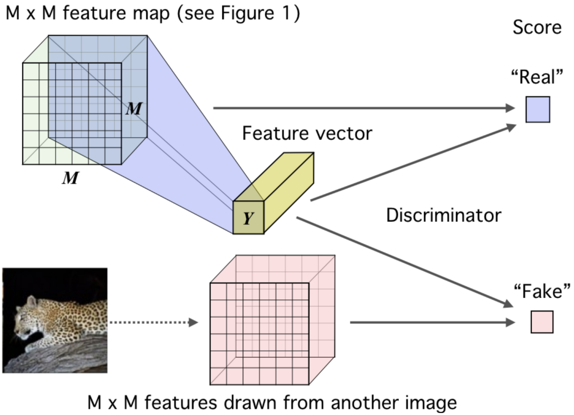
- Pre-encode an image into 3d-tensor $c = C_\psi(x)$ of "local" feature vectors.
- Compute a "global" feature vector $y = E_\psi(c)$ (by avg-pooling followed by MLP, for example).
- Train T_ω, C_ψ, E_ψ to maximize $I(c, y)$.

Deep InfoMax with *global* objective

- Pre-encode an image into 3d-tensor $c = C_\psi(x)$ of "local" feature vectors.
- Compute a "global" feature vector $y = E_\psi(c)$ (by avg-pooling followed by MLP, for example).
- Train T_ω, C_ψ, E_ψ to maximize $I(c, y)$.
- I.e. our T_ω takes c and y as inputs and outputs a scalar score.

Deep InfoMax with *global* objective

- Pre-encode an image into 3d-tensor $c = C_\psi(x)$ of "local" feature vectors.
- Compute a "global" feature vector $y = E_\psi(c)$ (by avg-pooling followed by MLP, for example).
- Train T_ω, C_ψ, E_ψ to maximize $I(c, y)$.
- I.e. our T_ω takes c and y as inputs and outputs a scalar score.



Deep InfoMax with *local* objective

Deep InfoMax with *local* objective

- Do the same as for the global objective, but split c into local features and maximize:

$$\frac{1}{M^2} \sum_{i,j} I(c_{i,j}(x), y)$$

Deep InfoMax with *local* objective

- Do the same as for the global objective, but split c into local features and maximize:

$$\frac{1}{M^2} \sum_{i,j} I(c_{i,j}(x), y)$$

- I.e. we maximize the MIs between local patches and a global vector.

Deep InfoMax with *local* objective

- Do the same as for the global objective, but split c into local features and maximize:

$$\frac{1}{M^2} \sum_{i,j} I(c_{i,j}(x), y)$$

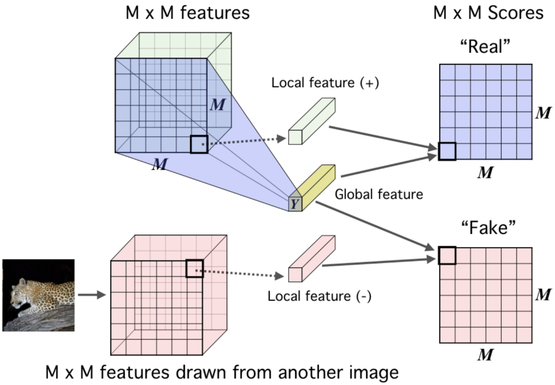
- I.e. we maximize the MIs between local patches and a global vector.
- This forces the model to encode useful global information.

Deep InfoMax with *local* objective

- Do the same as for the global objective, but split c into local features and maximize:

$$\frac{1}{M^2}\sum_{i,j} I(c_{i,j}(x), y)$$

- I.e. we maximize the MIs between local patches and a global vector.
- This forces the model to encode useful global information.



Some details and thoughts

Some details and thoughts

- In practice we combine both the global and local objectives during training.

Some details and thoughts

- In practice we combine both the global and local objectives during training.
- We also need to regularize the representations to be similar to some simple prior distribution.

Some details and thoughts

- In practice we combine both the global and local objectives during training.
- We also need to regularize the representations to be similar to some simple prior distribution.
- They conduct **a ton of** experiments on different datasets and for different downstream tasks demonstraing SotA results.

Some details and thoughts

- In practice we combine both the global and local objectives during training.
- We also need to regularize the representations to be similar to some simple prior distribution.
- They conduct **a ton of** experiments on different datasets and for different downstream tasks demonstraing SotA results.
- There are some different variants on how to reformulate the LB to make the training more stable.

Some details and thoughts

- In practice we combine both the global and local objectives during training.
- We also need to regularize the representations to be similar to some simple prior distribution.
- They conduct **a ton of** experiments on different datasets and for different downstream tasks demonstraing SotA results.
- There are some different variants on how to reformulate the LB to make the training more stable.
- There are some gritty details on how to do negative sampling properly.

Some details and thoughts

- In practice we combine both the global and local objectives during training.
- We also need to regularize the representations to be similar to some simple prior distribution.
- They conduct **a ton of** experiments on different datasets and for different downstream tasks demonstraing SotA results.
- There are some different variants on how to reformulate the LB to make the training more stable.
- There are some gritty details on how to do negative sampling properly.
- The paper is written quite ambigously and a lot of important details are scattered all over the manuscript... (i.e. how we do summarization of c into y, what prior do we use, etc)