Adversarial Training

M. Saiful Bari

November 26, 2017

Table of Contents

Types of Probabilistic Model

There are two type of probabilistic model.

Types of Probabilistic Model

There are two type of probabilistic model.

- Generative Model.

Types of Probabilistic Model

There are two type of probabilistic model.

- Generative Model.
- Discriminative Model.

Generative Model

1. Generative Model

Generative Model

1. Generative Model

- Uses Bayesian Statistics.

Generative Model

1. Generative Model

- Uses Bayesian Statistics.
- Learn Joint Probability Distribution $p(y, \mathbf{x})$ applying bayes rule we calculate $p(y|\mathbf{x})$.

Generative Model

1. Generative Model

- Uses Bayesian Statistics.
- Learn Joint Probability Distribution $p(y, \mathbf{x})$ applying bayes rule we calculate $p(y|\mathbf{x})$.
- Calculates complete probability model performing inference condition on the data and model.

Generative Model

1. Generative Model

- Uses Bayesian Statistics.
- Learn Joint Probability Distribution $p(y, \mathbf{x})$ applying bayes rule we calculate $p(y|\mathbf{x})$.
- Calculates complete probability model performing inference condition on the data and model.
- Works better only with large amount of data.

Generative Model

1. Generative Model

- Uses Bayesian Statistics.
- Learn Joint Probability Distribution $p(y, \mathbf{x})$ applying bayes rule we calculate $p(y|\mathbf{x})$.
- Calculates complete probability model performing inference condition on the data and model.
- Works better only with large amount of data.
- Generally used in unsupervised learning.

Generative Model

1. Generative Model

- Uses Bayesian Statistics.
- Learn Joint Probability Distribution $p(y, \mathbf{x})$ applying bayes rule we calculate $p(y|\mathbf{x})$.
- Calculates complete probability model performing inference condition on the data and model.
- Works better only with large amount of data.
- Generally used in unsupervised learning.
- Gives the internal belief of data.

Generative Model

1. Generative Model

- Uses Bayesian Statistics.
- Learn Joint Probability Distribution $p(y, \mathbf{x})$ applying bayes rule we calculate $p(y|\mathbf{x})$.
- Calculates complete probability model performing inference condition on the data and model.
- Works better only with large amount of data.
- Generally used in unsupervised learning.
- Gives the internal belief of data.
- It specifies how to generate the data using the class-conditional density $p(\mathbf{x}|y = c)$ and the class prior $p(y = c)$.

Discriminative Model

2. Discriminative Model

Discriminative Model

2. Discriminative Model

- Learn Conditional Probability distribution $p(y|\mathbf{x})$ directly.

Discriminative Model

2. Discriminative Model

- Learn Conditional Probability distribution $p(y|\mathbf{x})$ directly.
- It's elegancy is Simplicity.

Discriminative Model

2. Discriminative Model

- Learn Conditional Probability distribution $p(y|\mathbf{x})$ directly.
- It's elegancy is Simplicity.
- Compared to Generative approach, most of the time discriminative model can learn from small amount of data.

Example : Generative Model

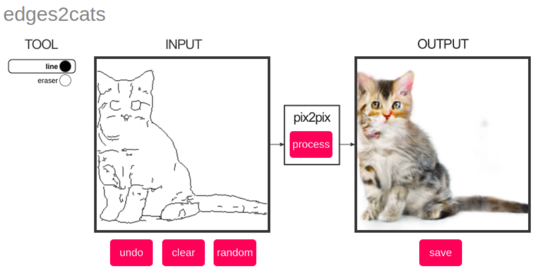

Figure 1: Edges are drawn and the model draws the similar shaped cat.

Live Example

Example : Generative Model

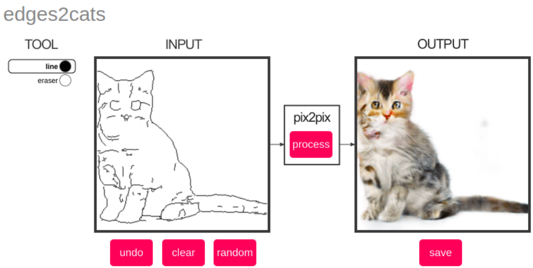

Figure 1: Edges are drawn and the model draws the similar shaped cat.

Live Example

Without internal belief this type of model can not be made.

Example : Discriminative Model

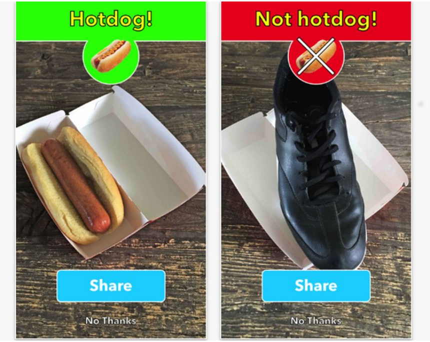

Figure 2: Two pictures are given and the model just comment it's hotdog or not.

Live Example

Summary

Summary

A discriminative model doesn't care about how data is generated, it's just **classify** the data.

Summary

A discriminative model doesn't care about how data is generated, it's just **classify** the data.

A generative model ask the question, which category most likely to **generate** the data.

Minimax Game : Adversary relation

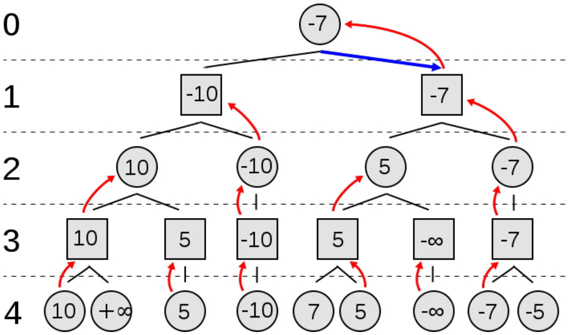

Figure 3: Minimax algorithm.Circles represent the moves of the player running the algorithm (maximizing player) and squares represent the moves of the opponent (minimizing player). The values inside the circles and squares represent the value of the minimax algorithm. The red arrows represent the chosen move, the numbers on the left represent the tree depth and the blue arrow the chosen move.

Generative Adversarial Nets

Generative Adversarial Nets- NIPS'14

Goodfellow et al.

Estimate generative models via an **adversarial process**, by **simultaneously** training a generative and a discriminative model. (paper link)

Generative Adversarial Nets

Generative Adversarial Nets- NIPS'14

Goodfellow et al.

Estimate generative models via an **adversarial process**, by **simultaneously** training a generative and a discriminative model. (paper link)

Experiments: Generate image form MNIST, TFD and CIFAR-10 dataset.

Generative Adversarial Nets

Generative Adversarial Nets- NIPS'14

Goodfellow et al.

Estimate generative models via an **adversarial process**, by **simultaneously** training a generative and a discriminative model. (paper link)

Experiments: Generate image form MNIST, TFD and CIFAR-10 dataset.

- Generative model (**G**) captures the data distribution.

Generative Adversarial Nets

Generative Adversarial Nets- NIPS'14

Goodfellow et al.

Estimate generative models via an **adversarial process**, by **simultaneously** training a generative and a discriminative model. (paper link)

Experiments: Generate image form MNIST, TFD and CIFAR-10 dataset.

- Generative model (**G**) captures the data distribution.
- Discriminative model (**D**) that estimates the probability that a **sample** came from the **training data** rather than **G**.

Generative Adversarial Nets

Generative Adversarial Nets- NIPS'14

Goodfellow et al.

Estimate generative models via an **adversarial process**, by **simultaneously** training a generative and a discriminative model. (paper link)

Experiments: Generate image form MNIST, TFD and CIFAR-10 dataset.

- Generative model (**G**) captures the data distribution.
- Discriminative model (**D**) that estimates the probability that a **sample** came from the **training data** rather than **G**.

Adversary relation are between G and D

Adversary relation

$$\textbf{Generative model } (G) \Longleftrightarrow \textbf{Counterfeiter}$$

$$\textbf{Discriminative model } (D) \Longleftrightarrow \textbf{Police}$$

Adversary relation

$$\textbf{Generative model } (G) \Longleftrightarrow \textbf{Counterfeiter}$$
$$\textbf{Discriminative model } (D) \Longleftrightarrow \textbf{Police}$$

G tries to produce something **fake**, **D** tries to **Counterfeit**.

Adversary relation

Generative model $(G) \Longleftrightarrow$ **Counterfeiter**

Discriminative model $(D) \Longleftrightarrow$ **Police**

G tries to produce something **fake**, **D** tries to **Counterfeit**.

How to shape product of G as close as original (sample)

by **Training**

Framework

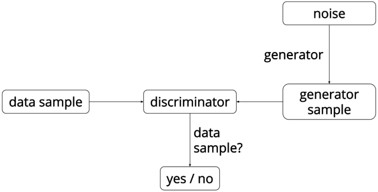

Figure 4: A graphical view of the GAN network

- A sample setup (MLP vs MLP)
  - Generative model passing sample noise through a MLP.
  - Discriminative model is also a MLP.

Framework

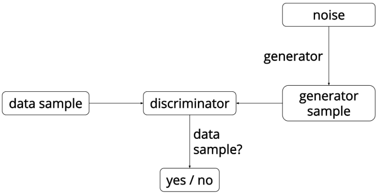

Figure 4: A graphical view of the GAN network

- A sample setup (MLP vs MLP)
  - Generative model passing sample noise through a MLP.
  - Discriminative model is also a MLP.
- Design model as you need. (ConvNet vs ConvNet etc.)

Framework

Some variables related to the model,

- $x =$ input data
- $p_g =$ generator (**G**) 's distribution
- $p_z(z) =$ prior on input noise variable
- $G(z; \theta_g) =$ a differentiable function represented by a MLP (generator) with parameter θ_g. (used for mapping to data space.)
- $D(x; \theta_d) =$ a MLP for discriminator.

Training

We train $\mathbf{D}$ to **maximize** the probability of assigning the correct label to both training examples and **fake generated** samples from generator $\mathbf{G}$.

Training

We train $\mathbf{D}$ to **maximize** the probability of assigning the correct label to both training examples and **fake generated** samples from generator $\mathbf{G}$.

simultaneously

Training

We train $\mathbf{D}$ to **maximize** the probability of assigning the correct label to both training examples and **fake generated** samples from generator $\mathbf{G}$.

simultaneously

We train $\mathbf{G}$ to **minimize** $log(1 - D(G(z))$

minimax Equation

$$\underset{G}{min}\,\underset{D}{max}\,V(D,G) = \mathbb{E}_{x \backsim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log(1 - D(G(z)))]$$

D and G play the two-player minimax game with value/loss function $V(G, D)$

Theoretical analysis

$$\underset{G}{min}\ \underset{D}{max}\ V(D,G) = \mathbb{E}_{x \backsim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log(1 - D(G(z))]$$

- The **training criterion** allows one to recover the data generating distribution as G.

Theoretical analysis

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \backsim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log(1 - D(G(z)))]$$

- The **training criterion** allows one to recover the data generating distribution as G.
- **minimax Equation** may not provide sufficient gradient for G to learn well.

Theoretical analysis

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \backsim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log(1 - D(G(z)))]$$

- The **training criterion** allows one to recover the data generating distribution as G.
- **minimax Equation** may not provide sufficient gradient for G to learn well.
- Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data saturating $\log(1 - D(G(z)))$.

Theoretical analysis

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \backsim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log(1 - D(G(z)))]$$

- The **training criterion** allows one to recover the data generating distribution as G.
- **minimax Equation** may not provide sufficient gradient for G to learn well.
- Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data saturating $\log(1 - D(G(z))$.
- Rather than training G to minimize $\log(1 - D(G(z))$ we can train G to maximize $\log D(G(z))$.

Notes for Implementing the Algorithm

- We must implement the game using an **iterative**, numerical approach.

Notes for Implementing the Algorithm

- We must implement the game using an **iterative**, numerical approach.
- Optimizing D to completion in the inner loop of training is computationally prohibitive.

Notes for Implementing the Algorithm

- We must implement the game using an **iterative**, numerical approach.
- Optimizing D to completion in the inner loop of training is computationally prohibitive.
- Finite datasets would result in overfitting.

Notes for Implementing the Algorithm

- We must implement the game using an **iterative**, numerical approach.
- Optimizing D to completion in the inner loop of training is computationally prohibitive.
- Finite datasets would result in overfitting.
- Instead, we alternate between $\mathbf{k}$ steps of optimizing $\mathbf{D}$ and one step of optimizing $\mathbf{G}$.

Notes for Implementing the Algorithm

- We must implement the game using an **iterative**, numerical approach.
- Optimizing D to completion in the inner loop of training is computationally prohibitive.
- Finite datasets would result in overfitting.
- Instead, we alternate between $\mathbf{k}$ steps of optimizing $\mathbf{D}$ and one step of optimizing $\mathbf{G}$.
- This results in D being maintained near its optimal solution, so long as G changes slowly enough.

Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 5: Algorithm for training Adversarial Nets

Important notes on Algorithm

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z_i)))]$$

- The gradient ascent expression for the discriminator.
- the ∇_{θ_d} denotes the gradient of the discriminator.
- m is the number of samples in a batch.
- The first term corresponds to optimizing the probability that real data is rated highly.
- The second term corresponds to optimizing the probability that the generated data $G(z)$ is rated poorly
- Notice we apply the gradient to the discriminator, not the generator.

Important notes on Algorithm

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z_i)))$$

- The gradient descent expression for the generator.
- the ∇_{θ_g} denotes the gradient of the generator.
- previous $\mathbf{k}$ steps are for optimizing discriminator.
- after $\mathbf{k}$ steps, we are using one steps for optimizing generator.
- The second term corresponds to optimizing the probability that the generated data $\mathbf{G}(\mathbf{z})$ is rated highly.
- Notice we apply the gradient to the generator, not the discriminator.

A sample GAN Training

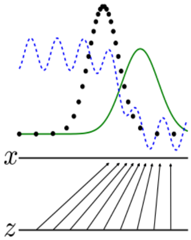

Figure 6: A sample GAN training.

Consider this is an adversarial pair near convergence.

A sample GAN Training

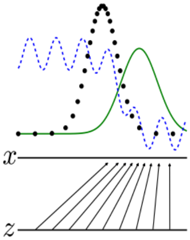

Figure 7: A sample GAN training.

Blue dashed line is discriminative distribution D.

A sample GAN Training

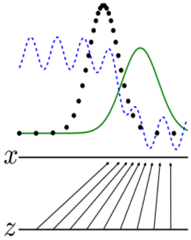

Figure 8: A sample GAN training.

Black dotted line is the real data x.(p_x is distribution of real data).

A sample GAN Training

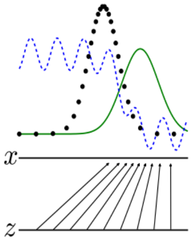

Figure 9: A sample GAN training.

Green line is the line generated by the generator G (p_g is distribution of generated data).

A sample GAN Training

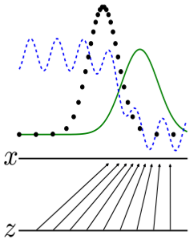

Figure 10: A sample GAN training.

Notice how green line separated the blue and black dotted distribution.

A sample GAN Training

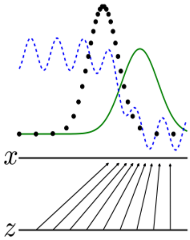

Figure 11: A sample GAN training.

The lower horizontal line is the domain from which z is sampled.

A sample GAN Training

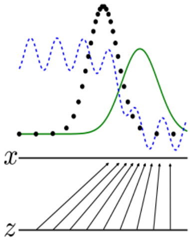

Figure 12: A sample GAN training.

The horizontal line above is part of the domain of x.

A sample GAN Training

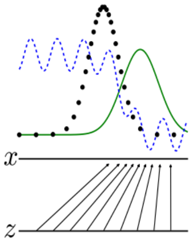

Figure 13: A sample GAN training.

The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution p_g on transformed samples. (transformed by who ??? remember **MLP** with generator input.)

A sample GAN Training

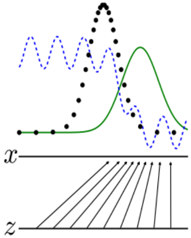

Figure 14: A sample GAN training.

Consider this is an adversarial pair near convergence: p_g is similar to p_{data} and D is a **partially accurate classifier**.

A sample GAN Training

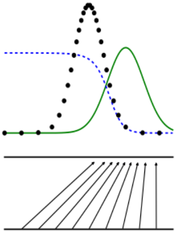

Figure 15: A sample GAN training.

In the inner loop of the algorithm, D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$.

A sample GAN Training

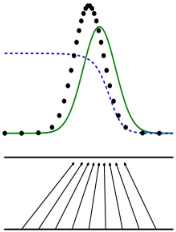

Figure 16: A sample GAN training.

After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data.

A sample GAN Training

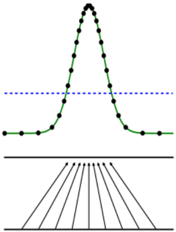

Figure 17: A sample GAN training.

After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. (overfitting ???)

A sample GAN Training

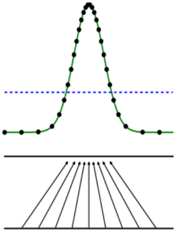

Figure 18: A sample GAN training.

The discriminator is unable to differentiate between the two distributions, $D(x) = \frac{1}{2}$ (equal probability to identify a sample as fake or real).

A sample GAN Training

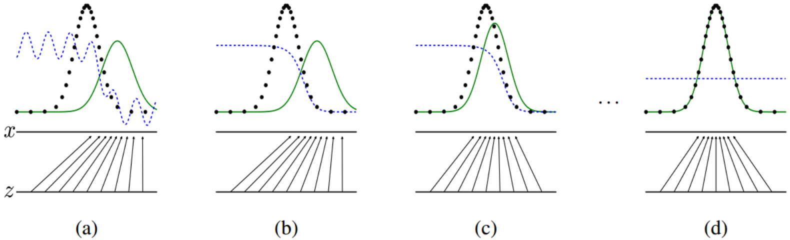

Figure 19: full picture of the training

Enough about GAN ... ?

so many GAN papers...

The Big Picture

using an **adversary model** (MLP,CNN,RNN etc) participating in **simultanious**/**collaborative** learning with **original model** (the task) is really a good idea to **controle** the learning procedure.

The Big Picture

using an **adversary model** (MLP,CNN,RNN etc) participating in **simultanious**/**collaborative** learning with **original model** (the task) is really a good idea to **controle** the learning procedure.

In this paper we use adversary to make noise become something we already have.

The Big Picture

using an **adversary model** (MLP,CNN,RNN etc) participating in **simultanious**/**collaborative** learning with **original model** (the task) is really a good idea to **controle** the learning procedure.

In this paper we use adversary to make noise become something we already have.

New Challenge

what if we can use an adversary model to stop generating some features while learning.

New Challenge

what if we can use an adversary model to stop generating some features while learning.

the **advarsay objective** will be,
somehow pass the information through **gradient update** to **maximize loss** for a **targeted feature** without interfering the **other features** to **learn** by **minimizing loss**
or
vice versa.

Domain Adaption

A common information could be in different distribution.

Domain Adaption

A common information could be in different distribution.

- Different language text with similar meaning.

Domain Adaption

A common information could be in different distribution.

- Different language text with similar meaning. Each language is a domain.

Domain Adaption

A common information could be in different distribution.

- Different language text with similar meaning. Each language is a domain.
- Positive negative review of a book and movie.

Domain Adaption

A common information could be in different distribution.

- Different language text with similar meaning. Each language is a domain.
- Positive negative review of a book and movie. Movie and Book are two different domain.

Domain Adaption

A common information could be in different distribution.

- Different language text with similar meaning. Each language is a domain.
- Positive negative review of a book and movie. Movie and Book are two different domain.

Predictions must be made based on features that cannot discriminate between the **training** (source) and **test** (target) domains.

Domain Adaption

A common information could be in different distribution.

- Different language text with similar meaning. Each language is a domain.
- Positive negative review of a book and movie. Movie and Book are two different domain.

Predictions must be made based on features that cannot discriminate between the **training** (source) and **test** (target) domains.

Domain Adaption

To extract the features form different distributions such that the features become **domain invarient**.

Domain Adaptation Influential Publication

Domain-Adversarial Training of Neural Networks-JMLR'16

Ganin et al.

A learning approach for **domain adaptation**, in which data at training and test time come from similar but different distributions. (paper link)

Domain Adaptation Influential Publication

Domain-Adversarial Training of Neural Networks-JMLR'16

Ganin et al.

A learning approach for **domain adaptation**, in which data at training and test time come from similar but different distributions. (paper link)

Experiment :

Document sentiment analysis (on book review and movie review) and image classification (inter-twining moons 2D problem)

Domain Adaptation Influential Publication

Domain-Adversarial Training of Neural Networks-JMLR'16

Ganin et al.

A learning approach for **domain adaptation**, in which data at training and test time come from similar but different distributions. (paper link)

Experiment :

Document sentiment analysis (on book review and movie review) and image classification (inter-twining moons 2D problem)

Proposed model : Domain-Adversarial Neural Network - DANN

General overview Ganin et al.

- The proposed approaches build mappings between the **source** (training-time) and the **target** (test-time).

General overview Ganin et al.

- The proposed approaches build mappings between the **source** (training-time) and the **target** (test-time).
- The **classifier** learned for the source domain, how it can also be applied to the target domain domains.

General overview Ganin et al.

- The proposed approaches build mappings between the **source** (training-time) and the **target** (test-time).
- The **classifier** learned for the source domain, how it can also be applied to the target domain domains.
- The **target** domain data are fully **unlabeled** (unsupervised domain annotation).

General overview Ganin et al.

- The proposed approaches build mappings between the **source** (training-time) and the **target** (test-time).
- The **classifier** learned for the source domain, how it can also be applied to the target domain domains.
- The **target** domain data are fully **unlabeled** (unsupervised domain annotation).
- focus on learning features,
  - Discriminativeness
  - Domain invariance

General overview Ganin et al.

- The model jointly optimizing the underlying features by adversary relation.

General overview Ganin et al.

- The model jointly optimizing the underlying features by adversary relation.
  - **The label predictor** : predicts class labels.

General overview Ganin et al.

- The model jointly optimizing the underlying features by adversary relation.
  - **The label predictor** : predicts class labels.
  - **The domain classifier** : discriminates between **source** and **target** domains during training. (– remember **adversary** ?).

General overview Ganin et al.

- The model jointly optimizing the underlying features by adversary relation.
  - **The label predictor** : predicts class labels.
  - **The domain classifier** : discriminates between **source** and **target** domains during training. (– remember **adversary** ?).
- **The label predictor** used both the training and test time.

General overview Ganin et al.

- The model jointly optimizing the underlying features by adversary relation.
  - **The label predictor** : predicts class labels.
  - **The domain classifier** : discriminates between **source** and **target** domains during training. (– remember **adversary** ?).
- **The label predictor** used both the training and test time.
- **The domain classifier** used in the training time.

General overview Ganin et al.

Adversary relation - for domain-invariant feature

General overview Ganin et al.

Adversary relation - for domain-invariant feature

- The parameters of the classifiers are optimized in order to minimize their error on the training set.

General overview Ganin et al.

Adversary relation - for domain-invariant feature

- The parameters of the classifiers are optimized in order to minimize their error on the training set.
- The parameters are optimized in order to,

General overview Ganin et al.

Adversary relation - for domain-invariant feature

- The parameters of the classifiers are optimized in order to minimize their error on the training set.
- The parameters are optimized in order to,
    - Minimize the loss of the **label classifier**.

General overview Ganin et al.

Adversary relation - for domain-invariant feature

- The parameters of the classifiers are optimized in order to minimize their error on the training set.
- The parameters are optimized in order to,
  - Minimize the loss of the **label classifier**.
  - Maximize the loss of the **domain classifier**.

General overview Ganin et al.

- The adaptation behaviour can be achieved in almost any feed-forward model.

General overview Ganin et al.

- The adaptation behaviour can be achieved in almost any feed-forward model.
- The model also augment a new **gradient reversal layer**.

Gradient Reversal later

Gradient reversal layer leaves the input unchanged during **forward propagation**.

Gradient Reversal later

Gradient reversal layer leaves the input unchanged during **forward propagation**.

Gradient reversal layer reverses the gradient by multiplying it by a **negative scalar** during the backpropagation.

Gradient Reversal later

Gradient reversal layer leaves the input unchanged during **forward propagation**.

Gradient reversal layer reverses the gradient by multiplying it by a **negative scalar** during the backpropagation.

Gradient reversal ensures that the feature distributions over the two domains are made similar.

General overview Ganin et al.

- The adaptation behaviour can be achieved in almost any feed-forward model.
- The model also augment a new **gradient reversal layer**.

General overview Ganin et al.

- The adaptation behaviour can be achieved in almost any feed-forward model.
- The model also augment a new **gradient reversal layer**.
- The resulting augmented architecture can be trained using standard backpropagation and stochastic gradient descent.

General overview Ganin et al.

- The adaptation behaviour can be achieved in almost any feed-forward model.
- The model also augment a new **gradient reversal layer**.
- The resulting augmented architecture can be trained using standard backpropagation and stochastic gradient descent.
- The approach is **generic** as a **DANN** version can be created for almost any existing **feed-forward architecture**.

Domain-Adversarial Neural Network - DANN

General overview Ganin et al.

The simplest **DANN** architecture have three parts (linear parts),

- Label predictor.
- Domain classifier.
- Feature extractor.

DANN model

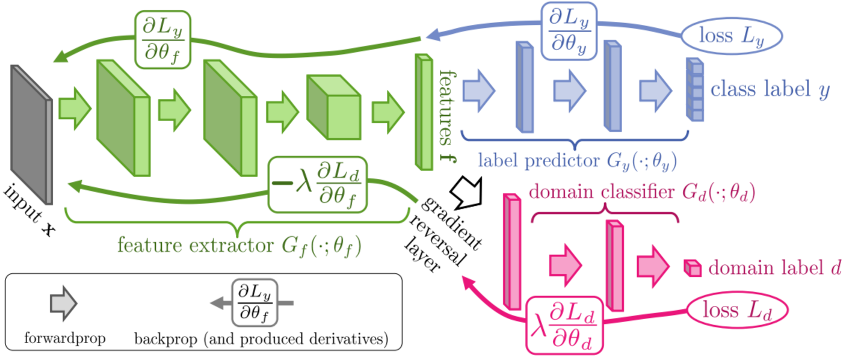

Figure 20: DANN Architecture

Setup

- $\mathbf{X}$ is the input space.

Setup

- $\mathbf{X}$ is the input space.
- $\mathbf{Y} = \{0, 1, ..., L-1\}$ is the set of $\mathbf{L}$ possible labels.

Setup

- $\mathbf{X}$ is the input space.
- $\mathbf{Y} = \{0, 1, ..., L-1\}$ is the set of $\mathbf{L}$ possible labels.
- We have two different distributions over XxY
  - $\mathcal{D}_S$ source domain.
  - $\mathcal{D}_T$ target domain.

Setup

- $\mathbf{X}$ is the input space.
- $\mathbf{Y} = \{0, 1, ..., L-1\}$ is the set of $\mathbf{L}$ possible labels.
- We have two different distributions over XxY
  - $\mathcal{D}_S$ source domain.
  - $\mathcal{D}_T$ target domain.

$$S = (x_i, y_i)_{i=1}^{n} \sim (\mathcal{D}_S)^n$$

$$T = (x_i)_{i=n+1}^{N} \sim (\mathcal{D}_X)^{n'}$$

$$N = n + n'$$

where,

- $(\mathcal{D}_S)^n$ and $(\mathcal{D}_X)^{n'}$ is the marginal distribution of $\mathcal{D}_T$ over X.

Target Risk

$$S = (\mathbf{x}_i, y_i)_{i=1}^{n} \sim (\mathcal{D}_S)^n$$

$$T = (\mathbf{x}_i)_{i=n+1}^{N} \sim (\mathcal{D}_X)^{n'}$$

$$N = n + n'$$

Target Risk

$$S = (\mathbf{x}_i, y_i)_{i=1}^{n} \sim (\mathcal{D}_S)^n$$

$$T = (\mathbf{x}_i)_{i=n+1}^{N} \sim (\mathcal{D}_X)^{n'}$$

$$N = n + n'$$

The goal of the learning algorithm is to build a classifier $\eta : X \to Y$ with a low **target risk**,

Target Risk

$$S = (\mathbf{x}_i, y_i)_{i=1}^{n} \sim (\mathcal{D}_S)^n$$

$$T = (\mathbf{x}_i)_{i=n+1}^{N} \sim (\mathcal{D}_X)^{n'}$$

$$N = n + n'$$

The goal of the learning algorithm is to build a classifier $\eta : X \to Y$ with a low **target risk**,

$$R_{\mathcal{D}_T}(\eta) = Pr_{(\mathbf{x},y)\sim\mathcal{D}_T}(\eta(\mathbf{x}) \neq y)$$

Target Risk

$$S = (\mathbf{x}_i, y_i)_{i=1}^{n} \sim (\mathcal{D}_S)^n$$

$$T = (\mathbf{x}_i)_{i=n+1}^{N} \sim (\mathcal{D}_X)^{n'}$$

$$N = n + n'$$

The goal of the learning algorithm is to build a classifier $\eta : X \to Y$ with a low **target risk**,

$$R_{\mathcal{D}_T}(\eta) = Pr_{(\mathbf{x},y)\sim\mathcal{D}_T}(\eta(\mathbf{x}) \neq y)$$

While having no information about the labels of $\mathcal{D}_T$.

Domain-Adversarial Neural Networks (DANN)

Example Case with a Shallow Neural Network

Domain-Adversarial Neural Networks (DANN)

Example Case with a Shallow Neural Network

- We suppose that the input space is formed by m-dimensional real vectors. $X = \mathbb{R}^m$

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Domain-Adversarial Neural Networks (DANN)

Example Case with a Shallow Neural Network

- We suppose that the input space is formed by m-dimensional real vectors. $X = \mathbb{R}^m$

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where, G_f is the **hidden layer**. that learn a function mapping examples into D-dimentional representation,

Domain-Adversarial Neural Networks (DANN)

Example Case with a Shallow Neural Network

- We suppose that the input space is formed by m-dimensional real vectors. $X = \mathbb{R}^m$

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$

where, G_f is the **hidden layer**. that learn a function mapping examples into D-dimentional representation,

$$G_f : X \to \mathbb{R}^D$$

$$sigma(a) = \left[\frac{1}{1 + exp(-a)} \right]_{i=1}^{|a|}$$

Predictor :: Shallow Neural Network with DANN

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Predictor :: Shallow Neural Network with DANN

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$G_y(G_f(x); \mathbf{V}, \mathbf{c}) = softmax(\mathbf{V}G_f(\mathbf{x}) + c)$$

Predictor :: Shallow Neural Network with DANN

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$G_y(G_f(x); \mathbf{V}, \mathbf{c}) = softmax(\mathbf{V}G_f(\mathbf{x}) + c)$$

where , $softmax(a) = \left[\frac{exp(a_i)}{\sum_{j=1}^{|a|} exp(a_j)} \right]_{i=1}^{|a|}$

Predictor :: Shallow Neural Network with DANN

$$G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = sigm(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$G_y(G_f(x); \mathbf{V}, \mathbf{c}) = softmax(\mathbf{V}G_f(\mathbf{x}) + c)$$
$$\text{where , } softmax(a) = \left[\frac{exp(a_i)}{\sum_{j=1}^{|a|} exp(a_j)}\right]_{i=1}^{|a|}$$

and G_y is the **prediction layer** that learns a function, $G_y : \mathbb{R}^D \rightarrow [0,1]^L$ G_y is parameterized by a pair $(\mathbf{V}, c) \in \mathbb{R}^{L \times D} \times \mathbb{R}^L$
here, G_y act as a **Label predictor**.

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$$

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$$

Training the neural network then leads to the following **optimization** problem on the source domain,

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b}) \right]$$

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$$

Training the neural network then leads to the following **optimization** problem on the source domain,

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b}) \right]$$

where,

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$$

Training the neural network then leads to the following **optimization** problem on the source domain,

$$\underset{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}{min}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) + \lambda \cdot R(\mathbf{W},\mathbf{b})\right]$$

where,

- $\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) = \mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i)$

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$$

Training the neural network then leads to the following **optimization** problem on the source domain,

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b}) \right]$$

where,

- $\mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) = \mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i)$
- λ is a weighted scalar hyperparameter.

Loss function of Predictor :: Shallow Neural Network with DANN

Given a source example $(\mathbf{x}_i, y_i)$, the natural classification loss to use is the **negative log-likelihood** of the correct label.

$$\mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i) = \log \frac{1}{G_y(G_f(\mathbf{x}))_{y_i}}$$

Training the neural network then leads to the following **optimization** problem on the source domain,

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b}) \right]$$

where,

- $\mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) = \mathcal{L}_y(G_y(G_f(\mathbf{x}_i)), y_i)$
- λ is a weighted scalar hyperparameter.
- $R(\mathbf{W}, \mathbf{b})$ is an optional **regularizer**.

Domain Classifier :: Shallow Neural Network with DANN

Recap

S represent **source domain** and T represents **target domain**.

Domain Classifier :: Shallow Neural Network with DANN

Recap

S represent **source domain** and T represents **target domain**.

The output of the hidden layer $G_f(\cdot)$ is the **internal representation** of the neural network.

$$S(G_f) = \{G_f(\mathbf{x})|\mathbf{x} \in S)$$
$$T(G_f) = \{G_f(\mathbf{x})|\mathbf{x} \in T)$$

Domain Classifier :: Shallow Neural Network with DANN

Recap

S represent **source domain** and T represents **target domain**.

The output of the hidden layer $G_f(\cdot)$ is the **internal representation** of the neural network.

$$S(G_f) = \{G_f(\mathbf{x}) | \mathbf{x} \in S)$$
$$T(G_f) = \{G_f(\mathbf{x}) | \mathbf{x} \in T)$$

from **Ben-David et al.**, the empirical $\mathcal{H}$-divergence of a symmetric hypothesis class $\mathcal{H}$ between samples $S(G_f)$ and $T(G_f)$ is given by,

$$\hat{\delta}(S(G_f), T(G_f)) = 2\left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{n}\sum_{i=1}^{n} I[\eta(G_f(\mathbf{x_i})) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(G(_f(\mathbf{x})) = 1]\right]\right)$$

Domain Classifier :: Shallow Neural Network with DANN

$$\hat{\delta}(S(G_f), T(G_f)) = 2\left(1 - \underset{\eta \in \mathcal{H}}{min}\Big[\frac{1}{n}\sum_{i=1}^{n} I[\eta(G_f(\mathbf{x_i})) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(G(_f(\mathbf{x})) = 1]\Big]\right)$$

Domain Classifier :: Shallow Neural Network with DANN

$$\hat{\delta}(S(G_f), T(G_f)) = 2\left(1 - \underset{\eta \in \mathcal{H}}{min}\Big[\frac{1}{n}\sum_{i=1}^{n} I[\eta(G_f(\mathbf{x_i})) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(G(_f(\mathbf{x})) = 1]\Big]\right)$$

the min part of this equation will be estimated by **domain classification** layerG_d .

Domain Classifier :: Shallow Neural Network with DANN

$$\hat{\delta}(S(G_f), T(G_f)) = 2\Big(1 - \min_{\eta \in \mathcal{H}} \Big[\frac{1}{n}\sum_{i=1}^{n} I[\eta(G_f(\mathbf{x_i})) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(G(_f(\mathbf{x})) = 1]\Big]\Big)$$

the min part of this equation will be estimated by **domain classification** layerG_d .

$$G_d : \mathbb{R}^D \to [0,1]^D$$

Here, G_d is parameterized by $(\mathbf{u}, z) \in \mathbb{R}^D \times \mathbb{R}$.

Domain Classifier :: Shallow Neural Network with DANN

$$\hat{\delta}(S(G_f), T(G_f)) = 2\left(1 - \min_{\eta \in \mathcal{H}}\Big[\frac{1}{n}\sum_{i=1}^{n} I[\eta(G_f(\mathbf{x_i})) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(G(_f(\mathbf{x})) = 1]\Big]\right)$$

the min part of this equation will be estimated by **domain classification** layerG_d .

$$G_d : \mathbb{R}^D \to [0,1]^D$$

Here, G_d is parameterized by $(\mathbf{u}, z) \in \mathbb{R}^D \times \mathbb{R}$.

$$G_d(G_f(\mathbf{x}); \mathbf{u}, z) = sigm(\mathbf{u}^T G_f(x) + z)$$

where G_d act as a **Domain classifier**.

Loss function of Domain Classifier :: Shallow Neural Network with DANN

Loss Function of Logistic Regressor (domain classifier) G_d,

Loss function of Domain Classifier :: Shallow Neural Network with DANN

Loss Function of Logistic Regressor (domain classifier) G_d,

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)), d_i) = d_i \log \frac{1}{G_d(G_f(\mathbf{x_i}))} + (1 - d_i) \log \frac{1}{1 - G_d(g_f((\mathbf{x_i}))}$$

Loss function of Domain Classifier :: Shallow Neural Network with DANN

Loss Function of Logistic Regressor (domain classifier) G_d,

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)), d_i) = d_i \log \frac{1}{G_d(G_f(\mathbf{x_i}))} + (1-d_i) \log \frac{1}{1 - G_d(g_f((\mathbf{x_i}))}$$

$$\mathbf{x_i} \sim \mathcal{D}_S^X \quad if \quad d_i = 0$$
$$\mathbf{x_i} \sim \mathcal{D}_T^X \quad if \quad d_i = 1$$

Shallow Neural Network with DANN

Shallow Neural Network with DANN

- The examples from the source distribution $(d_i = 0)$, the corresponding labels $y_i \in Y$ are known at **training time**.

Shallow Neural Network with DANN

- The examples from the source distribution $(d_i = 0)$, the corresponding labels $y_i \in Y$ are known at **training time**.
- The examples from the target domains, we do not know the labels at training time.
    - We want to predict such labels at test time.

Regularizer :: Shallow Neural Network with DANN

Ooptimization objective

$$\underset{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}{min}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c})+\lambda\cdot R(\mathbf{W},\mathbf{b})\right]$$

Loss of the Domain regressor

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)),d_i)=d_i\log\frac{1}{G_d(G_f(\mathbf{x_i}))}+(1-d_i)\log\frac{1}{1-G_d(g_f((\mathbf{x_i}))}$$

Regularizer :: Shallow Neural Network with DANN

Ooptimization objective

$$\underset{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}{min}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c})+\lambda\cdot R(\mathbf{W},\mathbf{b})\right]$$

Loss of the Domain regressor

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)),d_i)=d_i\log\frac{1}{G_d(G_f(\mathbf{x_i}))}+(1-d_i)\log\frac{1}{1-G_d(g_f((\mathbf{x_i}))}$$

We can add add loss of **Domain regressor** as the **regularizer** in the optimazation problem,

Regularizer :: Shallow Neural Network with DANN

Ooptimization objective

$$\underset{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}{min}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c})+\lambda\cdot R(\mathbf{W},\mathbf{b})\right]$$

Loss of the Domain regressor

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)),d_i)=d_i\log\frac{1}{G_d(G_f(\mathbf{x_i}))}+(1-d_i)\log\frac{1}{1-G_d(g_f((\mathbf{x_i}))}$$

We can add add loss of **Domain regressor** as the **regularizer** in the optimazation problem,

$$R(\mathbf{W},\mathbf{b})=\underset{(\mathbf{u},z)}{max}\left[-\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)-\frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\right]$$

Regularizer :: Shallow Neural Network with DANN

Ooptimization objective

$$\underset{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}}{min}\left[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c})+\lambda\cdot R(\mathbf{W},\mathbf{b})\right]$$

Loss of the Domain regressor

$$\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)),d_i)=d_i\log\frac{1}{G_d(G_f(\mathbf{x_i}))}+(1-d_i)\log\frac{1}{1-G_d(g_f((\mathbf{x_i}))}$$

We can add add loss of **Domain regressor** as the **regularizer** in the optimazation problem,

$$R(\mathbf{W},\mathbf{b})=\underset{(\mathbf{u},z)}{max}\left[-\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)-\frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\right]$$

where, $\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)=\mathcal{L}_d(G_d(G_f(\mathbf{x}_i)),d_i)$

optimization objective

We can now re-write the whole **optimization objective**

optimization objective

We can now re-write the whole **optimization objective**

$$\begin{aligned} E(W,b,V,c,u,z) &= \Big[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) + \lambda\cdot R(\mathbf{W},\mathbf{b})\Big] \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) - \lambda\Bigg(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z) \\ &\qquad + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\Bigg) \end{aligned}$$

$$(\hat{\mathbf{W}},\hat{\mathbf{V}},\hat{\mathbf{b}},\hat{\mathbf{c}}) = \underset{\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}}{argmin}\ \ E(\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c},\hat{\mathbf{u}},\hat{z})$$

$$(\hat{\mathbf{u}},\hat{z}) = \underset{\mathbf{u},z}{argmax}\ \ E(\hat{\mathbf{W}},\hat{\mathbf{V}},\hat{\mathbf{b}},\hat{\mathbf{c}},\mathbf{u},z)$$

optimization objective

We can now re-write the whole **optimization objective**

$$
\begin{aligned}
E(W, b, V, c, u, z) &= \Big[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b})\Big] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) - \lambda\Bigg(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) \\
&\qquad + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z)\Bigg)
\end{aligned}
$$

optimization objective

We can now re-write the whole **optimization objective**

$$
\begin{aligned}
E(W, b, V, c, u, z) &= \Big[\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b})\Big] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) - \lambda\Bigg(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z) \\
&\qquad + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{u}, z)\Bigg)
\end{aligned}
$$

$$(\hat{\mathbf{W}}, \hat{\mathbf{V}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}) = \underset{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}}{argmin}\ \ E(\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}, \hat{\mathbf{u}}, \hat{z})$$

$$(\hat{\mathbf{u}}, \hat{z}) = \underset{\mathbf{u}, z}{argmax}\ \ E(\hat{\mathbf{W}}, \hat{\mathbf{V}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}, \mathbf{u}, z)$$

Algorithm

Algorithm 1 Shallow DANN – Stochastic training update

Input:
— samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and $T = \{\mathbf{x}_i\}_{i=1}^{n'}$,
— hidden layer size D,
— adaptation parameter λ,
— learning rate μ,
Output: neural network $\{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$

$\mathbf{W}, \mathbf{V} \leftarrow \text{random\_init}(\,D\,)$
$\mathbf{b}, \mathbf{c}, \mathbf{u}, d \leftarrow 0$
while stopping criterion is not met **do**
  for i from 1 to n **do**
    `# Forward propagation`
    $G_f(\mathbf{x}_i) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_i)$
    $G_y(G_f(\mathbf{x}_i)) \leftarrow \text{softmax}(\mathbf{c} + \mathbf{V}G_f(\mathbf{x}_i))$
    `# Backpropagation`
    $\Delta_{\mathbf{c}} \leftarrow -(\mathbf{e}(y_i) - G_y(G_f(\mathbf{x}_i)))$
    $\Delta_{\mathbf{V}} \leftarrow \Delta_{\mathbf{c}}\ G_f(\mathbf{x}_i)^\top$
    $\Delta_{\mathbf{b}} \leftarrow \left(\mathbf{V}^\top \Delta_{\mathbf{c}}\right) \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{b}} \cdot (\mathbf{x}_i)^\top$
    `# Domain adaptation regularizer...`
    `# ...from current domain`
    $G_d(G_f(\mathbf{x}_i)) \leftarrow \text{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_i))$
    $\Delta_d \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
    $\Delta_{\mathbf{u}} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))G_f(\mathbf{x}_i)$
    $\text{tmp} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
      $\times\, \mathbf{u} \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
    $\Delta_{\mathbf{b}} \leftarrow \Delta_{\mathbf{b}} + \text{tmp}$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{W}} + \text{tmp} \cdot (\mathbf{x}_i)^\top$
    `# ...from other domain`
    $j \leftarrow \text{uniform\_integer}(1, \ldots, n')$
    $G_f(\mathbf{x}_j) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_j)$
    $G_d(G_f(\mathbf{x}_j)) \leftarrow \text{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_j))$
    $\Delta_d \leftarrow \Delta_d - \lambda G_d(G_f(\mathbf{x}_j))$
    $\Delta_{\mathbf{u}} \leftarrow \Delta_{\mathbf{u}} - \lambda G_d(G_f(\mathbf{x}_j))G_f(\mathbf{x}_j)$
    $\text{tmp} \leftarrow -\lambda G_d(G_f(\mathbf{x}_j))$
      $\times\, \mathbf{u} \odot G_f(\mathbf{x}_j) \odot (1 - G_f(\mathbf{x}_j))$
    $\Delta_{\mathbf{b}} \leftarrow \Delta_{\mathbf{b}} + \text{tmp}$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{W}} + \text{tmp} \cdot (\mathbf{x}_j)^\top$
    `# Update neural network parameters`
    $\mathbf{W} \leftarrow \mathbf{W} - \mu\Delta_{\mathbf{W}}$
    $\mathbf{V} \leftarrow \mathbf{V} - \mu\Delta_{\mathbf{V}}$
    $\mathbf{b} \leftarrow \mathbf{b} - \mu\Delta_{\mathbf{b}}$
    $\mathbf{c} \leftarrow \mathbf{c} - \mu\Delta_{\mathbf{c}}$
    `# Update domain classifier`
    $\mathbf{u} \leftarrow \mathbf{u} + \mu\Delta_{\mathbf{u}}$
    $d \leftarrow d + \mu\Delta_d$
  end for
end while

Note: In this pseudo-code, $\mathbf{e}(y)$ refers to a "one-hot" vector, consisting of all 0s except for a 1 at position y, and $\odot$ is the element-wise product.

Algorithm

Algorithm 1 Shallow DANN – Stochastic training update

Input:
— samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and $T = \{\mathbf{x}_i\}_{i=1}^{n'}$,
— hidden layer size D,
— adaptation parameter λ,
— learning rate μ,
Output: neural network $\{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$

$\mathbf{W}, \mathbf{V} \leftarrow \text{random\_init}(\,D\,)$
$\mathbf{b}, \mathbf{c}, \mathbf{u}, d \leftarrow 0$
while stopping criterion is not met **do**
  for i from 1 to n **do**
    `# Forward propagation`
    $G_f(\mathbf{x}_i) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_i)$
    $G_y(G_f(\mathbf{x}_i)) \leftarrow \text{softmax}(\mathbf{c} + \mathbf{V}G_f(\mathbf{x}_i))$

    `# Backpropagation`
    $\Delta_{\mathbf{c}} \leftarrow -(\mathbf{e}(y_i) - G_y(G_f(\mathbf{x}_i)))$
    $\Delta_{\mathbf{V}} \leftarrow \Delta_{\mathbf{c}}\; G_f(\mathbf{x}_i)^\top$
    $\Delta_{\mathbf{b}} \leftarrow \left(\mathbf{V}^\top \Delta_{\mathbf{c}}\right) \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{b}} \cdot (\mathbf{x}_i)^\top$

    `# Domain adaptation regularizer...`
    `# ...from current domain`
    $G_d(G_f(\mathbf{x}_i)) \leftarrow \text{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_i))$
    $\Delta_d \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
    $\Delta_{\mathbf{u}} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))G_f(\mathbf{x}_i)$
    $\text{tmp} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
        $\times\, \mathbf{u} \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
    $\Delta_{\mathbf{b}} \leftarrow \Delta_{\mathbf{b}} + \text{tmp}$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{W}} + \text{tmp} \cdot (\mathbf{x}_i)^\top$

    `# ...from other domain`
    $j \leftarrow \text{uniform\_integer}(1, \ldots, n')$
    $G_f(\mathbf{x}_j) \leftarrow \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_j)$
    $G_d(G_f(\mathbf{x}_j)) \leftarrow \text{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_j))$
    $\Delta_d \leftarrow \Delta_d - \lambda G_d(G_f(\mathbf{x}_j))$
    $\Delta_{\mathbf{u}} \leftarrow \Delta_{\mathbf{u}} - \lambda G_d(G_f(\mathbf{x}_j))G_f(\mathbf{x}_j)$
    $\text{tmp} \leftarrow -\lambda G_d(G_f(\mathbf{x}_j))$
        $\times\, \mathbf{u} \odot G_f(\mathbf{x}_j) \odot (1 - G_f(\mathbf{x}_j))$
    $\Delta_{\mathbf{b}} \leftarrow \Delta_{\mathbf{b}} + \text{tmp}$
    $\Delta_{\mathbf{W}} \leftarrow \Delta_{\mathbf{W}} + \text{tmp} \cdot (\mathbf{x}_j)^\top$

    `# Update neural network parameters`
    $\mathbf{W} \leftarrow \mathbf{W} - \mu\Delta_{\mathbf{W}}$
    $\mathbf{V} \leftarrow \mathbf{V} - \mu\Delta_{\mathbf{V}}$
    $\mathbf{b} \leftarrow \mathbf{b} - \mu\Delta_{\mathbf{b}}$
    $\mathbf{c} \leftarrow \mathbf{c} - \mu\Delta_{\mathbf{c}}$

    `# Update domain classifier`
    $\mathbf{u} \leftarrow \mathbf{u} + \mu\Delta_{\mathbf{u}}$
    $d \leftarrow d + \mu\Delta_d$
  end for
end while

Note: In this pseudo-code, $\mathbf{e}(y)$ refers to a "one-hot" vector, consisting of all 0s except for a 1 at position y, and $\odot$ is the element-wise product.

Summary

Goodfellow et al.

the **advarsay objective** will be,
somehow pass the information through **gradient update** to **maximize loss** for a **targeted feature** without interfering the **other features** to **learn** by **minimizing loss**
or
vice versa.

Ganin et al.

By backpropagating a **negative gradient** of a certain feature, we can remove the effect of that certain feature from a **mixtured feature vector**.

Challenge

Controlling the effect of the **regulatizer** by λ is very complicated.

Challenge

Controlling the effect of the **regulatizer** by λ is very complicated.

We can not **overtrain** the domain classifier. (caussing feature reduction)

Challenge

Controlling the effect of the **regulatizer** by λ is very complicated.

We can not **overtrain** the domain classifier. (caussing feature reduction)

We can not train (domain classifier) less . (reduce domain invarient property)

Tweaking DANN

What if we could tweak DANN model so that it learns well and regularizer become stable at a learning point or the model gets more robust.

Tweaking DANN

Learning sleep stages from Radio Signals: A conditional Adversarial Architecture - PMLR'17

Tweaking DANN

Learning sleep stages from Radio Signals: A conditional Adversarial Architecture - PMLR'17

Zhao et al.

CNN+LSTM based model for predicting sleep stages from radio measurement with **modified adversarial training** that discards extra information. (paper link)

General overview

- Sleep progresses in cycles that involve multiple sleep stages:

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake
  - Light sleep

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake
  - Light sleep
  - Deep sleep

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake
  - Light sleep
  - Deep sleep
  - Rapid Eye Movement (REM)

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake
  - Light sleep
  - Deep sleep
  - Rapid Eye Movement (REM)
- The gold standard for monitoring sleep imvolves wearable devices that may cause discomfort.

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake
  - Light sleep
  - Deep sleep
  - Rapid Eye Movement (REM)
- The gold standard for monitoring sleep imvolves wearable devices that may cause discomfort.
- Wireless signal like low powered **radio frequency**(RF) can extract a person's **breathing** and **heart beats**.

General overview

- Sleep progresses in cycles that involve multiple sleep stages:
  - Awake
  - Light sleep
  - Deep sleep
  - Rapid Eye Movement (REM)
- The gold standard for monitoring sleep imvolves wearable devices that may cause discomfort.
- Wireless signal like low powered **radio frequency**(RF) can extract a person's **breathing** and **heart beats**.
- Wireless system would provide better sleep stage monitoring.

General overview

- We must learn **RF** signal features that capture the sleep stages and their temporal progression.

General overview

- We must learn **RF** signal features that capture the sleep stages and their temporal progression.
- The features should be **transferable** to new **subjects** and **different environments**

General overview

- We must learn **RF** signal features that capture the sleep stages and their temporal progression.
- The features should be **transferable** to new **subjects** and **different environments**
- **RF** signals carry much information that is **irrelevant** to sleep staging, and are highly dependent on the individuals and the measurement conditions.

Representation Learning

- Main objectives to learn invariant representations in deep adversarial networks.
- Goal of this paper is to remove **conditional dependencies** rather than making the representation domain independent

Model

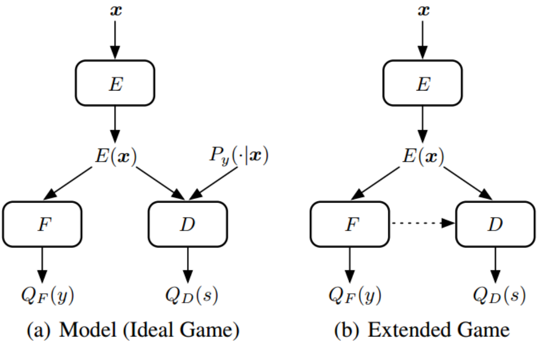

Figure 21: Model and Extended Game. Dotted arrow indicates that the information does not propagate back on this link. E is encoder, F is label predictor and D is the discriminator.

Model

- RF signals carry information about the

Model

- RF signals carry information about the
  - Subject (Domain 1)
  - Measurement environment (Domain 2)

Model

- RF signals carry information about the
  - Subject (Domain 1)
  - Measurement environment (Domain 2)
- Our goal is to learn a latent representation (i.e., an encoder) that can be used to predict label y(sleep stage).

Model

- RF signals carry information about the
  - Subject (Domain 1)
  - Measurement environment (Domain 2)
- Our goal is to learn a latent representation (i.e., an encoder) that can be used to predict label y(sleep stage).
- We want this representation to generalize well to predict sleep stages for new subjects without having labeled data from them.

Model

- RF signals carry information about the
  - Subject (Domain 1)
  - Measurement environment (Domain 2)
- Our goal is to learn a latent representation (i.e., an encoder) that can be used to predict label y(sleep stage).
- We want this representation to generalize well to predict sleep stages for new subjects without having labeled data from them.
- **Simply making the representation invariant to the source domains could hamper the accuracy of the predictive task** (why ??)

Model

This paper proposed to **remove conditional dependencies** between the representation and the source domains.

Ideal Game

The ideal game is like below,

Ideal Game

The ideal game is like below,

$$\mathcal{V}(E, F, D) = \mathcal{L}_f(F, E) - \lambda \cdot \mathcal{L}_d(D; E)$$

Ideal Game

The ideal game is like below,

$$\mathcal{V}(E, F, D) = \mathcal{L}_f(F, E) - \lambda \cdot \mathcal{L}_d(D; E)$$

The training procedure can be viewed as a three-player minimax game of E, F and D.

Ideal Game

The ideal game is like below,

$$\mathcal{V}(E, F, D) = \mathcal{L}_f(F, E) - \lambda \cdot \mathcal{L}_d(D; E)$$

The training procedure can be viewed as a three-player minimax game of E, F and D.

$$\underset{E}{min}\ \underset{F}{min}\ \underset{D}{max} \mathcal{V}(E, F, D) = \underset{E,F}{min}\ \underset{D}{max}\ \mathcal{V}(E, F, D)$$

Extended Game

apart from the mathematics, the extended game reduced to,

Extended Game

apart from the mathematics, the extended game reduced to,

- Connection in **forward propagation** with **label predictor** and **discriminator**.

Extended Game

apart from the mathematics, the extended game reduced to,

- Connection in **forward propagation** with **label predictor** and **discriminator**.
- Disconnection in **back propagation** with **label predictor** and **discriminator**.

Extended Game

apart from the mathematics, the extended game reduced to,

- Connection in **forward propagation** with **label predictor** and **discriminator**.
- Disconnection in **back propagation** with **label predictor** and **discriminator**.
- learn the Ideal game.

Algorithm

Algorithm 1 Encoder, predictor and discriminator training

Input: Labeled data $\{(\boldsymbol{x}_i, y_i, s_i)\}_{i=1}^{M}$, learning rate η.
Compute stop criterion for inner loop: $\delta_d \leftarrow H(s)$
for number of training iterations **do**
  Sample a mini-batch of training data $\{(\boldsymbol{x}_i, y_i, s_i)\}_{i=1}^{m}$

$$\mathcal{L}_f^i \leftarrow -\log Q_F(y_i|E(\boldsymbol{x}_i))$$

$$\boldsymbol{w}_i \leftarrow Q_F(\cdot|E(\boldsymbol{x}_i)) \;\blacktriangleright\; \textit{stop gradient along this link}$$

$$\mathcal{L}_d^i \leftarrow -\log Q_D(s_i|E(\boldsymbol{x}_i), \boldsymbol{w}_i)$$

$$\mathcal{V}^i = \mathcal{L}_f^i - \lambda \cdot \mathcal{L}_d^i$$

  Update encoder E:

$$\theta_e \leftarrow \theta_e - \eta_e \nabla_{\theta_e} \frac{1}{m} \sum_{i=1}^{m} \mathcal{V}^i$$

  Update predictor F:

$$\theta_f \leftarrow \theta_f - \eta_f \nabla_{\theta_f} \frac{1}{m} \sum_{i=1}^{m} \mathcal{V}^i$$

  repeat
    Update discriminator D:

$$\theta_d \leftarrow \theta_d + \eta_d \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \mathcal{V}^i$$

  until $\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_d^i \leq \delta_d$
end for

Training Algorithm

- Training structure is like GAN.

Training Algorithm

- Training structure is like GAN.
- The number of training steps in the inner loop usually needs to be carefully chosen.

Training Algorithm

- Training structure is like GAN.
- The number of training steps in the inner loop usually needs to be carefully chosen.
- A large number of steps is computationally inefficient but a small one will cause the model to collapse.

Discussion of the model benefits

- It guarantees an **equilibrium solution**.

Discussion of the model benefits

- It guarantees an **equilibrium solution**.
- It allows to properly leverage the adversarial feedback even when the **target labels are uncertain**.

Aspects transfer

Aspect-augmented Adversarial Networks for Domain Adaptation -TACL'17

Zhang et al.

A neural method for **transfer** learning between two (source and target) classification tasks or aspects over the **same domain** (semi-supervised). (paper link)

General overview

- Rather than training on **target labels**, this paper uses a few **keywords** pertaining to source and target aspects indicating sentence relevance instead of document class labels.

General overview

- Rather than training on **target labels**, this paper uses a few **keywords** pertaining to source and target aspects indicating sentence relevance instead of document class labels.
- Values extracted for different fields from the same document are often dependent as they share the same context.

General overview

- Rather than training on **target labels**, this paper uses a few **keywords** pertaining to source and target aspects indicating sentence relevance instead of document class labels.
- Values extracted for different fields from the same document are often dependent as they share the same context.
- In this paper, they consider a version of a problem where there is a clear dependence between **two tasks** but annotations are available only for the **source task**.

General overview

- **Aspect transfer** is the objective is to learn to classify examples differently, focusing on different aspects, without access to target aspect labels.

General overview

- **Aspect transfer** is the objective is to learn to classify examples differently, focusing on different aspects, without access to target aspect labels.
- Such transfer learning is possible only with **auxiliary information** relating the tasks together.

General overview

- **Aspect transfer** is the objective is to learn to classify examples differently, focusing on different aspects, without access to target aspect labels.
- Such transfer learning is possible only with **auxiliary information** relating the tasks together.
- The key challenge is to articulate and incorporate **commonalities** across the tasks.

General overview

- **Aspect transfer** is the objective is to learn to classify examples differently, focusing on different aspects, without access to target aspect labels.
- Such transfer learning is possible only with **auxiliary information** relating the tasks together.
- The key challenge is to articulate and incorporate **commonalities** across the tasks.
- Both the source and the target classifiers operate over the **same domain**,

General overview

- **Aspect transfer** is the objective is to learn to classify examples differently, focusing on different aspects, without access to target aspect labels.
- Such transfer learning is possible only with **auxiliary information** relating the tasks together.
- The key challenge is to articulate and incorporate **commonalities** across the tasks.
- Both the source and the target classifiers operate over the **same domain**,
- To learn this invariant representation, this paper introduce an **adversarial domain classifier**.

General overview

- Adversarial training of domain and label classifiers can be challenging to stabilize.

General overview

- Adversarial training of domain and label classifiers can be challenging to stabilize.
- the aspect transfer problem and the method we develop to solve it work the same even when source and target documents are a priori different, something we will demonstrate later.

General overview

- Adversarial training of domain and label classifiers can be challenging to stabilize.
- the aspect transfer problem and the method we develop to solve it work the same even when source and target documents are a priori different, something we will demonstrate later.
- source and target tasks, both of these tasks are defined over the same set of examples.

General overview

- Adversarial training of domain and label classifiers can be challenging to stabilize.
- the aspect transfer problem and the method we develop to solve it work the same even when source and target documents are a priori different, something we will demonstrate later.
- source and target tasks, both of these tasks are defined over the same set of examples.
- The set of possible labels are the same across aspects. Like, postive,negative labels of having two different clinical report(aspects).

General Overview

- This paper also pre-calculates the **relevance** is given per sentence, for some subset of sentences across the documents

General Overview

- This paper also pre-calculates the **relevance** is given per sentence, for some subset of sentences across the documents
- **Relevance** indicates the possibility that the answer for that document would be found in the sentence but without indicating which way the answer goes.

General Overview

- This paper also pre-calculates the **relevance** is given per sentence, for some subset of sentences across the documents
- **Relevance** indicates the possibility that the answer for that document would be found in the sentence but without indicating which way the answer goes.
- Relevance is always **aspect dependent** yet often easy to provide in the form of simple keyword rules.

General Overview

- This paper also pre-calculates the **relevance** is given per sentence, for some subset of sentences across the documents
- **Relevance** indicates the possibility that the answer for that document would be found in the sentence but without indicating which way the answer goes.
- Relevance is always **aspect dependent** yet often easy to provide in the form of simple keyword rules.
- remember **sentence level relevence**.

General Overview

- This paper also pre-calculates the **relevance** is given per sentence, for some subset of sentences across the documents
- **Relevance** indicates the possibility that the answer for that document would be found in the sentence but without indicating which way the answer goes.
- Relevance is always **aspect dependent** yet often easy to provide in the form of simple keyword rules.
- remember **sentence level relevence**.
- the **aspect transfer** problem and solution works even when source and target documents are a **priori different**.

Model

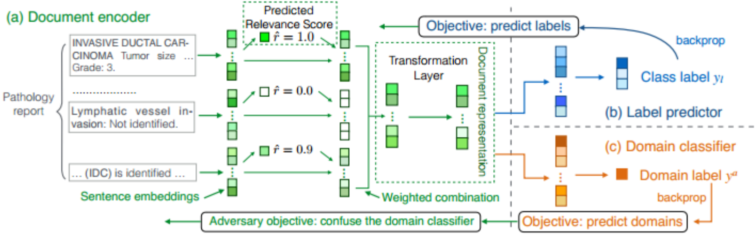

Figure 22: model representation

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

$$\hat{x}_{i,j} = \tanh(\mathbf{W}^c\mathbf{h}_{i,j} + \mathbf{b}^c)$$

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

$$\hat{x}_{i,j} = \tanh(\mathbf{W}^c\mathbf{h}_{i,j} + \mathbf{b}^c)$$

where $x_{i,j}$ in the convolutionla input and $\hat{x_{i,j}}$ is the convolutional output.

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

$$\hat{x}_{i,j} = \tanh(\mathbf{W}^c\mathbf{h}_{i,j} + \mathbf{b}^c)$$

where $x_{i,j}$ in the convolutionla input and $x\hat{_{i,j}}$ is the convolutional output.
both of the activation function must be same.

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

$$\mathcal{L}^{rel} = \sum_{(a,l,i)\in R} \left(r^a_{l,i} - \hat{r}^a_{l,i}\right)^2$$

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

$$\mathcal{L}^{rel} = \sum_{(a,l,i)\in R} \left(r^a_{l,i} - \hat{r}^a_{l,i}\right)^2$$

$$\Omega^{tr} = \lambda^{tr}||W^{tr} - I||^2_F$$

Unique Feature

This paper promotes an unique feature, adding reconstruction loss.

$$\mathcal{L}^{all} = \mathcal{L}^{rec} + \mathcal{L}^{rel} + \Omega^{tr} + \mathcal{L}^{lab} - \rho\mathcal{L}^{dom}$$

$$\mathcal{L}^{rec}(d) = \frac{1}{n}\sum_{i,j} ||\hat{x}_{i,j} - \tanh(x_{i,j}||^2$$

$$\mathcal{L}^{rel} = \sum_{(a,l,i)\in R} \left(r^a_{l,i} - \hat{r}^a_{l,i}\right)^2$$

$$\Omega^{tr} = \lambda^{tr}||W^{tr} - I||^2_F$$

here, $\mathcal{L}^{lab}$ and $\mathcal{L}^{dom}$ are the class label and domain classifier loss. For both of them we use cross entropy.

Summary

This paper provides an unique way of adding **regularizer** to an adversary learning.

Summary

This paper provides an unique way of adding **regularizer** to an adversary learning.

This paper provides an way to **add external information** to the model as relavance.

Combining All four paper

Combining All four paper

- GAN : What is adversary and how it's applied to a model to **generative model**.

Combining All four paper

- GAN : What is adversary and how it's applied to a model to **generative model**.
- DANN : How adversary applied to a NN and used it for **domain invariance**.

Combining All four paper

- GAN : What is adversary and how it's applied to a model to **generative model**.
- DANN : How adversary applied to a NN and used it for **domain invariance**.
- CAA : Works on removing **conditional dependencies** of two domains instead of **domain invariance**.

Combining All four paper

- GAN : What is adversary and how it's applied to a model to **generative model**.
- DANN : How adversary applied to a NN and used it for **domain invariance**.
- CAA : Works on removing **conditional dependencies** of two domains instead of **domain invariance**.
- AANDA : works on adding better **looss**. Also applied for feature transfer within **same domain**.