Semi-supervised Learning with Deep Generative Models

Diedrik P. Kingma, Danilo J. Rezende, Shakir Mohamed, Max Welling

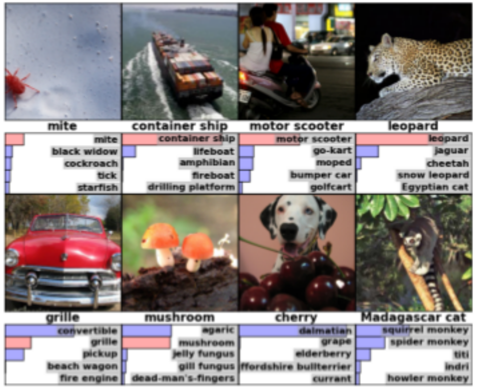
What is Deep Learning very good at?

Classifying highly structured data

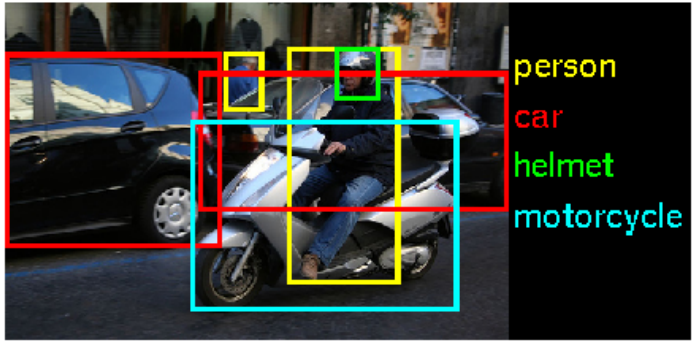
-ImageNet

-Part of Speech Tagging

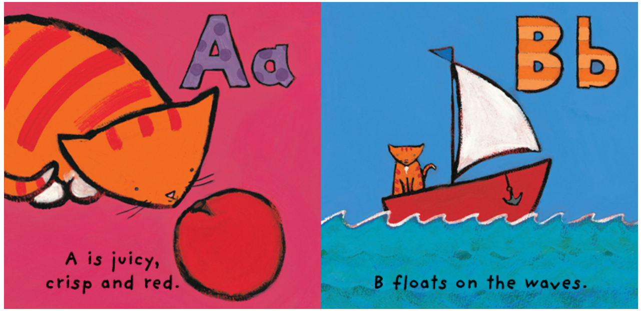
-MNIST



Sensitive to signals even in obscured or translated scenarios



How smart are Neural Nets?



?

Constrained to training classes

Labeled data is costly

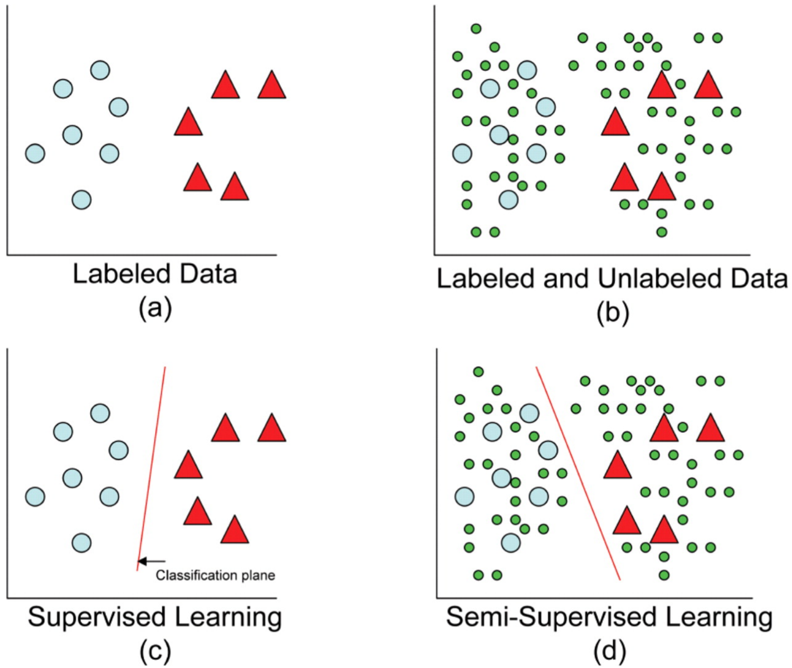
How do we generalize to more classes? More complex concepts?

Solution: Semi-supervised Learning

Learning in the situation of very little labeled (supervised) data

Use accessible data to improve decision boundaries and better classify unlabeled data

A real attempt at inductive reasoning?



Previous Work

Self Training Scheme (Rosenberg et al.)

Transductive SVMs (Joachims)

Graph Based Methods (Blum et al., Zhu et al.)

Manifold Tangent Classifier (Ranzato and Szummer)

Significant Contributions

Semi-supervised learning with generative models formed by the fusion of both:

-Probabilistic Models

-Deep Neural Networks

Stochastic Variational Inference for both model **and** variational parameters

Results: State of the art-classification, learns to separate content types from styles

Components

M1-Latent Feature Discriminative Model

M2-Generative Semi-Supervised Model

M1+M2 Stacked Generative Semi-Supervised Model

Optimization of Model using Variational Inference

Latent-Feature Discriminative Model

$$(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

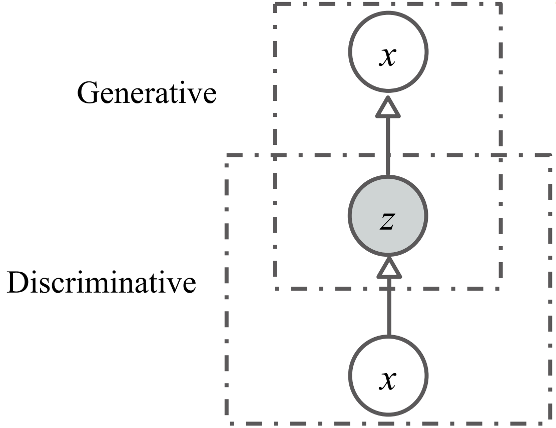
$$\mathbf{x}_i \in \mathbb{R}^D$$
$$y_i \in \{1, \ldots, L\}$$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}); \qquad p_\theta(\mathbf{x}|\mathbf{z}) = f(\mathbf{x}; \mathbf{z}, \boldsymbol{\theta})$$

$f(\mathbf{x}; \mathbf{z}, \boldsymbol{\theta})$ is a suitable likelihood function (e.g., a Gaussian or Bernoulli distribution)

The probabilities are formed by a non-linear transformations of a set of latent variables **z**.

Non-linear functions are neural networks!



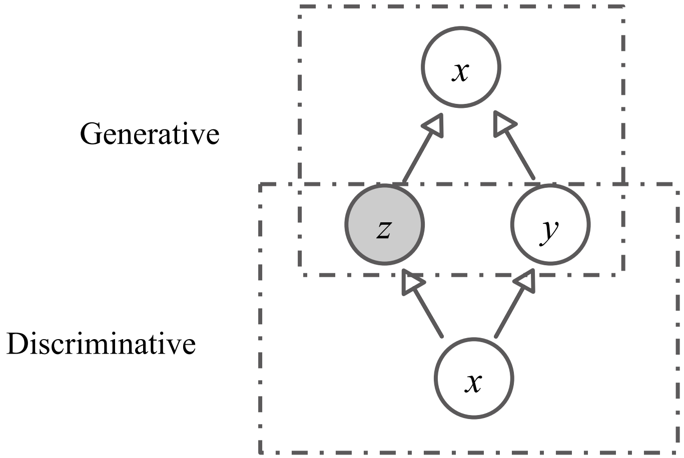
Generative semi-supervised Model

$$p(y) = \mathrm{Cat}(y|\boldsymbol{\pi}); \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}); \qquad p_\theta(\mathbf{x}|y, \mathbf{z}) = f(\mathbf{x}; y, \mathbf{z}, \boldsymbol{\theta})$$

$\mathrm{Cat}(y|\boldsymbol{\pi})$ is the multinomial distribution

Class labels are treated as latent variables, and $\mathbf{z}$ is an additional latent variable

Again, the likelihood function is parameterized by a non-linear transformation of latent variables, which are deep neural networks



Stacked Model (M1+M2)

Use the latent variables from M1 (z_1), to learn M2. Instead of raw data (x).

$$p_\theta(\mathbf{x}, y, \mathbf{z}_1, \mathbf{z}_2) = p(y)p(\mathbf{z}_2)p_\theta(\mathbf{z}_1|y, \mathbf{z}_2)p_\theta(\mathbf{x}|\mathbf{z}_1)$$

where

$$p(y) = \mathrm{Cat}(y|\boldsymbol{\pi}) \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$$

Conditionals are parameterized as deep neural nets as in previous models.

Optimization via Variational Inference

Posteriors are non-linear dependencies between random variables and thus extremely difficult to compute

$$p(z \mid x, \alpha) = \frac{p(z, x \mid \alpha)}{\int_z p(z, x \mid \alpha)} \longrightarrow q_\phi(\mathbf{z}|\mathbf{x})$$

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

Approximate with another function that's "close" and computable

(Jensen's Inequality) $f(\mathrm{E}[X]) \geq \mathrm{E}[f(X)]$

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]$$

Establish a lower bound objective

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

In our case...

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - KL[q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})] = -\mathcal{J}(\mathbf{x})$$

$$\log p_\theta(\mathbf{x}, y) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},y)}\left[\log p_\theta(\mathbf{x}|y,\mathbf{z}) + \log p_\theta(y) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x},y)\right] = -\mathcal{L}(\mathbf{x}, y)$$

$$\sum_y q_\phi(y|\mathbf{x})(-\mathcal{L}(\mathbf{x}, y)) + \mathcal{H}(q_\phi(y|\mathbf{x})) = -\mathcal{U}(\mathbf{x})$$

$$\mathcal{J} = \sum_{(\mathbf{x},y)\sim\widetilde{p}_l} \mathcal{L}(\mathbf{x}, y) + \sum_{\mathbf{x}\sim\widetilde{p}_u} \mathcal{U}(\mathbf{x})$$

$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \cdot \mathbb{E}_{\widetilde{p}_l(\mathbf{x},y)}\left[-\log q_\phi(y|\mathbf{x})\right]$$

Optimization Algorithms (EM variant)

$$\nabla_{\{\theta,\phi\}}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I})}\left[\nabla_{\{\theta,\phi\}}\log p_\theta(\mathbf{x}|\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x})\odot\boldsymbol{\epsilon})\right]$$

Algorithm 1 Learning in model M1

while generativeTraining() **do**
  $\mathcal{D} \leftarrow$ getRandomMiniBatch()
  $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{x}_i) \quad \forall \mathbf{x}_i \in \mathcal{D}$
  $\mathcal{J} \leftarrow \sum_n \mathcal{J}(\mathbf{x}_i)$
  $(\mathbf{g}_\theta, \mathbf{g}_\phi) \leftarrow (\frac{\partial \mathcal{J}}{\partial \theta}, \frac{\partial \mathcal{J}}{\partial \phi})$
  $(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{\phi}) + \boldsymbol{\Gamma}(\mathbf{g}_\theta, \mathbf{g}_\phi)$
end while
while discriminativeTraining() **do**
  $\mathcal{D} \leftarrow$ getLabeledRandomMiniBatch()
  $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{x}_i) \quad \forall \{\mathbf{x}_i, y_i\} \in \mathcal{D}$
  trainClassifier($\{\mathbf{z}_i, y_i\}$)
end while

Algorithm 2 Learning in model M2

while training() **do**
  $\mathcal{D} \leftarrow$ getRandomMiniBatch()
  $y_i \sim q_\phi(y_i|\mathbf{x}_i) \quad \forall \{\mathbf{x}_i, y_i\} \notin \mathcal{O}$
  $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|y_i, \mathbf{x}_i)$
  $\mathcal{J}^\alpha \leftarrow$ eq. (9)
  $(\mathbf{g}_\theta, \mathbf{g}_\phi) \leftarrow (\frac{\partial \mathcal{L}^\alpha}{\partial \theta}, \frac{\partial \mathcal{L}^\alpha}{\partial \phi})$
  $(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{\phi}) + \boldsymbol{\Gamma}(\mathbf{g}_\theta, \mathbf{g}_\phi)$
end while

Results MNIST

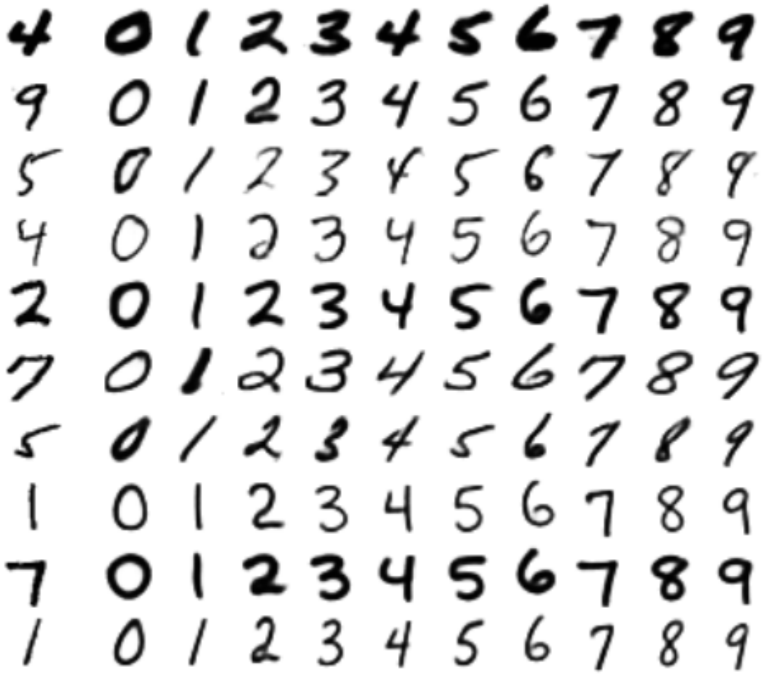
Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 ($\pm$ 0.95)	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.33** ($\pm$ 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.59** ($\pm$ 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 ($\pm$ 0.12)	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.40** ($\pm$ 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.18** ($\pm$ 0.04)

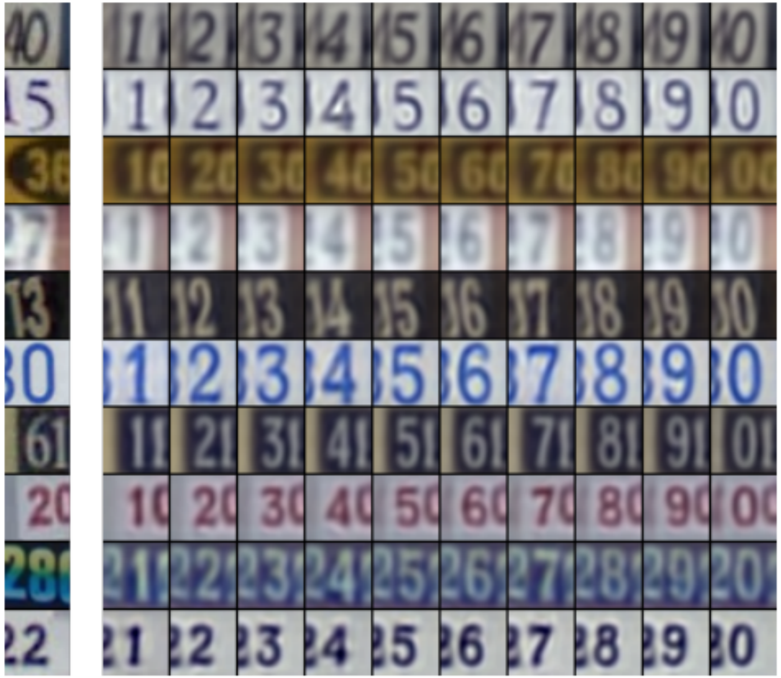
Classes vs. Styles

(a) Handwriting styles for MNIST obtained by fixing the class label and varying the 2D latent variable **z**

Other Data Sets



(b) MNIST analogies



(c) SVHN analogies

Classification

Table 2: Semi-supervised classification on the SVHN dataset with 1000 labels.

KNN	TSVM	M1+KNN	M1+TSVM	M1+M2
77.93 ($\pm$ 0.08)	66.55 ($\pm$ 0.10)	65.63 ($\pm$ 0.15)	54.33 ($\pm$ 0.11)	**36.02** ($\pm$ 0.10)

Table 3: Semi-supervised classification on the NORB dataset with 1000 labels.

KNN	TSVM	M1+KNN	M1+TSVM
78.71 ($\pm$ 0.02)	26.00 ($\pm$ 0.06)	65.39 ($\pm$ 0.09)	**18.79** ($\pm$ 0.05)

Conclusion

Innovative model design, especially using generative models to perform classification tasks

Implementation of variational inference

Results in powerful model with intra-class variation understanding

Could these be used with Convolutional Neural Nets?