Variational Sequential Labelers for Semi-Supervised Learning [1]

Mingda Chen    Qingming Tang    Karen Livescu    Kevin Gimpel

Toyota Technological Institute at Chicago
Chicago, IL
{mchen,qmtang,klivescu,kgimpel}@ttic.edu

March 7, 2019
Almost all papers in this seminar have focused on **discriminative models**

**Basic Definition:** Given some *existing* example $x$, *perform some task* that yields output $y$

Both papers today touch on **generative models**.

**Basic Definition:** Construct a *brand new* example $x$ that has *similar properties* as the training set

**Most Popular Techniques:**
- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)
This paper introduces a *bunch* of new ideas we have not discussed in this seminar...

*My Goal*: Focus on fostering intuitions

*My Decision*: Provide extra background information not in the paper to ensure common foundation

Let’s review some key terminology...
Review of Foundational Concepts
Supervised Learning

- “Traditional” mental model of machine learning
- Given a set of labeled examples: $L = \{(x_i, y_i)\}$, learn a function $f : x \rightarrow y$.
  - Preferred approach when you have lots of labeled data
  - What do you do when labeled data is limited and difficult to collect?
Semi-Supervised Learning

- Given a set of labeled examples (tuples) $L = \{(x_i, y_i)\}$ and a set of unlabeled examples $U = \{x_j\}$, still learn a function $f : x \rightarrow y$.
- **Generally**: $|U| \gg |L|$
- **Question**: When is semi-supervised learning useful? *Let’s visualize an example*

**Intuition**: Use semi-supervised techniques when $U$ has key *information* not in $L$
Representation Learning

- **Formally:** Automated learning of ways to represent data that “[makes] it easier to extract useful information when building classifiers or other predictors” [2]

- **Question:** Have we seen/used representation learning in this course?

- **Answer:** Absolutely. Word embeddings (e.g., GloVe, Word2Vec)
“Vanilla” Autoencoder

To understand variational autoencoders, it helps to understand “vanilla” autoencoders...

A neural network used for representation learning — typically dimensionality reduction

![Diagram of autoencoder with encoder, bottleneck, and decoder]

Objective Function \( J = \min ||x - \hat{x}|| \)
Formalizing an Autoencoder (with some notation abuse)

- **Input**: $x \in \mathbb{R}^{\text{dim}(x)}$
- **Encoder**: $q : x \rightarrow z$
- **Latent Vector**: $z \in \mathbb{R}^d$
  - $d$: Hyperparameter where generally $d < \text{dim}(x)$
  - **Usage**: Usually the input to another learner
- **Decoder**: $p : z \rightarrow \hat{x}$
- **Output**: $D(x) = D(\hat{x})$
“Vanilla” Autoencoder: Learns a dimensionality reduction function $\mathbb{R}^{\text{dim}(x)} \rightarrow \mathbb{R}^d$

Variational Autoencoder: Learns a probability distribution $p_\theta(\cdot)$
  - Combines ideas from probabilistic graphical models and neural networks
  - Based on concepts in “variational” inference

Question: Why learn $p_\theta(z)$ if $p(x)$ already exists?

Answer: $p_\theta(z)$ is forced to be tractable (simple) allowing for easy generation of new samples.
  - $p_\theta(z)$ is form a multivariate Gaussian, $\mathcal{N}(\mu, \Sigma)$
Variational Autoencoder Architecture

- Network architecture largely the same as a “vanilla” autoencoder
- Primary change is the latent vector $z$ is now the concatenation of a vector of means $\mu$ and a vector of standard deviations $\sigma$
- **Updated Objective Function:** $J = \min \left\{ -\log p_\theta(x) \right\}$
Estimating the Loss Function

Question: How do you calculate $p_\theta(x)$?

Answer: $p_\theta(x)$ is generally intractable. We need a better approach.

Alternate Approach: Maximize $p_\theta(x)$’s variational lower bound

Two key mathematical concepts needed:

- **Expectation**: Probabilistic average value for some function $f(x)$ with probability distribution $p(x)$

  $$\mathbb{E}\{f(x)\} \triangleq \int f(x)p(x)dx$$

- **KL Divergence**: Quantifies difference of probability distribution $p$ with respect to $q$

  $$D_{\text{KL}}(p \parallel q) \triangleq -\sum_x p(x) \log \frac{q(x)}{p(x)}$$
Variational Autoencoder Loss

\[ U = \mathbb{E}_{z \sim q(\cdot)} \left\{ \log p_\theta(x|z) \right\} - D_{KL} \left( q_\phi(z|x_{1:T}) \parallel p_\theta(z) \right) \]

- **Reconstruction Quality**
- **Distribution Divergence Regularizer**

\( p_\theta(z) \) is multivariate Gaussian:

\[ q(z) \triangleq q(\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \]

where:
- \( \mu \): Mean vector in \( z \)
- \( \Sigma \): Covariance matrix formed from \( \text{diag}(\sigma) \) in \( z \)
Proof for Variational Inference Function

Based on [3].

\[
\log p_\theta(x) = \log \left( \int p_\theta(x, z) dz \right) \\
= \log \left( \int p_\theta(x, z) \frac{q(z)}{q(z)} dz \right) \\
= \log \mathbb{E}_{z \sim q(\cdot)} \left\{ \frac{p_\theta(x, z)}{q(z)} \right\} \\
\geq \mathbb{E}_{z \sim q(\cdot)} \left\{ \log \frac{p_\theta(x, z)}{q(z)} \right\} \\
= \mathbb{E}_{z \sim q(\cdot)} \left\{ \log \frac{p_\theta(x | z)p_\theta(z)}{q(z)} \right\} \\
= \mathbb{E}_{z \sim q(\cdot)} \left\{ \log p_\theta(x | z) \right\} - D_{KL} \left( q(z) \parallel p_\theta(z) \right)
\]

Bayes’ Rule
Def. of expectation
Jensen’s Inequality for concave func. log
Bayes’ Rule
Def. KL Divergence
Reparameterization Trick

- The variational autoencoder architecture is slightly more complicated than I showed.
- Variational autoencoder training randomly samples from the distribution defined by $z$.
- Stochastic sampling is non-differentiable blocking backpropagation.
- The *reparameterization trick* is a clever way to enable both backpropagation and random sampling.
  - If there is interest, we can go through the trick on the whiteboard at the end. Just ask.
Sequence Labeling

Assign a **sequence of labels** to a **sequence of objects**.

Common NLP Sequence Labeling Tasks:

- **Part of Speech (POS) Tagging**: Mark each word as noun, verb, adjective, etc.

<table>
<thead>
<tr>
<th>Bob</th>
<th>made</th>
<th>a</th>
<th>book</th>
<th>collector</th>
<th>happy</th>
<th>the</th>
<th>other</th>
<th>day [4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Compound Noun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Compound Adverb</td>
<td></td>
</tr>
</tbody>
</table>

- **Named-Entity Recognition (NER)**: Identify names of real-world objects like people, locations, products, etc. denoted with proper nouns.

<table>
<thead>
<tr>
<th>President</th>
<th>Teddy</th>
<th>Roosevelt</th>
<th>Jr.</th>
<th>is</th>
<th>the</th>
<th>namesake</th>
<th>of</th>
<th>teddy</th>
<th>bears</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out</td>
<td>Begin</td>
<td>Inside</td>
<td>End</td>
<td>Out</td>
<td>Out</td>
<td>Out</td>
<td>Out</td>
<td>Out</td>
<td>Out</td>
</tr>
</tbody>
</table>
Variational Sequential Labeler (VSL)
VSLs & Notation

**Question**: What is a variational sequence learner (VSL)?

**Answer**: Sequence labeler based on the variational autoencoder architecture

Before we go further, let’s agree on some notation...

- $x_{1:T}$: Ordered word sequence of length $T$
- $x_t$: Word at position $t$ in $x_{1:T}$
- $x_{-t}$: Words in $x_{1:T}$ other than $x_t$
- $\ell_{1:T}$: One-to-one ground-truth sequential labels of the words in $x_{1:T}$
- $\ell_t$: Label for word $x_t$ at position $t$
- $h_t$: BiGRU output vector at position $t$
- $z$ & $y$: VAE latent variables
Variational Architectures

Chen et al. [1] propose three variational sequential labeler (VSL) architectures:

- VSL-G
- VSL-GG Flat
- VSL-GG Hierarchical

Note: Each “G” represents a single multivariate, Gaussian latent variables

We will go through each architecture individually focusing on the differences from the preceding architecture...
Base VSL-G(aussian) Architecture

Three Primary Components:
1. Bidirectional RNN (concatenation of word & character-level GRUs)
2. Variational Autoencoder with one Gaussian latent variable
3. Labeler classifier
BiGRU

- Input is a concatenation of word embedding for $x_t$ and hidden state of character-level BiGRU.
You may be wondering how the VSL uses the generative property of the VAE architecture. *It doesn’t.*

Variational architecture is used for its loss properties, in particular the regularizer.
Feedforward with no hidden layer and one output per label (same as baselines)

**Loss Function for \( \hat{l}_t \):**

\[
C(x_{1:T}, l_t) = \mathbb{E}_{z_t \sim q_\phi(\cdot|x_{1:T}, t)} \left[ -\log f(l_t|z_t) \right]
\]
Parametrization of Priors

The traditional approach assumes prior $p_\theta(z)$ to be multivariate, standard Gaussian.

Similar to recent work [5, 6, 7], Chen et al. attempt to learn the prior $p(z|x_{-t})$.

- Since VSLs are non-generative, requirement for simple prior can be loosened.

**Approach:** Update (per example) prior iteratively based on previous posterior

- *Initial State* ($k = 0$): Use standard multivariate Gaussian

- *Subsequent Update Steps*:

  $$p_\theta^{(k)}(z|x_{-t}) \approx \sum_x q_\phi^{(k-1)}(z|X_t = x, x_{-t}, t)p_{\text{data}}(X_t = x|x_{-t})$$

  - $p_{\text{data}}$: Empirical probability for $x$ given $x_{-t}$ in training set
  - $X_t$: Random variable corresponding to observed word at position $t$

**Question:** Any intuition for this formula? *Just marginalizing over $X_t$*
How is Unlabeled Data Used?

- Used to train variational autoencoder only, i.e., improve prediction of $x_T$
- No direct effect on labeler classifier
Variational Sequence Labeler (VSL) Loss

Combination of variational autoencoder & labeler losses

- **Variational Loss**: Tuned slightly based on VSL architecture

\[
U = \mathbb{E}_{z \sim q(\cdot)} \left\{ \log p_\theta(x | z) \right\} - D_{\text{KL}} \left( q_\phi(z | x_{1:T}) \parallel p_\theta(z) \right)
\]

- **Reconstruction Quality**
- **Distribution Divergence Regularizer**

- **Classifier Loss**: Same for all three architectures

\[
C(x_{1:T}, l_t) = \mathbb{E}_{z_t \sim q_\phi(\cdot | x_{1:T}, t)} \left[ -\log f(l_t | z_t) \right]
\]

- **Total Sequence Loss**: Same for all three architectures

\[
L(x_{1:T}, l_{1:t}) = \sum_{t=1}^{T} \left[ C(x_{1:T}, l_t) - \alpha U(x_{1:T}, t) \right]
\]

- **\( \alpha \)**: Trade-off hyperparameter tuned using validation set

**Question**: Why is the variational loss subtracted?
Downside of Variational Regularizers

Concerns with Variational Weights:

- Need to tune the weight of the variational regularizer
- Simple priors can hinder learning in early stages of training.

Solution: Weight annealing

- Iterative prior updating is an inherent form of weight annealing. Why?
- Add a weight to the $KL$ divergence terms in $U$ that starts at small value at increases to 1 over training
VSL-GG Flat Architecture

- Two multivariate Gaussian (GG) latent variables
- Variable $y$ used in both generator and classifier blocks
- Requires a second distribution divergence regularizer, $D_{KL}(q(y|x_{1:T}) \parallel p_\theta(\cdot))$, in variational loss $U$
**VSL-GG Hierarchical Architecture**

Single FF layer that yields $z$ from concatenation of $y$ & output of encoder not shown

**Intuition:** *Word-specific* latent information in $z$ may differ depending on *class specific* latent information in $y$
Experiments
Datasets

- **Named Entity Recognition Dataset:**
  - *Name:* CoNLL 2003 English [8]
  - *Labeling Scheme:* BIOES (Begin, Inside, Outside, End, Single)
  - *Size:* 17 million words (10% used for training)

- **Part of Speech Tagging:**
  - **Twitter POS**
    - # of Tags: 25
  - **Universal Dependencies (UD)**
    - # of Tags: 17
    - *Languages Used:* French, German, Indonesian, Spanish, Russian, Croatian (out of 17 total)
Results
## Named Entity Recognition Results

<table>
<thead>
<tr>
<th></th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F_1$</td>
<td>ULΔ</td>
</tr>
<tr>
<td>BiGRU Baseline</td>
<td>87.6</td>
<td>—</td>
</tr>
<tr>
<td>VSL-G</td>
<td>87.8</td>
<td>+0.1</td>
</tr>
<tr>
<td>VSL-GG-Flat</td>
<td>88.0</td>
<td>+0.1</td>
</tr>
<tr>
<td>VSL-GG-Hier</td>
<td><strong>88.4</strong></td>
<td><strong>+0.2</strong></td>
</tr>
</tbody>
</table>

- **ULΔ**: Improvement when using unlabeled data in training
- **Summary**: VSL-Hierarchical got the best results
- **Note**: Only this dataset had large generalization error and no (test) improvement with unlabeled data
## Twitter POS Results

<table>
<thead>
<tr>
<th></th>
<th>Validation</th>
<th></th>
<th>Test</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>ULΔ</td>
<td>Acc.</td>
<td>ULΔ</td>
</tr>
<tr>
<td>BiGRU Baseline</td>
<td>90.8</td>
<td>—</td>
<td>90.6</td>
<td>—</td>
</tr>
<tr>
<td>VSL-G</td>
<td>91.1</td>
<td>+0.1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>VSL-GG-Flat</td>
<td>91.4</td>
<td>+0.1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>VSL-GG-Hier</td>
<td><strong>91.6</strong></td>
<td><strong>+0.3</strong></td>
<td><strong>91.6</strong></td>
<td><strong>+0.3</strong></td>
</tr>
</tbody>
</table>

- **ULΔ**: Improvement when using unlabeled data in training
- **Summary**: VSL-Hierarchical got the best results. Total accuracy improvement: 1.3%
### Universal Dependencies POS Results

<table>
<thead>
<tr>
<th></th>
<th>German</th>
<th></th>
<th>French</th>
<th></th>
<th>Russian</th>
<th></th>
<th>Croatian</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>ULΔ</td>
<td>Acc.</td>
<td>ULΔ</td>
<td>Acc.</td>
<td>ULΔ</td>
<td>Acc.</td>
<td>ULΔ</td>
</tr>
<tr>
<td>NCRF</td>
<td>93.4</td>
<td>—</td>
<td>90.4</td>
<td>—</td>
<td>86.6</td>
<td>—</td>
<td>86.1</td>
<td>—</td>
</tr>
<tr>
<td>NCRF-AE</td>
<td>93.7</td>
<td>+0.2</td>
<td>90.8</td>
<td>+0.2</td>
<td>87.8</td>
<td>+1.1</td>
<td>87.9</td>
<td>+1.2</td>
</tr>
<tr>
<td>BiGRU Baseline</td>
<td>95.9</td>
<td>—</td>
<td>92.6</td>
<td>—</td>
<td>95.2</td>
<td>—</td>
<td>95.6</td>
<td>—</td>
</tr>
<tr>
<td>VSL-G</td>
<td>96.1</td>
<td>+0.0</td>
<td>92.8</td>
<td>+0.0</td>
<td>95.3</td>
<td>+0.0</td>
<td>95.6</td>
<td>+0.1</td>
</tr>
<tr>
<td>VSL-GG-Flat</td>
<td>96.1</td>
<td>+0.0</td>
<td>93.0</td>
<td>+0.1</td>
<td>95.5</td>
<td>+0.1</td>
<td>95.8</td>
<td>+0.1</td>
</tr>
<tr>
<td>VSL-GG-Hier</td>
<td><strong>96.4</strong></td>
<td><strong>+0.1</strong></td>
<td><strong>93.3</strong></td>
<td><strong>+0.1</strong></td>
<td><strong>95.9</strong></td>
<td><strong>+0.1</strong></td>
<td><strong>96.3</strong></td>
<td><strong>+0.2</strong></td>
</tr>
</tbody>
</table>

- NCRF & NCRF-AE (from [9]) natively support semi-supervised learning
- Only four of six language results shown
Discussion of Results & Experiments
t-SNE Digression

Question: Is anyone here familiar with t-SNE [10]?

Answer: Dimensionality reduction technique almost exclusively used for visualizing data.

Advantages of t-SNE:
- Often useful for visually building intuitions about the non-linear separability of data
- Widespread usage in research publications
- Scales to large datasets (e.g., millions of samples)

Disadvantages of t-SNE:
- Does not learn a function from high dimensional to low dimensional space (i.e., mapping cannot be applied to new data)
- Not distance based but instead relies on labeling probabilities
Each point represents one Twitter POS sample & color is the point’s label.

- Color separation hints labeler *may* perform better over that latent vector.
- Which results visually looks like it would perform best?
- These plots represent <1% accuracy diff. *Take these plots with a grain of salt*.
Variational vs. “Vanilla” Sequence Labeling

**Question**: What are the quantifiable benefits of using a variational framework over a “vanilla” autoencoder?

Switching to a “vanilla” autoencoder required two architectural changes:

- Removal of *variational regularization*
  - KL divergence(s) in objective function like $D_{KL}(q(z_t|x_{1:T}, t) \parallel p(z_t|x_{-t}))$
- Removal of the randomness (e.g., reparametrization trick) from the autoencoder
Performance Benefit of Variational Architecture

<table>
<thead>
<tr>
<th></th>
<th>Twitter</th>
<th>NER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>no VR</td>
</tr>
<tr>
<td>BiGRU baseline</td>
<td>90.8</td>
<td>–</td>
</tr>
<tr>
<td>VSL-G</td>
<td>91.1</td>
<td>90.9</td>
</tr>
<tr>
<td>VSL-GG-Flat</td>
<td>91.4</td>
<td>90.9</td>
</tr>
<tr>
<td>VSL-GG-Hier</td>
<td>91.6</td>
<td>91.0</td>
</tr>
</tbody>
</table>

“no VR” represents non-variational architecture

Takeaways:
- When one latent variable (i.e., no $y_t$), variational network has little benefit (0.1%).
- Max Accuracy Improvement due to Variational Architecture: 0.5%–0.6%
Effect of Unlabeled Set Size

Relationship between size of unlabeled set and POS % accuracy on Twitter dataset.

Takeaways:
- Accuracy improvement is very minimal <0.2%
- Improvement plateaus quickly after ~3K unlabeled examples
Discussion of Results & Experiments

1. Review of Foundational Concepts
   - Learning
   - “Vanilla” & Variational Autoencoders
   - Sequence Labeling

2. Variational Sequential Labeler (VSL)
   - Notation
   - Architectures
     - VSL-G
     - VSL-GG Flat
     - VSL-GG Hierarchical

3. Experiments

4. Results

5. Discussion of Results & Experiments
   - Effect of Latent Hierarchy
   - Effect of Variational Regularization
   - Effect of Unlabeled Data

6. References
References


X. Yang, “Understanding the variational lower bound.”

“Parts of speech and functions: ”bob made a book collector happy the other day”,” Jan 2015.


References II


$F_1$ Score

$F_1$ Score:

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$