Shogun Workshop 2013

Structured Output Learning

Fernando José Iglesias García
fernando.iglesias@shogun-toolbox.org

July 12, 2013

About me

- Involved with Shogun for about one and half years.
- Participated in two GSoC:
  - 2012 - Structured output learning (this talk!).
  - Ongoing this summer - Distance metric learning.

Outline

Introduction
- Structured Output Learning
- Examples

Outline

Outline

Outline

Structured Output Learning

The Structured Output (SO) setting presented here is an instance of supervised learning.

Structured Output Learning

The Structured Output (SO) setting presented here is an instance of supervised learning.

- The data is composed of pairs (input, output), or (observation, label), generally denoted by

$$(x, y) \quad \text{where } x \in \mathcal{X},\ y \in \mathcal{Y}.$$

Structured Output Learning

The Structured Output (SO) setting presented here is an instance of supervised learning.

- The data is composed of pairs (input, output), or (observation, label), generally denoted by

$$(x, y) \quad \text{where } x \in \mathcal{X},\ \ y \in \mathcal{Y}.$$

- The aim is to obtain a function which maps observations to labels,

$$f : \mathcal{X} \to \mathcal{Y}.$$

Structured Output Learning

The Structured Output (SO) setting presented here is an instance of supervised learning.

- The data is composed of pairs (input, output), or (observation, label), generally denoted by

$$(x, y) \quad \text{where } x \in \mathcal{X},\ \ y \in \mathcal{Y}.$$

- The aim is to obtain a function which maps observations to labels,

$$f : \mathcal{X} \to \mathcal{Y}.$$

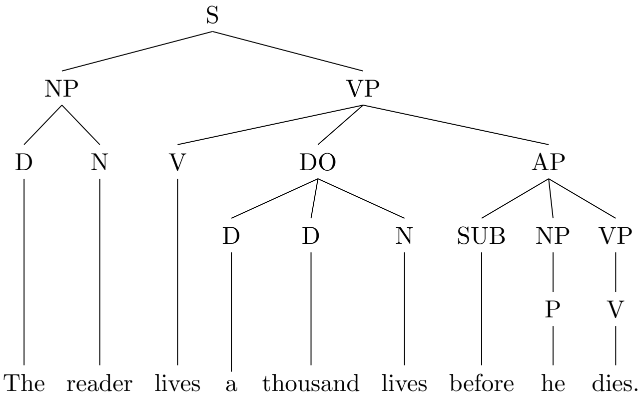
- To obtain this function f, we use *training* data $\{(x_i, y_i)\}_{i=1}^{n}$.

Structured Output Learning

- Non-structured output learning
  - The inputs in $\mathcal{X}$ can be any kind of objects.
  - The outputs in $\mathcal{Y}$ are singular, each label is represented by a single number.
- Structured output learning
  - The inputs in $\mathcal{X}$ can be any kind of objects.
  - The outputs in $\mathcal{Y}$ are complex, each label has multiple variables.

Natural Language Processing (NLP) – Parse trees

- $\mathcal{X}$ = sentences.
- $\mathcal{Y}$ = parse trees.



Computer Vision – Segmentation

- $\mathcal{X}$ = images.
- $\mathcal{Y}$ = segmented images.

Structured Prediction and Learning in Computer Vision (Nowozin and Lampert, 2011).

Structured Output SVM (SO-SVM or SSVM)

Structured Output SVM (SO-SVM or SSVM)

- The objects $y \in \mathcal{Y}$ are composed of multiple random variables (Tsochantaridis et al., 2005).

Structured Output SVM (SO-SVM or SSVM)

- The objects $y \in \mathcal{Y}$ are composed of multiple random variables (Tsochantaridis et al., 2005).
- Approaches to extend the SVM to structured learning:

Structured Output SVM (SO-SVM or SSVM)

- The objects $y \in \mathcal{Y}$ are composed of multiple random variables (Tsochantaridis et al., 2005).
- Approaches to extend the SVM to structured learning:
  - Consider each $y \in \mathcal{Y}$ as a different class.

Structured Output SVM (SO-SVM or SSVM)

- The objects $y \in \mathcal{Y}$ are composed of multiple random variables (Tsochantaridis et al., 2005).
- Approaches to extend the SVM to structured learning:
  - Consider each $y \in \mathcal{Y}$ as a different class. ☹
    - Too many classes!
    - E.g.: $\mathcal{Y}$ = binary sequences of length 250 $\rightarrow 2^{250}$ distinct sequences.
    - $2^{250} = 2^{10 \times 25} > 1000^{25} = 10^{75}$ (there are $\approx 10^{80}$ atoms in the universe).

Structured Output SVM (SO-SVM or SSVM)

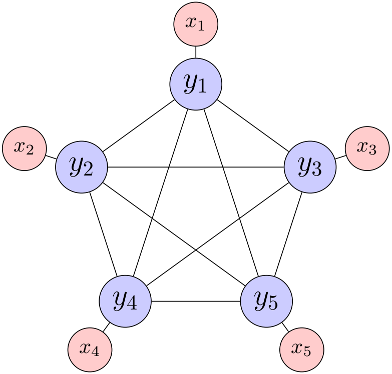
- The objects $y \in \mathcal{Y}$ are composed of multiple random variables (Tsochantaridis et al., 2005).
- Approaches to extend the SVM to structured learning:
  - Consider each $y \in \mathcal{Y}$ as a different class. ☹
    - Too many classes!
    - E.g.: $\mathcal{Y}$ = binary sequences of length 250 $\rightarrow 2^{250}$ distinct sequences.
    - $2^{250} = 2^{10 \times 25} > 1000^{25} = 10^{75}$ (there are $\approx 10^{80}$ atoms in the universe).
  - Map pairs of (x, y) jointly onto a fixed dimensional space.

Structured Output SVM (SO-SVM or SSVM)

- The objects $y \in \mathcal{Y}$ are composed of multiple random variables (Tsochantaridis et al., 2005).
- Approaches to extend the SVM to structured learning:
  - Consider each $y \in \mathcal{Y}$ as a different class. ☹
    - Too many classes!
    - E.g.: $\mathcal{Y}$ = binary sequences of length 250 $\rightarrow 2^{250}$ distinct sequences.
    - $2^{250} = 2^{10\times 25} > 1000^{25} = 10^{75}$ (there are $\approx 10^{80}$ atoms in the universe).
  - Map pairs of (x, y) jointly onto a fixed dimensional space. ☺
    - It becomes a geometrical problem, where SVM principles can be applied.

Joint feature map

- Observation-label interactions.
- Label-label interactions.



$$\psi(x, y) = \begin{bmatrix} \sum_i \psi_u(x_i, y_i) \\ \\ \sum_{i,j} \psi_p(y_i, y_j) \end{bmatrix}$$

SO-SVM prediction: inference or arg max

Given an observation $x \in \mathcal{X}$, the SO-SVM finds its corresponding label $y^* \in \mathcal{Y}$ maximizing a score:

$$y^* = \arg\max_{y \in \mathcal{Y}} \left(\mathbf{w}^T \psi(x, y) \right) .$$

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

- Exhaustive enumeration is not feasible.

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

- Exhaustive enumeration is not feasible.
- We have to exploit the structure of $\mathcal{Y}$.

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

- Exhaustive enumeration is not feasible.
- We have to exploit the structure of $\mathcal{Y}$.
  - If the labels $y \in \mathcal{Y}$ are sequences, Viterbi.

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

- Exhaustive enumeration is not feasible.
- We have to exploit the structure of $\mathcal{Y}$.
    - If the labels $y \in \mathcal{Y}$ are sequences, Viterbi.
    - For $y \in \mathcal{Y}$ trees, CYK (Cocke-Younger-Kasami).

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

- Exhaustive enumeration is not feasible.
- We have to exploit the structure of $\mathcal{Y}$.
  - If the labels $y \in \mathcal{Y}$ are sequences, Viterbi.
  - For $y \in \mathcal{Y}$ trees, CYK (Cocke-Younger-Kasami).
  - For $y \in \mathcal{Y}$ graphs, no general efficient algorithm (Koller and Friedman, 2009).

How is the $\arg\max_{y \in \mathcal{Y}}$ actually solved?

- Exhaustive enumeration is not feasible.
- We have to exploit the structure of $\mathcal{Y}$.
  - If the labels $y \in \mathcal{Y}$ are sequences, Viterbi.
  - For $y \in \mathcal{Y}$ trees, CYK (Cocke-Younger-Kasami).
  - For $y \in \mathcal{Y}$ graphs, no general efficient algorithm (Koller and Friedman, 2009).
    - Use approximations.
    - Find a subset within $\mathcal{Y}$.

Structured or delta loss: Δ

Consider again the label sequence learning example. During training, the SO-SVM may predict two different labels for the same input:

Structured or delta loss: Δ

Consider again the label sequence learning example. During training, the SO-SVM may predict two different labels for the same input:

- One that is completely different from the ground truth label.
- Another one that only differs in one element.

Structured or delta loss: Δ

Consider again the label sequence learning example. During training, the SO-SVM may predict two different labels for the same input:

- One that is completely different from the ground truth label.
- Another one that only differs in one element.

The traditional zero-one loss ranks both predictions in the same way.

Structured or delta loss: Δ

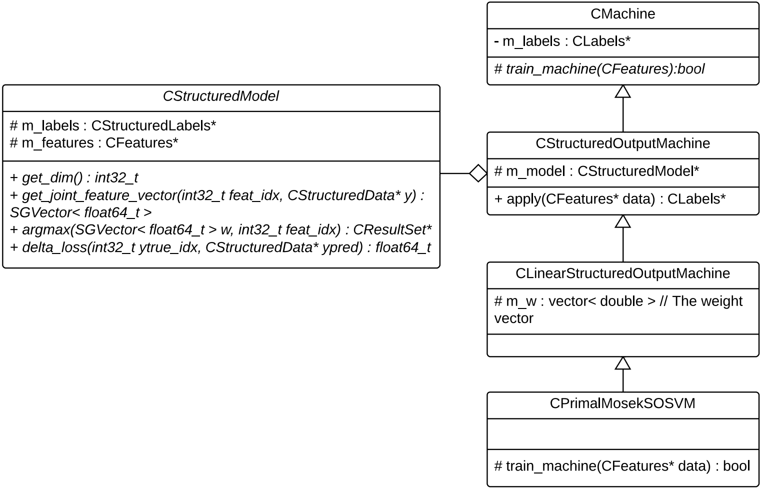
Consider again the label sequence learning example. During training, the SO-SVM may predict two different labels for the same input:

- One that is completely different from the ground truth label.
- Another one that only differs in one element.

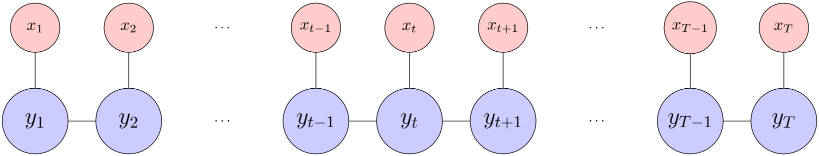
The traditional zero-one loss ranks both predictions in the same way.

- This is not a good idea.
- Use instead a loss that takes into account the structure of the labels. Typically, the Hamming loss is used.

Simplified class hierarchy



Label sequence learning



$$y^* = \arg\max_{y \in \mathcal{Y}} \left(\mathbf{w}^T \psi(x, y) \right) \rightarrow \text{Viterbi}$$

C++ example: training data setup

```
// Create binary label sequences
CSequenceLabels* labels =
    new CSequenceLabels(num_examples);
// Fill in the sequences, one could look like [0 0 1 1]

// Create matrix features
CMatrixFeatures<float>* features =
    new CMatrixFeatures<float>(num_examples);
// Fill in the features, one could look like
//     [0 1 2 1]
//     [1 2 0 2]
//     [2 1 0 0]
```

C++ example: training

```
CHMSVMModel* model =
    new CHMSVMModel(features, labels, SMT_TWO_STATE);
CPrimalMosekSOSVM* sosvm =
    new CPrimalMosekSOSVM(model, labels);
sosvm->train();

sosvm->get_w().display_vector("w");
sosvm->get_slacks().display_vector("slacks");

// Free memory
SG_UNREF(sosvm);
```

Python example

```
from shogun.Structure  import TwoStateModel, \
    PrimalMosekSOSVM
from shogun.Evaluation import StructuredAccuracy

model = TwoStateModel.simulate_data(num_examples, \
    example_length, num_features, num_noise_features)

sosvm = PrimalMosekSOSVM(model, model.get_labels())
sosvm.train()
predicted = sosvm.apply()

evaluator = StructuredAccuracy()
acc = evaluator.evaluate(predicted, labels)
print('Train accuracy = %.4f' % acc)
```

Extensions

Some possible extensions of the SO learning framework are twofold:

Extensions

Some possible extensions of the SO learning framework are twofold:

- Parameter estimation or learning (aka training).
  - Solvers in the dual to leverage kernels, online solvers, other approaches different from margin maximization, etc.

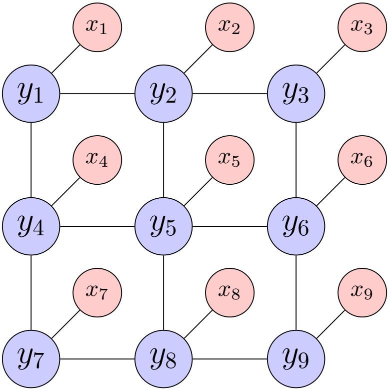
Extensions

Some possible extensions of the SO learning framework are twofold:

- Parameter estimation or learning (aka training).
  - Solvers in the dual to leverage kernels, online solvers, other approaches different from margin maximization, etc.
- Structured models.
  - What if there is no structured model to represent the labels of my application at hand?
  - SWIG director classes may help.

New structured model: motivation

Graph labelling for image segmentation.



SWIG directors example: inheritance

```
from shogun.Structure import DirectorStructuredModel

class MyGridGraphModel(DirectorStructuredModel):

  def get_joint_feature_vector(self, feat_idx, y):

  def delta_loss(self, y1, y2):

  def argmax(self, w, feat_idx, training):
```

SWIG directors example: using the new model

```
from subgradient_sosvm import StochasticSubgradientSOSVM

model = MyGridGraphModel(train_features, train_labels)
sosvm = StochasticSubgradientSOSVM(model, train_labels)
sosvm.train()
predicted = sosvm.apply(test_features)

evaluator = StructuredAccuracy()
acc = evaluator.evaluate(predicted, test_labels)

print('Test accuracy = %.4f' % acc)
```

Thank you for coming!
Questions, doubts and suggestions are most welcome!!

References

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Secaucus, NJ, USA, 2006.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(2):265–292, 2002.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Sebastian Nowozin and Christoph H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4):185–365, 2011.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.