Part 5: Structured Support Vector Machines

Sebastian Nowozin and Christoph H. Lampert

Providence, 21st June 2012
Problem (Loss-Minimizing Parameter Learning)

Let \( d(x, y) \) be the (unknown) true data distribution.
Let \( D = \{(x^1, y^1), \ldots, (x^N, y^N)\} \) be i.i.d. samples from \( d(x, y) \).
Let \( \phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^D \) be a feature function.
Let \( \Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \) be a loss function.

▶ Find a weight vector \( w^* \) that leads to minimal expected loss

\[
\mathbb{E}_{(x,y) \sim d(x,y)} \{ \Delta(y, f(x)) \}
\]

for \( f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle \).

Pro:

▶ We directly optimize for the quantity of interest: expected loss.
▶ No expensive-to-compute partition function \( Z \) will show up.

Con:

▶ We need to know the loss function already at training time.
▶ We can’t use probabilistic reasoning to find \( w^* \).
Reminder: learning by regularized risk minimization

For compatibility function $g(x, y; w) := \langle w, \phi(x, y) \rangle$ find $w^*$ that minimizes

$$
E_{(x,y) \sim d(x,y)} \Delta(y, \text{argmax}_y g(x, y; w)).
$$

Two major problems:

- $d(x, y)$ is unknown
- $\text{argmax}_y g(x, y; w)$ maps into a discrete space
  - $\Delta(y, \text{argmax}_y g(x, y; w))$ is discontinuous, piecewise constant
Task:

$$\min_w \mathbb{E}_{(x,y) \sim d(x,y)} \Delta( y, \text{argmax}_y g(x, y; w) ).$$

Problem 1:

- $d(x, y)$ is unknown

Solution:

- Replace $\mathbb{E}_{(x,y) \sim d(x,y)} ( \cdot )$ with empirical estimate $\frac{1}{N} \sum_{(x^n,y^n)} ( \cdot )$
- To avoid overfitting: add a regularizer, e.g. $\lambda \|w\|^2$.

New task:

$$\min_w \lambda \|w\|^2 + \frac{1}{N} \sum_{n=1}^{N} \Delta( y^n, \text{argmax}_y g(x^n, y; w) ).$$
Task:

\[
\min_w \lambda \|w\|^2 + \frac{1}{N} \sum_{n=1}^{N} \Delta(y^n, \text{argmax}_y g(x^n, y; w))
\]

Problem:

- \( \Delta(y, \text{argmax}_y g(x, y; w)) \) discontinuous w.r.t. \( w \).

Solution:

- Replace \( \Delta(y, y') \) with well behaved \( \ell(x, y, w) \).
- Typically: \( \ell \) upper bound to \( \Delta \), continuous and convex w.r.t. \( w \).

New task:

\[
\min_w \lambda \|w\|^2 + \frac{1}{N} \sum_{n=1}^{N} \ell(x^n, y^n, w))
\]
Regularized Risk Minimization

\[
\min_w \quad \lambda \|w\|^2 + \frac{1}{N} \sum_{n=1}^{N} \ell(x^n, y^n, g)
\]

Regularization + Loss on training data

Hinge loss: maximum margin training

\[
\ell(x^n, y^n, w) := \max_{y \in \mathcal{Y}} \left[ \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle \right]
\]

- \(\ell\) is maximum over linear functions \(\rightarrow\) continuous, convex.
- \(\ell\) bounds \(\Delta\) from above.

Proof: Let \(\bar{y} = \arg\max_y g(x, y, w)\)

\[
\Delta(y^n, \bar{y}) \leq \Delta(y^n, \bar{y}) + g(x^n, \bar{y}, w) - g(x^n, y^n, w) \\
\leq \max_{y \in \mathcal{Y}} \left[ \Delta(y^n, y) + g(x^n, y, w) - g(x^n, y^n, w) \right]
\]
Structured Output Support Vector Machine

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \left[ \max_{y \in Y} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^*) \rangle \right]$$

Conditional Random Field

$$\min_w \frac{\|w\|^2}{2\sigma^2} + \sum_{n=1}^{N} \left[ \log \sum_{y \in Y} \exp \left( \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^*) \rangle \right) \right]$$

CRFs and SSVMs have more in common than usually assumed.
- both do regularized risk minimization
- $\log \sum_{y} \exp(\cdot)$ can be interpreted as a *soft-max*
Solving the Training Optimization Problem Numerically

Structured Output Support Vector Machine:

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \left[ \max_{y \in Y} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle \right]$$

Unconstrained optimization, convex, non-differentiable objective.
Structured Output SVM (equivalent formulation):

$$\min_{w, \xi} \quad \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi^n$$

subject to, for $n = 1, \ldots, N$,

$$\max_{y \in \mathcal{Y}} \left[ \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle \right] \leq \xi^n$$

non-linear constraints, convex, differentiable objective.
Structured Output SVM (also equivalent formulation):

\[
\min_{w, \xi} \quad \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi^n
\]

subject to, for \(n = 1, \ldots, N\),

\[
\Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle \leq \xi^n, \quad \text{for all } y \in \mathcal{Y}
\]

\(N|\mathcal{Y}|\) linear constraints, convex, differentiable objective.
Example: Multiclas SVM

\[ \mathcal{Y} = \{1, 2, \ldots, K\}, \quad \Delta(y, y') = \begin{cases} 1 & \text{for } y \neq y' \\ 0 & \text{otherwise} \end{cases} \]

\[ \phi(x, y) = \left( [y = 1] \phi(x), [y = 2] \phi(x), \ldots, [y = K] \phi(x) \right) \]

Solve:
\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^{N} \xi^n
\]
subject to, for \( i = 1, \ldots, n \),
\[
\langle w, \phi(x^n, y^n) \rangle - \langle w, \phi(x^n, y) \rangle \geq 1 - \xi^n \quad \text{for all } y \in \mathcal{Y} \setminus \{y^n\}.
\]

Classification:
\[ f(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle. \]
Example: Hierarchical SVM

Hierarchical Multiclass Loss:

\[ \Delta(y, y') := \frac{1}{2} \text{(distance in tree)} \]
\[ \Delta(\text{cat, cat}) = 0, \quad \Delta(\text{cat, dog}) = 1, \]
\[ \Delta(\text{cat, bus}) = 2, \quad etc. \]

Solve:

\[
\min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi^n
\]

subject to, for \( i = 1, \ldots, n, \)

\[
\langle w, \phi(x^n, y^n) \rangle - \langle w, \phi(x^n, y) \rangle \geq \Delta(y^n, y) - \xi^n \quad \text{for all } y \in \mathcal{Y}.
\]

[A. Binder, K.-R. Müller, M. Kawanabe: "On taxonomies for multi-class image categorization", IJCV, 2011]
Solving the Training Optimization Problem Numerically

We can solve SSVM training like CRF training:

\[
\min_w \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \left[ \max_{y \in Y} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle \right]
\]

- continuous 😊
- unconstrained 😊
- convex 😊
- non-differentiable 😞
  → we can’t use gradient descent directly.
  → we’ll have to use subgradients
**Definition**

Let \( f : \mathbb{R}^D \rightarrow \mathbb{R} \) be a convex, not necessarily differentiable, function.

A vector \( v \in \mathbb{R}^D \) is called a **sub-gradient** of \( f \) at \( w_0 \), if

\[
f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.
\]

For differentiable \( f \), the gradient \( v = \nabla f(w_0) \) is the only subgradient.
Sub-gradient descent works basically like gradient descent:

**Sub-gradient Descent Minimization – minimize** $F(w)$

1. **require**: tolerance $\epsilon > 0$
2. $w_{\text{cur}} \leftarrow 0$
3. **repeat**
   - $v \in \nabla_{\text{sub}} F(w_{\text{cur}})$
   - $\eta \leftarrow \argmin_{\eta \in \mathbb{R}} F(w_{\text{cur}} - \eta v)$
   - $w_{\text{cur}} \leftarrow w_{\text{cur}} - \eta v$
4. **until** $\|v\| < \epsilon$
5. **return** $w_{\text{cur}}$

Converges to global minimum, but rather inefficient if $F$ non-differentiable.

Computing a subgradient:

$$\min_w \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \ell^n(w)$$

with $$\ell^n(w) = \max_y \ell^n_y(w)$$, and

$$\ell^n_y(w) := \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle$$

Subgradient of $$\ell^n$$ at $$w_0$$: find maximal (active) $$y$$, use $$v = \nabla \ell^n_y(w_0)$$. 
Subgradient Descent S-SVM Training

**input** training pairs \{\((x^1, y^1), \ldots, (x^n, y^n)\) \} \subset \mathcal{X} \times \mathcal{Y},
**input** feature map \(\phi(x, y)\), loss function \(\Delta(y, y')\), regularizer \(C\),
**input** number of iterations \(T\), stepsizes \(\eta_t\) for \(t = 1, \ldots, T\)

1: \(w \leftarrow 0\)
2: \(\textbf{for } t=1, \ldots, T \textbf{ do}\)
3: \(\textbf{for } i=1, \ldots, n \textbf{ do}\)
4: \(\hat{y} \leftarrow \arg\max_{y \in \mathcal{Y}} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle\)
5: \(v^n \leftarrow \phi(x^n, \hat{y}) - \phi(x^n, y^n)\)
6: \(\textbf{end for}\)
7: \(w \leftarrow w - \eta_t(w - \frac{C}{N} \sum_n v^n)\)
8: \(\textbf{end for}\)

**output** prediction function \(f(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle\).

Observation: each update of \(w\) needs 1 \(\arg\max\)-prediction per example.
We can use the same tricks as for CRFs, e.g. **stochastic updates**:

**Stochastic Subgradient Descent S-SVM Training**

**input** training pairs \( \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y} \),

**input** feature map \( \phi(x, y) \), loss function \( \Delta(y, y') \), regularizer \( C \),

**input** number of iterations \( T \), stepsizes \( \eta_t \) for \( t = 1, \ldots, T \)

1: \( w \leftarrow \vec{0} \)
2: **for** \( t = 1, \ldots, T \) **do**
3: \( (x^n, y^n) \leftarrow \) randomly chosen training example pair
4: \( \hat{y} \leftarrow \arg\max_{y \in \mathcal{Y}} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle - \langle w, \phi(x^n, y^n) \rangle \)
5: \( w \leftarrow w - \eta_t (w - \frac{C}{N} [\phi(x^n, \hat{y}) - \phi(x^n, y^n)]) \)
6: **end for**

**output** prediction function \( f(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle \).

Observation: each update of \( w \) needs only 1 argmax-prediction (but we’ll need many iterations until convergence)
Solving the Training Optimization Problem Numerically

We can solve an S-SVM like a linear SVM:

One of the equivalent formulations was:

$$\min_{w \in \mathbb{R}^D, \xi \in \mathbb{R}^N_+} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi^n$$

subject to, for $i = 1, \ldots, n$,

$$\langle w, \phi(x^n, y^n) \rangle - \langle w, \phi(x^n, y) \rangle \geq \Delta(y^n, y) - \xi^n, \quad \text{for all } y \in Y'.$$

Introduce feature vectors $\delta \phi(x^n, y^n, y) := \phi(x^n, y^n) - \phi(x^n, y)$. 
Solve

\[
\min_{w \in \mathbb{R}^D, \xi \in \mathbb{R}_+^n} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi^n
\]

subject to, for \(i = 1, \ldots, n\), for all \(y \in \mathcal{Y}\),

\[
\langle w, \delta \phi(x^n, y^n, y) \rangle \geq \Delta(y^n, y) - \xi^n.
\]

This has the same structure as an ordinary SVM!

- quadratic objective 😊
- linear constraints 😊

**Question:** Can’t we use a ordinary SVM/QP solver?

**Answer:** Almost! We could, if there weren’t \(N|\mathcal{Y}|\) constraints.

- E.g. 100 binary 16 × 16 images: \(10^{79}\) constraints
Solution: working set training

- It’s enough if we enforce the **active constraints**. The others will be fulfilled automatically.
- We don’t know which ones are active for the optimal solution.
- But it’s likely to be only a small number → can of course be formalized.

Keep a set of potentially active constraints and update it iteratively:

### Working Set Training

- Start with working set $S = \emptyset$ (no contraints)
- Repeat until convergence:
  - Solve S-SVM training problem with constraints from $S$
  - Check, if solution violates any of the **full** constraint set
    - if no: we found the optimal solution, **terminate**.
    - if yes: add most violated constraints to $S$, **iterate**.

**Good practical performance and theoretic guarantees:**

- polynomial time convergence $\epsilon$-close to the global optimum

Working Set S-SVM Training

**input** training pairs \{((x^1, y^1), \ldots, (x^n, y^n)) \subset X \times Y, input feature map \phi(x, y), loss function \Delta(y, y'), regularizer C

1: \text{\(S \leftarrow \emptyset\)}
2: \text{repeat}
3: \((w, \xi) \leftarrow \text{solution to QP only with constraints from } S\)
4: \text{for } i=1, \ldots, n \text{ do}
5: \ \hat{y} \leftarrow \text{argmax}_{y \in Y} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle
6: \ \text{if } \hat{y} \neq y^n \text{ then}
7: \ \ S \leftarrow S \cup \{ (x^n, \hat{y}) \} 
8: \ \text{end if}
9: \ \text{end for}
10: \text{until } S \text{ doesn't change anymore.}

**output** prediction function \( f(x) = \text{argmax}_{y \in Y} \langle w, \phi(x, y) \rangle. \)

Observation: each update of \(w\) needs \(1\ \text{argmax}\)-prediction per example. (but we solve globally for next \(w\), not by local steps)
One-Slack Formulation of S-SVM:
(equivalent to ordinary S-SVM formulation by \( \xi = \frac{1}{N} \sum_n \xi^n \))

\[
\min_{w \in \mathbb{R}^D, \xi \in \mathbb{R}^+} \frac{1}{2} \|w\|^2 + C\xi
\]

subject to, for all \((\hat{y}^1, \ldots, \hat{y}^N) \in \mathcal{Y} \times \cdots \times \mathcal{Y},\)

\[
\sum_{n=1}^N \left[ \Delta(y^n, \hat{y}^N) + \langle w, \phi(x^n, \hat{y}^n) \rangle - \langle w, \phi(x^n, y^n) \rangle \right] \leq N \xi,
\]

\(|\mathcal{Y}|^N\) linear constraints, convex, differentiable objective.

We blew up the constraint set even further:

- 100 binary \(16 \times 16\) images: \(10^{177}\) constraints (instead of \(10^{79}\)).
Working Set One-Slack S-SVM Training

**input** training pairs \( \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset \mathcal{X} \times \mathcal{Y} \),

**input** feature map \( \phi(x, y) \), loss function \( \Delta(y, y') \), regularizer \( C \)

1. \( S \leftarrow \emptyset \)
2. repeat
3. \( (w, \xi) \leftarrow \text{solution to QP only with constraints from } S \)
4. for \( i=1, \ldots, n \) do
5. \( \hat{y}^n \leftarrow \arg\max_{y \in \mathcal{Y}} \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle \)
6. end for
7. \( S \leftarrow S \cup \{((x^1, \ldots, x^n), (\hat{y}^1, \ldots, \hat{y}^n))\} \)
8. until \( S \) doesn’t change anymore.

**output** prediction function \( f(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle \).

Often faster convergence:
We add one *strong* constraint per iteration instead of \( n \) weak ones.
We can solve an S-SVM like a non-linear SVM: compute Lagrangian dual

- min becomes max,
- original (primal) variables $w, \xi$ disappear,
- new (dual) variables $\alpha_{iy}$: one per constraint of the original problem.

**Dual S-SVM problem**

$$\max_{\alpha \in \mathbb{R}^{|Y|}} \sum_{n=1, \ldots, N} \alpha_{ny} \Delta(y^n, y) - \frac{1}{2} \sum_{n,\bar{n}=1, \ldots, N} \alpha_{ny} \alpha_{\bar{n}\bar{y}} \langle \delta\phi(x^n, y^n, y), \delta\phi(x^{\bar{n}}, y^{\bar{n}}, \bar{y}) \rangle$$

subject to, for $n = 1, \ldots, N$,

$$\sum_{y \in Y} \alpha_{ny} \leq \frac{C}{N}.$$
We can kernelize:

- Define joint kernel function \( k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R} \)

\[
k( (x, y), (\bar{x}, \bar{y}) ) = \langle \phi(x, y), \phi(\bar{x}, \bar{y}) \rangle.
\]

- \( k \) measure similarity between two (input,output)-pairs.

- We can express the optimization in terms of \( k \):

\[
\langle \delta \phi(x^n, y^n, y), \delta \phi(x^\bar{n}, y^\bar{n}, \bar{y}) \rangle \\
= \langle \phi(x^n, y^n) - \phi(x^n, y), \phi(x^\bar{n}, y^\bar{n}) - \phi(x^\bar{n}, \bar{y}) \rangle \\
= \langle \phi(x^n, y^n), \phi(x^\bar{n}, y^\bar{n}) \rangle - \langle \phi(x^n, y^n), \phi(x^\bar{n}, \bar{y}) \rangle \\
- \langle \phi(x^n, y), \phi(x^\bar{n}, y^\bar{n}) \rangle + \langle \phi(x^n, y), \phi(x^\bar{n}, \bar{y}) \rangle \\
= k((x^n, y^n), (x^\bar{n}, y^\bar{n})) - k((x^n, y^n), \phi(x^\bar{n}, \bar{y})) \\
- k((x^n, y), (x^\bar{n}, y^\bar{n})) + k((x^n, y), \phi(x^\bar{n}, \bar{y})) \\
= K_{i\bar{i}y\bar{y}}
\]
Kernelized S-SVM problem:

$$\max_{\alpha \in \mathbb{R}^{|\mathcal{Y}|}_+} \sum_{i=1,\ldots,n} \alpha_{iy} \Delta(y^n, y) - \frac{1}{2} \sum_{y,\bar{y} \in \mathcal{Y}} \alpha_{iy} \alpha_{i\bar{y}} K_{iy\bar{y}}$$

subject to, for $i = 1, \ldots, n$,

$$\sum_{y \in \mathcal{Y}} \alpha_{iy} \leq \frac{C}{N}.$$ 

- too many variables: train with working set of $\alpha_{iy}$.

Kernelized prediction function:

$$f(x) = \arg\max_{y \in \mathcal{Y}} \sum_{iy'} \alpha_{iy'} k((x_i, y_i), (x, y))$$
What do "joint kernel functions" look like?

\[ k((x, y), (\bar{x}, \bar{y})) = \langle \phi(x, y), \phi(\bar{x}, \bar{y}) \rangle. \]

As in **graphical model**: easier if \( \phi \) decomposes w.r.t. factors:

- \( \phi(x, y) = (\phi_F(x, y_F))_{f \in \mathcal{F}} \)

Then the kernel \( k \) decomposes into sum over factors:

\[
k((x, y), (\bar{x}, \bar{y})) = \left\langle \left( \phi_F(x, y_F) \right)_{f \in \mathcal{F}}, \left( \phi_F(x', y'_F) \right)_{f \in \mathcal{F}} \right\rangle \\
= \sum_{f \in \mathcal{F}} \left\langle \phi_F(x, y_F), \phi_F(x', y'_F) \right\rangle \\
= \sum_{f \in \mathcal{F}} k_F((x, y_F), (x', y'_F))
\]

We can define kernels for each factor (e.g. nonlinear).
Example: figure-ground segmentation with grid structure

\[(x, y) \hat{=} (x', y')\]

Typical kernels: arbitrary in \(x\), linear (or at least simple) w.r.t. \(y\):

- **Unary factors:**
  \[k_p((x_p, y_p), (x'_p, y'_p)) = k(x_p, x'_p) [y_p = y'_p]\]
  with \(k(x_p, x'_p)\) local image kernel, e.g. \(\chi^2\) or histogram intersection

- **Pairwise factors:**
  \[k_{pq}((y_p, y_q), (y'_p, y'_q)) = [y_q = y'_q] [y_q = y'_q]\]

More powerful than all-linear, and argmax-prediction still possible.
Example: object localization

\[
(x, y) \overset{\sim}{=} (\text{left}, \text{top}), \quad (x', y') \overset{\sim}{=} (\text{right}, \text{bottom})
\]

Only one factor that includes all \(x\) and \(y\):

\[
k((x, y), (x', y')) = k_{\text{image}}(x|y, x'|y')
\]

with \(k_{\text{image}}\) image kernel and \(x|y\) is image region within box \(y\).

argmax-prediction as difficult as object localization with \(k_{\text{image}}\)-SVM.
Summary – S-SVM Learning

Given:

- training set \( \{(x^1, y^1), \ldots, (x^n, y^n)\} \subset X \times Y \)
- loss function \( \Delta : Y \times Y \rightarrow \mathbb{R} \).

Task: learn parameter \( w \) for \( f(x) := \text{argmax}_y \langle w, \phi(x, y) \rangle \) that minimizes expected loss on future data.

S-SVM solution derived by maximum margin framework:

- enforce correct output to be better than others by a margin:
  \[
  \langle w, \phi(x^n, y^n) \rangle \geq \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle \quad \text{for all } y \in Y.
  \]

- convex optimization problem, but non-differentiable
- many equivalent formulations \( \rightarrow \) different training algorithms
- training needs repeated \( \text{argmax} \) prediction, no probabilistic inference
Extra I: Beyond Fully Supervised Learning

So far, training was *fully supervised*, all variables were observed. In real life, some variables are *unobserved* even during training.

- **missing labels in training data**
- **latent variables, e.g. part location**
- **latent variables, e.g. part occlusion**
- **latent variables, e.g. viewpoint**
Three types of variables:

- $x \in \mathcal{X}$ always observed,
- $y \in \mathcal{Y}$ observed only in training,
- $z \in \mathcal{Z}$ never observed (latent).

Decision function:  
$$f(x) = \arg\max_{y \in \mathcal{Y}} \max_{z \in \mathcal{Z}} \langle w, \phi(x, y, z) \rangle$$

**Maximum Margin Training with Maximization over Latent Variables**

Solve:  
$$\min_{w, \xi} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi^n$$

subject to, for $n = 1, \ldots, N$, for all $y \in \mathcal{Y}$

$$\Delta(y^n, y) + \max_{z \in \mathcal{Z}} \langle w, \phi(x^n, y, z) \rangle - \max_{z \in \mathcal{Z}} \langle w, \phi(x^n, y^n, z) \rangle$$

Problem: not a convex problem $\rightarrow$ can have local minima

[C. Yu, T. Joachims, "Learning Structural SVMs with Latent Variables", ICML, 2009]

similar idea: [Felzenszwalb, McAllester, Ramaman. A Discriminatively Trained, Multiscale, Deformable Part Model, CVPR’08]
Structured Learning is full of Open Research Questions

▶ How to train faster?
  ▶ CRFs need many runs of probabilistic inference,
  ▶ SSVMs need many runs of arg\text{max}-predictions.

▶ How to reduce the necessary amount of training data?
  ▶ semi-supervised learning? transfer learning?

▶ How can we better understand different loss function?
  ▶ when to use probabilistic training, when maximum margin?
  ▶ CRFs are “consistent”, SSVMs are not. Is this relevant?

▶ Can we understand structured learning with approximate inference?
  ▶ often computing $\nabla \mathcal{L}(w)$ or $\text{argmax}_y \langle w, \phi(x, y) \rangle$ exactly is infeasible.
  ▶ can we guarantee good results even with approximate inference?

▶ More and new applications!