Statistical Machine Learning

https://cvml.ist.ac.at/courses/SML_W18

Christoph Lampert

IST AUSTRIA

Institute of Science and Technology

Winter Semester 2018/2019
Lecture 11

(lots of material courtesy of S. Nowozin, http://www.nowozin.net)

Overview (tentative)

Date		no.	Topic
Oct 08	Mon	1	A Hands-On Introduction
Oct 10	Wed	–	self-study (Christoph traveling)
Oct 15	Mon	2	Bayesian Decision Theory
			Generative Probabilistic Models
Oct 17	Wed	3	Discriminative Probabilistic Models
			Maximum Margin Classifiers
Oct 22	Mon	4	Generalized Linear Classifiers, Optimization
Oct 24	Wed	5	Evaluating Predictors; Model Selection
Oct 29	Mon	–	self-study (Christoph traveling)
Oct 31	Wed	6	Overfitting/Underfitting, Regularization
Nov 05	Mon	7	Learning Theory I: classical/Rademacher bounds
Nov 07	Wed	8	Learning Theory II: miscellaneous
Nov 12	Mon	9	Probabilistic Graphical Models I
Nov 14	Wed	10	Probabilistic Graphical Models II
Nov 19	Mon	11	Probabilistic Graphical Models III
Nov 21	Wed	12	Probabilistic Graphical Models IV
until Nov 25			final project

Structured Loss Functions

$$\Delta(\bar{y}, y)$$

Loss function

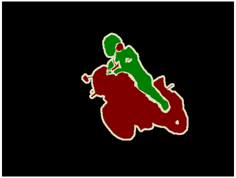
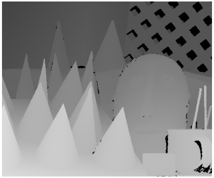
How to judge if a (structured) prediction is good?

- Define a *loss function*

$$\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+,$$

$\Delta(\bar{y}, y)$ measures the loss incurred by predicting y when $\bar{y}$ is correct.

- The *loss function* is application dependent


Example 1: 0/1 loss

Loss is 0 for perfect prediction, 1 otherwise:

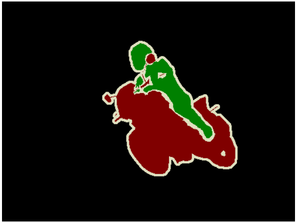
$$\Delta_{0/1}(\bar{y}, y) = [\![\bar{y} \neq y]\!] = \begin{cases} 0 & \text{if } \bar{y} = y \\ 1 & \text{otherwise} \end{cases}$$

Every mistake is equally bad. Usually not very useful in *structured prediction*.

Example 2: Hamming loss

Count the number of mislabeled variables:

$$\Delta_H(\bar{y}, y) = \frac{1}{|V|} \sum_{i \in V} [\![\bar{y}_i \neq y_i]\!]$$



Used, e.g., for graph labeling tasks

Example 3: Squared error

If we can add elements in $\mathcal{Y}_i$
(pixel intensities, optical flow vectors, etc.).

Sum of squared errors

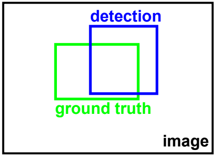
$$\Delta_Q(\bar{y}, y) = \frac{1}{|V|} \sum_{i \in V} \|\bar{y}_i - y_i\|^2.$$

Used, e.g., in stereo reconstruction, part-based object detection.

Example 4: Task specific losses

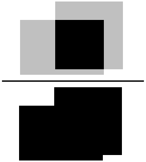
Object detection

- bounding boxes, or
- arbitrarily shaped regions



Intersection-over-union loss:

$$\Delta_{\text{IoU}}(bary, y) = 1 - \frac{\text{area}(\bar{y} \cap y)}{\text{area}(\bar{y} \cup y)} \qquad = 1 -$$



Used, e.g., in PASCAL VOC challenges for object detection, because its scale-invariance (no bias for or against big objects).

Making Bayes-optimal Predictions

Given a distribution $p(y|x)$, what is the best way to predict $f : \mathcal{X} \to \mathcal{Y}$?

Bayesian decision theory: pick $f(x)$ that causes minimal expected loss:

$$f(x) = \operatorname*{\mathbf{argmin}}_{y \in \mathcal{Y}} \mathcal{R}_\Delta(y)$$

$$\text{for} \quad \mathcal{R}_\Delta(y) = \mathop{\mathbb{E}}_{\bar{y} \sim p(y|x)} \{\Delta(\bar{y}, y)\} = \sum_{\bar{y} \in \mathcal{Y}} \Delta(\bar{y}, y) p(\bar{y}|x)$$

For many loss functions not tractable, but some exceptions:

- $\mathcal{R}_{\Delta_{0/1}}(y) = 1 - p(y|x)$, so $f(x) = \mathbf{argmax}_y\, p(y|x)$
- $\mathcal{R}_{\Delta_H}(y) = 1 - \sum_{i \in V} p(y_i|x)$, so $f(x) = (y_1, \dots, y_n)$
  for $y_i = \mathbf{argmax}_{k \in \mathcal{Y}_i}\, p(y_i = k|x)$

Structured Support Vector Machines

$$\mathbf{min}_f \;\; \mathbb{E}_{(x,y)}\, \Delta(y, f(x))$$

Loss-Minimizing Parameter Learning

- $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ i.i.d. training set
- $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ be a feature function, like for CRF
- $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a loss function.

- Find a weight vector w^* that minimizes the expected loss

$$\mathop{\mathbb{E}}_{(x,y)} \Delta(y, f(x))$$

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

Loss-Minimizing Parameter Learning

- $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ i.i.d. training set
- $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ be a feature function, like for CRF
- $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a loss function.

- Find a weight vector w^* that minimizes the expected loss

$$\mathop{\mathbb{E}}_{(x,y)} \Delta(y, f(x))$$

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

Advantage:

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

Disadvantage:

- We need to know the loss function already at training time.
- We can't use probabilistic reasoning to find w^*.

Inspiration: multi-class SVM

- $\mathcal{X}$ anything, $\quad \mathcal{Y} = \{1, 2, \ldots, K\}$,
- feature map $\phi : \mathcal{X} \to \mathcal{H}$ (explicit or implicit via kernel)
- training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
- goal: learn functions $g_k(x) = \langle w_k, \phi(x) \rangle$ for $k = 1, \ldots, K$.

Prediction: $\quad f(x) = \operatorname*{\mathbf{argmax}}_{k=1,\ldots,K} g_k(x) = \operatorname*{\mathbf{argmax}}_{k=1,\ldots,K} \langle w_k, \phi(x) \rangle$

Enforce a margin between the correct and all incorrect labels:

$$\min_{w_1,\ldots,w_K,\boldsymbol{\xi}} \quad \frac{1}{2} \sum_{k=1}^{K} \|w_k\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i$$

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

$$\langle w_{y^i}, \phi(x^i) \rangle \geq 1 + \langle w_k, \phi(x^i) \rangle - \xi^i, \quad \text{for all } k \neq y_i.$$

Crammer-Singer Multiclass SVM

Equivalent parameterization:

- $\mathcal{X}$ anything, $\quad \mathcal{Y} = \{1, 2, \ldots, K\}$,
- feature map $\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ (explicit or implicit via kernel)
- $\psi(x, y) = \Big([\![y = 1]\!]\phi(x),\ [\![y = 2]\!]\phi(x), \ldots,\ [\![y = K]\!]$
- $w = (w_1, \ldots, w_K) \in \mathbb{R}^{KD}$
- goal: learn a function $g(x, y) = \langle w, \psi(x, y) \rangle$

Prediction: $\quad f(x) = \underset{k=1,\ldots,M}{\mathbf{argmax}} \ \langle w, \psi(x, y) \rangle$

Enforce a margin of 1 between the correct and any incorrect label:

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

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

$$\langle w, \psi(x_i, y_i) \rangle \geq 1 + \langle w, \psi(x_i, \bar{y}) \rangle - \xi_i, \quad \text{for all } \bar{y} \neq y_i.$$

Observation:

- for structure outputs, not all "incorrect" labels are equally bad
  $\rightarrow$ margin between y_i and $\bar{y}$ should depend on $\Delta(y_i, \bar{y})$

Structured (Output) Support Vector Machine

Goal: learn a function $g(x, y) = \langle w, \psi(x, y) \rangle$

Prediction: $f(x) = \operatorname*{\mathbf{argmax}}_{k=1,\dots,M} \; \langle w, \psi(x, y) \rangle$

Enforce a margin $\Delta(y_i, y)$ between the correct and any incorrect label:

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

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

$$\langle w, \psi(x_i, y_i) \rangle \geq \Delta(y_i, \bar{y}) + \langle w, \psi(x_i, \bar{y}) \rangle - \xi_i, \quad \text{for all } \bar{y} \in \mathcal{Y}.$$

Structured Output Support Vector Machine

Equivalent unconstrained formulation (solve for optimal $\xi_1, \ldots, \xi_n$):

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{\bar{y}\in\mathcal{Y}} \; \Big[\Delta(y_i,\bar{y}) + \langle w, \psi(x_i,\bar{y})\rangle - \langle w, \psi(x_i,y_i)\rangle\Big]$$

Conditional Random Field

Regularized conditional log-likelihood:

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log \sum_{\bar{y}\in\mathcal{Y}} \exp\big(\langle w, \psi(x_i,\bar{y})\rangle - \langle w, \phi(x_i,y_i)\rangle\big)$$

CRFs and SSVMs have more in common than usually assumed.

- $\log\sum_y \exp(\cdot)$ can be interpreted as a soft-max (differentiable)
- SSVM training takes loss function into account
- CRF is trained without specific loss, loss enters at prediction time

Structured Output Support Vector Machine

Equivalent unconstrained formulation (solve for optimal $\xi_1, \dots, \xi_n$):

$$\min_w \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{\bar{y}\in\mathcal{Y}} \; \Big[\Delta(y_i,\bar{y}) + \langle w, \psi(x_i,\bar{y})\rangle - \langle w, \psi(x_i,y_i)\rangle\Big]$$

Conditional Random Field

Regularized conditional log-likelihood:

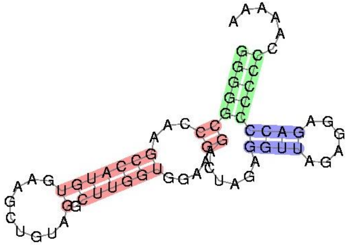
$$\min_w \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log \sum_{\bar{y}\in\mathcal{Y}} \exp\big(\langle w, \psi(x_i,\bar{y})\rangle - \langle w, \phi(x_i,y_i)\rangle\big)$$

CRFs and SSVMs have more in common than usually assumed.

- $\log\sum_y \exp(\cdot)$ can be interpreted as a soft-max (differentiable)
- SSVM training takes loss function into account
- CRF is trained without specific loss, loss enters at prediction time

Example: RNA Secondary Structure Prediction De Bona *et al.*, 2007]

AAAAACCCCCCCCAGAGGAGAUUG
GAGAUCAAAGGUGGUUCGGAUGUC $\rightarrow$
GAAGUGUACCGAACCCGGGGG

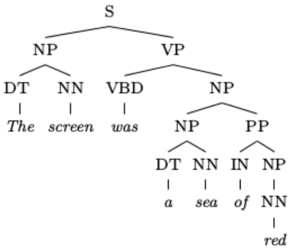


- $\mathcal{X} = \Sigma^*$ for $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$ (nucleotide sequence)
- $\mathcal{Y} = \{(i, j) : i, j \in \mathbb{N}, i < j\}$ $\quad (i, j)$ mean "x_i binds with x_j"
- $\psi(x, y)$ domain-specific features: binding energy of $x_i \leftrightarrow x_j$, prefered patterns (motifs), loop properties, ...
- $\Delta(\bar{y}, y)$: number of wrong/missing bindings (Hamming loss)

$$\min_w \ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{\bar{y}\in\mathcal{Y}} \Big[\Delta(y_i, \bar{y}) + \langle w, \psi(x_i, \bar{y})\rangle - \langle w, \psi(x_i, y_i)\rangle \Big]$$

Example: Sentence Parsing [Taskar *et al.*, 2004]

`The screen was a sea of red.` $\rightarrow$



- $\mathcal{X} = \{\text{English sentences}\}$
- $\mathcal{Y} = \{\text{parse tree}\}$
- $\psi(x, y)$ domain-specific features:
  - word properties, e.g. "· starts with capital letter", "· ends in `ing`"
  - grammatical rules: $NP \rightarrow DT + NN$
- $\Delta(\bar{y}, y)$: number of wrong assignments

Solving S-SVM Training in Practice

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max_{\bar{y}\in\mathcal{Y}} \Big[\Delta(y_i, \bar{y}) + \langle w, \psi(x_i, \bar{y})\rangle - \langle w, \psi(x_i, y_i)\rangle \Big]$$

- continuous
- unconstrained
- convex
- non-differentiable

Computing a subgradient:

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y \, \ell_y(x_i, y_i, w)$, and

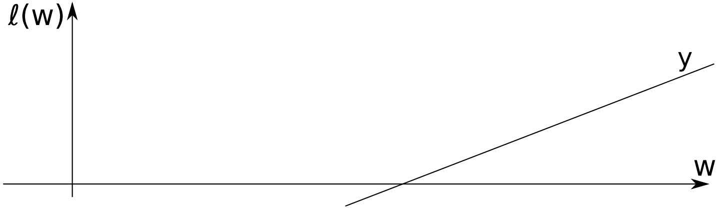
$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$

Computing a subgradient:

$$\min_{w} \ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y\, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



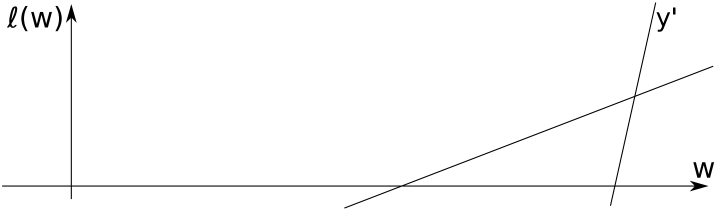
For each $y \in \mathcal{Y}$, $\ell_y(w)$ is a linear function of w.

Computing a subgradient:

$$\min_{w} \ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y \, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



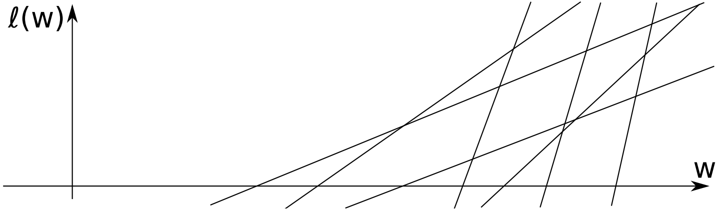
For each $y \in \mathcal{Y}$, $\ell_y(w)$ is a linear function of w.

Computing a subgradient:

$$\min_w \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y\, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



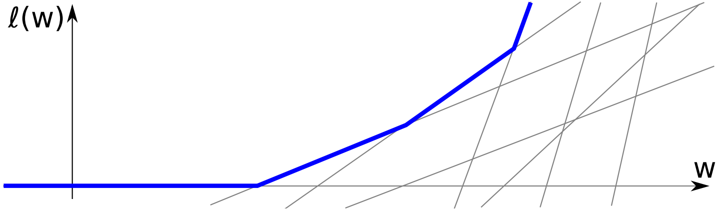
For each $y \in \mathcal{Y}$, $\ell_y(w)$ is a linear function of w.

Computing a subgradient:

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y\, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



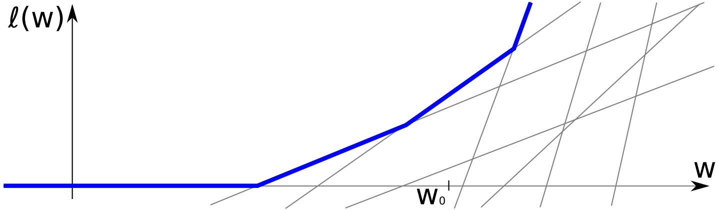
$\mathbf{max}$ over finite $\mathcal{Y}$: piece-wise linear

Computing a subgradient:

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y \, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



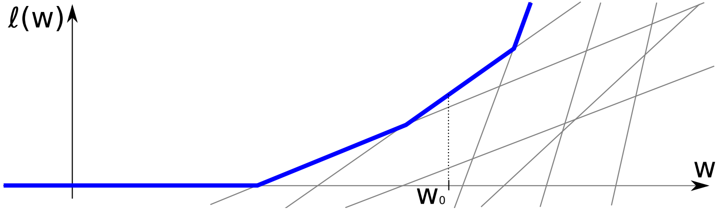
Subgradient of ℓ at w_0:

Computing a subgradient:

$$\min_w \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y\, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



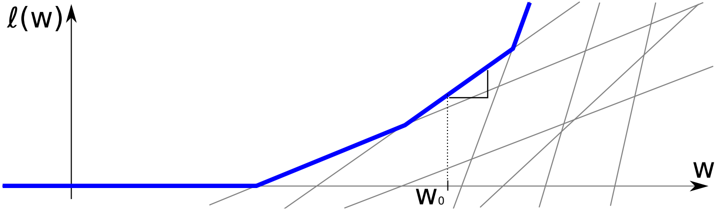
Subgradient of ℓ at w_0: find maximal (active) y.

Computing a subgradient:

$$\min_{w} \; \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y \, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



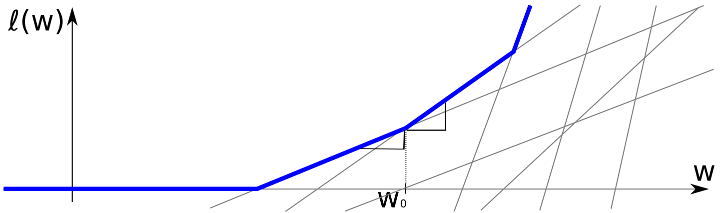
Subgradient of ℓ at w_0: find maximal (active) y, use $v = \nabla \ell_y(w_0)$.

Computing a subgradient:

$$\min_{w} \ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, w)$$

with $\ell(x_i, y_i, w) = \mathbf{max}_y \, \ell_y(x_i, y_i, w)$, and

$$\ell_y(x_i, y_i, w) := \Delta(y_i, y) + \langle w, \psi(x_i, y)\rangle - \langle w, \psi(x_i, y_i)\rangle$$



Not necessarily unique, but $v = \nabla \ell_y(w_0)$ works for any maximal y

Solving S-SVM Training Numerically – Subgradient Method

Subgradient Method S-SVM Training

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

```
w ← 0⃗
for t=1,...,T do
    for i=1,...,n do
        ŷ ← argmax_{y∈𝒴}  Δ(yⁿ, y) + ⟨w, φ(xⁿ, y)⟩ − ⟨w, φ(xⁿ, yⁿ)⟩
        vⁿ ← φ(xⁿ, ŷ) − φ(xⁿ, yⁿ)
    end for
    w ← w − η_t(λw − (1/N) Σ_n vⁿ)
end for
```

$w \leftarrow \vec{0}$
for t=1,...,T **do**
  for i=1,...,n **do**
    $\hat{y} \leftarrow \mathbf{argmax}_{y \in \mathcal{Y}} \quad \Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle$
    $v^n \leftarrow \phi(x^n, \hat{y}) - \phi(x^n, y^n)$
  end for
  $w \leftarrow w - \eta_t(\lambda w - \frac{1}{N}\sum_n v^n)$
end for

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

Obs: each update of w needs N $\mathbf{argmax}$-prediction (one per example).
Obs: computing the $\mathbf{argmax}$ is (loss augmented) **energy minimization**.

Example: Image Segmenatation

- $\mathcal{X}$ images, $\mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Big($  ,  $\Big)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

Example: Image Segmenatation

- $\mathcal{X}$ images, $\mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Big($  ,  $\Big)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

$t = 1$: $w = 0$,

$$\hat{y} = \operatorname*{\mathbf{argmax}}_{y} \Big[\langle w, \phi(x^n, y) \rangle + \Delta(y^n, y) \Big]$$

$$\overset{w=0}{=} \operatorname*{\mathbf{argmax}}_{y} \Delta(y^n, y) \quad = \text{"the opposite of } y^n \text{"}$$

Example: Image Segmenatation

- $\mathcal{X}$ images, $\mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Bigg($ ,  $\Bigg)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

$t = 1$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $-$, blue $-$, gray $-$

Example: Image Segmenatation

- $\mathcal{X}$ images, $\mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Bigg($  ,  $\Bigg)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

$t = 1$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $-$, blue $-$, gray $-$

$t = 2$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $=$, blue $=$, gray $-$

Example: Image Segmenatation

- $\mathcal{X}$ images, $\mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Big($ ,  $\Big)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

$t = 1$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $-$, blue $-$, gray $-$

$t = 2$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $=$, blue $=$, gray $-$

$t = 3$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $=$, white $=$, green $-$, blue $-$, gray $-$

- $\mathcal{X}$ images, $\mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Big($ ,  $\Big)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

$t = 1$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $-$, blue $-$, gray $-$

$t = 2$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $=$, blue $=$, gray $-$

$t = 3$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $=$, white $=$, green $-$, blue $-$, gray $-$

$t = 4$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $=$, white $=$, green $-$, blue $=$, gray $=$

Example: Image Segmenatation

- $\mathcal{X}$ images, $\quad \mathcal{Y} = \{$ binary segmentation masks $\}$.
- Training example(s): $(x^n, y^n) = \Big($  ,  $\Big)$
- $\Delta(y, \bar{y}) = \sum_p [\![y_p \neq \bar{y}_p]\!]$ (Hamming loss)

$t = 1$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $-$, blue $-$, gray $-$

$t = 2$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $+$, white $+$, green $=$, blue $=$, gray $-$

$t = 3$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $=$, white $=$, green $-$, blue $-$, gray $-$

$t = 4$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $=$, white $=$, green $-$, blue $=$, gray $=$

$t = 5$: $\hat{y} =$  $\phi(y^n) - \phi(\hat{y})$: black $=$, white $=$, green $=$, blue $=$, gray $=$

$t = 6, \ldots$: no more changes.

Images: [Carreira, Li, Sminchisescu, "Object Recognition by Sequential Figure-Ground Ranking", IJCV 2010]

Solving S-SVM Training Numerically – Subgradient Method

Same trick as for CRFs: **stochastic updates**:

Stochastic Subgradient Method S-SVM Training

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

$w \leftarrow \vec{0}$
for t=1,...,T **do**
  $(x^n,y^n)\ \leftarrow$ randomly chosen training example pair
  $\hat{y} \leftarrow \mathbf{argmax}_{y\in\mathcal{Y}}\,\Delta(y^n,y) + \langle w,\phi(x^n,y)\rangle - \langle w,\phi(x^n,y^n)\rangle$
  $w \leftarrow w - \eta_t(\lambda w - \frac{1}{N}[\phi(x^n,\hat{y}) - \phi(x^n,y^n)])$
end for

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

Observation: each update of w needs only 1 $\mathbf{argmax}$-prediction
(but we'll need many iterations until convergence)

Structured Support Vector Machine:

$$\min_{w} \quad \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{n=1}^{N} \max_{y\in\mathcal{Y}} \Big[\Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle\Big]$$

Subgradient method converges slowly. Can we do better?

Solving S-SVM Training Numerically

Structured Support Vector Machine:

$$\min_{w} \quad \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{n=1}^{N} \max_{y\in\mathcal{Y}} \Big[\Delta(y^n, y) + \langle w, \phi(x^n, y)\rangle - \langle w, \phi(x^n, y^n)\rangle\Big]$$

Subgradient method converges slowly. Can we do better?

We can use **inequalities** and **slack variables** to reformulate the optimization.

Solving S-SVM Training Numerically

Structured SVM (equivalent formulation):

Idea: slack variables

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

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

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

Note: $\xi^n \geq 0$ automatic, because left hand side is non-negative.

Differentiable objective, convex, N non-linear contraints,

Structured SVM (also equivalent formulation):

Idea: expand $\mathbf{max}$ term into individual constraints

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

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

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

Differentiable objective, convex, $N|\mathcal{Y}|$ linear constraints

Solve an S-SVM like a linear Support Vector Machine:

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

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

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

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

Solving S-SVM Training Numerically

Solve

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

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

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

Same structure as an ordinary SVM!

- quadratic objective ☺
- linear constraints ☺

Solving S-SVM Training Numerically

Solve

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

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

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

Same structure as an ordinary SVM!

- quadratic objective ☺
- linear constraints ☺

Question: Can we use an ordinary SVM/QP solver?

Solving S-SVM Training Numerically

Solve

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

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

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

Same structure as an ordinary SVM!

- quadratic objective ☺
- linear constraints ☺

Question: Can we use an 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

Solving S-SVM Training Numerically – Working Set

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 $\leftarrow$ can of course be formalized.

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

Solving S-SVM Training Numerically – Working Set

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:

Solving S-SVM Training Numerically – Working Set

- 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.

Solving S-SVM Training Numerically – Working Set

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 $\leftarrow$ can of course be formalized.

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

Solving S-SVM Training Numerically – Working Set

- 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 ϵ-close to the global optimum

Working Set 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 λ

$w \leftarrow 0$, $S \leftarrow \emptyset$
repeat
  $(w, \xi) \leftarrow$ *solution to QP only with constraints from* S
  for i=1,. . . ,n **do**
    $\hat{y} \leftarrow \mathbf{argmax}_{y \in \mathcal{Y}} \quad \Delta(y^n, y) + \langle w, \phi(x^n, y) \rangle$
    if $\hat{y} \neq y^n$ **then**
      $S \leftarrow S \cup \{(x^n, \hat{y})\}$
    end if
  end for
until S doesn't change anymore.

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

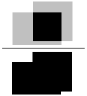
Obs: each update of w needs N $\mathbf{argmax}$-predictions (one per example), but we solve globally for next w, not by local steps.

Example: Object Localization

- $\mathcal{X}$ images, $\quad \mathcal{Y} = \{$ object bounding box $\} \subset \mathbb{R}^4$.
- Training examples:    
- Goal: $f : \mathcal{X} \to \mathcal{Y}$

$\longmapsto$

- Loss function: area overlap $\Delta(y, y') = 1 - \frac{\text{area}(y \cap y')}{\text{area}(y \cup y')}$



[Blaschko, Lampert: "Learning to Localize Objects with Structured Output Regression", ECCV 2008]

Structured SVM:

- $\phi(x,y) :=$ "bag-of-words histogram of region y in image x"

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

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

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

Interpretation:

- For every image, the correct bounding box, y^n, should have a higher score than any wrong bounding box.
- Less overlap between the boxes $\rightarrow$ bigger difference in score

Example: Object Localization

Working set training – Step 1:

- $w \leftarrow 0$.

For every example:

- $\hat{y} \leftarrow \mathbf{argmax}_{y \in \mathcal{Y}} \quad \Delta(y^n, y) + \underbrace{\langle w, \phi(x^n, y) \rangle}_{=0}$

  maximal Δ-loss $\quad \equiv \quad$ minimal overlap with y^n $\quad \equiv \quad \hat{y} \cap y^n = \emptyset$

- add constraint

$$\langle w, \phi(x^n, y^n) \rangle - \langle w, \phi(x^n, \hat{y}) \rangle \geq 1 \; - \; \xi^n$$

Note: similar to binary SVM training for object detection:

- positive examples: ground truth bounding boxes
- negative examples: random boxes from 'image background'

Example: Object Localization

Working set training – Later Steps:

For every example:

- $\hat{y} \leftarrow \mathbf{argmax}_{y \in \mathcal{Y}} \underbrace{\Delta(y^n, y)}_{\text{bias towards 'wrong' regions}} + \underbrace{\langle w, \phi(x^n, y) \rangle}_{\text{object detection score}}$
- if $\hat{y} = y^n$: do nothing,
  else: add constraint

$$\langle w, \phi(x^n, y^n) \rangle - \langle w, \phi(x^n, \hat{y}) \rangle \geq \Delta(y^n, \hat{y}) \; - \; \xi^n$$

  enforces $\hat{y}$ to have lower score after re-training.

Note: similar to hard negative mining for object detection:

- perform detection on training image
- if detected region is far from ground truth, add as negative example

Difference: S-SVM handles regions that overlap with ground truth.

Dual S-SVM

We can also dualize the S-SVM optimization:

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

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

$$\alpha_{ny} \geq 0, \qquad \text{and} \qquad \sum_{y\in\mathcal{Y}} \alpha_{ny} \leq \frac{2}{\lambda N}.$$

Quadratic (convex) objective, linear constraints, $N|\mathcal{Y}|$ unknowns

Dual S-SVM

We can also dualize the S-SVM optimization:

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

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

$$\alpha_{ny} \geq 0, \qquad \text{and} \qquad \sum_{y \in \mathcal{Y}} \alpha_{ny} \leq \frac{2}{\lambda N}.$$

Quadratic (convex) objective, linear constraints, $N|\mathcal{Y}|$ unknowns

Recover weight vector from dual coefficients:

$$w = \sum_{n,\alpha} \alpha_{ny} \phi(x^n, y)$$

State-of-the-art: solve dual with **Frank-Wolfe algorithm**.