Nonlinear SVMs

Kernels, Mercer's theorem
Examples: protein folding

Joachim M. Buhmann

November 15, 2019

Non–Linear SVMs

Feature extraction by non linear transformation $\mathbf{y} = \phi(\mathbf{x})$

Problem with non-linear feature transformations:

For very high dimensional spaces the non-linear transformation $\phi(\mathbf{x})$ might be computationally too difficult to compute when we only require the inner product

$$\mathbf{y}_i^{\mathsf{T}} \mathbf{y}_j = \phi^{\mathsf{T}}(\mathbf{x}_i)\phi(\mathbf{x}_j) \ .$$

A **kernel function** is defined by

$$\forall \mathbf{x}, \mathbf{z} \in \Omega : \quad K(\mathbf{x}, \mathbf{z}) = \phi^{\mathsf{T}}(\mathbf{x})\phi(\mathbf{z})$$

Using the kernel function the discriminant function becomes

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i z_i \underbrace{K(\mathbf{x}_i, \mathbf{x})}_{\text{replaces } \mathbf{y}_i^{\mathsf{T}} \mathbf{y}}$$

What are Kernels?

Similarity based reasoning in machine learning

- Structure detection in data sets requires two concepts: (i) identity/equivalence and (ii) similarity
- A notion of identity is necessary to decide if a pattern has already been selected for the set of admissible solutions.
- A notion of similarity endows the solution space with a local structure.
- Many pattern recognition problems are formulated based on a metric.
- Dissimilarities are often chosen as the squared norm of difference vectors $\|\mathbf{x}-\mathbf{y}\|^2 = \mathbf{x}^\mathsf{T}\mathbf{x} - 2\mathbf{x}^\mathsf{T}\mathbf{y} + \mathbf{y}^\mathsf{T}\mathbf{y},\ \forall \mathbf{x},\mathbf{y} \in \mathbb{R}^d$.

$$\text{dissimilarity}(\mathbf{x},\mathbf{y}) = \text{similarity}(\mathbf{x},\mathbf{x}) + \text{similarity}(\mathbf{y},\mathbf{y}) - 2\text{similarity}(\mathbf{x},\mathbf{y})$$

Characterization of Kernels

For any symmetric matrix $K(\mathbf{x}_i, \mathbf{x}_j)|_{i,j=1}^n$ (Gram matrix) there exists an eigenvector decomposition

$$K = V \Lambda V^\mathsf{T}.$$

V: orthogonal matrix of eigenvectors $(v_{ti})|_{i=1}^n$
Λ: diagonal matrix of eigenvalues λ_t

Assume all eigenvalues are nonnegative and consider mapping

$$\phi : \mathbf{x}_i \rightarrow \left(\sqrt{\lambda_t} v_{ti}\right)_{t=1}^n \in \mathbb{R}^n, i = 1, \ldots, n$$

Then it follows

$$\phi^\mathsf{T}(\mathbf{x}_i)\phi(\mathbf{x}_j) = \sum_{t=1}^n \lambda_t v_{ti} v_{tj} = \left(V \Lambda V^\mathsf{T}\right)_{ij} = K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$$

Positivity of Kernels

Theorem: Let Ω be a finite input space $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with $K(\mathbf{x}, \mathbf{z})$ a symmetric function on Ω. Then $K(\mathbf{x}, \mathbf{z})$ is a kernel function if and only if the matrix

$$K = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^{n}$$

is positive semi-definite (has only non-negative eigenvalues).

The important restriction for this theorem is the finiteness of the space. The property of positive semi-definiteness has to be proved for all pairs $(\mathbf{x}, \mathbf{z}) \in \Omega \times \Omega$.

The extension to infinite dimensional Hilbert Spaces might also include a non-negative weighting $\{\lambda_i \ : \ 1 \leq i \leq n\}$

$$< \phi(\mathbf{x}), \phi(\mathbf{z}) >= \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{z}) \ .$$

Mercer's Theorem

Theorem (Mercer 1909)

Let Ω be a compact subset of $\mathbb{R}^n$. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(X) \to L_2(X)$,

$$(T_K f)(\cdot) = \int_\Omega K(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x},$$

is positive, that is

$$\int_{\Omega \times \Omega} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x} d\mathbf{z} > 0 \quad \forall f \in L_2(\Omega)$$

Then we can expand $K(\mathbf{x}, \mathbf{z})$ in a uniformly convergent series in terms of T_K's eigen-functions $\phi_j \in L_2(\Omega)$, with $||\phi_j||_{L_2} = 1$ and positive associated eigenvalues $\lambda_j > 0$.

Proof: theory of Fredholm integral equations.

Possible Kernels

Remark: Each kernel function, that hold Mercer's conditions describes an inner product in a high dimensional space. The kernel function replaces the inner product.

Possible Kernels:

$$a) \quad K(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}\right) \qquad \text{(RBF Kernel)}$$

$$b) \quad K(\mathbf{x},\mathbf{z}) = \tanh \kappa \mathbf{x}\mathbf{z} - b \qquad \text{(Sigmoid Kernel)}$$

$$c) \quad K(\mathbf{x},\mathbf{z}) = (\mathbf{x}\mathbf{z})^d \qquad \text{(Polynomial Kernel)}$$

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}\mathbf{z}+1)^d$$

$$d) \quad K(\mathbf{x},\mathbf{z}) : \text{string kernels for sequences}$$

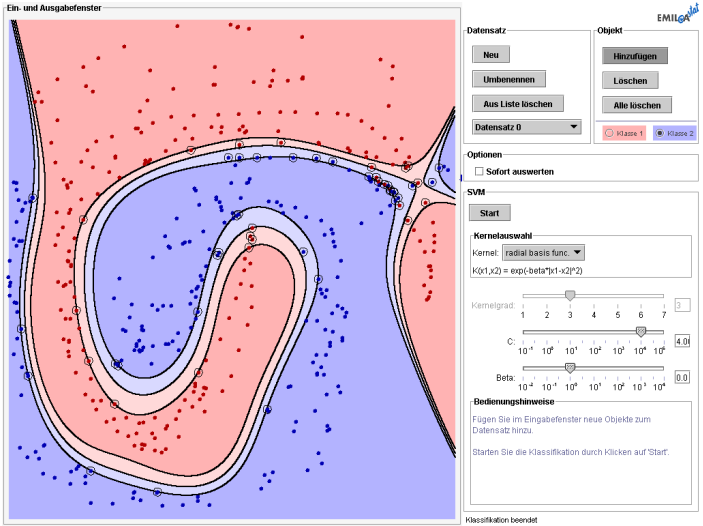
Kernel Engineering

Kernel composition rules

Let K_1, K_2 be kernels over $\Omega \times \Omega, \Omega \subseteq \mathbb{R}^d$, $a \in \mathbb{R}^+$, $f(.)$ a real-vealued function $\phi : \Omega \to \mathbb{R}^e$ with K_3 a kernel over $\mathbb{R}^e \times \mathbb{R}^e$.
Then the following functions are kernels:

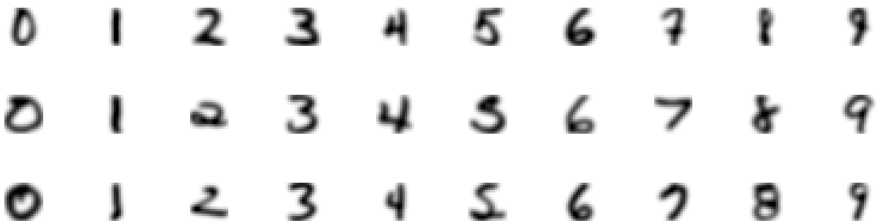
1. $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z}) + K_2(\mathbf{x}, \mathbf{z})$
2. $K(\mathbf{x}, \mathbf{z}) = aK_1(\mathbf{x}, \mathbf{z})$
3. $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$
4. $K(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$
5. $K(\mathbf{x}, \mathbf{z}) = K_3(\phi(\mathbf{x}), \phi(\mathbf{z}))$
6. $K(\mathbf{x}, \mathbf{z}) = p(K_1(\mathbf{x}, \mathbf{z}))$, ($p(x)$ is a polynomial with positive coefficients)
7. $K(\mathbf{x}, \mathbf{z}) = \exp(K_1(\mathbf{x}, \mathbf{z}))$

Applet HTML Page



Example: Hand Written Digit Recognition

- 7291 training images und 2007 test images (16x16 pixel, 256 gray values)



Classification method	test error
human classification	2.7 %
perceptron	5.9 %
support vector machines	4.0 %

SVMs for Secondary Structure Prediction

Proteins are represented in 0^{th} order by the percentage of amino-acids in polypeptide chains; $\rightsquigarrow$ "vectorial" representation in $\mathbb{R}^{20}$

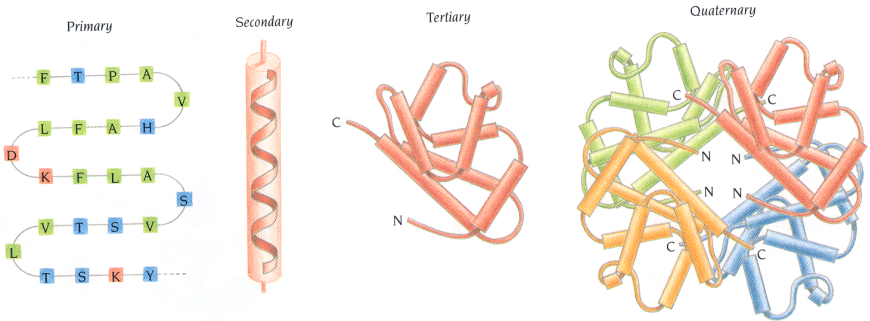
Protein structure problem: **sequence** as primary structure, **local motives** as secondary structure, **protein folds** as ternary structure.

SVM classification typically uses the RBF kernel

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{z}\|^2\right)$$

Secondary structure prediction as a multiclass problem: Detect classes helix (H), sheet (E) and coil (C)

Types of Protein Structure



Primary Structure	$\widehat{=}$	sequence of AAs
Secondary Structure	$\widehat{=}$	folding of parts of the AA chain
Tertiary Structure	$\widehat{=}$	real 3-D conformation
Quaternary Structure	$\widehat{=}$	3-D arrangement of several domains

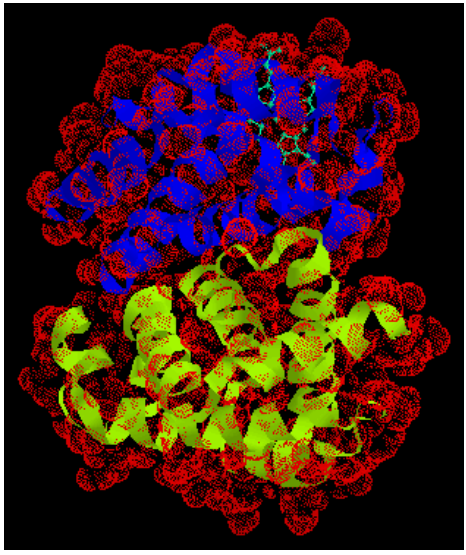
Protein Folding

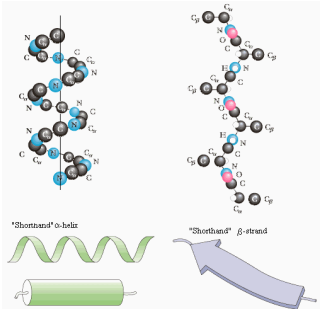
- The whole structure (primary to quaternary) is determined by the primary sequence and their physico-chemical interactions in the medium.
- Therefore, the folding of a protein is defined by the genetic material itself, as the three-dimensional structure with the minimal free energy.
- The structure of a protein determines its functionality.

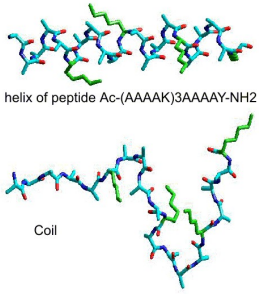
Holy grail of molecular biochemistry

Predict protein folding based on sequence information alone.

Structure of Hemoglobin







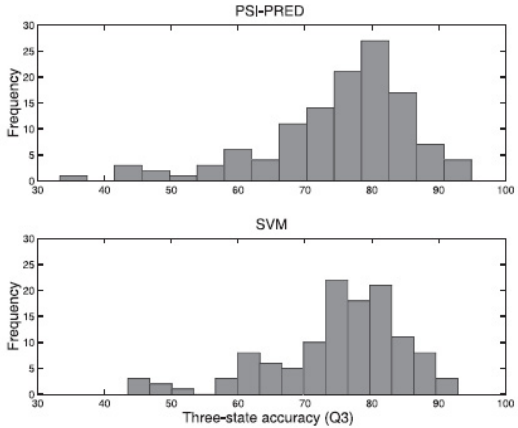
Accuracy measure Q_3 is the percentage of correct 3-state symbols, i.e.

$$Q_3 = \frac{\#\text{correctly predicted residues}}{\text{total \# of residues}} \cdot 100$$

Practical Problem: How to apply SVMs for $k > 2$ classes?

Experimental Results

- PHD (by B. Rost et al., Neural Network based approach) – 72-74% Q_3
- Psi-pred (by D. T. Jones et al., Neural Network based approach) – 76-78% Q_3
- The extensive study by Ward et al. (Bioinformatics, 2003) with different SVM realization reports results 73-77% Q_3
- Two-layer classification strategy with position-specific scoring scheme (Guo et al., Proteins, 2004)). Accuracy ranges from 78% – 80%.



Histogram of Q_3 scores for 121 test proteins

(Ward et al., Bioinformatics 19:13, 2003)

Structural SVMs

From margins to score functions

Joachim M. Buhmann

November 15, 2019

Extensions to the SVM

Up to now we only considered binary (two classes) SVMs, i.e. $z \in \{-1, +1\}$.
Possible extensions:

- Multiclass SVMs: Each sample $\mathbf{y}$ is assigned to one of M classes, i.e. $z \in \{1, 2, \ldots, M\}$.
- Structured SVMs: Each sample $\mathbf{y}$ is assigned to a structured output label $z \in \mathbb{K}$, e.g. partitions, trees, lists, etc.

As an example, the sample $\mathbf{y}$ could correspond to the pixels of an image and the output label to a segmentation of that image into foreground and background (the assignment of all pixels to either foreground or background represents the output label).

Multiclass SVMs (linear discriminant function)

Binary SVMs are based on notion of the margin m:

$$z_i g(\mathbf{y}) = z_i(\mathbf{w}^\mathsf{T}\mathbf{y}_i + w_0) \geq m \quad \text{for all } \mathbf{y}_i \in \mathcal{Y}$$

The notion of the margin must be generalized to multi-class problems:* For every class $z \in \{1, 2, \ldots, M\}$, we introduce a weight vector $\mathbf{w}_z$ and define the margin as the maximum m s.t.

$$(\mathbf{w}_{z_i}^\mathsf{T}\mathbf{y}_i + w_{z_i,0}) - \max_{z \neq z_i}(\mathbf{w}_z^\mathsf{T}\mathbf{y}_i + w_{z,0}) \geq m \quad \text{for all } \mathbf{y}_i \in \mathcal{Y}$$

* This can be achieved in different ways. We follow the approach proposed in: K. Crammer and Y. Singer, "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines", JMLR 2001

Multiclass SVMs: Formulation

Similar to before, we can define an optimization problem to learn hard functional margin multiclass SVMs ($\mathbf{w}^\mathsf{T} := (\mathbf{w}_1^\mathsf{T}, \ldots, \mathbf{w}_M^\mathsf{T})$ is a concatenation of the individual class weights vectors)

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} = \min_{\{\mathbf{w}_z\}_{z=1}^M} \frac{1}{2}\sum_{z=1}^{M} \mathbf{w}_z^\mathsf{T}\mathbf{w}_z \\ \text{s.t.} \quad & (\mathbf{w}_{z_i}^\mathsf{T}\mathbf{y}_i + w_{z_i,0}) - \max_{z \neq z_i}(\mathbf{w}_z^\mathsf{T}\mathbf{y}_i + w_{z,0}) \geq 1 \quad \forall \mathbf{y}_i \in \mathcal{Y} \end{aligned}$$

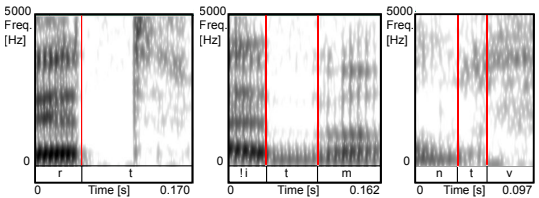
For non-separable data we can again introduce slacks $\xi_i \geq 0$:

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}\geq 0} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_i \xi_i \\ \text{s.t.} \quad & (\mathbf{w}_{z_i}^\mathsf{T}\mathbf{y}_i + w_{z_i,0}) - \max_{z \neq z_i}(\mathbf{w}_z^\mathsf{T}\mathbf{y}_i + w_{z,0}) \geq 1 - \xi_i \quad \forall \mathbf{y}_i \in \mathcal{Y} \end{aligned}$$

Multiclass SVMs: Example

Phoneme classification

Task: Given a (segmented) speech utterance, classify each segment as one out of (typically) 39 phonemes.



The difficulty of this task is partly caused by the large variability of speech utterances and the dependencies between neighbouring phonemes.

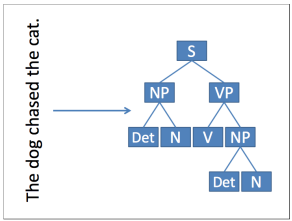
- Input: Features based on the mel-frequency cepstrum (averaged within every segment)
- Output: Phonemes

Structural SVMs (SSVMs)

Structural SVMs can be used to predict complex objects.

Example.

A parser takes as input a natural language sentence. The task is to predict a parse tree decomposing the sentence into its constituents.



- S . . . sentence
- NP . . . noun phrase (*the dog, the cat*)
- *VP . . . verbal phrase (chased the cat)*
- *N . . . noun (dog, cat)*
- *V . . . verb (chased)*
- *Det . . . determiners (the)*

(the number of possible parse trees is tremendous)

SSVMs: More examples

- **Natural language processing:**
  - Automatic translation (output: sentences)
- **Bioinformatics:**
  - Secondary structure prediction (output: bipartite graphs)
  - Enzyme function prediction (output: path in a tree)
- **Speech processing:**
  - Automatic transcription (output: sentences)
  - Text-to-speech (output: audio signal)
- **Computer Vision:**
  - semantic segmentation of images (output: partitioning of an image)
- **Robotics:**
  - Planning (output: sequence of actions)

From Christoph Lampert, "Learning with Structured Inputs and Outputs"

SSVMs: Key Problems

Four key problems to overcome for SSVMs:

1. Compact representation of the output space
   If we allowed even just one parameter for each class, we would already have more parameters than we could ever hope to have enough training data for.

2. Efficient prediction
   Sequentially enumerating all possible classes may be infeasible, hence making a single prediction on a new example may be computationally challenging.

3. Prediction error
   The notion of a prediction error must be refined – for example, predicting a parse tree that is almost correct should be treated differently from predicting a tree that is completely wrong. Furthermore, the training data may be inconsistent.

4. Efficient training

   Efficient training algorithms are needed that have a run-time complexity sub-linear in the number of classes.

SSVMs: Output Space Representation

SSVMs could be formulated as multiclass SVMs (one weight vector per class), however this would lead to a blow-up of the overall number of parameters.

Key ideas to overcome this:

1. Define a joint feature map $\mathbf{\Psi}(z, \mathbf{y})$ that combines properties of inputs and outputs.
2. Define a scoring function $f_{\mathbf{w}}(z, \mathbf{y}) = \mathbf{w}^{\mathsf{T}} \mathbf{\Psi}(z, \mathbf{y})$ ($\Rightarrow$ the number of features depends on the dimensionality of the joint feature map only and is "independent" of the number of classes).
3. Perform classification via

$$\hat{z} := h(\mathbf{y}) = \arg\max_{z \in \mathbb{K}} f_{\mathbf{w}}(z, \mathbf{y}).$$

SSVMs: Formulation

We define the margin as the maximum m that satisfies

$$\mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \max_{z \neq z_i} \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z, \mathbf{y}_i) \geq m \quad \text{for all } \mathbf{y}_i \in \mathcal{Y}.$$

This yields the optimization problem for hard functional margin SSVMs:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \max_{z \neq z_i} \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z, \mathbf{y}_i) \geq 1 \quad \forall \mathbf{y}_i \in \mathcal{Y} \end{aligned}$$

SSVMs: Efficient Prediction

Classification requires computing

$$\hat{z} := h(\mathbf{y}) = \arg\max_{z \in \mathbb{K}} f_{\mathbf{w}}(z, \mathbf{y}).$$

This is in general a hard combinatorial problem.

For efficient classification, there must be some kind of structural matching between the (given) compositional structure of the outputs z and the (designed) joint feature map $\mathbf{\Psi}$. For example:

- Decomposable output spaces
  The output space $\mathbb{K}$ can be decomposed into non-overlapping independent parts s.t. $\mathbb{K} = \mathbb{K}_1 \times \ldots \times \mathbb{K}_m$ (and $\mathbf{\Psi}$ respects this decomposition), then maximization can be performed part-wise.
- Specific dependency structures
  A more general case is captured by Markov networks. Let z be a vector of random variables and $\mathbf{\Psi}(z, \mathbf{y})$ represent sufficient statistics of a conditional exponential model $P(z \mid \mathbf{y})$. Then, maximizing $f_{\mathbf{w}}(z, \mathbf{y})$ corresponds to finding the most probable output $\arg\max_z P(z \mid \mathbf{y})$. Fast inference methods available (e.g. Junction tree algorithm, Viterbi algorithm), depending on the dependency structure of z.

SSVMs: Prediction Error

The hard margin optimization problem

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \max_{z \neq z_i} \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z, \mathbf{y}_i) \geq 1 \quad \forall \mathbf{y}_i \in \mathcal{Y} \end{aligned}$$

may not be feasible (inconsistent training data, or because the model class is not powerful enough).

To allow and quantify errors we introduce a loss function $\Delta \colon \mathbb{K} \times \mathbb{K} \to \mathbb{R}$, where $\Delta(z', z)$ is the loss of predicting z' when the correct output is z.

Reformulated problem with slacks $\xi_i \geq 0$ (margin rescaling):

$$\begin{aligned} \min_{\mathbf{w}, \boldsymbol{\xi} \geq 0} \quad & \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z, \mathbf{y}_i) \geq \Delta(z, z_i) - \xi_i \quad \forall \mathbf{y}_i \in \mathcal{Y}, z \neq z_i \\ & \Leftrightarrow \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \max_{z \neq z_i} [\Delta(z, z_i) + \mathbf{w}^\mathsf{T}\boldsymbol{\Psi}(z, \mathbf{y}_i)] \geq -\xi_i \quad \forall \mathbf{y}_i \in \mathcal{Y} \end{aligned}$$

SSVMs: Efficient Training

Since not all constraints can be enumerated (the number of classes is often too large), we cannot use a standard QP solver. A cutting-plane algorithm as follows can be used:

Algorithm Training SSVMs with margin-rescaling

Require: training data $((z_1, \mathbf{y}_1), \ldots, (z_n, \mathbf{y}_n))$, tradeoff parameter C, precision ϵ

repeat

for $i = 1, \ldots, n$ **do**

$\tilde{z} \leftarrow \arg\max_{z' \in \mathbb{K}} \left(\Delta(z', z_i) + \mathbf{w}^\mathsf{T} \boldsymbol{\Psi}(z', \mathbf{y}_i)\right)$ // Loss augmented inference

if $\mathbf{w}^\mathsf{T}[\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \boldsymbol{\Psi}(\tilde{z}, \mathbf{y}_i)] < \Delta(\tilde{z}, z_i) - \xi_i - \epsilon$ **then**

$\mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{w}^\mathsf{T}[\boldsymbol{\Psi}(z_i, \mathbf{y}_i) - \boldsymbol{\Psi}(\tilde{z}, \mathbf{y}_i)] \geq \Delta(\tilde{z}, z_i) - \xi_i\}$ // Add constraint to constraint set $\mathcal{W}$

$(\mathbf{w}, \boldsymbol{\xi}) \leftarrow \arg\min_{\mathbf{w}', \boldsymbol{\xi}' \geq 0} \frac{1}{2}\mathbf{w}'^\mathsf{T}\mathbf{w}' + C\sum_{i=1}^{n} \xi_i'$ s.t. $\mathcal{W}$

end if

end for

until $\mathcal{W}$ has not changed during iteration

return $\mathbf{w}$ // Return weights

SSVMs: Applying SSVMs to New Tasks

Required functions.

Need design/implementation of the following four functions:

- Joint feature map $\mathbf{\Psi}(z, \mathbf{y})$
- Loss function $\Delta(z', z)$
- Prediction rule $h(\mathbf{y}) = \arg\max_{z \in \mathbb{K}} \ \mathbf{w}^\mathsf{T} \mathbf{\Psi}(z, \mathbf{y})$
- Loss augmented inference $\arg\max_{z' \in \mathbb{K}} \left(\Delta(z', z_i) + \mathbf{w}^\mathsf{T} \mathbf{\Psi}(z', \mathbf{y}_i) \right)$

SSVMs: Optimizing Diversity in Search Engines*

Task.

Given a set of documents $\mathcal{D} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{|\mathcal{D}|}\}$, select a subset $S \subseteq \mathcal{D}$ of these documents s.t. $|S| \leq c$. The set S should maximize subtopic coverage.

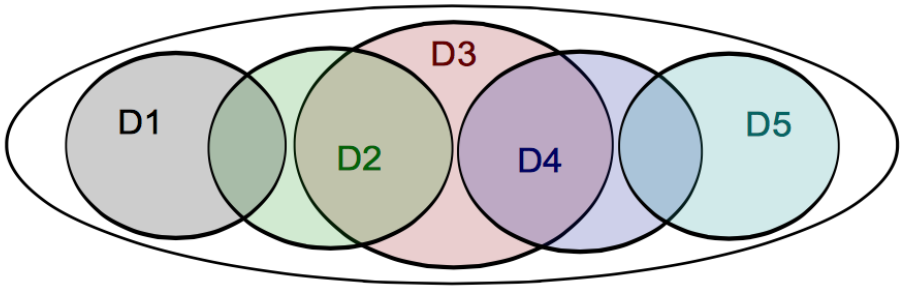
Loss function.

Given a ground truth labeling of the documents corresponding to presented subtopics, i.e. $\mathbf{T} = \{T_1, \ldots, T_n\}$, define $\Delta(\mathbf{T}, S)$ to be the weighted fraction of subtopics not covered by S.

* T. Joachims et al. "Predicting structured objects with support vector machines." Communications of the ACM 52.11 (2009): 97-104.

SSVMs: Prediction Rule

Even if ground truth subtopics where known, computing the best S is difficult.



Problem corresponds to budgeted maximum coverage problem $\Rightarrow$ approximately solvable using greedy algorithm

SSVMs: Feature Maps

In practice, subtopics of new queries are unknown:

- Use words in documents to represent information diversity.
- Covering more (distinct) words $\Leftrightarrow$ covering more subtopics.

Feature map construction.

- Feature vector of word frequencies amongst documents:

$$\phi(v, \mathcal{D}) = \begin{pmatrix} \mathbf{1}[v \text{ appears in at least 15\% of } \mathcal{D}] \\ \mathbf{1}[v \text{ appears in at least 35\% of } \mathcal{D}] \\ \cdots \end{pmatrix}$$

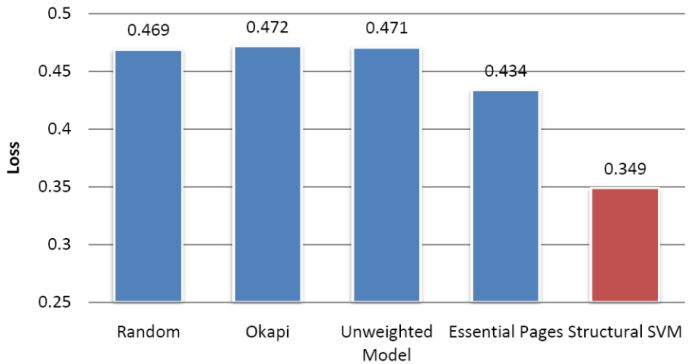
- Feature map:

$$\mathbf{\Psi}(S, \mathcal{D}) = \sum_{v \in V(S)} \phi(v, \mathcal{D}),$$

where $V(S)$ is the union of set of words in documents in S.

SSVMs: Results

Experiments on TREC 6-8 Interactive Track queries.



(Loss for retrieving 5 documents.)

OKAPI: Retrieval function not accounting for diversity
Unweighted Model: All words have equal benefit
Essential Pages: Pre-defined word weighting functions

SSVMs: Image Collection Summarization*

Task.

Given a set (collection) of images $\mathcal{I} = \{I_1, \ldots, I_{|\mathcal{I}|}\}$, select a subset $S \subseteq \mathcal{I}$ of these images s.t. $|S| \leq c$. The set S should summarize the collection as good as possible.

Approach.

Learn a scoring function $f_{\mathbf{w}}(S, \mathcal{I}) = \sum_{i=1}^{m} w_i p_i(S, \mathcal{I})$, where $p_i(S, \mathcal{I})$ are functions measuring coverage/diversity of S in the context of $\mathcal{I}$. Then, summarization of image collection $\mathcal{I}$ is performed by computing

$$S^* = \arg \max_{S \subseteq V, |S| \leq c} f_{\mathbf{w}}(S, \mathcal{I}).$$

* Tschiatschek, S., Bilmes, J., Haochen, W., & Iyer, R. (2014). "Learning Mixtures of Submodular Functions for Image Collection Summarization". NIPS 2014.

SSVMs: Data & Training

Data.

- 14 image collections with 100 images each
- $\sim$ 400 human generated summaries for each collection (collected on Amazon mechanical turk)

Training.

- $m = 594$

  The function f is the sum of 594 functions modeling coverage/likelihood and diversity.
- Visual-ROUGE used as loss function Δ

  ROUGE is a recall oriented measure commonly used to assess the quality of textual summaries. V-ROUGE adopts the ROUGE score for images.
- Training using cutting-plane algorithm

  We trained on 13 collections, and tested the learned SSVM on the 14^{th} collection.

SSVMs: Testing

Generate summary of size 10 (there are $\binom{100}{10} = 17.310.309.456.440$ different candidate subsets)

whole collection

$\Rightarrow$

summary

SSVMs: Complete Results

Cross-validation experiments

	Limits		Our Method			Baseline Methods				
No.	Min	Max	ℓ_{1-R}	ℓ_c	ℓ_S	FL	FL_{pen}	MMR	GC_{pen}	kM
1	-2.55	2.78	**1.51**	0.87	-0.36	**1.45**	1.31	-0.51	0.85	-0.84
2	-2.06	2.22	**1.27**	1.26	0.44	0.18	**0.79**	0.65	-0.16	-0.41
3	-2.07	2.24	**1.46**	0.95	0.23	0.47	0.74	**0.85**	-0.66	-0.81
4	-3.20	2.04	**1.04**	0.81	-0.18	0.71	**1.13**	0.51	-0.01	-0.10
5	-1.65	1.92	**1.11**	1.06	0.58	0.96	**1.10**	0.95	-0.64	0.21
6	-2.83	2.40	**1.47**	0.65	0.27	**1.26**	0.88	-0.08	-0.46	-0.91
7	-2.44	2.07	**1.07**	0.96	0.15	**0.93**	0.81	-0.33	-0.58	-0.58
8	-1.66	2.04	**1.13**	0.96	0.07	**0.62**	**0.62**	0.57	-1.28	0.00
9	-2.32	2.59	**1.21**	1.13	0.51	**0.81**	0.71	0.09	0.09	-0.86
10	-1.46	2.34	**1.06**	0.78	0.14	**1.58**	1.42	-0.26	0.50	-0.14
11	-1.55	1.85	**0.95**	0.92	-0.08	0.43	**0.60**	-0.29	-0.11	0.35
12	-1.74	2.39	**1.11**	0.58	0.12	0.78	**1.04**	0.02	0.30	-0.42
13	-0.94	1.72	0.32	**0.53**	0.14	0.02	0.11	**0.52**	-0.09	0.49
14	-1.46	1.75	**1.08**	0.97	0.77	0.23	**0.37**	0.22	-0.32	0.12
Avg.	-2.00	2.17	**1.13**	0.89	0.20	0.75	**0.83**	0.21	-0.18	-0.28