Digit Recognition

with Support Vector Machines

Lisa Gaedke-Merzhäuser
Paul Korsmeier
Lisa Mattrisch
Vanessa Schreck

Freie Universität Berlin, Mathematical Aspects of Machine Learning

July 19, 2017

Outline

1. Introduction to Our Data Set
2. Our Approach
3. Sequential Minimal Optimization (SMO)
4. Multi-Class Classification
5. Results & Conclusions

Introduction to Our Data Set

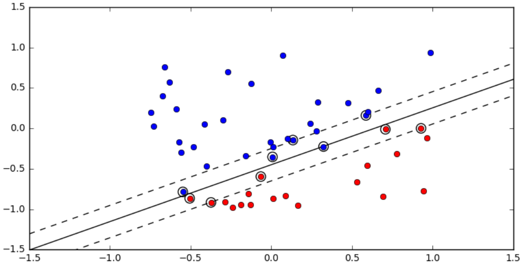
Main Goal: train algorithm to recognize handwritten digits

Data:

- 42,000 greyscale images
- 28 by 28 pixels each
- partitioned into ten classes

Our Approach

We want to use the concept of SVMs.



- **Problem I:** SVMs are binary classifiers
- **Problem II:** Need to solve optimization problem

Our Approach

1. Implement solver for our QP
   - 3 versions
2. Implement basic SVM algorithm
   - linear kernel / Gaussian kernel
   - Parameter optimization
3. Combine individual SVMs in different ways
   - 3 versions
4. Validate and compare results

Sequential Minimal Optimization (SMO)

- The primal Soft Margin SVM QP is equivalent to solving the **dual problem:**

$$\begin{aligned} \text{minimize} \qquad & d(\alpha) := \tfrac{1}{2}\alpha^T Q\alpha - 1^T\alpha \\ \text{s.t.} \quad & 0 \leq \alpha \leq C \quad \text{and} \quad y^T\alpha = 0, \end{aligned}$$

  where $q_{ij} = y_i y_j k(x_i, x_j)$, x_i the data, y_i the labels, k the kernel function and C the penalty term

- Since Q is spsd, satisfying the KKT conditions guarantees a solution to (1).

Sequential Minimal Optimization (SMO)

- Lagrangian of dual objective d:
  $\mathcal{L}(\alpha, \delta, \mu, \beta) = d(\alpha) - \delta^T \alpha + \mu^T (\alpha - C) - \beta \alpha^T y$
- **KKT conditions** for dual Lagrangian:

$$\left.\begin{aligned} \nabla_\alpha \mathcal{L}(\alpha^*, \delta^*, \mu^*, \beta^*) &= 0 \\ \delta_i^* &\geq 0 \\ \delta_i^* \alpha_i^* &= 0 \\ \mu_i^* &\geq 0 \\ \mu_i^* (\alpha_i^* - C) &= 0 \\ \alpha_i^* & \text{ feasible} \end{aligned}\right\} \text{for all } i \in \{1, \ldots, l\}$$

- Define $F_i(\alpha) := y_i (\partial_i d)(\alpha) = \sum_{i=1}^{l} \alpha_j y_j k(x_i, x_j) - y_i$.

Sequential Minimal Optimization (SMO)

- The KKT conditions are equivalent to:

$$b_{up}(\alpha) := \min_{i \in I_{up}(\alpha)} F_i(\alpha) \geq \max_{j \in I_{low}(\alpha)} F_j(\alpha) =: b_{low}(\alpha),$$

  where
  - $I_{up}(\alpha) := \{i \mid \alpha_i < C, y_i = 1 \text{ or } \alpha_i > 0, y_i = -1\}$
  - $I_{low}(\alpha) := \{j \mid \alpha_j < C, y_j = -1 \text{ or } \alpha_j > 0, y_j = 1\}$.
- Relax to $b_{up}(\alpha) \geq b_{low}(\alpha) - \tau$ for some tolerance $\tau > 0$.
- A pair $(i, j) \in I_{up}(\alpha) \times I_{low}(\alpha)$ with $F_i(\alpha) < F_j(\alpha) - \tau$ is called **τ-violating.**

Sequential Minimal Optimization (SMO)

Algorithm (General SMO type algorithm)

Let $\tau > 0$. *Initialize* $k = 0$ *and* $\alpha^0 = 0$ *and generate iterates* α^k, $k \in \mathbb{N}$, *as follows:*

1. *If* α^k *satisfies* $b_{up}(\alpha^k) \geq b_{low}(\alpha^k) - \tau$, *stop. Else* ***pick*** *a* τ*-violating pair* $(i, j) \in I_{up}(\alpha^k) \times I_{low}(\alpha^k)$.
2. ***Minimize*** d *only in* α_i^k *and* α_j^k, *leaving* α_n^k *fixed for* $n \notin \{i, j\}$ *and respecting constraints.* $\rightarrow$ *Obtain* α^{new}.
3. *Set* $k := k + 1$, $\alpha^k := \alpha^{new}$ *and go to Step 1.*

Sequential Minimal Optimization (SMO)

- Each step of GSMO is only a (clipped) **one-dimensional QP** → analytic solution known → **cheap**.
- Two heuristics for choosing violating pair:
  - WSS1: steepest possible **gradient**

$$(i_{up}, i_{low}) \in \operatorname{argmin}_{i \in I_{up}(\alpha)} F_i(\alpha) \times \operatorname{argmax}_{j \in I_{low}(\alpha)} F_j(\alpha)$$

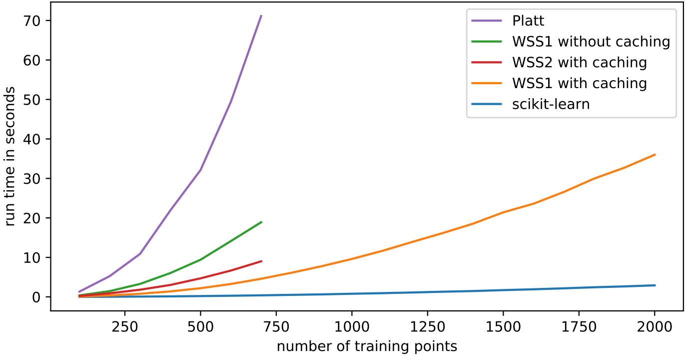
  - WSS2: maximal possible **decrease** in d

$$(i, j) \in \operatorname{argmin}_{i \in I_{up}(\alpha)} F_i(\alpha) \times I_{low}(\alpha) : d(\alpha^{new}) - d(\alpha) \to \min$$

- WSS2 seems promising, but is too expensive.

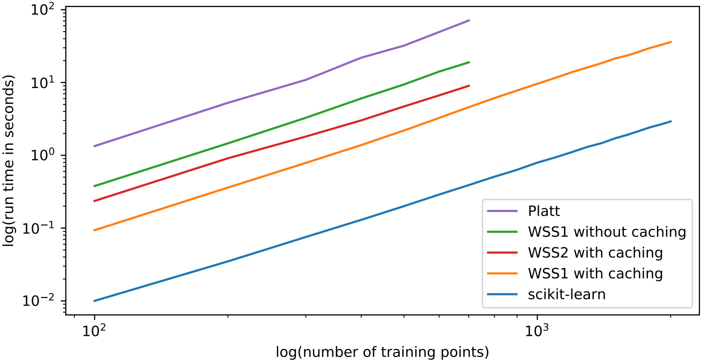
Sequential Minimal Optimization (SMO)

- **Run time comparison** of algorithms with Gaussian kernel and labels by first ECOC classifier on our digits:



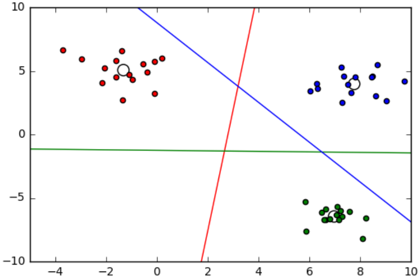
Sequential Minimal Optimization (SMO)

- All algorithms seem to have **polynomial order 2.**
- Our WSS1 with caching runs about 5.8 times as long as scikit-learn SVC.



Multi-Class Classification

- Choose k groups of the classes
- Train k SVMs that separate each group from the rest
- Compare outcome to what would arise for each digit.
- **Problem:** Points may not be classified uniquely.
- Handle overlappings by minimizing distance to barycenters



Multi-Class Classification

1. One-vs-All

Idea: For each $i \in \{0, 1, \ldots, 9\}$, train an SVM that separates class i from the rest

Class	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
0	-1	1	1	1	1	1	1	1	1	1
1	1	-1	1	1	1	1	1	1	1	1
2	1	1	-1	1	1	1	1	1	1	1
3	1	1	1	-1	1	1	1	1	1	1
4	1	1	1	1	-1	1	1	1	1	1
5	1	1	1	1	1	-1	1	1	1	1
6	1	1	1	1	1	1	-1	1	1	1
7	1	1	1	1	1	1	1	-1	1	1
8	1	1	1	1	1	1	1	1	-1	1
9	1	1	1	1	1	1	1	1	1	-1

Multi-Class Classification

2. Error Correcting Output Codes

Idea: Relabeling with large Hamming distance according to:

Class	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	−1	−1	−1	−1	1	−1	1	−1	−1	1	1	−1	1
1	−1	−1	1	1	1	1	−1	1	−1	1	1	−1	−1	1	−1
2	1	−1	−1	1	−1	−1	−1	1	1	1	1	−1	1	−1	1
3	−1	−1	1	1	−1	1	1	1	−1	−1	−1	−1	1	−1	1
4	1	1	1	−1	1	−1	1	1	−1	−1	1	1	−1	−1	1
5	−1	1	−1	−1	1	1	−1	−1	1	1	−1	−1	−1	−1	1
6	1	−1	1	1	1	−1	−1	−1	−1	1	−1	1	−1	−1	1
7	−1	−1	−1	1	1	1	1	−1	1	−1	1	1	−1	−1	1
8	1	1	−1	1	−1	1	1	−1	−1	1	−1	−1	−1	1	1
9	−1	1	1	1	−1	−1	−1	−1	1	−1	1	−1	−1	1	1

Results & Conclusions

# training points	One-vs-All uniquely classfied, linear	One-vs-All with bary-centers, linear	One-vs-All uniquely classfied, Gaussian	One-vs-All with bary-centers, Gaussian	ECOC, linear	ECOC, Gaussian
500						
1000						
2000						
5000						
10000						

Table: Correctly Classified Digits

Results & Conclusions

# training points	One-vs-All uniquely classfied, linear	One-vs-All with bary-centers, linear	One-vs-All uniquely classfied, Gaussian	One-vs-All with bary-centers, Gaussian	ECOC, linear	ECOC, Gaussian
500	65.9%	74.1%	75.4%	83.3%	74.2%	87.4%
1000	68.2%	75.0%	84.3%	89.0%	78.0%	92.7%
2000	70.2%	76.4%	89.8%	91.9%	77.8%	94.3%
5000	70.0%	73.8%	88.9%	91.6%	82.0%	95.2%
10000	64.6%	67.5%	88.0%	90.6%	82.5%	95.4%

Table: Correctly Classified Digits

Results & Conclusions

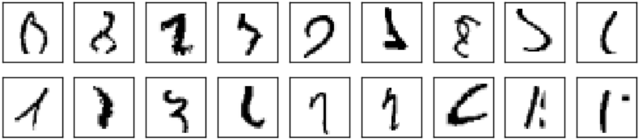


Figure: Visualizing very illegible digits