Machine Learning 2

3. (Advanced) Support Vector Machines (SVMs)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

		A. Advanced Supervised Learning
Fri. 12.4.	(1)	A.1 Generalized Linear Models
Fri. 26.4.	(2)	A.2 Gaussian Processes
Fri. 3.5.	(3)	A.2b Gaussian Processes (ctd.)
Fri. 10.5.	(4)	A.3 Advanced Support Vector Machines
		B. Ensembles
Fri. 17.5.	(5)	B.1 Stacking
Fri. 24.5.	(6)	B.2 Boosting
Fri. 31.5.	(7)	B.3 Mixtures of Experts
		C. Sparse Models
Fri. 7.6.	(8)	C.1 Homotopy and Least Angle Regression
Fri. 14.6.	—	— Pentecoste Break —
Fri. 21.6.	(9)	C.2 Proximal Gradients
Fri. 28.6.	(10)	C.3 Laplace Priors
Fri. 29.6.	(11)	C.4 Automatic Relevance Determination
		D. Complex Predictors
Fri. 6.7.	(12)	D.1 Latent Dirichlet Allocation (LDA)
Fri. 12.7.	(13)	Q & A

Outline

Outline

1. Stochastic (Sub)gradient Descent

2. Dual Coordinate Descent

3. The Adaptive Multi Hyperplane Machine

SVM Optimization Problem / Slack Variables

$$
\begin{aligned}
\text{minimize } & \frac{1}{2}||\beta||^2 + \gamma \sum_{n=1}^{N} \xi_n \\
\text{w.r.t. } y_n(\beta_0 + \beta^T x_n) \geq & 1 - \xi_n, \quad n = 1, \dots, N \\
\xi \geq & 0 \\
\beta \in & \mathbb{R}^p, \quad \beta_0 \in \mathbb{R}
\end{aligned}
$$

SVM Optimization Problem / Slack Variables

$$
\begin{aligned}
\text{minimize } & \frac{1}{2}||\beta||^2 + \gamma \sum_{n=1}^{N} \xi_n \\
\text{w.r.t. } y_n(\beta_0 + \beta^T x_n) \geq & 1 - \xi_n, \quad n = 1, \dots, N \\
\xi \geq & 0 \\
\beta \in & \mathbb{R}^p, \quad \beta_0 \in \mathbb{R}
\end{aligned}
$$

can be rewritten:

$$
\begin{aligned}
\text{minimize } f(\beta) := & \frac{1}{2}||\beta||^2 + \gamma \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) \\
\propto & \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2}\lambda||\beta||^2, \quad \lambda := \frac{1}{\gamma N}
\end{aligned}
$$

SVM Optimization Problem / Hinge Loss

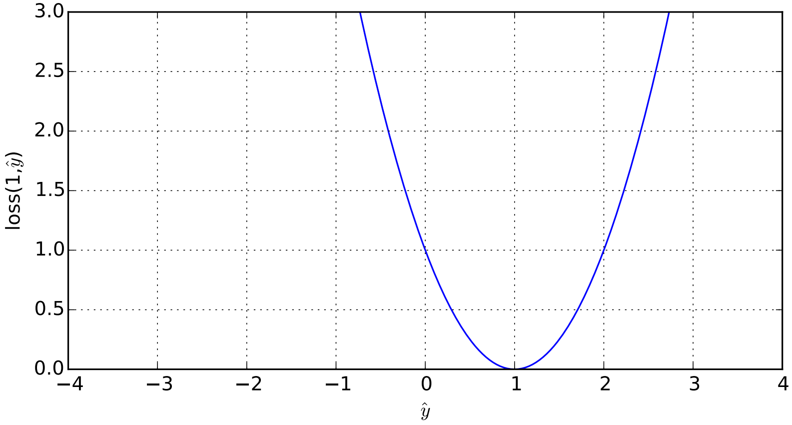
can be rewritten (ctd.):

$$
\begin{aligned}
\text{minimize } f(\beta) :=& \frac{1}{2}||\beta||^2 + \gamma \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) \\
\propto& \frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2}\lambda||\beta||^2, \quad \lambda := \frac{1}{\gamma N} \\
=& \frac{1}{N} \sum_{n=1}^{N} \ell_{\text{hinge}}(y_n, \beta_0 + \beta^T x_n) + \frac{1}{2}\lambda||\beta||^2
\end{aligned}
$$

with

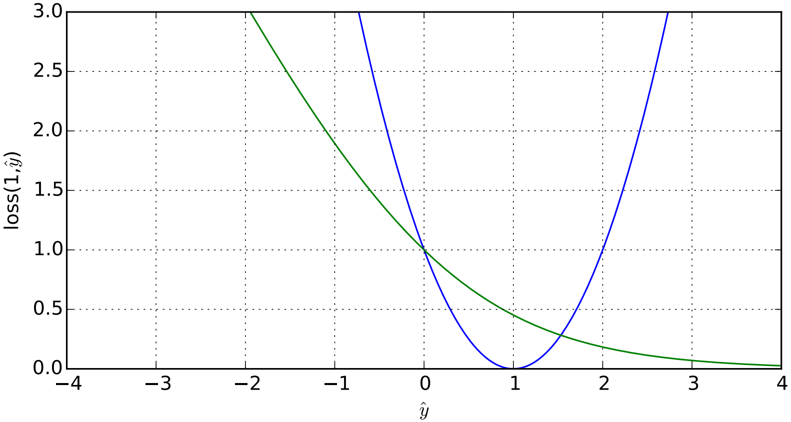
$$\ell_{\text{hinge}}(y, \hat{y}) := \max(0, 1 - y\hat{y})$$

Brief Digression: Losses $\ell(y = 1, \hat{y})$



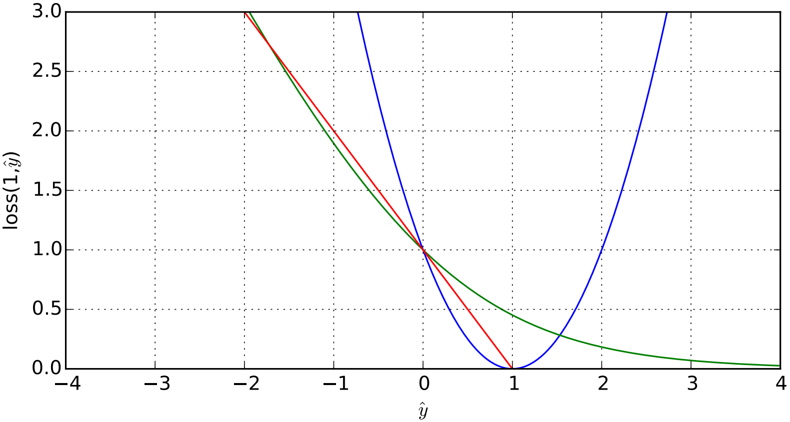
blue: squared error: $(1 - \hat{y})^2$
green: logistic loss: $\ln(1 + e^{-1 \cdot \hat{y}}) / \ln(2)$
red: hinge loss: $\max(0, 1 - 1 \cdot \hat{y})$

Brief Digression: Losses $\ell(y = 1, \hat{y})$



blue: squared error: $(1 - \hat{y})^2$
green: logistic loss: $\ln(1 + e^{-1 \cdot \hat{y}}) / \ln(2)$
red: hinge loss: $\max(0, 1 - 1 \cdot \hat{y})$

Brief Digression: Losses $\ell(y = 1, \hat{y})$



blue: squared error: $(1 - \hat{y})^2$
green: logistic loss: $\ln(1 + e^{-1\cdot\hat{y}})/\ln(2)$
red: hinge loss: $\max(0, 1 - 1 \cdot \hat{y})$

(Sub)gradients

$$
\begin{aligned}
f(\beta) &:= \frac{1}{N}\sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2}\lambda||\beta||^2 \\
&= \frac{1}{N} \sum_{\substack{n=1 \\ y_n(\beta_0+\beta^T x_n)<1}}^{N} 1 - y_n(\beta_0 + \beta^T x_n) + \frac{1}{2}\lambda||\beta||^2
\end{aligned}
$$

(Sub)gradients

$$f(\beta) := \frac{1}{N}\sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2}\lambda||\beta||^2$$

$$= \frac{1}{N} \sum_{\substack{n=1 \\ y_n(\beta_0+\beta^T x_n)<1}}^{N} 1 - y_n(\beta_0 + \beta^T x_n) + \frac{1}{2}\lambda||\beta||^2$$

subgradients:

$$\frac{\partial f}{\partial \beta} = \frac{1}{N} \sum_{\substack{n=1 \\ y_n(\beta_0+\beta^T x_n)<1}}^{N} -y_n x_n + \lambda\beta$$

(Sub)gradients

$$f(\beta) := \frac{1}{N}\sum_{n=1}^{N} \max(0, 1 - y_n(\beta_0 + \beta^T x_n)) + \frac{1}{2}\lambda||\beta||^2$$

$$= \frac{1}{N} \sum_{\substack{n=1 \\ y_n(\beta_0+\beta^T x_n)<1}}^{N} 1 - y_n(\beta_0 + \beta^T x_n) + \frac{1}{2}\lambda||\beta||^2$$

subgradients:

$$\frac{\partial f}{\partial \beta} = \frac{1}{N} \sum_{\substack{n=1 \\ y_n(\beta_0+\beta^T x_n)<1}}^{N} -y_n x_n + \lambda\beta$$

stochastic subgradients:

$$\frac{\partial f}{\partial \beta}|_{\mathcal{D}^{(t)}} = \frac{1}{|\mathcal{D}^{(t)}|} \sum_{\substack{(x,y)\in\mathcal{D}^{(t)} \\ y(\beta_0+\beta^T x)<1}}^{N} -yx + \lambda\beta, \quad \mathcal{D}^{(t)} \subseteq \mathcal{D}^{\text{train}}, \text{iteration } t$$

Bound on Parameter Norm

The optimal parameters are bound from above:

$$||\beta^*|| \leq \frac{1}{\sqrt{\lambda}}$$

Trivially,

$$\frac{1}{2}\lambda||\beta^*||^2 \leq f(\beta^*) \leq f(0) = 1$$

$$\rightsquigarrow ||\beta^*|| \leq \frac{\sqrt{2}}{\sqrt{\lambda}}$$

[SSS07] have a more complex proof to show the tighter bound (p. 4, end of proof of theorem 1).

Primal Estimated Subgradient Solver for SVMs (Pegasos)

- use **stochastic (sub)gradient descent**

$$\tilde{\beta}^{(t+1)} := \beta^{(t)} - \eta^{(t)} \frac{\partial f}{\partial \beta}|_{\mathcal{D}^{(t)}}$$

  - use gradient sample size K (aka **mini batches**)
    - though no empirical evidence that $K > 1$ has any benefits

- after each SGD step, **reproject/rescale** β:

$$\beta^{(t+1)} := \tilde{\beta}^{(t+1)} \frac{1}{\max(1, \sqrt{\lambda}\,||\tilde{\beta}^{(t+1)}||)}$$

- use **fixed hyperbola schedule as learning rate**:

$$\eta^{(t)} := \frac{1}{\lambda t}$$

- see [SSS07]

Performance Comparison

Table 1. Training time in CPU-seconds

	Pegasos	SVM-Perf	SVM-Light
CCAT	2	77	20,075
Covertype	6	85	25,514
astro-ph	2	5	80

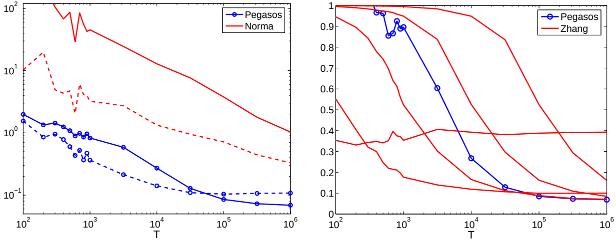


Figure 2. Comparisons of Pegasos to Norma (left) and Pegasos to stochastic gradient descent with a fixed learning rate (right) on the Astro-Physics datset. In the left plot, the solid lines designate the objective value and the dashed lines depict the loss on the test set.

[SSS07]

Performance Comparison (2/2)

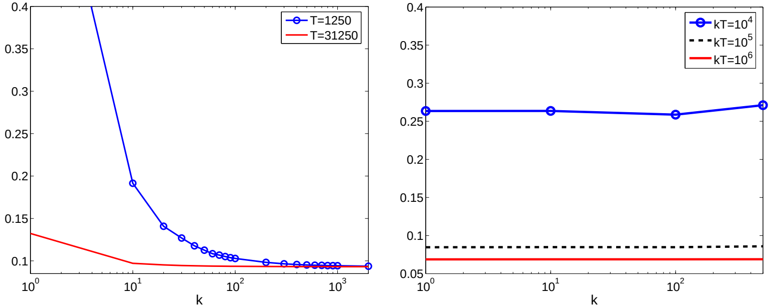


Figure 3. The effect of k on the objective value of Pegasos on the Astro-Physics dataset. Left: T is fixed. Right: kT is fixed.

[SSS07]

(Non-Linear) Kernels

SGD in the primal first works for **linear kernels**.

Any linear model can be kernelized by **representing instances in terms of kernel features**:

original feature representation:

$$x_n \in \mathbb{R}^M, \quad n \in \{1, \ldots, N\}$$

kernel feature representation:

$$\tilde{x}_n \in \mathbb{R}^N, x_{n,m} := k(x_n, x_m), \quad m \in \{1, \ldots, N\}$$

then:

$$\begin{aligned}\hat{y}_{\text{linear}}(\tilde{x}_n; \beta) =& \beta^T \tilde{x}_n = \sum_{m=1}^{N} \beta_m \tilde{x}_{n,m} \\ &= \sum_{m=1}^{N} \alpha_m k(x_m, x_n) = \hat{y}_{\text{kernel } k}(x_n; \alpha), \quad \alpha_m := \beta_m\end{aligned}$$

Outline

Dual Problem

Remember, the dual problem was:

$$\text{minimize } f(\alpha) := \frac{1}{2}\alpha^T Q\alpha - 1^T\alpha, \quad Q_{n,m} := y_n y_m x_n^T x_m$$

$$\text{w.r.t. } \alpha \in [0, \frac{1}{N\lambda}]$$

Dual Problem

Remember, the dual problem was:

$$\text{minimize } f(\alpha) := \frac{1}{2}\alpha^T Q\alpha - 1^T\alpha, \quad Q_{n,m} := y_n y_m x_n^T x_m$$

$$\text{w.r.t. } \alpha \in [0, \frac{1}{N\lambda}]$$

coordinate descent w.r.t. coordinate α_n:

$$\begin{aligned} f_n(\alpha_n) :=& f(\alpha_n; \alpha_{-n}) \propto \frac{1}{2}Q_{n,n}\alpha_n^2 + Q_{n,-n}\alpha_{-n}\alpha_n - \alpha_n \\ \frac{\partial f_n}{\partial \alpha_n} =& Q_{n,n}\alpha_n + Q_{n,-n}\alpha_{-n} - 1 \overset{!}{=} 0 \\ \rightsquigarrow \quad \alpha_n =& \frac{1 - Q_{n,-n}\alpha_{-n}}{Q_{n,n}} \end{aligned}$$

possibly clip α_n:

$$\alpha_n = \max(0, \min(\frac{1}{N\lambda}, \frac{1 - Q_{n,-n}\alpha_{-n}}{Q_{n,n}}))$$

Avoid Computing $Q_{n,-n}\alpha_{-n}$

$$
\begin{aligned}
\alpha_n^{(t+1)} :=& \frac{1 - Q_{n,-n}\alpha_{-n}^{(t)}}{Q_{n,n}} \\
=& \frac{1 - Q_{n,.}\alpha^{(t)} + Q_{n,n}\alpha_n^{(t)}}{Q_{n,n}} \\
=& \alpha_n^{(t)} - \frac{Q_{n,.}\alpha^{(t)} - 1}{Q_{n,n}} \\
=& \alpha_n^{(t)} - \frac{y_n \hat{y}_n^{(t)} - 1}{Q_{n,n}}
\end{aligned}
$$

Avoid Computing $Q_{n,-n}\alpha_{-n}$

$$\alpha_n^{(t+1)} = \alpha_n^{(t)} - \frac{y_n \hat{y}_n^{(t)} - 1}{Q_{n,n}}$$

with

$$\hat{y}_n^{(t)} = (\beta^{(t)})^T x_n$$

and due to

$$\beta^{(t)} = \sum_{n=1}^{N} \alpha_n^{(t)} y_n x_n$$

as only $\alpha_n^{(t)}$ changes:

$$\beta^{(t+1)} := \beta^{(t)} + (\alpha_n^{(t+1)} - \alpha_n^{(t)}) y_n x_n$$

- accelerates from $O(N)$ to $O(M)$
- even $O(M_{\text{nz}})$ for sparse predictor vectors x (M_{nz} being the average number of nonzeros)

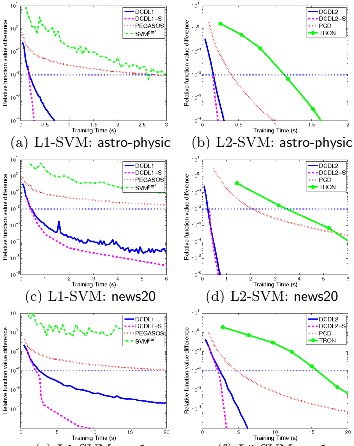
Performance Comparison

Table 2. On the right training time for a solver to reduce the primal objective value to within 1% of the optimal value; see (20). Time is in seconds. The method with the shortest running time is boldfaced. Listed on the left are the statistics of data sets: l is the number of instances and n is the number of features.

Data set	Data statistics			Linear L1-SVM			Linear L2-SVM		
	l	n	# nonzeros	DCDL1	Pegasos	SVM^{perf}	DCDL2	PCD	TRON
a9a	32,561	123	451,592	**0.2**	1.1	6.0	0.4	**0.1**	0.1
astro-physic	62,369	99,757	4,834,550	**0.2**	2.8	2.6	**0.2**	0.5	1.2
real-sim	72,309	20,958	3,709,083	**0.2**	2.4	2.4	**0.1**	0.2	0.9
news20	19,996	1,355,191	9,097,916	**0.5**	10.3	20.0	**0.2**	2.4	5.2
yahoo-japan	176,203	832,026	23,506,415	**1.1**	12.7	69.4	**1.0**	2.9	38.2
rcv1	677,399	47,236	49,556,258	**2.6**	21.9	72.0	**2.7**	5.1	18.6
yahoo-korea	460,554	3,052,939	156,436,656	**8.3**	79.7	656.8	**7.1**	18.4	286.1

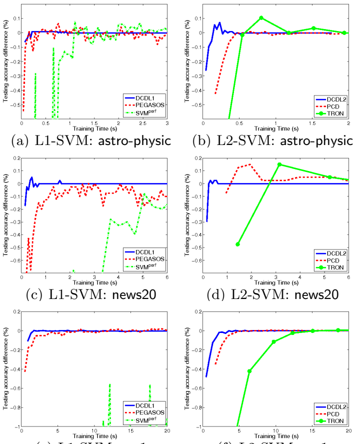
[HCL+08]

Performance Comparison (2/2)



(a) L1-SVM: astro-physic (b) L2-SVM: astro-physic
(c) L1-SVM: news20 (d) L2-SVM: news20
(e) L1-SVM: rcv1 (f) L2-SVM: rcv1

Figure 1. Time versus the relative error (20). DCDL1-S, DCDL2-S are DCDL1, DCDL2 with shrinking. The dotted line indicates the relative error 0.01. Time is in seconds.



(a) L1-SVM: astro-physic (b) L2-SVM: astro-physic
(c) L1-SVM: news20 (d) L2-SVM: news20
(e) L1-SVM: rcv1 (f) L2-SVM: rcv1

Figure 2. Time versus the difference of testing accuracy between the current model and the reference model (obtained using strict stopping conditions). Time is in seconds.

[HCL+08]

Outline

Multi-Class SVM

multi-class SVM:

$$\hat{y}(x) := \arg\max_{y \in \mathcal{Y}} s_y(x)$$

$$s_y(x;\beta) := \beta_y^T x, \quad \beta_y \in \mathbb{R}^M \quad \forall y \in \mathcal{Y} = \{y_1, \ldots, y_L\}$$

$$f(\beta) := \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, x_n) + \frac{\lambda}{2}||\beta||^2, \quad \beta := (\beta_{y_1}, \beta_{y_2}, \ldots, \beta_{y_L})$$

margin-based loss:

$$\ell(y, x; \beta) := \max(0, \quad 1 + \max_{y' \in \mathcal{Y}, y' \neq y} s_{y'}(x) - s_y(x))$$

Multi-Hyperplane Machine

multi-hyperplane score function:

$$s_y(x;\beta) := \max_{k=1,\ldots,K} \beta_{y,k}^T x, \quad \beta_{y,k} \in \mathbb{R}^M, k \in \{1,\ldots,K\}$$

margin-based loss:

$$\ell(y,x;\beta) := \max(0, \quad 1 + \max_{y' \in \mathcal{Y}, y' \neq y} s_{y'}(x) - s_y(x))$$

relaxation / convex upper bound:

$$\ell(y_n,x_n;\beta,z_n) := \max(0, \quad 1 + \max_{y' \in \mathcal{Y}, y' \neq y_n} s_{y'}(x_n) - \beta_{y_n,z_n}^T x_n)$$

- block coordinate descent / EM type training (β, z)
- use SGD to train β.

SGD for Training the Multi-Hyperplane Machine

relaxation / convex upper bound:

$$\ell(y_n, x_n; \beta, z_n) := \max(0, \quad 1 + \max_{y' \in \mathcal{Y}, y' \neq y_n} s_{y'}(x_n) - \beta_{y_n, z_n}^T x_n)$$

gradient:

$$\frac{\partial \ell}{\partial \beta_{y,k}}(y_n, x_n; z_n) = \begin{cases} x_n, & \text{if } (y,k) = \arg\max_{\substack{y' \in \mathcal{Y}, y' \neq y_n \\ k'=1,\ldots,K}} \beta_{y',k'}^T x_n \\ -x_n, & \text{if } (y,k) = (y_n, z_n) \\ 0, & \text{otherwise} \end{cases}$$

Adaptive Multi-Hyperplane Machine:

- initialize $\beta \equiv 0$.
- if all $\beta_{y',k'}^T x < 0^T x = 0$, create a new hyperplane $K+1$ with $\beta_{y,K+1} = 0$.
  (conceptually infinite number of hyperplanes)

Performance Comparison

Table 3: Error rate and training time comparison with large-scale algorithms (RBF SVM is solved by LibSVM unless specified otherwise. Poly2 and LibSVM results are from [5]).

Datasets	Error rate (%)					Training time (seconds)[1]				
	AMM *batch*	AMM *online*	Linear (Pegasos)	Poly2 SVM	RBF SVM	AMM *batch*	AMM *online*	Linear (Pegasos)	Poly2 SVM	RBF SVM
a9a	15.03±0.11	16.44±0.23	15.04±0.07	14.94	14.97	2	0.2	1	2	99
ijcnn	2.40±0.11	3.02±0.14	7.76±0.19	2.16	1.31	2	0.1	1	11	27
webspam	4.50±0.24	6.14±1.08	7.28±0.09	1.56	0.80	80	4	12	3,228	15,571
mnist_bin	0.53±0.05	0.54±0.03	2.03±0.04	NA	0.43[2]	3084	300	277	NA	2 days[2]
mnist_mc	3.20±0.16	3.36±0.20	8.41±0.11	NA	0.67[3]	13864	1200	1180	NA	8 days[3]
rcv1_bin	2.20±0.01	2.21±0.02	2.29±0.01	NA	NA	1100	80	25	NA	NA
url	1.34±0.21	2.87±1.49	1.50±0.39	NA	NA	400	24	100	NA	NA

[1] excludes data loading time.
[2] achieved by parallel training P-packSVMs on 512 processors; results from [28].
[3] achieved by LaSVM; results from [12].

[WDCV11]

Outlook

See [DLVW13] for

- two further scalable learning algorithms for non-linear SVMs,
- an implementation, and
- an evaluation

Summary

- Linear SVMs can be interpreted as linear models with a specific classification loss, the **hinge loss**.
  - not penalizing scores for positive labels > 1 (as squared error) nor encouraging such scores (as logistic loss).
- Linear SVMs simply can be learned by **stochastic (sub)gradient descent**.
  - an additional **reprojection step** can accelerate convergence.
- Linear and nonlinear SVMs can be trained using **coordinate descent** in the dual.
  - for nonlinear SVMs each step is expensive: $O(N)$
  - for linear SVMs, the **primal parameters can be maintained**, yielding a training procedure in $O(M)$ or even $O(M_{\text{nonzero}})$
- Both learning algorithms for linear SVMs are among the fastest currently known.
- Nonlinear SVMs can be approximated by **multiple hyperplanes**.
  - always using the most positive one (maximum over score functions)
  - hyperplanes can be added as needed, once a point is on the wrong side of all hyperplanes.

Further Readings

- See the cited original papers.
- Multi-class SVM:
  - [WW98]
  - [Mur12, section 14.5.2.4]

  and

Acknowledgements: Thanks to Randolf Scholz for pointing out a mistake in an earlier version of these slides.

References

Nemanja Djuric, Liang Lan, Slobodan Vucetic, and Zhuang Wang.
BudgetedSVM: A Toolbox for Scalable SVM Approximations.
J. Mach. Learn. Res., 14(1):3813–3817, December 2013.

C. J Hsieh, K. W Chang, C. J Lin, S. S Keerthi, and S. Sundararajan.
A dual coordinate descent method for large-scale linear SVM.
In *Proceedings of the 25th International Conference on Machine Learning*, pages 408–415, 2008.

Kevin P. Murphy.
Machine Learning: A Probabilistic Perspective.
The MIT Press, 2012.

S. Shalev-Shwartz, Y. Singer, and N. Srebro.
Pegasos: Primal estimated sub-gradient solver for svm.
In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, 2007.

Zhuang Wang, Nemanja Djuric, Koby Crammer, and Slobodan Vucetic.
Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification.
In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 24–32. ACM, 2011.

Jason Weston and Chris Watkins.
Multi-class support vector machines.
Technical report, Royal Holloway, University of London, 1998.