Large-Scale Kernel Methods - II

Sanjiv Kumar, Google Research, NY

EECS-6898, Columbia University - Fall, 2010

Kernel Methods

Provide a flexible way to generate nonlinear decision functions

Suppose $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) \qquad x \in \Re^d, \alpha \in \Re^n$

Kernel SVM $\hat{y} = \mathrm{sgn}[f(x)] \qquad y \in \{-1, 1\}$

Kernel regression $\hat{y} = f(x) \qquad y \in \Re$

Kernel Logistic Regression $p(\hat{y} = 1 \mid x) = \sigma(f(x)) \quad y \in \{-1, 1\}$

Kernel Methods

Provide a flexible way to generate nonlinear decision functions

Suppose $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$ $\quad x \in \Re^d, \alpha \in \Re^n$

Kernel SVM $\hat{y} = \mathrm{sgn}[f(x)]$ $\quad y \in \{-1, 1\}$

Kernel regression $\hat{y} = f(x)$ $\quad y \in \Re$

Kernel Logistic Regression $p(\hat{y} = 1 \mid x) = \sigma(f(x))$ $\quad y \in \{-1, 1\}$

Goal in Learning

Find the best α that minimizes a L_2-regularized loss function

$$J(\alpha) = \sum_{i=1}^{n} L(f(x_i), y_i) + \lambda \alpha^T K \alpha$$

Kernel Methods

Provide a flexible way to generate nonlinear decision functions

Suppose $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) \qquad x \in \Re^d, \alpha \in \Re^n$

Kernel SVM $\hat{y} = \mathrm{sgn}[f(x)] \qquad y \in \{-1, 1\}$

Kernel regression $\hat{y} = f(x) \qquad y \in \Re$

Kernel Logistic Regression $p(\hat{y} = 1 \mid x) = \sigma(f(x)) \quad y \in \{-1, 1\}$

Goal in Learning

Find the best α that minimizes a L_2-regularized loss function

$$J(\alpha) = \sum_{i=1}^{n} L(f(x_i), y_i) + \lambda \alpha^T K \alpha$$

Kernel SVM $L = \max\{0, 1 - \overbrace{y_i f(x_i)}^{\text{margin}}]$

hinge-loss: arises from margin constraints → lead to sparse α

Kernel Methods

Provide a flexible way to generate nonlinear decision functions

Suppose $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) \qquad x \in \Re^d, \alpha \in \Re^n$

Kernel SVM $\hat{y} = \mathrm{sgn}[f(x)] \qquad y \in \{-1, 1\}$

Kernel regression $\hat{y} = f(x) \qquad y \in \Re$

Kernel Logistic Regression $p(\hat{y} = 1 \mid x) = \sigma(f(x)) \quad y \in \{-1, 1\}$

Goal in Learning

Find the best α that minimizes a L_2-regularized loss function

$$J(\alpha) = \sum_{i=1}^{n} L(f(x_i), y_i) + \lambda \alpha^T K \alpha$$

Kernel SVM $L = \max\{0, 1 - \overbrace{y_i f(x_i)}^{\text{margin}}]$ hinge-loss: arises from margin constraints → lead to sparse α

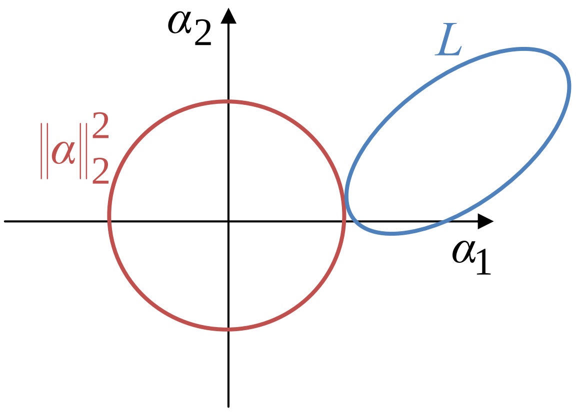
Kernel regression $L = (y_i - f(x_i))^2$

Kernel Logistic Regression $L = -\log\{\sigma(f(x_i))\}$

Need to induce sparsity in α, e.g., by replacing $\alpha^T K \alpha$ with $\|\alpha\|_1$

L_2 vs L_1 Regularizer

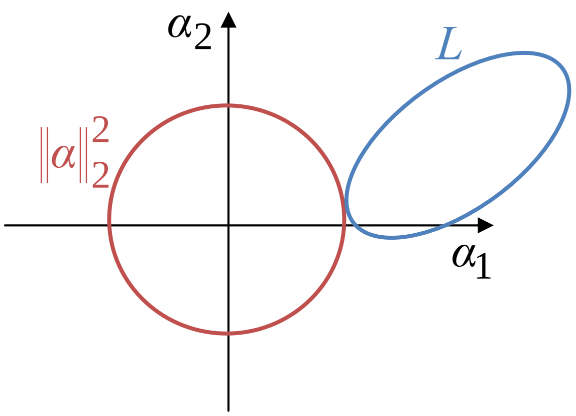
A 2-D illustration

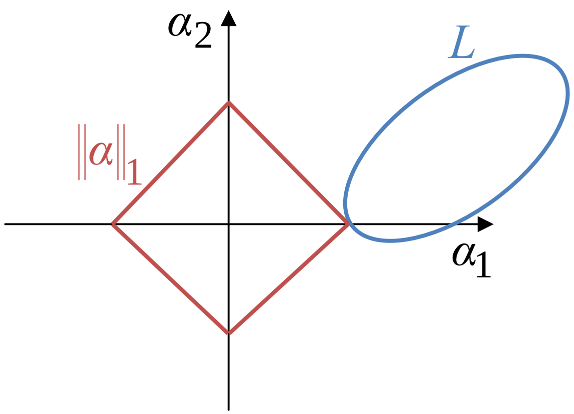


Both α_1 and α_2 are nonzero

L_2 vs L_1 Regularizer

A 2-D illustration





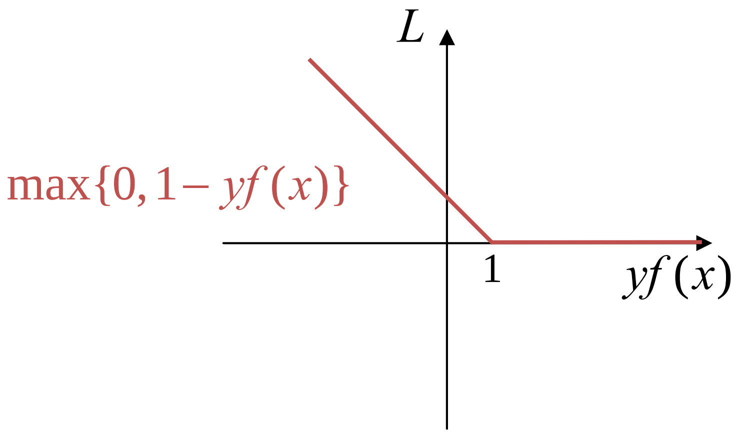
Both α_1 and α_2 are nonzero

Only α_1 is nonzero

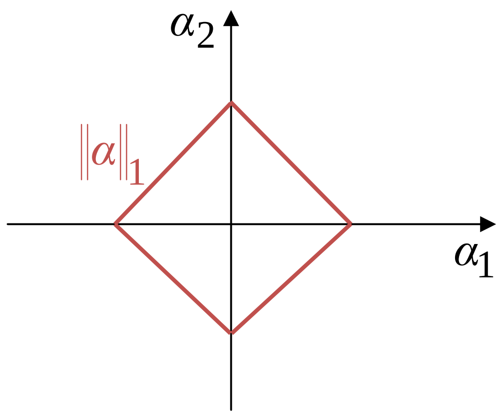
sparse solution !

Non-differentiability

Both hinge-loss and L_1 regularizer are non-differentiable



Hinge Loss



L_1-regularization

Gradients cannot be computed at kinks !

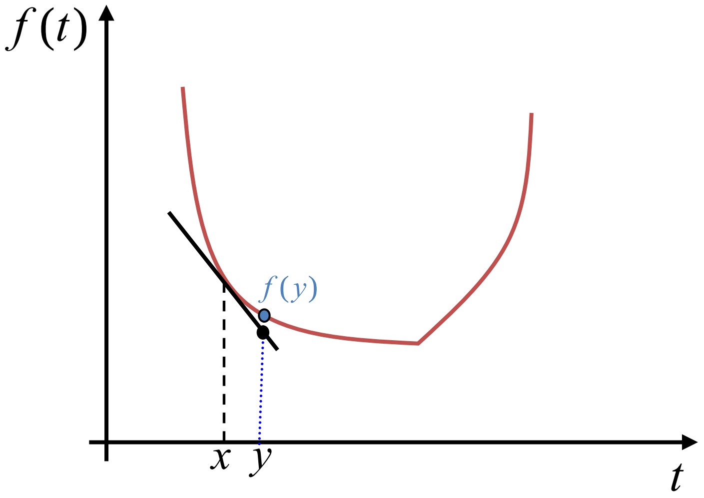
Focus on max-margin formulations, L_1 regularization later

Subgradient

For a convex, differentiable function f,

$$f(y) \geq f(x) + \nabla f^T (y - x)$$

RHS is a global underestimator of f

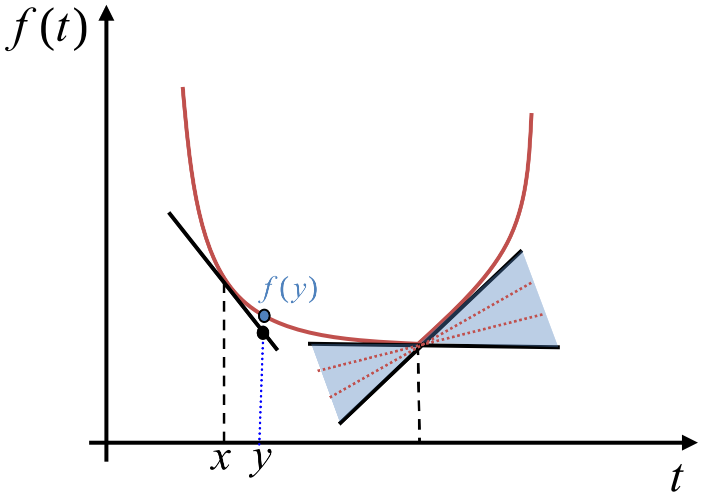


Subgradient

For a convex, differentiable function f,

$$f(y) \geq f(x) + \nabla f^T (y - x) \quad \forall y$$

RHS is a global underestimator of f



Subgradient

- A vector g is called subgradient at x if,

$$f(y) \geq f(x) + g^T (y - x) \quad \forall y$$

- A subgradient can exist even if a function is non-differentiable at x
- Set of all subgradients at x is called sub-differential $\partial f(x)$
- For a convex function,
  - sub-differential is always nonempty and a closed convex set
  - If f is differentiable at x, $\partial f(x) = \{\nabla f\}$

Subgradient Example - L_1 Norm

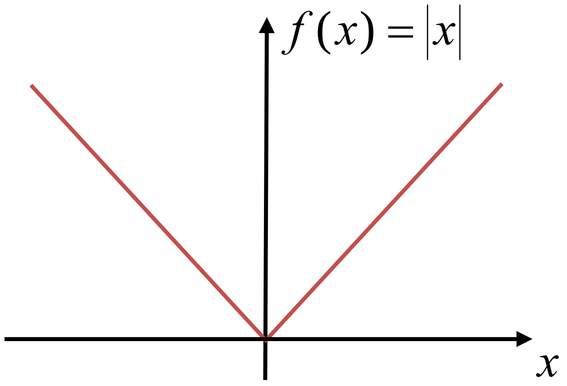
1-Dim case

$$f(x) = |x|, \quad x \in R$$

$$\text{if } x > 0 \quad g = 1$$

$$\text{if } x < 0 \quad g = -1$$

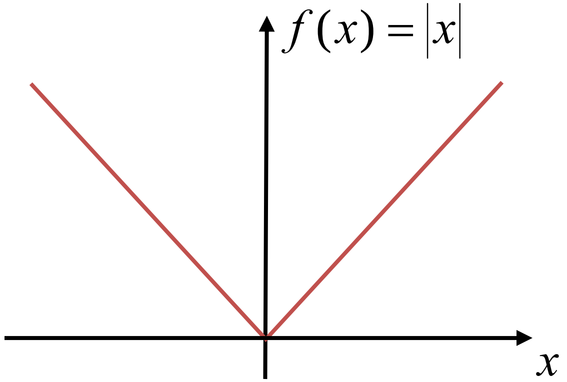
$$\text{if } x = 0 \quad |y| \geq g\,y \Rightarrow g \in [-1, 1]$$



Subgradient Example - L_1 Norm

1-Dim case

$$f(x) = |x|, \quad x \in R$$



$$\text{if } x > 0 \quad g = 1$$

$$\text{if } x < 0 \quad g = -1$$

$$\text{if } x = 0 \quad |y| \geq g\, y \Rightarrow g \in [-1, 1]$$

d-Dim case

$$f(x) = \|x\|_1 = \sum_{j=1}^{d} |x_j|, \quad x \in R^d$$

Rewrite $\|x\|_1 = \max\{s^T x \mid s_i \in \{-1, 1\}\}$

Want to find an s, such that $\|x\|_1 = s^T x$

A simple choice $s_j = 1 \text{ if } x_j > 0 \quad s_j = -1 \text{ if } x_j < 0 \quad s_j = 1 \text{ or } -1 \text{ if } x_j = 0$

$$g_j = \begin{cases} +1 & \text{if } x_j > 0 \\ -1 & \text{if } x_j < 0 \\ +1 \text{ or } -1 & \text{if } x_j = 0 \end{cases}$$

Subgradient Method

Want to minimize $J(\alpha)$

$$\boxed{\alpha_{(k+1)} = \alpha_{(k)} - \eta_{(k)} g_{(k)}} \qquad g_{(k)} \in \partial J(\alpha_{(k)}) \quad \underbrace{\eta_{(k)} = a / \sqrt{k} \text{ or } a/k \quad a > 0}$$

satisfy "square summable but not summable" constraints

$$\sum_{k=1}^{\infty} \eta_{(k)} = \infty \qquad \sum_{k=1}^{\infty} \eta_{(k)}^2 < \infty$$

Subgradient Method

Want to minimize $J(\alpha)$

$$\boxed{\alpha_{(k+1)} = \alpha_{(k)} - \eta_{(k)} g_{(k)}} \qquad g_{(k)} \in \partial J(\alpha_{(k)}) \quad \eta_{(k)} = a/\sqrt{k} \text{ or } a/k \quad a > 0$$

- If the convex function $J(.)$ is differentiable at $\alpha_{(k)}$, the only subgradient is the gradient

  → reduces to gradient descent

Subgradient Method

Want to minimize $J(\alpha)$

$$\boxed{\alpha_{(k+1)} = \alpha_{(k)} - \eta_{(k)} g_{(k)}} \qquad g_{(k)} \in \partial J(\alpha_{(k)}) \quad \eta_{(k)} = a/\sqrt{k} \text{ or } a/k \quad a > 0$$

- If the convex function $J(.)$ is differentiable at $\alpha_{(k)}$, the only subgradient is the gradient

  → reduces to gradient descent

- Subgradient method is not a descent method,

  → common to keep track of the best point found so far at each iteration

  → at each step, one sets

$$J^*_{(k)} = \min\{J^*_{(k-1)}, J_{(k)}\}$$

Also use the corresponding α

- Convergence guarantees

  → For diminishing step size rule, guaranteed to weakly converge to the optimum

Projected Subgradient

To solve constrained optimization problem

$$\boxed{\begin{array}{l}\text{minimize } J(\alpha) \\ \text{subject to } \alpha \in C\end{array}}$$

C is a convex set

$$\alpha_{(k+1)} = P(\alpha_{(k)} - \eta_{(k)} g_{(k)})$$

Euclidean projection on C

Projected Subgradient

To solve constrained optimization problem

minimize $J(\alpha)$
subject to $\alpha \in C$

C is a convex set

$$\alpha_{(k+1)} = P(\alpha_{(k)} - \eta_{(k)} g_{(k)})$$

Euclidean projection on C

- Minimization with general constraints

minimize $J(\alpha)$
subject to $f_i(\alpha) \leq 0$ $\quad f_i(.)$ are convex $\forall i = 1,\dots,m$

$$\alpha_{(k+1)} = \alpha_{(k)} - \eta_{(k)} g_{(k)}$$

$$g_{(k)} \in \begin{cases} \partial J(\alpha) & \text{if current point is feasible} \\ \partial f_j(\alpha) & \text{if jth constraint is violated} \end{cases}$$

Cutting Plane Methods for Max-Margin

Binary linear SVM   training set $\{x_i, y_i\}_{i=1}^n \quad x \in \Re^d, y \in \{1, -1\}$

$$\hat{y} = \mathrm{sgn}[w^T x]$$

augment vectors to incorporate bias

scaled C

Primal Formulation $\quad \min \; w^T w + \frac{C}{n} \sum_i \xi_i$

$$\text{s.t.} \quad y_i (w^T x_i) \geq 1 - \xi_i \quad \forall\, i = 1, \ldots, n$$

$$\xi_i \geq 0$$

Cutting Plane Methods for Max-Margin

Binary linear SVM   training set $\{x_i, y_i\}_{i=1}^n \quad x \in \Re^d, y \in \{1, -1\}$

$$\hat{y} = \text{sgn}[w^T x]$$

scaled C

Primal Formulation $$\min \; w^T w + \frac{C}{n}\sum_i \xi_i$$

$$\text{s.t.} \quad y_i(w^T x_i) \geq 1 - \xi_i \quad \forall\, i = 1, \ldots, n$$

$$\xi_i \geq 0$$

Alternative Formulation: Based on Structured-SVMs (i.e., data may not be i.i.d.)

Given a feature map $\psi(x, y)$ e.g., for linear SVMs $\psi(x_i, y_i) = (1/2) y_i x_i$

and a loss function $\Delta(y, \tilde{y})$ e.g., 0/1 loss in SVMs $\Delta(y_i, \tilde{y}_i) = 1, \text{ if } y_i \neq \tilde{y}_i$
$= 0, \text{ otherwise}$

Cutting Plane Methods for Max-Margin

Binary linear SVM training set $\{x_i, y_i\}_{i=1}^n \quad x \in \Re^d, y \in \{1, -1\}$

$$\hat{y} = \text{sgn}[w^T x]$$

scaled C

Primal Formulation
$$\min \; w^T w + \frac{C}{n} \sum_i \xi_i$$
$$\text{s.t.} \quad y_i(w^T x_i) \geq 1 - \xi_i \quad \forall\, i = 1, \ldots, n$$
$$\xi_i \geq 0$$

sum n constraints
$(1/n)\sum_{i=1}^n \xi_i = \xi$

Alternative Formulation: Based on Structured-SVMs (i.e., data may not be i.i.d.)

Given a feature map $\psi(x, y)$ e.g., for linear SVMs $\psi(x_i, y_i) = (1/2) y_i x_i$

and a loss function $\Delta(y, \tilde{y})$ e.g., 0/1 loss in SVMs $\Delta(y_i, \tilde{y}_i) = 1, \text{ if } y_i \neq \tilde{y}_i$
$= 0, \text{ otherwise}$

$$\min \; w^T w + C\xi$$
$$\text{s.t.} \quad \left\langle w, (1/n)\sum_{i=1}^n (\psi(x_i, y_i) - \psi(x_i, \tilde{y}_i)) \right\rangle \geq (1/n)\sum_{i=1}^n \Delta(y_i - \tilde{y}_i) - \xi \quad \forall\, \tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_n) \in \{-1, 1\}^n$$
$$\xi \geq 0$$

2^n (decomposable) constraints
→ n constraints

Single slack variable instead of n !

Cutting Plane Algorithm

Key Idea – Keep only a very small number of (active) constraints at each iteration and solve a small QP problem

$$\min \; w^T w + C\xi$$

$$\text{s.t.} \quad \left\langle w, \underbrace{(1/n)\sum_{i=1}^{n}(\psi(x_i, y_i) - \psi(x_i, \tilde{y}_i))}_{\overline{\Psi}} \right\rangle \geq \underbrace{(1/n)\sum_{i=1}^{n}\Delta(y_i - \tilde{y}_i)}_{\overline{\Delta}} - \xi \qquad \forall \tilde{y}, \; \xi \geq 0$$

function of $\tilde{y}$

Cutting Plane Algorithm

Key Idea – Keep only a very small number of (active) constraints at each iteration and solve a small QP problem

$$\min \quad w^T w + C\xi$$

$$\text{s.t.} \quad \Big\langle w, \underbrace{(1/n)\textstyle\sum_{i=1}^{n}(\psi(x_i, y_i) - \psi(x_i, \tilde{y}_i))}_{\overline{\Psi}} \Big\rangle \geq \underbrace{(1/n)\textstyle\sum_{i=1}^{n}\Delta(y_i - \tilde{y}_i)}_{\overline{\Delta}} - \xi \qquad \forall \tilde{y},\ \xi \geq 0$$

function of $\tilde{y}$

Algorithm

1. Given a constraint set W (containing at most m vectors $\overline{\Psi}_k$)

solve
$$\arg\min_{w,\xi} \quad w^T w + C\xi \qquad O(m^3 + md)$$

$$\text{s.t.} \quad w^T \overline{\Psi}_k \geq \overline{\Delta}_k - \xi \quad \forall\, k = 1,\ldots,m$$

Cutting Plane Algorithm

Key Idea – Keep only a very small number of (active) constraints at each iteration and solve a small QP problem

$$\min \; w^T w + C\xi$$

$$\text{s.t.} \quad \Big\langle w, \underbrace{(1/n)\textstyle\sum_{i=1}^{n}(\psi(x_i, y_i) - \psi(x_i, \tilde{y}_i))}_{\overline{\Psi}} \Big\rangle \geq \underbrace{(1/n)\textstyle\sum_{i=1}^{n}\Delta(y_i - \tilde{y}_i)}_{\overline{\Delta}} - \xi \qquad \forall \tilde{y}, \; \xi \geq 0$$

function of $\tilde{y}$

Algorithm

1. Given a constraint set W (containing at most m vectors $\overline{\Psi}_k$)

$$\text{solve} \quad \arg\min_{w,\xi} \; w^T w + C\xi \qquad O(m^3 + md)$$

$$\text{s.t.} \quad w^T \overline{\Psi}_k \geq \overline{\Delta}_k - \xi \quad \forall\, k = 1, \ldots, m$$

2. Find the most violated constraint and add to the constraint set, remove inactive ones

$$\text{for } i = 1, \ldots, n \quad \tilde{y}_i \leftarrow \arg\max_r \{\Delta(y_i, r) + w^T \psi(x_i, r)\} \qquad r \in \{1, -1\} \qquad O(nd)$$

Cutting Plane Algorithm

Key Idea – Keep only a very small number of (active) constraints at each iteration and solve a small QP problem

$$\min \; w^T w + C\xi$$

$$\text{s.t.} \quad \Big\langle w, \underbrace{(1/n)\textstyle\sum_{i=1}^{n}(\psi(x_i, y_i) - \psi(x_i, \tilde{y}_i))}_{\overline{\Psi}} \Big\rangle \geq \underbrace{(1/n)\textstyle\sum_{i=1}^{n}\Delta(y_i - \tilde{y}_i)}_{\overline{\Delta}} - \xi \qquad \forall \tilde{y}, \; \xi \geq 0$$

function of $\tilde{y}$

Algorithm

1. Given a constraint set W (containing at most m vectors $\overline{\Psi}_k$)

   solve $\quad \arg\min_{w,\xi} \; w^T w + C\xi \qquad O(m^3 + md)$

   $$\text{s.t.} \quad w^T \overline{\Psi}_k \geq \overline{\Delta}_k - \xi \quad \forall\, k = 1, \ldots, m$$

2. Find the most violated constraint and add to the constraint set, remove inactive ones

   $$\text{for } i = 1, \ldots, n \quad \tilde{y}_i \leftarrow \arg\max_r \{\Delta(y_i, r) + w^T \psi(x_i, r)\} \qquad r \in \{1, -1\} \qquad O(nd)$$

3. Compute new $\overline{\Psi}$ and $\overline{\Delta}$, and iterate until $w^T \overline{\Psi} \geq \overline{\Delta} - \xi - \varepsilon$

Guaranteed to converge, more intuitive stopping criterion, kernel extensions easy

Experiment

Linear SVM

	n	d	s	Classification SVM-Perf	Classification SVM-Light
Reuters CCAT	804,414	47,236	0.16%	149.7	20,075.5
Reuters C11	804,414	47,236	0.16%	178.9	5,187.4
Arxiv astro-ph	62,369	99,757	0.08%	16.9	80.1
Covertype 1	522,911	54	22.22%	171.7	25,514.3
KDD04 Physics	150,000	78	38.42%	31.9	1,040.2

cutting-plane method

uses kernel-approach with decomposition

Gains mainly due to solving linear SVM updating w explicitly rather than using (linear) kernel !

Joachims [13]

Online Learning

Primarily based on some form of Stochastic Gradient Descent

– Have been applied mostly to linear problems, kernel extensions easy

Example: Recall SVM formulation as regularized loss function

$$\min_{w} \; (1/n)\sum_{i=1}^{n} \underbrace{l(x_i, y_i; w)}_{\max\{0,\, 1-y_i w^T x_i\}} + \lambda \underbrace{R(w)}_{(1/2)w^T w} \qquad x \in \Re^d, y \in \{1, -1\}$$

Online Learning

Primarily based on some form of Stochastic Gradient Descent

– Have been applied mostly to linear problems, kernel extensions easy

Example: Recall SVM formulation as regularized loss function

$$\min_{w} \; (1/n)\sum_{i=1}^{n} \underbrace{l(x_i, y_i; w)}_{\max\{0,1-y_i w^T x_i\}} + \lambda \underbrace{R(w)}_{(1/2)w^T w} \qquad x \in \Re^d, y \in \{1,-1\}$$

- Use stochastic subgradient since hinge-loss is non-differentiable

- Usually an ε-accurate solution $\hat{w}$ is obtained

$$f(\hat{w}) \le \min_w f(w) + \varepsilon$$

- Saves significant training time in practice since solving training loss beyond a precision usually does not affect the generalization performance

- More important to spend time in finding good setting of λ

Projected Subgradient Approach for SVM

Key Idea: After each (sub)gradient step, project w in L_2-ball of radius $1/\sqrt{\lambda}$

- Allows aggressive decrease in learning rate and hence faster convergence → $O(1/\varepsilon)$

Why projection?

Projected Subgradient Approach for SVM

Key Idea: After each (sub)gradient step, project w in L_2-ball of radius $1/\sqrt{\lambda}$

- Allows aggressive decrease in learning rate and hence faster convergence → $O(1/\varepsilon)$

Why projection?

The optimal solution lives within a ball of radius $1/\sqrt{\lambda}$

Proof Sketch

Comparing dual and primal at optimum

$$\|\hat{\alpha}\|_1 - (1/2)\|\hat{w}\|^2 = (1/2)\|\hat{w}\|^2 + C\sum_i \hat{\xi}_i$$

Projected Subgradient Approach for SVM

Key Idea: After each (sub)gradient step, project w in L_2-ball of radius $1/\sqrt{\lambda}$

- Allows aggressive decrease in learning rate and hence faster convergence → $O(1/\varepsilon)$

Why projection?

The optimal solution lives within a ball of radius $1/\sqrt{\lambda}$

Proof Sketch

Comparing dual and primal at optimum

$$\|\hat{\alpha}\|_1 - \underbrace{(1/2)\|\hat{w}\|^2}_{\text{since } \hat{w}=\sum_i \hat{\alpha}_i y_i x_i} = (1/2)\|\hat{w}\|^2 + C\sum_i \hat{\xi}_i$$

$$0 \le \hat{\xi}_i$$
$$0 \le \hat{\alpha}_i \le C = 1/\lambda n$$
$$\Rightarrow \|\hat{\alpha}_i\|_1 \le 1/\lambda$$

Projected Subgradient Approach for SVM

Key Idea: After each (sub)gradient step, project w in L_2-ball of radius $1/\sqrt{\lambda}$

- Allows aggressive decrease in learning rate and hence faster convergence → $O(1/\varepsilon)$

Why projection?

The optimal solution lives within a ball of radius $1/\sqrt{\lambda}$

Proof Sketch

Comparing dual and primal at optimum

$$\|\hat{\alpha}\|_1 - \underbrace{(1/2)\|\hat{w}\|^2}_{\text{since } \hat{w} = \sum_i \hat{\alpha}_i y_i x_i} = (1/2)\|\hat{w}\|^2 + C\sum_i \hat{\xi}_i$$

$$0 \le \hat{\xi}_i$$

$$0 \le \hat{\alpha}_i \le C = 1/\lambda n$$

$$\Rightarrow \|\hat{\alpha}_i\|_1 \le 1/\lambda$$

$$(1/2)\|\hat{w}\|^2 \le (1/2)\|\hat{w}\|^2 + C\sum_i \hat{\xi}_i = \|\hat{\alpha}\|_1 - (1/2)\|\hat{w}\|^2$$

$$\boxed{\|\hat{w}\|^2 \le \|\hat{\alpha}\|_1 \le 1/\lambda}$$

Projected Subgradient Approach for SVM

Algorithm

1. Initialize the initial vector $\|w_1\| \leq 1/\sqrt{\lambda}$
2. Compute a subgradient at the current estimate w_t using k data points for which loss is nonzero, i.e. margin $y_i w_t^T x_i \leq 1$

$$J_k(w) = (\lambda/2)\|w\|^2 + (1/k')\sum_{i=1}^{k'} \max(0, 1 - y_i w^T x_i)$$

$$\boxed{w'_{t+1} = w_t - \eta_t g_t}$$

Projected Subgradient Approach for SVM

Algorithm

1. Initialize the initial vector $\|w_1\| \leq 1/\sqrt{\lambda}$
2. Compute a subgradient at the current estimate w_t using k data points for which loss is nonzero, i.e. margin $y_i w_t^T x_i \leq 1$

$$J_k(w) = (\lambda/2)\|w\|^2 + (1/k')\sum_{i=1}^{k'} \max(0, 1 - y_i w^T x_i)$$

$$\boxed{w'_{t+1} = w_t - \eta_t g_t}$$

$$g_t = \lambda w_t - (1/k) \sum_{y_i w^T x_i < 1} y_i x_i$$

Apply subgradient computation on max of two convex functions

$$\eta_t = 1/(\lambda t)$$

Projected Subgradient Approach for SVM

Algorithm

1. Initialize the initial vector $\|w_1\| \leq 1/\sqrt{\lambda}$
2. Compute a subgradient at the current estimate w_t using k data points for which loss is nonzero, i.e. margin $y_i w_t^T x_i \leq 1$

$$J_k(w) = (\lambda/2)\|w\|^2 + (1/k')\sum_{i=1}^{k'} \max(0, 1 - y_i w^T x_i)$$

$$\boxed{w'_{t+1} = w_t - \eta_t g_t}$$

$$g_t = \lambda w_t - (1/k) \sum_{y_i w^T x_i < 1} y_i x_i$$

Apply subgradient computation on max of two convex functions

$$\eta_t = 1/(\lambda t)$$

3. Project the new estimate in the L_2 ball of radius $1/\sqrt{\lambda}$

$$w_{t+1} = w'_{t+1} \left(\min(1, \frac{1/\sqrt{\lambda}}{\|w'_{t+1}\|}) \right)$$

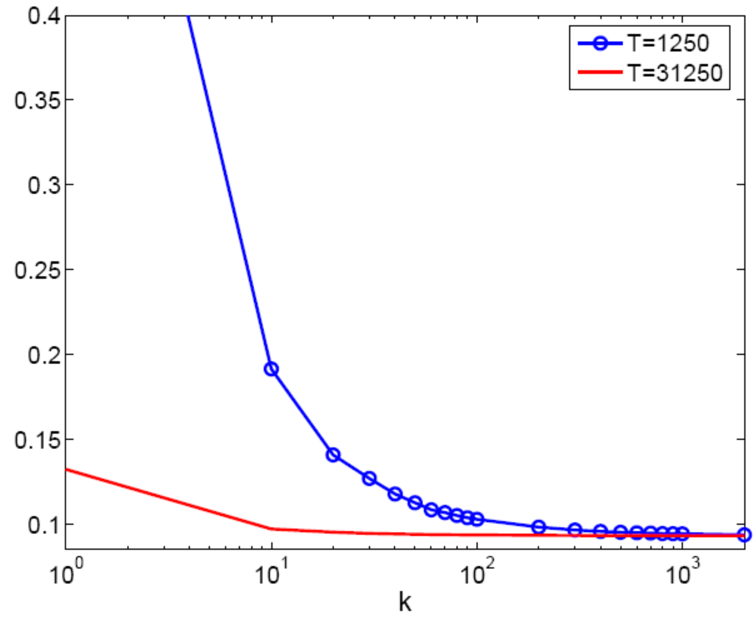
Experiment

Linear SVM

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

	n	d	s
Reuters CCAT	804,414	47,236	0.16%
Reuters C11	804,414	47,236	0.16%
Arxiv astro-ph	62,369	99,757	0.08%
Covertype 1	522,911	54	22.22%
KDD04 Physics	150,000	78	38.42%

T: # of iterations



Effect of batchsize k on objective value

S. Shalev-Shwartz [14]

Multi-class Extensions

For L-class classification problem, $\{x_i, y_i\}_{i=1}^{n}$ $x \in \Re^d$ $y = \{1, 2,..., L\}$

Prediction function $\hat{y} = \arg\max_j [w_{(j)}^T x]$ $w_{(j)} \in \Re^d, j = 1,..., L$

Multi-class Extensions

For L-class classification problem, $\{x_i, y_i\}_{i=1}^n$ $x \in \Re^d$ $y = \{1, 2, \ldots, L\}$

Prediction function $\hat{y} = \arg\max_j [w_{(j)}^T x]$ $\quad w_{(j)} \in \Re^d, j = 1, \ldots, L$

$$\min_w \; (1/n)\sum_{i=1}^n \underbrace{l(x_i, y_i; w)}_{\max\{0,\, 1 - w_{(y_i)}{}^T x_i + w_{(r_i)}{}^T x_i\}} + \lambda \underbrace{R(w)}_{(1/2) w^T w}$$

$$r_i = \arg\max_{j \neq y_i} w_{(j)}^T x_i$$

Multi-class Extensions

For L-class classification problem, $\{x_i, y_i\}_{i=1}^n \quad x \in \Re^d \quad y = \{1, 2, \ldots, L\}$

Prediction function $\hat{y} = \arg\max_j [w_{(j)}^T x] \qquad w_{(j)} \in \Re^d, j = 1, \ldots, L$

$$\min_w \; (1/n) \sum_{i=1}^n \underbrace{l(x_i, y_i; w)}_{\max\{0,\, 1 - w_{(y_i)}^T x_i + w_{(r_i)}^T x_i\}} + \lambda \underbrace{R(w)}_{(1/2) w^T w}$$

$$r_i = \arg\max_{j \neq y_i} w_{(j)}^T x_i$$

Parameter space: One parameter vector per class → Ld parameters

Algorithm: same update for each vector as for the binary case except,

$$g_{(j)}^t = \begin{cases} \lambda w_{(j)}^t - x_t, & \text{if } j = y_t \\ \lambda w_{(j)}^t + x_t, & \text{if } j = r_t \\ \lambda w_{(j)}^t, & \text{otherwise} \end{cases}$$

Kernel Perceptrons

Recall, perceptron algorithm for linear binary classifier $y = \{-1, 1\}$

$$f(x) = \text{sgn}(w^T x)$$

Update Rule $w_{t+1} = \begin{cases} w_t + y_t\, x_t & \text{if } x_t \text{ is misclassified, i.e., } y_t \neq f(x_t) \\ w_t & \text{otherwise} \end{cases}$

stocahstic (sub)gradient descent !

Kernel Perceptrons

Recall, perceptron algorithm for linear binary classifier $y = \{-1, 1\}$

$$f(x) = \mathrm{sgn}(w^T x)$$

Update Rule $w_{t+1} = \begin{cases} w_t + y_t\, x_t & \text{if } x_t \text{ is misclassified, i.e., } y_t \neq f(x_t) \\ w_t & \text{otherwise} \end{cases}$

If initial parameter setting is $w_0 = 0$

$$w_t = \sum_{m=1}^{k} y_m x_m$$

where $\{x_m\}_{m=1,\ldots,k}$ are k misclassified points and y_m are the corresponding labels

Kernel Perceptrons

Recall, perceptron algorithm for linear binary classifier $y = \{-1, 1\}$

$$f(x) = \mathrm{sgn}(w^T x)$$

Update Rule $w_{t+1} = \begin{cases} w_t + y_t\, x_t & \text{if } x_t \text{ is misclassified, i.e., } y_t \neq f(x_t) \\ w_t & \text{otherwise} \end{cases}$

If initial parameter setting is $w_0 = 0$

$$w_t = \sum_{m=1}^{k} y_m x_m$$

where $\{x_m\}_{m=1,\ldots,k}$ are k misclassified points and y_m are the corresponding labels

Prediction based on $\mathrm{sgn}[w_t^T x] = \mathrm{sgn}[\sum_{m=1}^{k} y_m x_m^T x]$

Kernel Perceptrons

Recall, perceptron algorithm for linear binary classifier $y = \{-1, 1\}$

$$f(x) = \mathrm{sgn}(w^T x)$$

Update Rule $w_{t+1} = \begin{cases} w_t + y_t\, x_t & \text{if } x_t \text{ is misclassified, i.e., } y_t \neq f(x_t) \\ w_t & \text{otherwise} \end{cases}$

If initial parameter setting is $w_0 = 0$

$w_t = \sum_{m=1}^{k} y_m x_m$ where $\{x_m\}_{m=1,\dots,k}$ are k misclassified points and y_m are the corresponding labels

Prediction based on $\mathrm{sgn}[w_t^T x] = \mathrm{sgn}[\sum_{m=1}^{k} y_m x_m^T x]$

Kernel Perceptron $f(x) = \mathrm{sgn}[\sum_{m=1}^{k} y_m k(\underbrace{x_m}_{\text{active vectors or "Support Vectors"}}, x)]$

Issue: The number of "support vectors" tend to increase linearly with iterations !
→ Storage and run-time increase linearly !

Kernel Perceptron Experiment

Linear vs Kernel Perceptron

MNIST: 60K training, 10K testing,

Linear

Kernel

$k(a,b) = (1 + a^T b)^d$

		$T =$	0.1	1	2
$d = 1$	Vote		10.7	8.5	8.3
	Avg.	(unnorm)	10.9	8.7	8.5
		(norm)	10.9	8.5	8.3
	Last	(unnorm)	16.0	14.7	13.6
		(norm)	15.4	14.1	13.1
$d = 2$	Vote		6.0	2.8	2.4
	Avg.	(unnorm)	6.0	2.8	2.4
		(norm)	6.2	3.0	2.5
	Last	(unnorm)	8.6	4.0	3.4
		(norm)	8.4	3.9	3.3
	Rand.	(unnorm)	13.4	5.9	4.7
		(norm)	13.2	5.9	4.7
	SupVec		1,639	8,190	9,888
	Mistake		2,150	10,201	15,290

Freund and Schapire [6]

Kernel Perceptron with Budget

Key Idea: Keep only a fixed number of support vectors

Simple Strategies

- "Forget" the oldest support vectors if beyond budget
- Remove the ones with largest margin first
- May cause big change in prediction as more support vectors are removed

Kernel Perceptron with Budget

Key Idea: Keep only a fixed number of support vectors

Simple Strategies

- "Forget" the oldest support vectors if beyond budget
- Remove the ones with largest margin first
- May cause big change in prediction as more support vectors are removed

Alternative Strategy

- Use weighted combination of kernels

$$f(x) = \mathrm{sgn}[\textstyle\sum_{m=1}^{k} \sigma_m y_m k(x_m, x)] \qquad \sigma_m \in [0,1]$$

- Weights are decayed exponentially as a support vector becomes old

$$\sigma_{m,t} = \varphi \sigma_{m,t-1} \qquad \sigma_{m,1} = 1$$

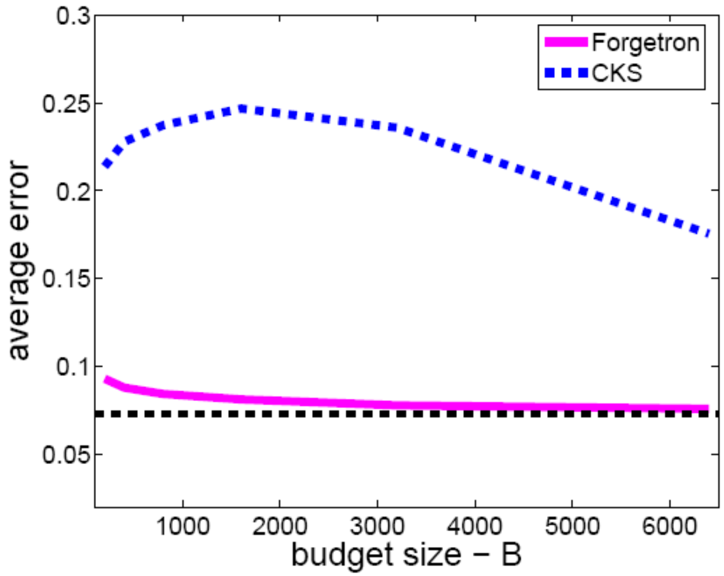
- If number of support vectors becomes more than budget, remove the oldest
- Selection of decay coefficient based on bound on number of mistakes

Budgeted Kernel Perceptron Experiment

$$k(a,b) = (1 + a^T b)^5$$

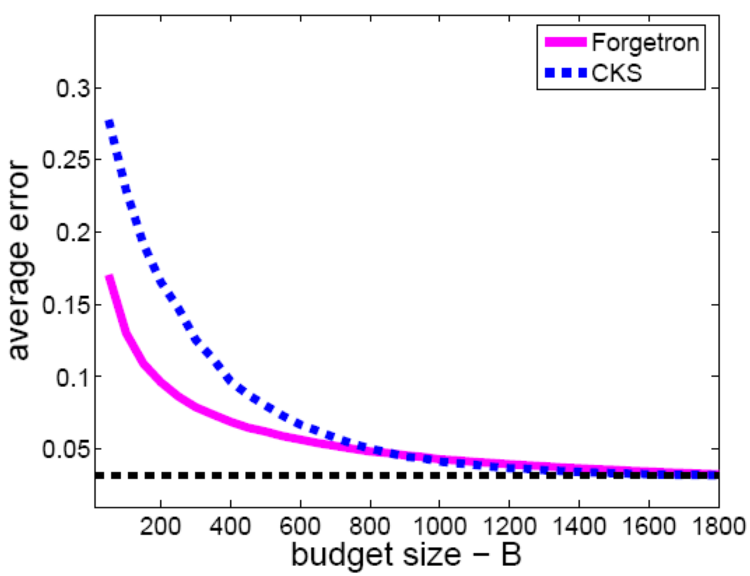
census-income: 200K training

Support Vec: 14,626



MNIST: 60K training

Support Vec: 1,886



CKS: Removes the point with largest margin

Dekel et al. [12]

Online Passive-Aggressive Algorithms

Key Idea: At each iteration try to achieve zero loss for a given data point

Example: Binary classification with ε-margin loss

$$l(x, y; w) = \max\{0, \varepsilon - yw^T x\} \qquad y \in \{-1, 1\}$$

Online Update (Separable Case): initialize $w_1 = 0$

$$w_{t+1} = \arg\min_w (1/2)\|w - w_t\|^2 \quad \text{s.t. } l(x_t, y_t; w) = 0$$

Online Passive-Aggressive Algorithms

Key Idea: At each iteration try to achieve zero loss for a given data point

Example: Binary classification with ε-margin loss

$$l(x, y; w) = \max\{0, \varepsilon - yw^T x\} \quad y \in \{-1, 1\}$$

Online Update (Separable Case): initialize $w_1 = 0$

$$w_{t+1} = \arg\min_{w} (1/2)\|w - w_t\|^2 \text{ s.t. } l(x_t, y_t; w) = 0$$

$$w_{t+1} = \begin{cases} w_t \text{ if } l(x_t, y_t; w_t) = 0 & \text{Passive update} \\ w_t + \tau_t y_t x_t, \text{ otherwise} & \text{Aggressive update} \end{cases}$$

$\tau_t = l(x_t, y_t; w_t) / \|x_t\|^2$ by Lagrangian optimization

Online Passive-Aggressive Algorithms

Key Idea: At each iteration try to achieve zero loss for a given data point

Example: Binary classification with ε-margin loss

$$l(x, y; w) = \max\{0, \varepsilon - y w^T x\} \quad y \in \{-1, 1\}$$

Online Update (Separable Case): initialize $w_1 = 0$

$$w_{t+1} = \arg\min_w (1/2)\|w - w_t\|^2 \quad \text{s.t.} \quad l(x_t, y_t; w) = 0$$

$$w_{t+1} = \begin{cases} w_t \text{ if } l(x_t, y_t; w_t) = 0 & \text{Passive update} \\ w_t + \tau_t y_t x_t, \text{ otherwise} & \text{Aggressive update} \end{cases}$$

$\tau_t = l(x_t, y_t; w_t)/\|x_t\|^2$ by Lagrangian optimization

Guaranteed to find a separating hyperplane whose margin is at least half of the best margin achievable by a batch algorithm !

Online Update (Inseparable Case): Same updates as above except

$$\tau_t = \min\{\gamma, \ l(x_t, y_t; w_t)\}/\|x_t\|^2 \qquad \gamma > 0$$

References

1. Notes on Subgradients, S. Boyd and L. Vanderberghe. 2003/2008.
   http://see.stanford.edu/materials/lsocoee364b/01-subgradients_notes.pdf
   http://www.stanford.edu/class/ee392o/subgrad_method.pdf
2. Rosenblatt F., “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,” Psychological Review, 1958.
3. A. B. J. Novikoff, “On convergence proofs on perceptrons,” Symposium on Mathematical Theory of Automata,” 615-622, 1962.
4. M. A. Aizerman, E. M., Braverman, L. I. Rozonoer L, “Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning,” Automation and Remote Control, 25, 821-837, 1964.
5. J. K. Anlauf, M. Biehl, “ The Adatron: An Adaptive Perceptron Algorithm,” Euro-Physics Letters, 1989.
6. Y. Freund, R. E. Schapire, “Large Margin Classification using the Perceptron Algorithm,” Machine Learning, 37, 277-296, 1999.
7. C. Gentile, “A new approximate maximal margin classification algorithm,” JMLR, 2001.
8. Y. Li and P. M. Long, “The relaxed online maximum margin algorithm,” Machine Learning, 2002.
9. J. Kivinen, A. Smola, R. C. Williamson, “Online Learning with Kernels,” IEEE Trans Signal Proc., 2002.
10. K. Crammer, O. Dekel, S. Shalev-Shwartz, Y. Singer, “Online Passive Aggressive Algorithms,” NIPS, 2003.
11. T. Zhang, “Solving large-scale linear prediction problems using stochastic gradient descent algorithms,” ICML 2004.
12. O. Dekel, S. Shalev-Shwartz, Y. Singer, “The Forgetron: A Kernel-Based Perceptron on a Fixed Budget,” NIPS, 2005.
13. T. Joachims, “Training Linear SVMs in Linear Time”, KDD, 2006.
14. S. Shalev-Schwartz, Y. Singer, N. Srebro, “Pegasos: Primal Estimated Sub-Gradient Solver for SVM,” ICML, 2007.
15. J. Yu, S. V. N. Vishwanathan, S. Gunter, N. N. Schraudolph, “A Quasi-Newton Approach to Nonsmooth Convex Optimization,” ICML 2008.
16. 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,” ICML, 2008.
17. T. Joachims, Chun-Nam John Yu, *Sparse Kernel SVMs via Cutting-Plane Training*, Machine Learning Journal, Special issue of ECML, 76(2-3):179-193, 2009.
18. Z. Wang, K. Crammer, S. Vucetic, “Multi-class Pegasos on a Budget,” ICML, 2010.