

Notes on Online Support Vector Machines

Kee Siong Ng

September 23, 2015

1 Preliminaries

The general Stochastic Gradient Descent template is

$$w := w - \eta_t \lambda \Omega'[w] - \eta_t \partial_w f_t(w),$$

where $f_t(w)$ is the loss function and $\Omega[x]$ is the regulariser.

2 NORMA and Related Algorithms

We write out the derivations of the online SVM algorithms given in [KSW01] and implemented in MADlib.

2.1 Preliminaries

Definition 1. Let $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) in that there exists a kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ and a dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ such that

1. k has the reproducing property

$$\langle f, k(x, \cdot) \rangle_{\mathcal{H}} = f(x) \quad \forall f \in \mathcal{H}, \forall x \in \mathcal{X} \quad (1)$$

2. $\mathcal{H}$ is the closure of the span of all $k(x, \cdot)$ with $x \in \mathcal{X}$.

In other words, each $f \in \mathcal{H}$ takes the form of a linear combination of kernel functions:

$$f(x) = \sum_i \alpha_i k(x_i, x).$$

The inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ induces a norm on $f \in \mathcal{H}$ in the usual way:

$$\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}.$$

We state the following without proof.

Proposition 1. *Let $J : \mathcal{H} \rightarrow \mathbb{R}$ be a functional. Then we have*

$$\langle \frac{\partial}{\partial f} J(f), g \rangle_{\mathcal{H}} = \lim_{\epsilon \rightarrow 0} \frac{J(f + \epsilon g) - J(f)}{\epsilon}. \quad (2)$$

The following result will be needed later.

Proposition 2. *For each $f \in \mathcal{H}$, we have*

$$\frac{\partial}{\partial f} f(x) = k(x, \cdot) \quad (3)$$

$$\frac{\partial}{\partial f} \|f\|_{\mathcal{H}}^2 = 2f. \quad (4)$$

Proof. For the first part, we have

$$\begin{aligned} \langle \frac{\partial}{\partial f} f(x), g \rangle_{\mathcal{H}} &= \lim_{\epsilon \rightarrow 0} \frac{f(x) + \epsilon g(x) - f(x)}{\epsilon} && \text{by (2)} \\ &= g(x) \\ &= \langle k(x, \cdot), g \rangle_{\mathcal{H}} && \text{by (1).} \end{aligned}$$

For the second part, we have

$$\begin{aligned} \langle \frac{\partial}{\partial f} \|f\|_{\mathcal{H}}^2, g \rangle_{\mathcal{H}} &= \lim_{\epsilon \rightarrow 0} \frac{\|f + \epsilon g\|_{\mathcal{H}}^2 - \|f\|_{\mathcal{H}}^2}{\epsilon} && \text{by (2)} \\ &= \lim_{\epsilon \rightarrow 0} \frac{\|f\|_{\mathcal{H}}^2 + \epsilon^2 \|g\|_{\mathcal{H}}^2 + 2\epsilon \langle f, g \rangle_{\mathcal{H}} - \|f\|_{\mathcal{H}}^2}{\epsilon} \\ &= \lim_{\epsilon \rightarrow 0} \epsilon \|g\|_{\mathcal{H}}^2 + 2\langle f, g \rangle_{\mathcal{H}} \\ &= \langle 2f, g \rangle_{\mathcal{H}} \end{aligned}$$

□

2.2 Stochastic Gradient Descent in RKHS

In the supervised learning setting, we assume there exist an (unknown) distribution P over $\mathcal{X} \times \mathcal{Y}$ and we seek an $f \in \mathcal{H}$ that minimises the expected risk

$$R[f, P] := \mathbb{E}_{(x,y) \sim P} [l(f(x), y)],$$

where $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ is a suitable loss function. In practice, since P is unknown, a standard approach is to instead minimise a regularised risk on a sample S drawn from P^m :

$$R_{\text{reg}}[f, S] := \frac{1}{m} \sum_{t=1}^m l(f(x_t), y_t) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2.$$

In the online setting, which deal with training examples one at a time, we can use stochastic gradient descent to find an $f \in \mathcal{H}$ that approximately minimises the regularised risk. We start with the zero initial hypothesis $f_1 = 0$ and use the following general rule to update the model at every step:

$$\begin{aligned}
f_{t+1} &:= f_t - \eta \frac{\partial}{\partial f} R_{reg}[f, \{(x_t, y_t)\}]|_{f=f_t} \\
&= f_t - \eta \frac{\partial}{\partial f} [l(f(x_t), y_t) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2]|_{f=f_t} \\
&= f_t - \eta [l'(f(x_t), y_t) \frac{\partial}{\partial f} [f(x)]|_{f=f_t} + \frac{\partial}{\partial f} [\frac{\lambda}{2} \|f\|_{\mathcal{H}}^2]|_{f=f_t}] \quad \text{chain rule} \\
&= f_t - \eta [l'(f(x_t), y_t) k(x_t, \cdot) + \lambda f_t] \quad \text{by (3),(4)} \\
&= (1 - \eta\lambda) f_t - \eta l'(f(x_t), y_t) k(x_t, \cdot),
\end{aligned}$$

where η is the learning rate and $l'(w, y) = \frac{\partial}{\partial w} l(w, y)$.

2.3 Applications

2.3.1 Classification

In (binary) classification, we have $\mathcal{Y} = \{-1, +1\}$. A suitable loss function from the classification with the ν -trick setting [SSWB00] is

$$l_{\rho, b}(f(x), y) := \max(0, \rho - y(f(x) + b)) - \nu\rho,$$

where $\rho \geq 0$ is the margin parameter and b is a constant offset to $f(x)$. In this case, we have

$$\begin{aligned}
l'_{\rho, b_t}(w, y_t) &= \frac{\partial}{\partial w} [if \ y_t(w + b_t) \leq \rho_t \text{ then } \rho_t - y_t(w + b_t) - \nu\rho_t \text{ else } -\nu\rho_t] \\
&= if \ y_t(w + b_t) \leq \rho_t \text{ then } -y_t \text{ else } 0.
\end{aligned}$$

In addition to updating f_t , we also need to update the ρ and b parameters at each time step using

$$\begin{aligned}
b_{t+1} &:= b_t - \eta \frac{\partial}{\partial b} R_{reg}[f_t, \{(x_t, y_t)\}]|_{b=b_t} \\
&= b_t - \eta \frac{\partial}{\partial b} [l_{\rho_t, b}(f_t(x_t), y_t)]|_{b=b_t} \\
&= b_t + \eta [if \ (y_t(f_t(x_t) + b_t) \leq \rho_t) \text{ then } y_t \text{ else } 0]
\end{aligned}$$

and

$$\begin{aligned}
\rho_{t+1} &:= \rho_t - \eta \frac{\partial}{\partial \rho} R_{reg}[f_t, \{x_t, y_t\}]|_{\rho=\rho_t} \\
&= \rho_t - \eta \frac{\partial}{\partial \rho} [l_{\rho, b_t}(f_t(x_t), y_t)]|_{\rho=\rho_t} \\
&= \rho_t - \eta [if \ y_t(f_t(x_t) + b_t) \leq \rho_t \ then \ (1 - \nu) \ else \ -\nu].
\end{aligned}$$

Letting α_i denote the weight associated with training example x_i , we obtain the following update rule (for $i = 1, \dots, t-1$)

$$(\alpha_i, \alpha_t, b, \rho) := ((1 - \lambda\eta)\alpha_i, \eta\sigma_t y_t, b + \eta\sigma_t y_t, \rho - \eta(\sigma_t - \nu)), \quad (5)$$

where $\sigma_t = 1$ if $y_t(f_t(x_t) + b_t) \leq \rho_t$ and zero otherwise. (Note that [KSW01] gives the incorrect ρ update formula: $\rho := \rho + \eta(\sigma_t - \nu)$.) It can be shown that the specific choice of λ has no influence in the ν -SV classification setting [SSWB00]. In practice, we would not want to set λ to a high value like 1, which will give us floating-point underflow problems as weights are discounted again and again.

2.3.2 Novelty Detection

An appropriate loss function for novelty detection in the ν -setting is

$$l_\rho(f(x), y) := \max(0, \rho - f(x)) - \nu\rho. \quad (6)$$

We thus have

$$\begin{aligned}
l'_{\rho_t}(w, y_t) &= \frac{\partial}{\partial w} [if \ w < \rho_t \ then \ \rho_t - w - \nu\rho_t \ else \ -\nu\rho_t] \\
&= if \ w < \rho_t \ then \ -1 \ else \ 0.
\end{aligned}$$

The margin parameter is updated as follows:

$$\begin{aligned}
\rho_{t+1} &:= \rho_t - \eta \frac{\partial}{\partial \rho} [l_\rho(f_t(x_t), y_t)]|_{\rho=\rho_t} \\
&= \rho_t - \eta [if \ f_t(x_t) < \rho_t \ then \ (1 - \nu) \ else \ -\nu].
\end{aligned}$$

The above yields the following update formula (for $i = 1, \dots, t-1$)

$$(\alpha_i, \alpha_t, \rho) := ((1 - \lambda\eta)\alpha_i, \eta\sigma_t, \rho - \eta(\sigma_t - \nu)), \quad (7)$$

where $\sigma_t = 1$ if $f_t(x_t) < \rho_t$ and zero otherwise. Note again that the ρ update formula in [KSW01] for this case is not quite right.

2.3.3 Regression

We will look at the ϵ -insensitive loss in the ν -setting, where the loss function is defined by

$$l_{\epsilon,b}(f(x), y) := \max(0, |y - (f(x) + b)| - \epsilon) + \nu\epsilon. \quad (8)$$

We thus have

$$\begin{aligned} l'_{\epsilon_t, b_t}(w, y_t) &= \frac{\partial}{\partial w} [\text{if } |y_t - (w + b_t)| > \epsilon_t \text{ then } |y_t - (w + b_t)| - \epsilon_t + \nu\epsilon_t \text{ else } \nu\epsilon_t] \\ &= \text{if } |y_t - (w + b_t)| > \epsilon_t \text{ then } -\text{sgn}(y_t - (w + b_t)) \text{ else } 0. \end{aligned}$$

The ϵ parameter is updated as follows:

$$\begin{aligned} \epsilon_{t+1} &:= \epsilon_t - \eta \frac{\partial}{\partial \epsilon} [l_{\epsilon, b_t}(f_t(x_t), y_t)]|_{\epsilon=\epsilon_t} \\ &= \epsilon_t - \eta [\text{if } |y_t - (f_t(x_t) + b_t)| > \epsilon_t \text{ then } -1 + \nu \text{ else } \nu]. \end{aligned}$$

The b parameter is updated as follows:

$$\begin{aligned} b_{t+1} &:= b_t - \eta \frac{\partial}{\partial b} [l_{\epsilon_t, b}(f_t(x_t), y_t)]|_{b=b_t} \\ &= b_t - \eta \frac{\partial}{\partial b} [\text{if } |y - (f_t(x_t) + b_t)| > \epsilon \text{ then } |y - (f_t(x_t) + b_t)| - \epsilon + \nu\epsilon \text{ else } \nu\epsilon] \\ &= b_t - \eta [\text{if } |y - (f_t(x_t) + b_t)| > \epsilon \text{ then } -\text{sgn}(y_t - (f_t(x_t) + b_t)) \text{ else } 0]. \end{aligned}$$

The above yields the following update formula (for $i = 1, \dots, t-1$)

$$(\alpha_i, \alpha_t, b, \epsilon) := ((1 - \lambda\eta)\alpha_i, \eta\sigma_t \text{sgn}(\delta_t), b + \eta\sigma_t \text{sgn}(\delta_t), \epsilon - \eta(\nu - \sigma_t)), \quad (9)$$

where $\delta_t := y_t - (f_t(x_t) + b_t)$, and $\sigma_t = 1$ if $|\delta_t| > \epsilon_t$ and 0 otherwise.

2.4 Learning Rates

The learning rate is (deliberately) kept fixed in [KSW01]. In experiments, I found that this setting is suitable in the case where we know the underlying concept being tracked changes constantly. In the case where we want to use this online algorithm to process large datasets, with the explicit assumption that the underlying concept is stationary, it is better to decay the learning rate at the rate of $1/t$.

3 Pegasos

In this section, we describe the Pegasos family of algorithms described in [SSSSC11]. These are practically useful algorithms, in particular the linear SVM algorithms.

3.1 Linear SVM

3.1.1 Regression with ϵ -insensitive loss

In the case of regression, the optimisation problem we want to solve is

$$\min_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{(x,y) \in S} l(w; (x, y)),$$

where the loss function is defined by

$$l(w; (x, y)) := \max\{0, |y - \langle w, x \rangle| - \epsilon\}. \quad (10)$$

Plugging the subgradient of (10) into the general stochastic gradient descent algorithm yields the following algorithm.

PegasosReg(S, T, λ, ϵ)

1. $w_1 := 0$
2. for $t = 1 \dots T$
 - (a) Sample (x_t, y_t) from S
 - (b) $\eta_t := 1/\lambda t$
 - (c) If $\langle w_t, x_t \rangle - y_t > \epsilon$ Then $w_{t+1} := (1 - \eta_t \lambda) w_t - \eta_t x_t$
 - (d) Elseif $y_t - \langle w_t, x_t \rangle > \epsilon$ Then $w_{t+1} := (1 - \eta_t \lambda) w_t + \eta_t x_t$
 - (e) Else $w_{t+1} := (1 - \eta_t \lambda) w_t$
3. Output w_{T+1}

3.1.2 Binary Classification

In the case of binary classification, the optimisation problem we want to solve is

$$\min_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{m} \sum_{(x,y) \in S} l(w; (x, y)),$$

where the loss function is defined by

$$l(w; (x, y)) := \max\{0, 1 - y\langle w, x \rangle\}. \quad (11)$$

Plugging the subgradient of (11) into the general stochastic gradient descent algorithm yields the following algorithm.

PegasosCls(S, T, λ)

1. $w_1 := 0$
2. for $t = 1 \dots T$
 - (a) Sample (x_t, y_t) from S
 - (b) $\eta_t := 1/\lambda t$
 - (c) If $y_t \langle w_t, x_t \rangle < 1$ Then $w_{t+1} := (1 - \eta_t \lambda) w_t + \eta_t y_t x_t$
 - (d) Else $w_{t+1} := (1 - \eta_t \lambda) w_t$
3. Output w_{T+1}

3.1.3 Learning Rate

We will now try to understand where the $\eta_t := 1/\lambda t$ comes from.

Definition 2. Let $X \subset \mathbb{R}^b$ and $f : X \rightarrow \mathbb{R}$. Then $g \in \mathbb{R}^n$ is a subgradient of f at $x \in X$ if for all $y \in X$ we have

$$f(y) \geq f(x) - \langle g, (x - y) \rangle.$$

Geometrically, $\lambda y \cdot (f(x) + \langle g, (x - y) \rangle)$ is the tangential line to f at x .

Essentially, a convex function has subgradients at every point in the interior of its domain X .

Definition 3. A function $f : X \rightarrow \mathbb{R}$ is λ -strongly convex if for all $x, y \in X$, we have

$$f(y) \geq f(x) - \langle g, (x - y) \rangle + \frac{\lambda}{2} \|x - y\|^2,$$

where g is a subgradient of f at x .

Theorem 1 ([Bub14, SSSSC11, HKKA06]). *Let $f_1, f_2, \dots, f_T$ be a sequence of λ -strongly convex functions. Let $w_1, w_2, \dots, w_{T+1}$ be a sequence of vectors defined by*

$$w_{t+1} := w_t - \eta_t g_t,$$

where w_1 is arbitrary, $\eta_t := 1/\lambda t$ and g_t is a subgradient of f_t at w_t . If $\forall t. \|g_t\| \leq G$, then we have for all u

$$\frac{1}{T} \sum_{t=1}^T f_t(w_t) \leq \frac{1}{T} \sum_{t=1}^T f_t(u) + \frac{G^2(1 + \ln(T))}{2\lambda T}.$$

Proof. First, observe that by the strong convexity of f_t and the elementary identity $2\langle a, b \rangle = \|a\|^2 + \|b\|^2 - \|a - b\|^2$, we have

$$f_t(w_t) - f_t(u) \tag{12}$$

$$\leq \langle g_t, w_t - u \rangle - \frac{\lambda}{2} \|w_t - u\|^2 \tag{13}$$

$$= \left\langle \frac{1}{\eta_t} (w_t - w_{t+1}), w_t - u \right\rangle - \frac{\lambda}{2} \|w_t - u\|^2 \tag{14}$$

$$= \frac{1}{2\eta_t} (\|w_t - w_{t+1}\|^2 + \|w_t - u\|^2 - \|w_{t+1} - u\|^2) - \frac{\lambda}{2} \|w_t - u\|^2 \tag{15}$$

$$= \frac{\eta_t \|g_t\|^2}{2} + \left[\frac{1}{2\eta_t} - \frac{\lambda}{2} \right] \|w_t - u\|^2 - \frac{1}{2\eta_t} \|w_{t+1} - u\|^2 \tag{16}$$

$$\leq \frac{G^2}{2\lambda t} + \frac{\lambda}{2} (t-1) \|w_t - u\|^2 - \frac{\lambda t}{2} \|w_{t+1} - u\|^2. \tag{17}$$

Summing both sides of the inequality over t and then dividing by T yields

$$\frac{1}{T} \sum_{t=1}^T f_t(w_t) \tag{18}$$

$$\leq \frac{1}{T} \sum_{t=1}^T \left[f_t(u) + \frac{G^2}{2\lambda t} + \frac{\lambda}{2} (t-1) \|w_t - u\|^2 - \frac{\lambda t}{2} \|w_{t+1} - u\|^2 \right] \tag{19}$$

$$= \frac{1}{T} \left[\sum_{t=1}^T f_t(u) + \frac{G^2}{2\lambda} \sum_{t=1}^T \frac{1}{t} - \frac{\lambda T}{2} \|w_{T+1} - u\|^2 \right] \tag{20}$$

$$\leq \frac{1}{T} \sum_{t=1}^T f_t(u) + \frac{G^2(1 + \ln(T))}{2\lambda T}. \tag{21}$$

□

For the $\eta_t := 1/\lambda t$ learning rate to make sense in the PegasosReg and PegasosCls algorithms above, we need to check that the objective functions are strongly convex.

Proposition 3. *A convex function f is λ -strongly convex iff the function $f(x) - \frac{\lambda}{2}\|x\|^2$ is convex.*

In the case of PegasosCls, the objective function

$$f_t(w) := \frac{\lambda}{2}\|w\|^2 + \max\{0, 1 - y_t\langle w, x_t \rangle\}$$

is λ -strongly convex since $f_t(w) - \frac{\lambda}{2}\|w\|^2$ is just the hinge loss function $\max\{0, 1 - y_t\langle w, x_t \rangle\}$, which is easily checked to be convex since the function $\lambda w \cdot (1 - y_t\langle w, x_t \rangle)$ is convex, and the max of two convex functions is convex. Similarly for the objective function of PegasosReg.

The learning rate η_t can be tweaked to give a better convergence result.

Theorem 2 ([Bub14], Thm 3.5). *Let $f_1, f_2, \dots, f_T$ be a sequence of λ -strongly convex functions. Let $w_1, w_2, \dots, w_{T+1}$ be a sequence of vectors defined by*

$$w_{t+1} := w_t - \eta_t g_t,$$

where w_1 is arbitrary, $\eta_t := 2/\lambda(t+1)$ and g_t is a subgradient of f_t at w_t . If $\forall t. \|g_t\| \leq G$, then we have for all u

$$\sum_{t=1}^T \frac{2t}{T(T+1)} f_t(w_t) \leq \sum_{t=1}^T \frac{2t}{T(T+1)} f_t(u) + \frac{2G^2}{\lambda(T+1)}.$$

3.2 Kernelised Pegasos

We next look at how we can apply the kernel trick to learn non-linear functions using the Pegasos framework. First, note that the general form of the weight update is

$$\begin{aligned} w_{t+1} &:= w_t - \eta_t g_t(f_t(w_t)) \\ &= w_t - \frac{1}{\lambda t} [\lambda w_t + g_t(l_t(w_t))] \\ &= (1 - \frac{1}{t})w_t - \frac{1}{\lambda t} g_t(l_t(w_t)), \end{aligned}$$

where $l_t(w_t)$ is the loss function at time t , $f_t(w_t) = l_t(w_t) + \|w_t\|^2/2$, and $g(h(w))$ is a subgradient of h at w .

Proposition 4. *If $w_1 = 0$, then for all $T \geq 1$. we have*

$$w_{T+1} = -\frac{1}{\lambda T} \sum_{t=1}^T g_t(l_t(w_t)).$$

Proof. In the base case, $w_2 = -\frac{1}{\lambda}g_1(l_1(w_1))$ as expected. For the inductive step, we have

$$\begin{aligned} w_{t+1} &:= (1 - \frac{1}{t})w_t - \frac{1}{\lambda t}g_t(l_t(w_t)) \\ &= \frac{t-1}{t} \left(-\frac{1}{\lambda(t-1)} \sum_{i=1}^{t-1} g_i(l_i(w_i)) \right) - \frac{1}{\lambda t}g_t(l_t(w_t)) \\ &= -\frac{1}{\lambda t} \sum_{i=1}^t g_i(l_i(w_i)). \end{aligned}$$

□

Armed with Proposition 4 and the subgradient formula

$$g_t(l_t(w_t)) = -\mathbb{I}[y_t \langle w_t, x_t \rangle < 1] y_t x_t,$$

we can now rewrite PegasosCls as follows:

PegasosCls(S, T, λ)

1. Sample (x, y) from S and set $M := \{(x, y)\}$
2. For $t = 2 \dots T$
 - (a) Sample (x_t, y_t) from S
 - (b) If $\frac{y_t}{\lambda(t-1)} \sum_{(x_i, y_i) \in M} y_i \langle x_i, x_t \rangle < 1$ Then $M := M \cup \{(x_t, y_t)\}$
3. Output M

In the algorithm, M is a multiset. To use the model M to make a prediction on an individual x , we compute

$$\langle w, x \rangle = \langle \frac{1}{\lambda T} \sum_{(x_i, y_i) \in M} y_i x_i, x \rangle = \frac{1}{\lambda T} \sum_{(x_i, y_i) \in M} y_i \langle x_i, x \rangle$$

and return the sign. To kernelise the algorithm, we simply replace the inner product $\langle x_i, x_t \rangle$ in Step 2b with a Mercer kernel $k(x_i, x_t)$.

We next rewrite PegasosReg. The subgradient formula is

$$g_t(l_t(w_t)) = \mathbb{I}[\langle w_t, x_t \rangle - y_t > \epsilon] x_t - \mathbb{I}[y_t - \langle w_t, x_t \rangle > \epsilon] x_t.$$

Using Proposition 4, we get the following.

PegasosReg(S, T, λ, ϵ)

1. Sample (x, y) from S and set $M := \{(\mathbb{I}[y < -\epsilon] - \mathbb{I}[y > \epsilon], x)\}$
2. For $t = 2 \dots T$
 - (a) Sample (x_t, y_t) from S
 - (b) $p := -\frac{1}{\lambda(t-1)} \sum_{(\alpha_i, x_i) \in M} \alpha_i \langle x_i, x_t \rangle$
 - (c) If $p - y_t > \epsilon$ then $M := M \cup \{(1, x_t)\}$.
 - (d) If $y_t - p > \epsilon$ then $M := M \cup \{(-1, x_t)\}$.
3. Output M

Prediction can be computed as usual and the algorithm can also be kernelised by replacing the inner product $\langle x_i, x_t \rangle$ in Step 2b with a Mercer kernel $k(x_i, x_t)$.

References

- [Bub14] Sébastien Bubeck. *Theory of Convex Optimization for Machine Learning*. 2014.
- [EMM02] Yaakov Engel, Shie Mannor, and Ron Meir. Sparse online greedy support vector regression. In *13th European Conference on Machine Learning*, pages 84–96, 2002.
- [HKKA06] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *19th COLT*, pages 499–513, 2006.
- [KSW01] Jyrki Kivinen, Alex J. Smola, and Robert C. Williamson. Online learning with kernels. In *NIPS*, pages 785–792, 2001.
- [SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Math. Program.*, 127(1):3–30, 2011.
- [SSWB00] Bernhard Schölkopf, Alexander J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Computation*, pages 1207–1245, 2000.

A Dead Ends

A.1 Sparse Online Greedy Support Vector Regression

The SOG-SVR algorithm described in [EMM02] appears fragile and sensitive to the learning rate. A line search is apparently required to tune the learning rate, and this is computationally expensive. I have difficulty even getting the algorithm to learn simple linear functions with offsets.