Regularized Least Squares and Support Vector Machines

Lorenzo Rosasco

9.520 Class 06

About this class

Goal To introduce two main examples of Tikhonov regularization, deriving and comparing their computational properties.

Basics: Data

- Training set: $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Inputs: $\mathbf{X} = \{x_1, \ldots, x_n\}$.
- Labels: $\mathbf{Y} = \{y_1, \ldots, y_n\}$.

Basics: RKHS, Kernel

- RKHS $\mathcal{H}$ with a positive semidefinite *kernel function* K:

$$\begin{aligned} \text{linear:} \quad & K(x_i, x_j) = x_i^T x_j \\ \text{polynomial:} \quad & K(x_i, x_j) = (x_i^T x_j + 1)^d \\ \text{gaussian:} \quad & K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{\sigma^2}\right) \end{aligned}$$

- Define the kernel matrix $\mathbf{K}$ to satisfy $\mathbf{K}_{ij} = K(x_i, x_j)$.
- The kernel function with one argument fixed is $K_x = K(x, \cdot)$.
- Given an arbitrary input x_*, $\mathbf{K}_{x_*}$ is a vector whose ith entry is $K(x_i, x_*)$.

Tikhonov Regularization

We are interested into studying Tikhonov Regularization

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \{ \sum_{i=1}^{n} V(y_i, f(x_i))^2 + \lambda \|f\|_{\mathcal{H}}^2 \}.$$

Representer Theorem

The representer theorem guarantees that the solution can be written as

$$f = \sum_{j=1}^{n} c_j K_{x_j}$$

for some $c = (c_1, \ldots, c_n) \in \mathbb{R}^n$.
So $\mathbf{K}c$ is a vector whose ith element is $f(x_i)$:

$$f(x_i) = \sum_{j=1}^{n} c_j K_{x_i}(x_j) = \sum_{j=1}^{n} c_j \mathbf{K}_{ij}$$

and $\|f\|_{\mathcal{H}}^2 = c^T \mathbf{K} c$.

Since $f = \sum_{j=1}^{n} c_j K_{x_j}$, then

$$
\begin{aligned}
\|f\|_{\mathcal{H}}^2 &= \langle f, f \rangle_{\mathcal{H}} \\
&= \langle \sum_{i=1}^{n} c_i K_{x_i}, \sum_{j=1}^{n} c_j K_{x_j} \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \langle K_{x_i}, K_{x_j} \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) = c^t \mathbf{K} c
\end{aligned}
$$

Plan

- RLS
  - dual problem
  - regularization path
  - linear case
- SVM
  - dual problem
  - linear case
  - historical derivation

The RLS problem

Goal: Find the function $f \in \mathcal{H}$ that minimizes the weighted sum of the square loss and the RKHS norm

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \{ \frac{1}{2} \sum_{i=1}^{n} (f(x_i) - y_i)^2 + \frac{\lambda}{2} ||f||_{\mathcal{H}}^2 \}.$$

RLS and Representer Theorem

Using the representer theorem the RLS problem is:

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{Y} - \mathbf{K}c\|_2^2 + \frac{\lambda}{2} c^T \mathbf{K} c$$

The above functional is differentiable, we can find the minimum setting the gradient w.r.t c to 0:

$$\begin{aligned} -\mathbf{K}(\mathbf{Y} - \mathbf{K}c) + \lambda \mathbf{K}c &= 0 \\ (\mathbf{K} + \lambda I)c &= \mathbf{Y} \\ c &= (\mathbf{K} + \lambda I)^{-1}\mathbf{Y} \end{aligned}$$

We find c by solving a system of linear equations.

RLS and Representer Theorem

Using the representer theorem the RLS problem is:

$$\underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{Y} - \mathbf{K}c\|_2^2 + \frac{\lambda}{2} c^T \mathbf{K} c$$

The above functional is differentiable, we can find the minimum setting the gradient w.r.t c to 0:

$$\begin{aligned} -\mathbf{K}(\mathbf{Y} - \mathbf{K}c) + \lambda \mathbf{K}c &= 0 \\ (\mathbf{K} + \lambda I)c &= \mathbf{Y} \\ c &= (\mathbf{K} + \lambda I)^{-1}\mathbf{Y} \end{aligned}$$

We find c by solving a system of linear equations.

Solving RLS for fixed Parameters

$$(\mathbf{K} + \lambda I)c = \mathbf{Y}.$$

- The matrix $\mathbf{K} + \lambda I$ is symmetric positive definite, so the appropriate algorithm is Cholesky factorization.
- In Matlab, the "slash" operator seems to be using Cholesky, so you can just write $c = (\mathbf{K} + I * I)\backslash \mathtt{Y}$, but to be safe, (or in octave), I suggest $\mathtt{R} = chol(\mathbf{K} + I * I); c = (R\backslash(\mathtt{R}'\backslash \mathtt{Y}))$;.

The above algorithm has complexity $O(n^3)$.

The RLS Solution, Comments

$$c = (\mathbf{K} + \lambda I)^{-1}\mathbf{Y}$$

The prediction at a new input x_* is:

$$\begin{aligned} f(x_*) &= \sum_{j=1}^{n} c_j \mathbf{K}_{x_j}(x_*) \\ &= \mathbf{K}_{x_*} c \\ &= \mathbf{K}_{x_*} \mathbf{G}^{-1}\mathbf{Y}, \end{aligned}$$

where $\mathbf{G} = \mathbf{K} + \lambda I$.
Note that the above operation is $O(n^2)$.

RLS Regularization Path

Typically we have to choose λ and hence to compute the solutions corresponding to different values of λ.

- Is there a more efficent method than solving $c(\lambda) = (\mathbf{K} + \lambda I)^{-1}\mathbf{Y}$ anew for each λ?
- Form the eigendecomposition $\mathbf{K} = \mathbf{Q}\Lambda\mathbf{Q}^T$, where Λ is diagonal with $\Lambda_{ii} \geq 0$ and $\mathbf{Q}\mathbf{Q}^T = I$.
- Then

$$\begin{aligned} \mathbf{G} &= \mathbf{K} + \lambda I \\ &= \mathbf{Q}\Lambda\mathbf{Q}^T + \lambda I \\ &= \mathbf{Q}(\Lambda + \lambda I)\mathbf{Q}^T, \end{aligned}$$

which implies that $\mathbf{G}^{-1} = \mathbf{Q}(\Lambda + \lambda I)^{-1}\mathbf{Q}^T$.

RLS Regularization Path

Typically we have to choose λ and hence to compute the solutions corresponding to different values of λ.

- Is there a more efficent method than solving $c(\lambda) = (\mathbf{K} + \lambda I)^{-1}\mathbf{Y}$ anew for each λ?
- Form the eigendecomposition $\mathbf{K} = \mathbf{Q}\Lambda\mathbf{Q}^T$, where Λ is diagonal with $\Lambda_{ii} \geq 0$ and $\mathbf{Q}\mathbf{Q}^T = I$.
- Then

$$\begin{aligned} \mathbf{G} &= \mathbf{K} + \lambda I \\ &= \mathbf{Q}\Lambda\mathbf{Q}^T + \lambda I \\ &= \mathbf{Q}(\Lambda + \lambda I)\mathbf{Q}^T, \end{aligned}$$

which implies that $\mathbf{G}^{-1} = \mathbf{Q}(\Lambda + \lambda I)^{-1}\mathbf{Q}^T$.

RLS Regularization Path Cont'd

- $O(n^3)$ time to solve one (dense) linear system, *or* to compute the eigendecomposition (constant is maybe 4x worse). Given $\mathbf{Q}$ and Λ, we can find $c(\lambda)$ in $O(n^2)$ time:

$$c(\lambda) = \mathbf{Q}(\Lambda + \lambda I)^{-1}\mathbf{Q}^T\mathbf{Y},$$

  noting that $(\Lambda + \lambda I)$ is diagonal.
- Finding $c(\lambda)$ for many λ's is (essentially) free!

The Linear Case

- The linear kernel is $K(x_i, x_j) = x_i^T x_j$.
- The linear kernel offers many advantages for computation.
- Key idea: we get a decomposition of the kernel matrix for free: $\mathbf{K} = \mathbf{X}\mathbf{X}^T$.
- In the linear case, we will see that we have two different computation options.

Linear kernel, linear function

With a linear kernel, the function we are learning is linear as well:

$$\begin{aligned} f(x_*) &= \mathbf{K}_{x_*} c \\ &= x_*^T \mathbf{X}^T c \\ &= x_*^T w, \end{aligned}$$

where we define w to be $\mathbf{X}^T c$.

Linear kernel cont.

For the linear kernel,

$$\begin{aligned} & \min_{c \in \mathbb{R}^n} \frac{1}{2} ||\mathbf{Y} - \mathbf{K}c||_2^2 + \frac{\lambda}{2} c^T \mathbf{K} c \\ = & \min_{c \in \mathbb{R}^n} \frac{1}{2} ||\mathbf{Y} - \mathbf{X}\mathbf{X}^T c||_2^2 + \frac{\lambda}{2} c^T \mathbf{X}\mathbf{X}^T c \\ = & \min_{w \in \mathbb{R}^d} \frac{1}{2} ||\mathbf{Y} - \mathbf{X}w||_2^2 + \frac{\lambda}{2} ||w||_2^2. \end{aligned}$$

Taking the gradient with respect to w and setting it to zero

$$\mathbf{X}^T \mathbf{X} w - \mathbf{X}^T \mathbf{Y} + \lambda w = 0$$

we get

$$w = (\mathbf{X}^T \mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \mathbf{Y}.$$

Solution for fixed parameter

$$w = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{Y}.$$

Choleski decomposition allows to solve the above problem in $O(d^3)$ for any fixed λ.

- We can work with the *covariance matrix* $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d\times d}$.
- The algorithm is identical to solving a general RLS problem replacing the kernel matrix by $\mathbf{X}^T\mathbf{X}$ and the labels vector by $\mathbf{X}^T y$.

We can classify new points in $O(d)$ time, using w, rather than having to compute a weighted sum of n kernel products (which will usually cost $O(nd)$ time).

Regularization Path via SVD

To compute solutions corresponding to multiple values of λ we can again consider an eigend-ecomposition/svd.

The economy-size SVD of $\mathbf{X}$ can be written as $\mathbf{X} = USV^T$, with $U \in \mathbb{R}^{n \times d}$, $S \in \mathbb{R}^{d \times d}$, $V \in \mathbb{R}^{d \times d}$, $U^T U = V^T V = VV^T = I_d$, and S diagonal and positive semidefinite. (Note that $UU^T \neq I_n$).

- We need $O(nd)$ memory to store the data in the first place. The (economy-sized) SVD also requires $O(nd)$ memory, and $O(nd^2)$ time.

Compared to the nonlinear case, we have replaced an $O(n)$ with an $O(d)$, in both time and memory. If $n >> d$, this can represent a huge savings.

Regularization Path via SVD

To compute solutions corresponding to multiple values of λ we can again consider an eigend-ecomposition/svd.
The economy-size SVD of $\mathbf{X}$ can be written as $\mathbf{X} = USV^T$, with $U \in \mathbb{R}^{n \times d}$, $S \in \mathbb{R}^{d \times d}$, $V \in \mathbb{R}^{d \times d}$, $U^T U = V^T V = VV^T = I_d$, and S diagonal and positive semidefinite. (Note that $UU^T \neq I_n$).

- We need $O(nd)$ memory to store the data in the first place. The (economy-sized) SVD also requires $O(nd)$ memory, and $O(nd^2)$ time.

Compared to the nonlinear case, we have replaced an $O(n)$ with an $O(d)$, in both time and memory. If $n >> d$, this can represent a huge savings.

Summary So Far

- When can we solve one RLS problem? (I.e. what are the bottlenecks?)
- We need to form $\mathbf{K}$, which takes $O(n^2 d)$ time and $O(n^2)$ memory. We need to perform a Cholesky factorization or an eigendecomposition of $\mathbf{K}$, which takes $O(n^3)$ time.
- In the linear case we have replaced an $O(n)$ with an $O(d)$, in both time and memory. If $n >> d$, this can represent a huge savings.
- **Usually, we run out of memory before we run out of time.**
- **The practical limit on today's workstations is (more-or-less) 10,000 points (using Matlab).**

Summary So Far

- When can we solve one RLS problem? (I.e. what are the bottlenecks?)
- We need to form **K**, which takes $O(n^2 d)$ time and $O(n^2)$ memory. We need to perform a Cholesky factorization or an eigendecomposition of **K**, which takes $O(n^3)$ time.
- In the linear case we have replaced an $O(n)$ with an $O(d)$, in both time and memory. If $n >> d$, this can represent a huge savings.
- **Usually, we run out of memory before we run out of time.**
- **The practical limit on today's workstations is (more-or-less) 10,000 points (using Matlab).**

Parting Shot

"You should be asking how the answers will be used and what is *really* needed from the computation. Time and time again someone will ask for the inverse of a matrix when all that is needed is the solution of a linear system; for an interpolating polynomial when all that is needed is its values at some point; for the solution of an ODE at a sequence of points when all that is needed is the limiting, steady-state value. A common complaint is that least squares curve-fitting couldn't possibly work on *this* data set and some more complicated method is needed; in almost all such cases, least squares curve-fitting will work just fine because it is so very robust."

Leader, Numerical Analysis and Scientific Computation

Plan

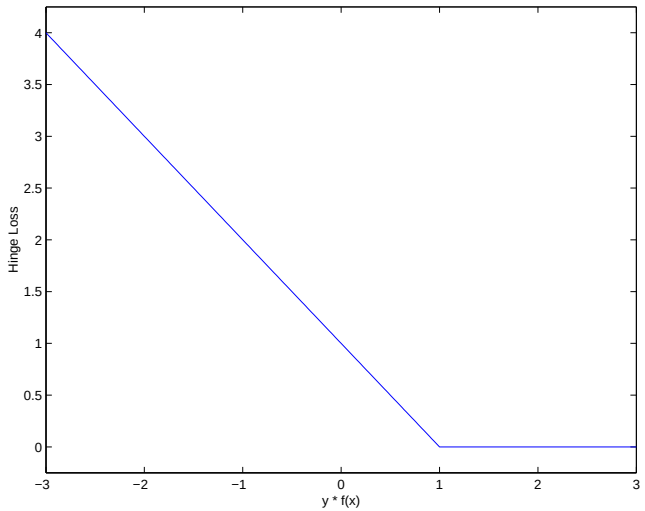
- RLS
  - dual problem
  - regularization path
  - linear case
- SVM
  - dual problem
  - linear case
  - historical derivation

The Hinge Loss

The support vector machine for classification arises considering the hinge loss

$$V(f(x,y)) \equiv (1 - yf(x))_+,$$

where $(s)_+ \equiv \max(s, 0)$.



SVM Standard Notation

With the hinge loss, our regularization problem becomes

$$\operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i f(x_i))_+ + \lambda \|f\|_{\mathcal{H}}^2.$$

In most of the SVM literature, the problem is written as

$$\operatorname*{argmin}_{f \in \mathcal{H}} \; C \sum_{i=1}^{n} V(y_i, f(x_i)) + \frac{1}{2} \|f\|_{\mathcal{H}}^2.$$

The formulations are equivalent setting $C = \frac{1}{2\lambda n}$.
This problem is non-differentiable (because of the "kink" in V).

SVM Standard Notation

With the hinge loss, our regularization problem becomes

$$\operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i f(x_i))_+ + \lambda \|f\|_{\mathcal{H}}^2.$$

In most of the SVM literature, the problem is written as

$$\operatorname*{argmin}_{f \in \mathcal{H}} \; C \sum_{i=1}^{n} V(y_i, f(x_i)) + \frac{1}{2} \|f\|_{\mathcal{H}}^2.$$

The formulations are equivalent setting $C = \frac{1}{2\lambda n}$.
This problem is non-differentiable (because of the "kink" in V).

Slack Variables Formulation

We rewrite the functional using slack variables ξ_i.

$$\begin{aligned} \underset{f \in \mathcal{H}}{\text{argmin}} \quad & C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \|f\|_{\mathcal{H}}^2 \\ \textit{subject to :} \quad & \xi_i \geq 1 - y_i f(x_i) \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad\qquad\quad i = 1, \ldots, n \end{aligned}$$

Applying the representer theorem we get a constrained quadratic programming problem:

$$\begin{aligned} \underset{c \in \mathbb{R}^n, \xi \in \mathbb{R}^n}{\text{argmin}} \quad & C \sum_{i=1}^{n} \xi_i + \frac{1}{2} c^T \mathbf{K} c \\ \text{subject to :} \quad & \xi_i \geq 1 - y_i \sum_{j=1}^{n} c_j K(x_i, x_j) \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, n \end{aligned}$$

Slack Variables Formulation

We rewrite the functional using slack variables ξ_i.

$$\begin{aligned} \underset{f \in \mathcal{H}}{\text{argmin}} \quad & C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \|f\|_{\mathcal{H}}^2 \\ \textit{subject to :} \quad & \xi_i \geq 1 - y_i f(x_i) \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad\qquad\quad i = 1, \ldots, n \end{aligned}$$

Applying the representer theorem we get a constrained quadratic programming problem:

$$\begin{aligned} \underset{c \in \mathbb{R}^n, \xi \in \mathbb{R}^n}{\text{argmin}} \quad & C \sum_{i=1}^{n} \xi_i + \frac{1}{2} c^T \mathbf{K} c \\ \text{subject to :} \quad & \xi_i \geq 1 - y_i \sum_{j=1}^{n} c_j K(x_i, x_j) \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, n \end{aligned}$$

How to Solve?

$$\begin{aligned} \underset{c\in\mathbb{R}^n,\xi\in\mathbb{R}^n}{\text{argmin}} \quad & C\sum_{i=1}^n \xi_i + \frac{1}{2}c^T\mathbf{K}c \\ \text{subject to :} \quad & \xi_i \geq 1 - y_i\left(\sum_{j=1}^n c_j K(x_i, x_j)\right) \quad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, n \end{aligned}$$

- This is a constrained optimization problem. The general approach:
  - Form the *primal* problem – we did this.
  - *Lagrangian* from primal – just like Lagrange multipliers.
  - *Dual* – one dual variable associated to each primal constraint in the Lagrangian.

Lagrangian and Dual

We derive the dual from the primal using the Lagrangian:

$$\underbrace{C\sum_{i=1}^{n}\xi_i + \frac{1}{2}c^T\mathbf{K}c - \sum_{i=1}^{n}\alpha_i(y_i\{\sum_{j=1}^{n}c_jK(x_i,x_j)\} - 1 + \xi_i) - \sum_{i=1}^{n}\zeta_i\xi_i}_{L(c,\xi,\alpha,\zeta)}$$

Dual problem is:

$$\underset{\alpha,\zeta\geq 0}{\operatorname{argmax}} \inf_{c,\xi} L(c,\xi,\alpha,\zeta)$$

First, minimize L w.r.t. (c,ξ):

$$(1) \quad \frac{\partial L}{\partial c} = 0 \implies c_i = \alpha_i y_i$$

$$(2) \quad \frac{\partial L}{\partial \xi_i} = 0 \implies C - \alpha_i - \zeta_i = 0$$

$$\implies 0 \leq \alpha_i \leq C$$

Lagrangian and Dual

We derive the dual from the primal using the Lagrangian:

$$\underbrace{C\sum_{i=1}^{n}\xi_i + \frac{1}{2}c^T\mathbf{K}c - \sum_{i=1}^{n}\alpha_i(y_i\{\sum_{j=1}^{n}c_jK(x_i,x_j)\} - 1 + \xi_i) - \sum_{i=1}^{n}\zeta_i\xi_i}_{L(c,\xi,\alpha,\zeta)}$$

Dual problem is:

$$\underset{\alpha,\zeta\geq 0}{\operatorname{argmax}}\ \inf_{c,\xi} L(c,\xi,\alpha,\zeta)$$

First, minimize L w.r.t. (c,ξ):

$$\begin{aligned}
(1)\quad & \frac{\partial L}{\partial c} = 0 && \Longrightarrow c_i = \alpha_i y_i \\
(2)\quad & \frac{\partial L}{\partial \xi_i} = 0 && \Longrightarrow C - \alpha_i - \zeta_i = 0 \\
& && \Longrightarrow 0 \leq \alpha_i \leq C
\end{aligned}$$

Towards the Dual I

From (2), plugging $\zeta_i = C - \alpha_i$ in the Lagrangian

$$\underbrace{C\sum_{i=1}^{n}\xi_i + \frac{1}{2}c^T\mathbf{K}c - \sum_{i=1}^{n}\alpha_i(y_i\{\sum_{j=1}^{n}c_jK(x_i,x_j)\} - 1 + \xi_i) - \sum_{i=1}^{n}\zeta_i\xi_i}_{L(c,\xi,\alpha,\zeta)}$$

we get

$$\underset{\alpha\geq 0}{\operatorname{argmax}}\inf_{c} L(c,\alpha) = \frac{1}{2}c^T\mathbf{K}c + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\sum_{j=1}^{n}K(x_i,x_j)c_j\right)$$

Towards the Dual II

$$\underset{\alpha \geq 0}{\operatorname{argmax}} \inf_{c} L(c, \alpha) = \frac{1}{2} c^T \mathbf{K} c + \sum_{i=1}^{n} \alpha_i \left(1 - y_i \sum_{j=1}^{n} K(x_i, x_j) c_j \right)$$

Next plugging in (1), i.e. $c_i = \alpha_i y_i$, we get

$$\begin{aligned} \underset{\alpha \geq 0}{\operatorname{argmax}} L(\alpha) &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i y_i K(x_i, x_j) \alpha_j y_j \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \alpha^T (\mathrm{diag}\mathbf{Y}) \mathbf{K} (\mathrm{diag}\mathbf{Y}) \alpha \end{aligned}$$

The Primal and Dual Problems Again

$$\begin{aligned} \underset{c\in\mathbb{R}^n,\xi\in\mathbb{R}^n}{\text{argmin}} \quad & C\sum_{i=1}^n \xi_i + \frac{1}{2}c^T\mathbf{K}c \\ \text{subject to :} \quad & \xi_i \geq 1 - y_i\left(\sum_{j=1}^n c_j K(x_i, x_j)\right) \quad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, n \end{aligned}$$

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^{\mathbf{n}}} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2}\alpha^T\mathbf{Q}\alpha \\ & 0 \leq \alpha_i \leq C \qquad i = 1, \ldots, n \end{aligned}$$

The dual problem is easier to solve: simple box constraints.

Support Vectors

The input input points with non zero coefficients are called support vectors.
We get a geometric interpretation using complementary slackness, primal/dual constraints.

Optimality Conditions

All optimal solutions must satisfy:

$$\sum_{j=1}^{n} c_j K(x_i, x_j) - \sum_{j=1}^{n} y_i \alpha_j K(x_i, x_j) = 0 \qquad i = 1, \ldots, n$$

$$C - \alpha_i - \zeta_i = 0 \qquad i = 1, \ldots, n$$

$$y_i(\sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j)) - 1 + \xi_i \geq 0 \qquad i = 1, \ldots, n$$

$$\alpha_i[y_i(\sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j)) - 1 + \xi_i] = 0 \qquad i = 1, \ldots, n$$

$$\zeta_i \xi_i = 0 \qquad i = 1, \ldots, n$$

$$\xi_i, \alpha_i, \zeta_i \geq 0 \qquad i = 1, \ldots, n$$

Optimality Conditions

These optimality conditions are both necessary and sufficient for optimality: (c, ξ, α, ζ) satisfy all of the conditions if and only if they are optimal for both the primal and the dual. (Also known as the Karush-Kuhn-Tucker (KKT) conditons.)

Interpreting the solution — sparsity

$$\alpha_i[y_i(\sum_{j=1}^{n} y_j\alpha_j K(x_i, x_j)) - 1 + \xi_i] = 0, \quad i = 1, \ldots, n.$$

Remember we defined $f(x) = \sum_{i=1}^{n} y_i\alpha_i K(x, x_i)$, so that

$$\begin{aligned} y_i f(x_i) > 1 \quad &\Rightarrow \quad (1 - y_i f(x_i)) < 0 \\ &\Rightarrow \quad \xi_i \neq (1 - y_i f(x_i)) \\ &\Rightarrow \quad \alpha_i = 0 \end{aligned}$$

Interpreting the solution — support vectors

Consider

$$C - \alpha_i - \zeta_i = 0 \qquad i = 1, \ldots, n$$
$$\zeta_i \xi_i = 0 \qquad i = 1, \ldots, n$$

$$\begin{aligned} y_i f(x_i) < 1 \quad &\Rightarrow \quad (1 - y_i f(x_i)) > 0 \\ &\Rightarrow \quad \xi_i > 0 \\ &\Rightarrow \quad \zeta_i = 0 \\ &\Rightarrow \quad \alpha_i = C \end{aligned}$$

Interpreting the solution — support vectors

So

$$y_i f(x_i) < 1 \Rightarrow \alpha_i = C.$$

Conversely, suppose $\alpha_i = C$. From

$$\alpha_i [y_i (\sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j)) - 1 + \xi_i] = 0, \quad i = 1, \ldots, n.$$

we have

$$\begin{aligned} \alpha_i = C &\implies \xi_i = 1 - y_i f(x_i) \\ &\implies y_i f(x_i) \leq 1 \end{aligned}$$

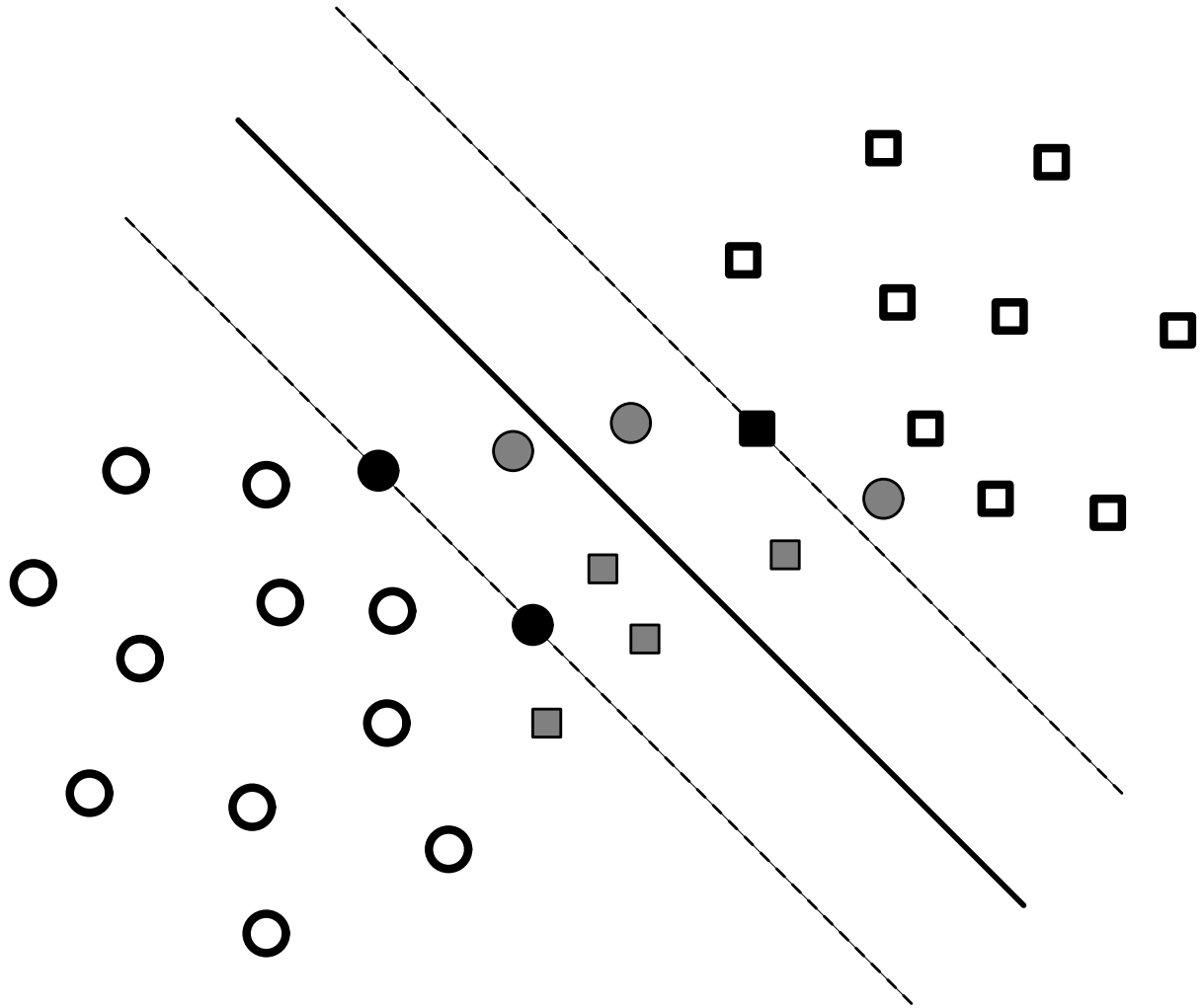
Interpreting the solution

Here are all of the derived conditions:

$$\begin{aligned} \alpha_i = 0 &\implies y_i f(x_i) \geq 1 \\ 0 < \alpha_i < C &\implies y_i f(x_i) = 1 \\ \alpha_i = C &\impliedby y_i f(x_i) < 1 \end{aligned}$$

$$\begin{aligned} \alpha_i = 0 &\impliedby y_i f(x_i) > 1 \\ \alpha_i = C &\implies y_i f(x_i) \leq 1 \end{aligned}$$

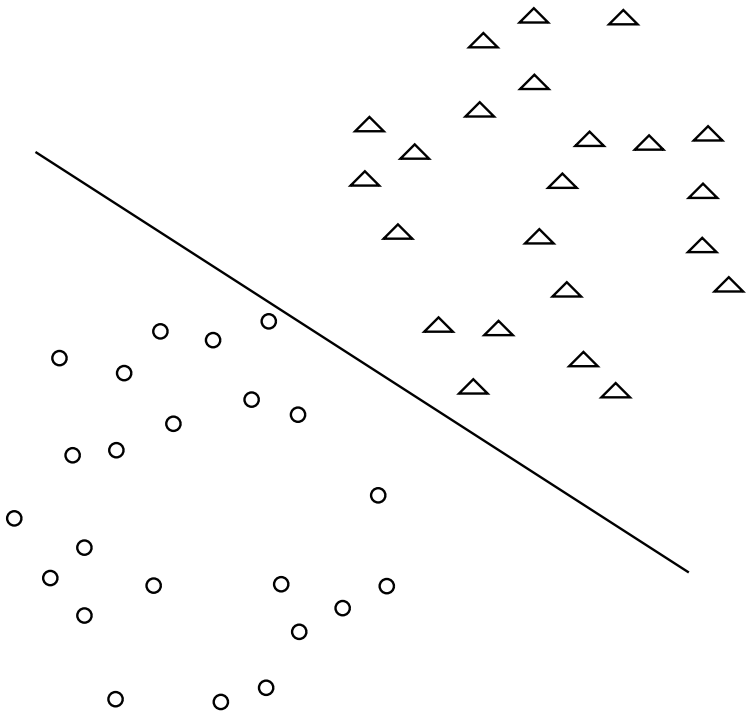
Geometric Interpretation of Reduced Optimality Conditions



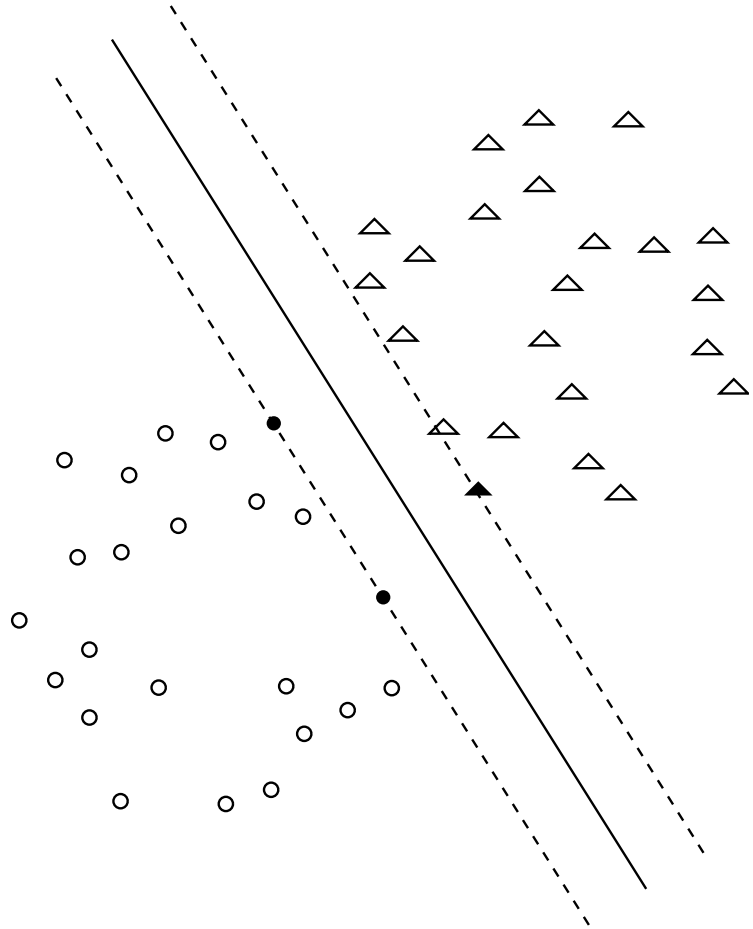
The Geometric Approach

The “traditional” approach to describe SVM is to start with the concepts of *separating hyperplanes* and *margin*. The theory is usually developed in a linear space, beginning with the idea of a perceptron, a linear hyperplane that separates the positive and the negative examples. Defining the margin as the distance from the hyperplane to the nearest example, the basic observation is that intuitively, we expect a hyperplane with larger margin to generalize better than one with smaller margin.

Large and Small Margin Hyperplanes



(a)



(b)

Maximal Margin Classification

Classification function:

$$f(x) = \text{sign}\,(w \cdot x). \tag{1}$$

w is a normal vector to the hyperplane separating the classes. We define the boundaries of the margin by $\langle w, x \rangle = \pm 1$.

What happens as we change $\|w\|$?

We push the margin in/out by rescaling w – the margin moves out with $\frac{1}{\|w\|}$. So maximizing the margin corresponds to minimizing $\|w\|$.

Maximal Margin Classification

Classification function:

$$f(x) = \text{sign}\,(w \cdot x). \tag{1}$$

w is a normal vector to the hyperplane separating the classes. We define the boundaries of the margin by $\langle w, x \rangle = \pm 1$.

What happens as we change $\|w\|$?

We push the margin in/out by rescaling w – the margin moves out with $\frac{1}{\|w\|}$. So maximizing the margin corresponds to minimizing $\|w\|$.

Maximal Margin Classification, Separable case

Separable means $\exists w$ s.t. all points are beyond the margin, i.e.

$$y_i \langle w, x_i \rangle \geq 1 \ , \ \forall i.$$

So we solve:

$$\begin{aligned} \underset{w}{\text{argmin}} \quad & \|w\|^2 \\ \text{s.t.} \quad & y_i \langle w, x_i \rangle \geq 1 \ , \ \forall i \end{aligned}$$

Maximal Margin Classification, Non-separable case

Non-separable means there are points on the wrong side of the margin, i.e.

$$\exists i \text{ s.t. } y_i \langle w, x_i \rangle < 1 \ .$$

We add slack variables to account for the wrongness:

$$\begin{aligned} \underset{\xi_i, w}{\text{argmin}} \quad & \sum_{i=1}^{n} \xi_i + \|w\|^2 \\ \text{s.t.} \quad & y_i \langle w, x_i \rangle \geq 1 - \xi_i \ , \ \forall i \end{aligned}$$

Historical Perspective

Historically, most developments begin with the geometric form, derived a dual program which was identical to the dual we derived above, and only then observed that the dual program required only dot products and that these dot products could be replaced with a kernel function.

More Historical Perspective

In the linearly separable case, we can also derive the separating hyperplane as a vector parallel to the vector connecting the closest two points in the positive and negative classes, passing through the perpendicular bisector of this vector. This was the “Method of Portraits”, derived by Vapnik in the 1970’s, and recently rediscovered (with non-separable extensions) by Keerthi.

Summary

- The SVM is a Tikhonov regularization problem, with the hinge loss.
- Solving the SVM means solving a constrained quadratic program, rouhgly $O(n^3)$
  - It's better to work with the dual program.
- Solutions can be *sparse* – few non-zero coefficients, this can have impact for memory and computational requirements.
- The non-zero coefficients correspond to points not classified correctly enough – a.k.a. "support vectors."
- There is alternative, geometric interpretation of the SVM, from the perspective of "maximizing the margin."

RLS and SVM Toolbox

- GURLS (Grand Unified Regularized Least Squares)
  `http://cbcl.mit.edu/gurls/`
- SVM Light: `http://svmlight.joachims.org`
- libSVM:
  `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`