Regularized Least Squares and Support Vector Machines

Francesca Odone and **Lorenzo Rosasco**

RegML 2013

About this class

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

Basics: Data

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

- 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

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

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

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

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

Using the representer theorem the RLS problem is:

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

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

Using the representer theorem the RLS problem is:

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

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

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

We find c by solving a system of linear equations.

Solving RLS for fixed Parameters

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

- The matrix $\mathbf{K} + \lambda n I$ is symmetric positive definite (with $\lambda > 0$), so the appropriate algorithm is Cholesky factorization.
- In Matlab, the operator $\backslash$ seems to be using Cholesky, so you can just write `c = (K +lambda*n*I)\Y;`
- To be safe (or in Octave)
  `R = chol(K +lambda*n*I); c = (R\(R'\Y));`.

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

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

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 $\mathbf{G} = \mathbf{K} + \lambda n I$.
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 $c(\lambda) = (\mathbf{K} + \lambda n I)^{-1}\mathbf{Y}$ anew for each λ?

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

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

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

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:

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

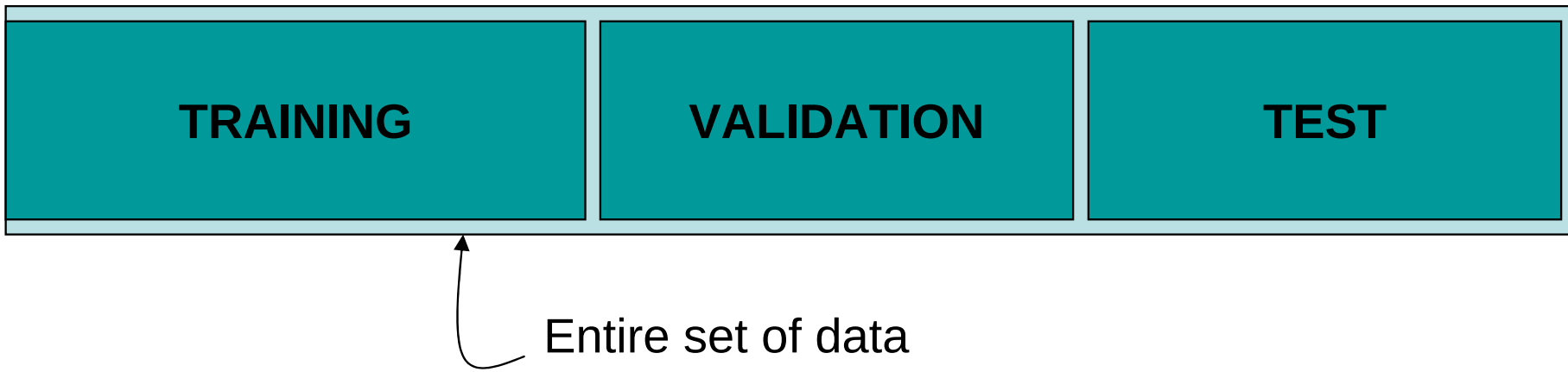
  noting that $(\Lambda + \lambda n I)$ is diagonal.
- Finding $c(\lambda)$ for many λ's is (essentially) free!

PARAMETER CHOICE

- idea: try different λ and see which one performs best
- How to try them? A simple choice is to use a validation set of data
- If we have "enough" training data we may sample out a training and a validation set.
- Otherwise a common practice is K-fold Cross Validation (KCV):
  1. Divide data into K sets of equal size: $S_1, \ldots, S_k$
  2. For each i train on the other $K - 1$ sets and test on the ith set
- If $K = n$ we get the leave-one-out strategy (LOO)

PARAMETER CHOICE

- Notice that some data should *always* be kept aside to be used as test set, to test the generalization performance of the system **after** parameter tuning took place



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: $\mathbf{K} = \mathbf{X}\mathbf{X}^T$
  — where $\mathbf{X} = [x_1^\top, \ldots, x_n^\top]$ is the data matrix $n \times d$
- 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 $\mathbf{X}^T c$.

For the linear kernel,

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

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

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

we get

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

Solution for fixed parameter

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

Choleski decomposition allows us 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 eigendecomposition/svd.

- We need $O(nd)$ memory to store the data in the first place. The 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?)

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).**

Plan

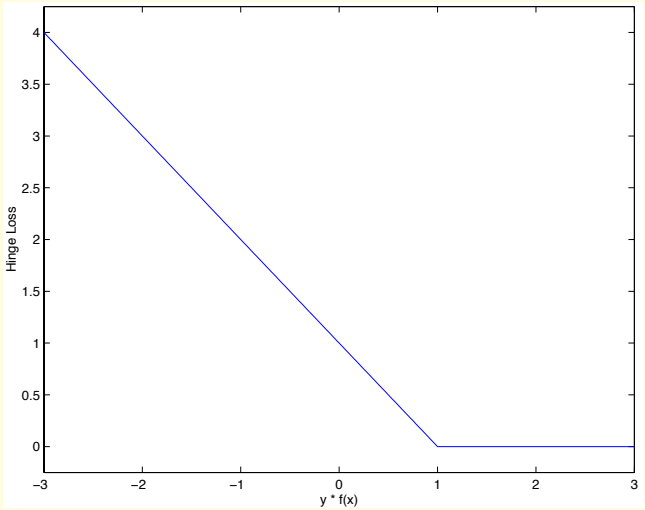
- RLS
  - dual problem
  - regularization path
  - linear case
- SVM
  - dual problem
  - linear case
  - historical derivation

The Hinge Loss

The support vector machine (SVM) 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

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

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

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

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.

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

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?

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

- 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.

We derive the dual from the primal using the Lagrangian:

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

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):

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

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

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

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

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

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

The Primal and Dual Problems

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

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

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

Support Vectors

- Basic idea: solve the dual problem to find the optimal α's, and use them to find c.
- The dual problem is easier to solve than the primal problem. It has simple box constraints and a single equality constraint, and the problem can be decomposed into a sequence of smaller problems.

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 (c, ξ) to the primal problem must satisfy the following conditions for some (α, ζ):

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

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \zeta_i = 0 \qquad i = 1, \ldots, n$$

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

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

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

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

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.)

We defined $f(x) = \sum_{i=1}^{n} y_i \alpha_i K(x, x_i)$. Now we can interpret the solution
From the condition

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

we derive

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

From the condition

$$\begin{aligned} C - \alpha_i - \zeta_i &= 0 \qquad i = 1, \ldots, n \\ \zeta_i \xi_i &= 0 \qquad i = 1, \ldots, n \end{aligned}$$

we derive

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

Conversely, suppose $\alpha_i = C$. For

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

to hold we must have

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

Similarly, if $\alpha_i = 0$. From

$$C - \alpha_i - \zeta_i = 0 \Rightarrow \zeta_i = C$$

For $\zeta_i \xi_i = 0$ we need $\xi_i = 0$.
The condition

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

becomes

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

then

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

Finally, if $0 < \alpha_i < C$. From

$$C - \alpha_i - \zeta_i = 0 \Rightarrow 0 < \zeta_i < C$$

For $\zeta_i \xi_i = 0$ we need $\xi_i = 0$.
The condition

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

becomes

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

then

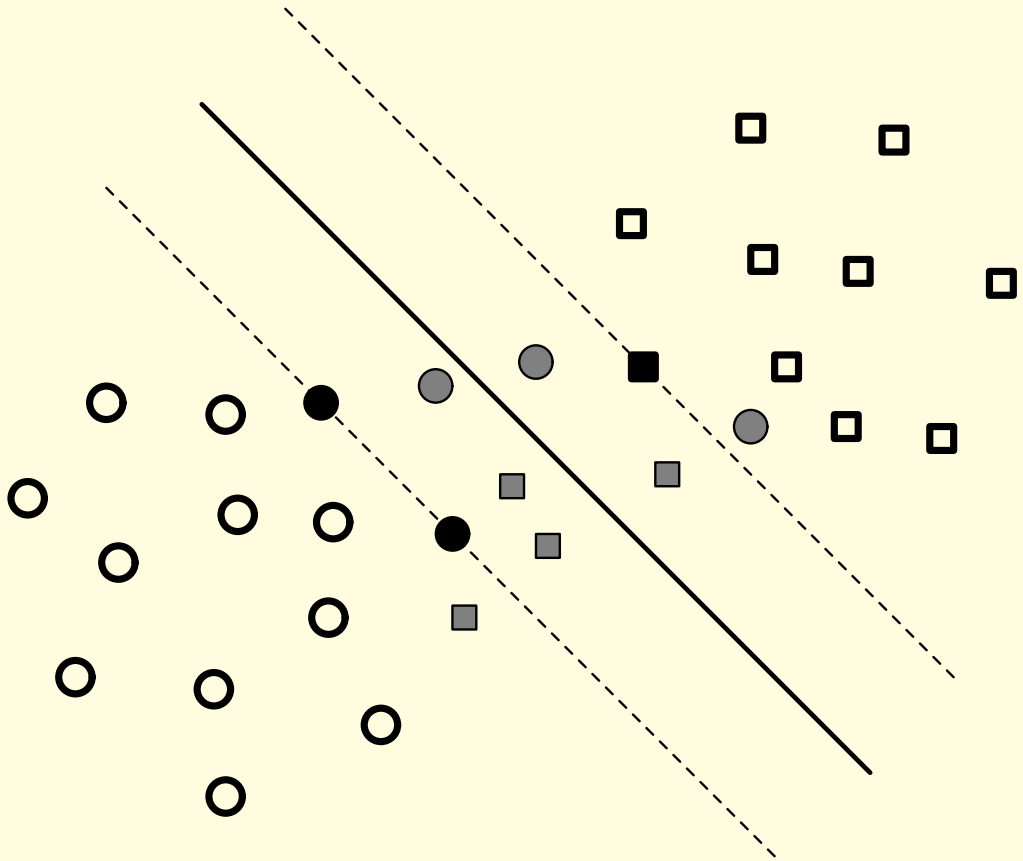
$$\begin{aligned} 0 < \alpha_i < C \quad &\Longrightarrow \quad \xi_i = 0 \\ &\Longrightarrow \quad y_i f(x_i) = 1 \end{aligned}$$

Here are all of the derived conditions:

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

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

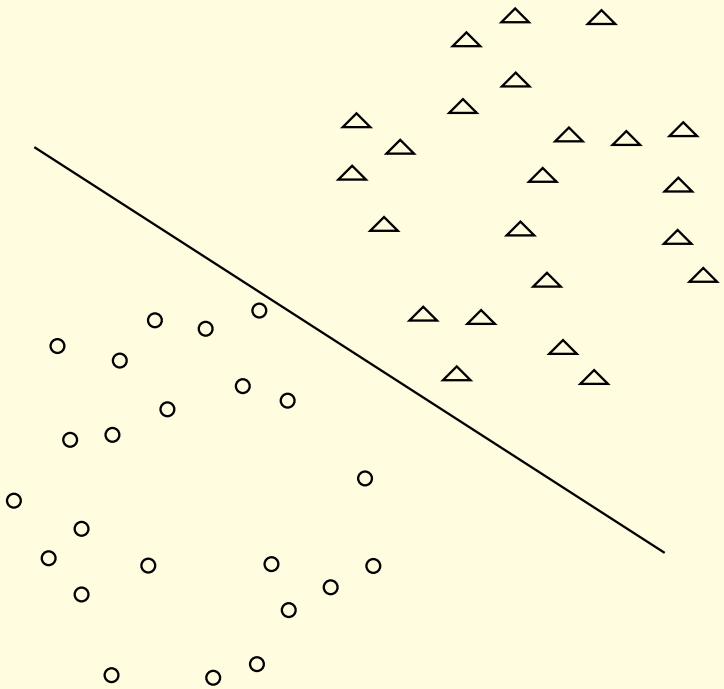
Geometric Interpretation of Reduced Optimality Conditions



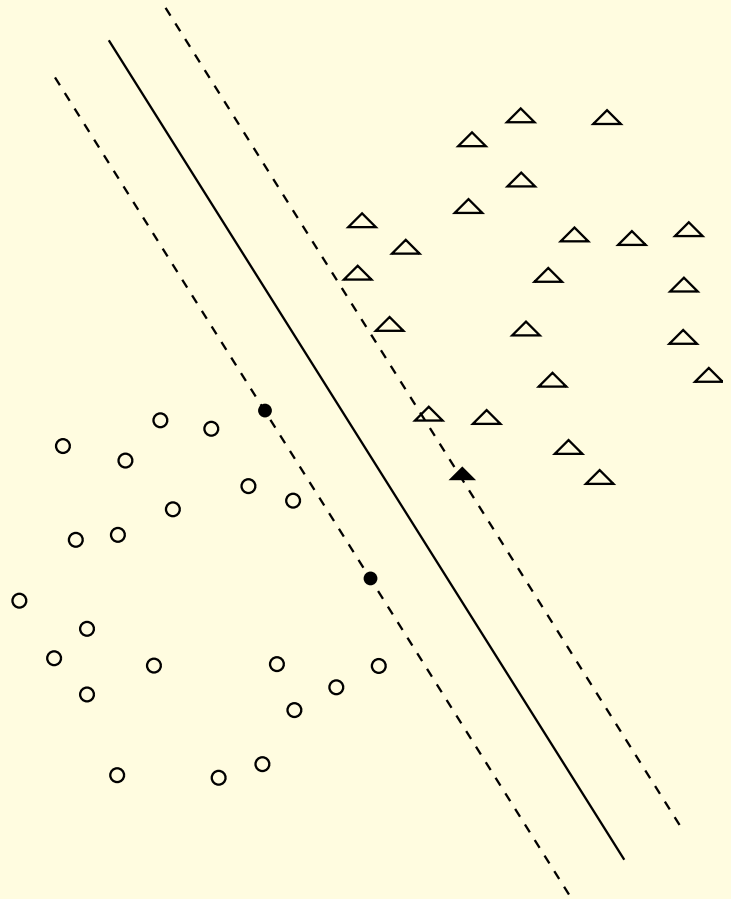
The Geometric Approach

- 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 $\|w\|$?

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."