Support Vector Machines

José Dorronsoro
Escuela Politécnica Superior
Universidad Autónoma de Madrid

2019

Contents

A Machine Learning Timeline?

- Multilayer Perceptrons: 1986–**2000?**
  - D.E. Rumelhart, G.E, Hinton, R.J. Williams. Learning representations by back-propagating errors Nature 323, 533-536, 1986
- Support Vector Machines: 1995–**2010?**
  - C Cortes, V Vapnik. Support-vector networks. Machine learning 20, 273-297, 1995
- Random Forests, Gradient Boosting Regression: 2000–**2015?**
  - L. Breiman. Random forests. Machine learning 45, 5-32, 2001
  - J.H. Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics 29, 1189-1232, 2001
- Deep Neural Networks: 2010–**20xx?**
  - E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. Neural Computation 18, 1527-1554, 2006
  - Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle. Greedy layer-wise training of deep networks. Advances in Neural Information Processing Systems 19 (NIPS'06), 153-160, 2007

1 Basic Classification

Classification Setup

- We have random patterns ω from M classes, $C_1, \ldots C_M$
- Over each pattern we "measure" d features $x = x(\omega) \in \mathbb{R}^d$
  - x inherits the randomness in ω and becomes a random variable
- A ω has a **prior probability** π_m of belonging to C_m
- Inside each class C_m there is a **conditional class density** $f(x|m)$ that "controls" the appearance of a given x
- The π_m and $f(x|m)$ determine the **posterior probability** $P(m|x)$ that x comes from class C_m
- **Intuition:** we should assign x to the class with the largest $P(m|x)$, that is, work with the classifier

$$\delta(x) = \text{arg max}_m P(m|x)$$

Computing Posterior Probabilities I

- If $\pi_m, f(x|m)$ and $f(x)$ are the prior probabilities and densities, we then have

$$P(m|x) = \frac{\pi_m \, f(x|m)}{f(x)}$$

- Therefore
$$\delta(x) = \text{arg max}_m \pi_m \ f(x|m)$$

- But computing $f(x|m)$ is usually hopeless
- Have to simplify things!
- Starting point
$$P(m|x) \simeq P(m|B_r(x)) = \frac{P(C_m \cap B_r(x))}{P(B_r(x))}$$

Computing Posterior Probabilities II

- Assume a sample with N patterns of which N_m are in C_m
- Choose an integer k and for a given x let $B(x, r)$ the smallest ball around x with k samples
- Let k_m the number of ball samples in class C_m
- Then
$$P(C_m \cap B_r(x)) = \frac{N_m}{N}\frac{k_m}{N_m}, \ P(B_r(x)) = \frac{k}{N}$$
and therefore
$$P(m|x) \simeq \frac{k_m}{N_m}\frac{N_m}{N}\frac{1}{\frac{k}{N}} = \frac{k_m}{k}$$

- We arrive at the k–**Nearest Neighbor** classifier
$$\delta_k^{NN}(x) = \text{arg max}_m \frac{k_m}{k} = \text{arg max}_m \ k_m$$

Computing Posterior Probabilities III

- Second option: **Logistic Regression**
- We assume
$$P(1|x) = \frac{1}{1 + e^{-(w_0 + w \cdot x)}}$$

- Then we have
$$P(0|x) = 1 + e^{w_0 + w \cdot x} \ \text{ and } \ \log \frac{P(1|x)}{P(0|x)} = w_0 + w \cdot x$$

- We estimate the optimal w^*, w_0^* by maximizing the sample's log likelihood
$$\begin{aligned} \ell(w_0, w; S) &= \log P(Y|X; w_0, w) \\ &= \sum_p y^p \ (w_0 + w \cdot x^p) - \sum_p \log(1 + e^{w_0 + w \cdot x^p}) \end{aligned}$$

2 Support Vector Classification

2.1 Classification and Margins

Revisiting the Classification Problem

- Basic problem: binary classification of a sample

$$S = \{(x^p, y^p), 1 \le p \le N\}$$

with d–dimensional x^p patterns and $y^p = \pm 1$

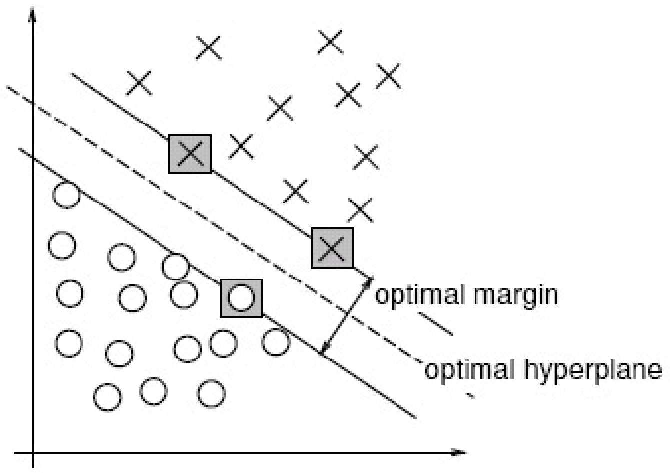
- We assume that S is **linearly separable**: for some w, b

$$\begin{aligned} w \cdot x^p + b &> 0 \quad \text{if } y^p = 1; \\ w \cdot x^p + b &< 0 \quad \text{if } y^p = -1 \end{aligned}$$

- More concisely, we want $y^p(w \cdot x^p + b) > 0$
- Q: When will such a model generalize well?
- Before that: can we find such a pair w, b?

Margins and Generalization

- A: Intuitively, when it has a large **margin**



- Q: How can we ensure a maximum margin?

Distance to a Hyperplane

- Recall that given the hyperplane $\pi : w \cdot x + b = 0$, w is orthogonal to the surface defined by π

- Thus, the distance $d(x, \pi)$ of a point x to π is computed by the projection on w of a vector $\overrightarrow{x^0 x}$, i.e.

$$d(x, \pi) = \frac{\left| w \cdot \overrightarrow{x^0 x} \right|}{\|w\|} = \frac{\left| w \cdot x - w \cdot x^0 \right|}{\|w\|} = \frac{|w \cdot x + b|}{\|w\|}$$

- We take absolute values to compensate for the orientation of w
- When the origin is in π (homogeneous π), the distance is

$$d(x, \pi) = \frac{|w \cdot x|}{\|w\|}$$

Learning and Margins

- If we assume w "points" to the positive patterns, we have $y^p(w \cdot x^p + b) = |w \cdot x^p + b|$
- The **margin** $m(w, b, S)$ is precisely the **minimum distance** between the sample S and π, i.e.,

$$m(w, b, S) = \min_p d(x^p, \pi) = \min_p \frac{y^p(w \cdot x^p + b)}{\|w\|}$$

- Notice that $(\lambda w, \lambda b)$ give the same margin than (w, b); we can thus normalize (w, b) as we see fit
- For instance, we could take $\|w\| = 1$ and have

$$m(w, b, S) = \min_p \frac{y^p(w \cdot x^p + b)}{\|w\|} = \min_p y^p(w \cdot x^p + b)$$

- The denominator disappears but we are left with an ugly numerator

Renormalizing the Hyperplane

- To avoid this, the following normalization of w, b is much more convenient

$$\min_p y^p(w \cdot x^p + b) \geq 1$$

- Since S is finite, we can work with w, b such that $y^{p_0}(w \cdot x^{p_0} + b) = 1$ for some p_0
- For a pair w, b so normalized we then have

$$m(w, b) = \min_p \left\{ \frac{y^p(w \cdot x^p + b)}{\|w\|} \right\} = \frac{1}{\|w\|}$$

- Thus, we maximize the overall margin working with these w and maximizing $1/\|w\|$, i.e., **minimizing** $\|w\|$ or, simply, $\frac{1}{2}\|w\|^2$

The Primal Problem

- We therefore rewrite the problem of finding a maximum margin separating hyperplane as

$$\min_{w,b} f(w,b) = \frac{1}{2}\|w\|^2$$

  s.t. $y^p(w \cdot x^p + b) \geq 1$

- This is the **SVM Primal Problem**: a **quadratic programming problem with linear restrictions** (actually affine)
- The function to minimize is very simple and also the constraints
- But there are too many of them for a direct attempt to minimization
- Solution within general theory of **constrained convex minimization**

2.2 Constrained Convex Optimization

The Lagrangian

- The Lagrangian of the primal problem is

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_p \alpha_p \left(y^p(w \cdot x^p + b) - 1\right),$$

  with $\alpha_p \geq 0$

- By construction, $L(w,b,\alpha) \leq f(w,b)$ and $L(w,b,0) = f(w,b)$
- Thus, for **feasible** w, b, α,

$$\min_{w,b \text{ feasible}} f(w,b) = \min_{w,b \text{ feasible}} \max_{\alpha \text{ feasible}} L(w,b,\alpha)$$

- Q: perhaps it holds that

$$\min_{w,b \text{ feasible}} \max_{\alpha \text{ feasible}} L(w,b,\alpha) = \max_{\alpha \text{ feasible}} \min_{w,b \text{ feasible}} L(w,b,\alpha)$$

- Let's hope so and define the **dual** function

The Dual Function

- The **dual** function is $\Theta(\alpha) = \min_{w,b} L(w,b,\alpha)$
  - Notice that we drop the requirement that w, b be feasible
- The **dual problem** D is now

$$\max \Theta(\alpha) \text{ s. t. } \alpha_p \geq 0$$

- Now we have for any feasible w, b, α

$$\Theta(\alpha) = \min_{w',b'} L(w',b',\alpha) \leq L(w,b,\alpha) \leq f(w,b)$$

- **Weak duality**: for primal optimal w^*, b^*, dual optimal α^* and any feasible w, b, α,

$$\Theta(\alpha) \leq \Theta(\alpha^*) \leq L(w^*, b^*, \alpha^*) \leq f(w^*, b^*) \leq f(w, b)$$

- **Dual gap**: $f(w^*, b^*) - \Theta(\alpha^*) \geq 0$

Strong Duality

- We have **strong duality** when the dual gap is 0 for some w^*, b^*, α^* feasible, i.e.,

$$f(w^*, b^*) = \Theta(\alpha^*)$$

- Then, w^*, b^* and α^* solve the primal and dual problems, respectively
- And moreover $\Theta(\alpha^*) = L(w^*, b^*, \alpha^*) = f(w^*, b^*)$
- **Theorem: The SVM problem has strong duality**
- Thus we can try the following:
  - Write an explicit dual problem with easier constraints
  - Solve the dual problem
  - Get the optimal primals w^*, b^* from the optimal dual α^*

Computing the Dual Function

- We first reorganize the (convex) Lagrangian as

$$L(w, b, \alpha) = w \cdot \left(\frac{1}{2} w - \sum_p \alpha_p y^p x^p \right) - b \sum_p \alpha_p y^p + \sum_p \alpha_p$$

- To minimize $L(w, b, \alpha)$ w.r. w and b, we just solve $\nabla_w L = 0$, $\frac{\partial L}{\partial b} = 0$
- From $\nabla_w L = 0$ we derive $w = \sum_p \alpha_p y^p x^p$
- From $\frac{\partial L}{\partial b} = 0$ we derive $\sum_p \alpha_p y^p = 0$

Computing the Dual Function II

- Substituting both into L we arrive at

$$\begin{aligned}\Theta(\alpha) &= \sum_p \alpha_p - \frac{1}{2} w \cdot \sum_p \alpha_p y^p x^p \\ &= \sum_p \alpha_p - \frac{1}{2} \sum_{p,q} \alpha_p \alpha_q y^p y^q \; x^p \cdot x^q = \sum_p \alpha_p - \frac{1}{2} \alpha^\tau Q \alpha\end{aligned}$$

with $Q_{p,q} = y^p y^q \;\; x^p \cdot x^q$

- The dual problem becomes

$$\max_{\alpha} \Theta(\alpha) = \max_{\alpha} \left\{ \sum_p \alpha_p - \frac{1}{2}\alpha^{\tau} Q \alpha \right\}$$

  subject to the constraints $\alpha_p \geq 0$, $\sum_p \alpha_p y^p = 0$

- As usual, we will minimize $-\Theta(\alpha)$ (and drop the $-$ from the notation)

Solving the Dual Problem

- We arrive again at a quadratic programming problem but with much simpler restrictions that we can try to simplify further
- The more difficult **linear** constraint $\sum_p \alpha_p y^p = 0$ comes from $\frac{\partial L}{\partial b} = 0$ and we can avoid it dropping b
- Thus, we try first to solve the **homogeneous** primal problem

$$\min \frac{1}{2}\|w\|^2 \quad \text{s.t.} \quad y^p \, w \cdot x^p \geq 1$$

  and its dual one

$$\min \frac{1}{2}\alpha^{\tau} Q \alpha - \sum_p \alpha_p \quad \text{s.t.} \quad \alpha_p \geq 0$$

Projected Gradient Descent

- To deal with the box constraints, we could simply apply gradient descent and clip it if needed
- The gradient of Θ is just

$$\nabla \Theta = Q\alpha - \mathbf{1}$$

  with **1** the all ones vector and we can apply it by **projected gradient descent**

- Projected (i.e., clipped) descent:
  - At step t update first α^t to α' as $\alpha'_p = \alpha^t_p - \rho\left((Q\alpha^t)_p - 1\right)$ for an appropriate step ρ
  - And then clip α' as $\alpha^{t+1}_p = \max\{\alpha'_p, 0\}$
- Nice and fine, but notice that $\dim(\alpha) = N$:
  - Computations have a cost of $O(N^2)$ per iteration
  - We need to keep Q in memory
  - Both very costly for large N

Dual Coordinate Gradient Descent

- Instead of updating the entire α we just cycle through its coordinates updating them one by one

- Since
$$\frac{\partial \Theta}{\partial \alpha_p} = (Q\alpha)_p - 1$$
we update α_p as
$$\alpha_p' = \alpha_p - \rho\left((Q\alpha)_p - 1\right)$$
- The optimal step is $\rho = \frac{1}{Q_{pp}}$ and the final update is
$$\alpha_p' = \max\left\{0, \alpha_p - \frac{(Q\alpha)_p - 1}{Q_{pp}}\right\}$$
- Maintaning the $(Q\alpha)_p$ is relatively simple and this is basically the approach followed in the LIBLINEAR package
  - It also has a primal counterpart

The SMO Algorithm

- Usually homogeneous SVMs give poorer results
- The simplest way to handle the linear constraint is
  - Start with an α^0 that verifies it
  - Update α^t to $\alpha^{t+1} = \alpha^t + \rho_t d^t$ with a direction d^t that also verifies it
  - Then $\sum_p \alpha^{t+1} y^p = \sum_p \alpha_p^t y^p + \rho_t \sum_p d_p^t y^p = 0$
- Simplest choice: select L_t, U_t so that $d^t = y^{L_t} e_{L_t} - y^{U_t} e_{U_t}$ is a maximal **descent direction**
- Since $\nabla_\alpha \Theta(\alpha^t) \cdot d^t = y^{L_t} \nabla\Theta(\alpha^t)_{L_t} - y^{U_t} \nabla\Theta(\alpha^t)_{U_t}$, the straightforward choice is
$$L_t = \arg\min_p y^p \nabla\Theta(\alpha^t)_p, \quad U_t = \arg\min_q y^q \nabla\Theta(\alpha^t)_q$$
- This is the basis of the **Sequential Minimal Optimization** (SMO) algorithm

Optimality Conditions

- Since L is convex in w, b and we have
$$\Theta(\alpha^*) = \min_{w,b} L(w, b, \alpha^*)$$
stationarity is necessary:
$$\nabla_w L(w^*, b^*, \alpha^*) = 0,\ \nabla_b L(w^*, b^*, \alpha^*) = 0$$
- By strong duality, $L(w^*, b^*, \alpha^*) = f(w^*, b^*)$ and **complementary slackness** follows
$$\alpha_p^* \left(y^p(w^* \cdot x^p + b^*) - 1\right) = 0 \ \text{ for all } \ p$$
- These conditions plus feasibility are together known as the **Karush–Kuhn–Tucker (KKT)** conditions, that are necessary and sufficient for w^*, b^*, α^* to be optimal

From Dual to Primal Solutions I

- We will use some of the KKT conditions to derive the optimal primal w^*, b^* after we obtain a dual optimal α^*
- Obvioulsy $w^* = \sum_p \alpha_p^* y^p x^p = \sum_{\alpha_p^* > 0} \alpha_p^* y^p x^p$
- What about b^*? Recall that the optimal α^*, w^*, b^* must satisfy the KKT conditions, that now are

$$\alpha_p^* \left(y^p (w^* \cdot x^p + b^*) - 1 \right) = 0$$

- Thus, if $\alpha_p^* > 0$, then $w^* \cdot x^p + b^* = y^p$ and, hence

$$b^* = y^p - w^* \cdot x^p$$

From Dual to Primal Solutions II

- In practice is better to average this formula over all $\alpha_p^* > 0$:

$$b^* = \frac{1}{N_S} \sum_{\{\alpha_q^* > 0\}} (y^q - w^* \cdot x^q)$$

  with $N_S = |\{q : \alpha_q^* > 0\}|$
- We have now completely solved linear SVM training;
- But there are more insights to be gained from the convex optimization perspective
- In particular, the KKT conditions have more information

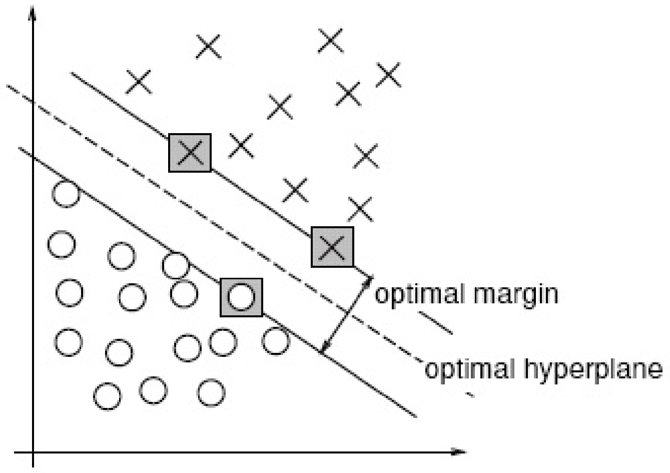
Support Vectors I

- Again, if $\alpha_p^* > 0$, then $y^p(w^* \cdot x^p + b^*) = 1$
  - Thus if $\alpha_p^* > 0$, x^p lies in one of the two **support hyperplanes** $w^* \cdot x^p + b^* = \pm 1$
- Vectors for which $\alpha_p^* > 0$ are thus called **support vectors** and the optimal w^* is a **linear combination** of them

$$w^* = \sum_{\{x^p \ \ SV\}} \alpha_p^* y^p x^p$$

- On the other hand, if x^p is not in a support hyperplane, then $y^p(w^* \cdot x^p + b^*) > 1$ and the KKT conditions imply $\alpha_p^* = 0$
- Notice that there may be x^p in the support hyperplanes that do not contribute to w^*

Support Vectors II



- In fact, while the optimal w^* is unique, the optimal α^* may be not
- In any case, **the support vectors completely determine the SVM classifier**

Takeaways on Linear SVMs I

- Maximum margins (MM) improve the generalization of linear classifiers
- To get a MM classifier we solve the primal problem

$$\min_{w,b} \frac{1}{2}\|w\|^2 \text{ s.t. } y^p(w \cdot x^p + b) \geq 1, 1 \leq p \leq N$$

- This is convex quadratic programming problem whose Lagrangian for $\alpha_p \geq 0$ is

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_p \alpha_p \left(y^p(w \cdot x^p + b) - 1\right),$$

- If $\mathcal{C} = \{\alpha : \alpha_p \geq 0, \sum \alpha_p y^p = 0\}$, the dual problem is

$$\max_{\alpha_p \in \mathcal{C}} \ \Theta(\alpha) = \sum_p \alpha_p - \frac{1}{2}\alpha^\tau Q \alpha$$

Takeaways on Linear SVMs II

- The dual gap $f(w^*, b^*) - \Theta(\alpha^*)$ is 0 and so we can
  - Obtain the optimal dual α^* and then
  - Derive the optimal primal w^*, b^*
- We solve the dual problem using the **SMO algorithm**, with a cost at least $\Omega(N^2)$

- The KKT conditions are used to obtain w^* and b^*
- For the optimal w^* we have $w^* = \sum_{SV} \alpha_p^* y^p x^p$
- For the optimal b^* we have $b^* = y^p - w^* \cdot x^p$ if $\alpha^* > 0$
- If $\alpha^* > 0$, $w^* \cdot x^p + b^* = y^p$, i.e., x^p is in one of the **support hyperplanes** $w^* \cdot x + b^* = \pm 1$

3 Non Linear SV Classification

3.1 Linear SVMs for Non Linear Problems

Cover's Theorem

- SVMs are simple and elegant, but also linear
- Q: Will linear SVM classifiers be powerful enough?
- Alternatively: Are linearly solvable classification problems **frequent enough**?
- A: No, because of **Cover's Theorem**
- The patterns in a size N sample S with dimension d are to be in **general position** if no $d+1$ points are in a $(d-1)$–dimensional hyperplane
- Then, if $N \le d+1$, **all 2–class problems** on S are linearly separable and if $N > d+1$, the **number of homogeneously linearly separable problems** is

$$2 \sum_{i=0}^{d} \binom{N-1}{i}$$

Counting Linearly Separable Problems

- Our current SVM classifiers will be useful if linearly separable 2–class problems are frequent enough
- It is relatively easy to show that for $N > d+1$

$$2 \sum_{i=0}^{d} \binom{N-1}{i} \le 2(d+1) \binom{N-1}{d} \le 2 \frac{d+1}{d!} N^d \lesssim N^d$$

- On the other hand, the total number of two–class problems over a sample of size N is 2^N
- And $\frac{N^d}{2^N} \to 0$ very fast when $N \to \infty$
- Since in many practical problems we will have $N \gg d$, essentially all such 2–class problems won't be linearly separable
- And our current SVMs will be useless on them

Linear SVMs for Non Linear Probems

- Q: What can we do?
- First step: make room for non linearly separable problems
- We no longer require perfect classification but **allow for error (slacks) in some patterns**
- We relax the previous requirement $y^p(w \cdot x^p + b) \geq 1$ to
$$y^p(w \cdot x^p + b) \geq 1 - \xi_p$$
where we impose a new constraint $\xi_p \geq 0$
- Notice that if $\xi_p \geq 1$, x^p will not be correctly classfied
- Thus, we allow for defective clasification but we also **penalize** it

L_k **Penalty SVMs**

- New primal problem: for $K \geq 1$ consider the cost function
$$\min \frac{1}{2}\|w\|^2 + \frac{C}{K}\sum \xi_p^K$$
now subject to $y^p(w \cdot x^p + b) \geq 1 - \xi_p$, $\xi_p \geq 0$
- Notice that if $C \to \infty$ we recover the previous slack-free approach
- Simplest choice $K = 2$: L_2 (i.e., square penalty) SVMs, that reduce to the previous set up
- Usual (and best) choice $K = 1$
  - We will concentrate on it

L_1 **SVMs**

- Primal problem
$$\min \frac{1}{2}\|w\|^2 + C\sum \xi_p$$
subject to $y^p(w \cdot x^p + b) \geq 1 - \xi_p$, $\xi_p \geq 0$
- The L_1 Lagrangian is then
$$\begin{aligned} L(w, b, \xi, \alpha, \beta) &= \frac{1}{2}\|w\|^2 + C\sum \xi_p - \\ &\quad \sum \alpha_p \left[y^p(w \cdot x^p + b) - 1 + \xi_p\right] - \sum \beta_p \xi_p \end{aligned}$$
with $\alpha_p, \beta_p \geq 0$

L_1 **SVM Lagrangian**

- Again we reorganize the L_1 Lagrangian as

$$\begin{aligned} L(w, b, \xi, \alpha, \beta) &= w \cdot \left(\frac{1}{2} w - \sum \alpha_p y^p \, x^p \right) + \\ & \sum \xi_p (C - \alpha_p - \beta_p) - b \sum \alpha_p y^p + \\ & \sum \alpha_p \end{aligned}$$

- The w and b partials yield as before $w = \sum \alpha_p y^p x^p$, $\sum \alpha_p y^p = 0$

The L_1 SVM Dual I

- From $\frac{\partial L}{\partial \xi_p} = C - \alpha_p - \beta_p = 0$ we see that

$$C = \alpha_p + \beta_p,$$

- Substituting things back into the Lagrangian we arrive at the L_1 dual function

$$\begin{aligned} \Theta(\alpha, \beta) &= \sum_p \alpha_p - \frac{1}{2} w \cdot \sum \alpha_p y^p x^p \\ &= \sum_p \alpha_p - \frac{1}{2} \alpha^\tau Q \alpha \end{aligned}$$

subject to $\sum_p \alpha_p y^p = 0, \alpha_p \geq 0, \beta_p \geq 0, \alpha_p + \beta_p = C$

The L_1 SVM Dual II

- In fact, we can drop β
  - Notice that, in fact, $\Theta(\alpha, \beta) = \Theta(\alpha)$
  - It is also clear that the constraints on α, β can be reduced to $0 \leq \alpha_p \leq C$
- Thus, we get essentially the same dual problem as before

$$\min_\alpha \; \frac{1}{2} \alpha^\tau Q \alpha - \sum_p \alpha_p$$

subject to $\sum \alpha_p y^p = 0, 0 \leq \alpha^p \leq C, 1 \leq p \leq N$

  - Notice that if $C \to \infty$ we recover the penalty free SVM
  - And here also $w^* = \sum \alpha_p^* y^p x^p$ for the optimal w^*
- We can solve it either a la SMO or by coordinate descent

KKT Conditions for L_1 SVMs

- The complementary slackness conditions are now

$$\begin{aligned} \alpha_p^* \left[y^p (w^* \cdot x^p + b^*) - 1 + \xi_p^* \right] &= 0 \\ \beta_p^* \xi_p^* &= 0 \end{aligned}$$

- Now, if $\xi_p^* > 0$, then $\beta_p^* = 0$ and, therefore, $\alpha_p^* = C$
  - We say that such an x^p is **at bound**
- Also, if $0 < \alpha_p^* < C$, then $\beta_p^* > 0$ and $\xi_p^* = 0$
  - Thus, if $0 < \alpha_p^* < C$, $y^p(w^* \cdot x^p + b^*) = 1$ and x^p lies in one of the support hyperplanes
  - We deduce b^* as before and, if needed, derive ξ_p^* as

$$\xi_p^* = 1 - y^p(w^* \cdot x^p + b^*) \text{ with } \alpha_p^* = C$$

Solving L_1 SVMs

- Dropping the b term we can also apply here Dual Coordinate Descent
- The update just becomes

$$\alpha'_p = \min\left\{C, \max\left\{0, \alpha_p - \frac{(Q\alpha)_p - 1}{Q_{pp}}\right\}\right\}$$

- We get a fast algorithm for large dimension problems but perhaps less precise because of the forced homogeneity
- The SMO algorithm for the slack–free case also extends easily here
  - We just have to take care of maintaining $0 \le \alpha_p \le C$

The Cost of SMO

- SMO can be applied to L_1 SVMs straightforwardly
  - We start with $\alpha^0 = 0$ for which trivially $\sum y^p \alpha_p^0 = 0$
  - At step t select $L_t = \arg\min_p y^p \nabla\Theta(\alpha^t)_p$, $\quad U_t = \arg\min_q y^q \nabla\Theta(\alpha^t)_q$
  - Update $\alpha^{t+1} = \alpha^t + \rho_t d^t$ with $d^t = y^{L_t} e_{L_t} - y^{U_t} e_{U_t}$ and clip it if needed to have $0 \le \alpha_{L_t}^{t+1}, \alpha_{U_t}^{t+1} \le C$
  - And iterate until a KKT–related stopping condition is met
- The cost of SMO is at least $\Omega(N^2)$ for
  - Each iteration has a $O(N)$ cost of selecting L, U and updating $\nabla\Theta(\alpha)$
  - At least $\Omega(N)$ iterations are needed for the number of SVs is usually $\Theta(N)$
- And the final number of iterations grows usually with C, so to train SVMs is costly: at least $\Omega(N^2)$

Good Option, But ...

- L_1 SVMs are (relatively) **sparse**, i.e., have (hopefully) few non–zero multipliers
- The bound $\alpha_p^* = C$ for $\xi_p^* > 0$ limits the effect of not correctly classified patterns
- And usually L_1 SVMs are much better than, say, L_2 SVMs
- But still they are linear ...
- We must thus somehow **introduce some kind of non–linear processing for SVMs to be truly effective**

3.2 The Kernel Trick

Back to Cover

- Recall that the number $L(N,D)$ of linearly separable dichotomies is

$$L(N,D) = \left\{ \begin{array}{cc} 2^N & \text{if } N \leq D+1 \\ 2\sum_{i=0}^{D} \binom{N-1}{i} & \text{if } N \geq D+1 \end{array} \right\}$$

- Notice that for D fixed, $\frac{L(N,D)}{2^N} \to 0$ as $N \to 0$
- In practice $N \gg D$ and the fraction of separable dichotomies will be very small
- But if $N \ll D$, **all dichotomies will be linearly separable**

The Kernel Trick

- Idea: (non linearly) augment pattern dimension going from $x \in \mathbf{R}^d$ to $\Phi(x) \in \mathbf{R}^D$ with $D \gg d$
- First option: do it explicitly as in $\Phi(x) = (x_1, \ldots, x_i, \ldots, x_i x_j, \ldots, x_i x_j x_k, \ldots)$
- Too cumbersome, so try to do it **implicitly**
- Observation: in SVMs we only need to compute dot products $x \cdot x'$
  - And the same is true for the SMO algorithm
- Thus we can work **implicitly** with extensions $\Phi(x)$ provided it is easy to compute $\Phi(x) \cdot \Phi(x')$
- Simplest case: $\Phi(x) \cdot \Phi(x') = k(x, x')$ for an appropriate **kernel** k

Example: Polynomial Kernels

- A simple option is to work with **polynomial** kernels $k(x, x') = (1 + x \cdot x')^m$
- Assume $m = 2$, $x = (x_1, x_2)$, $x' = (x'_1, x'_2)$; then

$$\begin{aligned} k(x,x') &= (1 + x_1 x'_1 + x_2 x'_2)^2 \\ &= 1 + 2x_1 x'_1 + 2x_2 x'_2 + x_1^2 (x'_1)^2 + \\ &\quad x_2^2 (x'_2)^2 + 2x_1 x_2 x'_1 x'_2 \\ &= \Phi(x) \cdot \Phi(x') \end{aligned}$$

with

$$\Phi(x_1, x_2) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

Positive Definite Kernels

- In fact, if the kernel is **positive definite** we can diagonalize it as

$$k(x, x') = \sum_0^\infty \lambda_k \varphi_k(x) \varphi_k(x')$$

  with $\lambda_k \geq 0$ and the (possibly infinitely many) $\{\varphi_k(x)\}$ orthonormal

- Defining then

$$\Phi(x) = (\sqrt{\lambda_0}\varphi_0(x), \sqrt{\lambda_1}\varphi_1(x), \ldots \ldots)$$

  we have $k(x, x') = \Phi(x) \cdot \Phi(x')$

- The dot product matrix Q is now the **kernel matrix** $Q_{p,q} = k(x^p, x^q)$

The Gaussian Kernel

- If we use the Gaussian kernel $k(x, x') = e^{-\gamma \|x - x'\|^2}$, $\Phi(x)$ has infinite dimension
  - So Cover's theorem no longer limits things
  - And overfitting is guaranteed unless we renounce perfect separability
  - And practical SVMs are (almost) always built using Gaussian kernels
- Thus we have to get effective SVMs that avoid overfit using a powerful kernel but also
  - Adequately adjusting the **penalty constant** C
  - And also the Gaussian **kernel's width** γ
- Notice that at each SV x^p the Gaussian kernel $e^{-\gamma \|x - x^p\|^2}$ defines an "influence region" around x^p
  - Thus we can see Gaussian SVC as a more flexible and effective way to exploit SV's neighbors

Selecting C for SVMs

- In all SVM models we have to choose an adequate C which acts as a regularization parameter:
  - Small C allow large slacks and a possible underfit
  - But large C imply very small slacks and possible overfit
- Notice that we can write the primal cost function as

$$\frac{1}{N} \sum \xi_p + \frac{1}{2} \frac{1}{CN} \|w\|^2$$

- Thus $\frac{1}{CN}$ behaves similarly to α in Ridge regression
- One usually explores values $10^k, \ -K_L \leq k \leq K_R$
  - Typical values are $K_L = 0$, i.e., $C_L = 1$, and $K_R = 3$ or 4, i.e., $C_R = 1,000$ or $10,000$

Selecting γ for Gaussian SVMs

- When working with Gaussian kernels, the features x_i are usually scaled to a $[0, 1]$ range
- Then $|x_i - x'_i| \leq 1$ and if d is pattern dimension

$$\|x - x'\|^2 = \sum_1^d (x_i - x'_i)^2 \lesssim d$$

- This suggests to explore γ values of the form

$$\frac{2^k}{d}, \ -K \leq k \leq K$$

- Large k values result in very sharp Gaussians
  - We may end up with a Gaussian for each sample x^p and, hence, overfit
- Small k values result in quite flat, nearly constant Gaussians
  - No x^p is relevant and, hence, underfit is quite likely

Linear Kernels?

- Recall that we use kernels to enlarge pattern dimension
  - We get better models but costlier training
  - And working with large datasets may become impractical
- We may try to avoid them if pattern dimension is already large and just use linear SVMs
- This is the approach followed by the LIBLINEAR package which offers
  - Dual-based solvers using coordinate descent methods
  - Primal-based solvers using Newton–type methods
- The constant term b is usually not considered, so data should be centered before training
- Only C has to be hyperparameterized

Other Things

- SVMs do not have an underlying probability model
  - Label prediction is the primary output
- The LIBSVM and its Scikit–learn wrapper can give probability predictions using an ad–hoc model
- SVM classification is intrinsically two-class
  - Multiclass problems are usually handled using a one–versus–rest (OVR) approach

- ν–SVMs (available in LIBSVM) can also be used for classification (and regression) usually with very similar results

Takeaways on Non Linear SVMs I

- The L_1 primal problem is

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum \xi_p$$

  s.t. $y^p(w \cdot x^p + b) \geq 1 - \xi_p$, $\xi_p \geq 0$, $1 \leq p \leq N$

- For $\alpha_p, \beta_p \geq 0$ the new Lagrangian is

$$\begin{aligned} L(w, b, \xi, \alpha, \beta) &= \frac{1}{2}\|w\|^2 - \sum_p \alpha_p \left(y^p(w \cdot x^p + b) - 1 + \xi_p\right) \\ &\quad - \sum \beta_p \xi_p \end{aligned}$$

- And for $\mathcal{C} = \{\alpha : 0 \leq \alpha_p \leq C, \sum \alpha_p y^p = 0\}$, the L_1 dual problem is

$$\max_{\alpha_p \in \mathcal{C}} \; \Theta(\alpha) = \sum_p \alpha_p - \frac{1}{2}\alpha^\tau Q \alpha$$

Takeaways on Non Linear SVMs II

- The new dual coincides essentially with the linear dual and can also be solved by the SMO algorithm, with a cost $\Omega(N^2)$
- The KKT conditions are again used to obtain w^* and b^*
- For the optimal w^* we have $w^* = \sum_{SVs} \alpha_p^* y^p x^p$
- If $0 < \alpha^* < C$ we have $b^* = y^p - w^* \cdot x^p$
- And if $\xi_p^* > 0$, $\alpha_p^* = C$
- All the dot products can be replaced by kernel operations $k(x^p, x^q)$
- Two hyperparameters appear: the penalty C and (if used) the Gaussian kernel width γ

4 Support Vector Regression

Back to the Primal Classification

- The slack ξ of a pattern x, y can be written as

$$\xi = \max\{0, -(y(w \cdot x + b) - 1)\} = h(y(w \cdot x + b) - 1)$$

  where $h(z) = \max\{0, -z\}$ is the **hinge loss**

- We can thus write the linear SVC primal problem as

$$\text{arg min}_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_p h(y^p(w \cdot x^p + b) - 1) \equiv$$
$$\text{arg min}_{w,b} \quad \sum_p h(y^p(w \cdot x^p + b) - 1) + \frac{1}{2C}\|w\|^2$$

- The hinge loss is not differentiable only at $z = 0$
- But this is also the case of the ReLUs in DNNs ...

Support Vector Regression

- In SV regression (SVR) we try to solve another regularized problem

$$\min_{w,b} f(w,b) = \sum_p \left[y^p - (w \cdot x^p + b)\right]_\epsilon + \frac{\lambda}{2}\|w\|^2$$

or, equivalently,

$$\min_{w,b} \frac{1}{N}\sum_p \left[y^p - (w \cdot x^p + b)\right]_\epsilon + \frac{1}{2}\frac{\lambda}{N}\|w\|^2$$

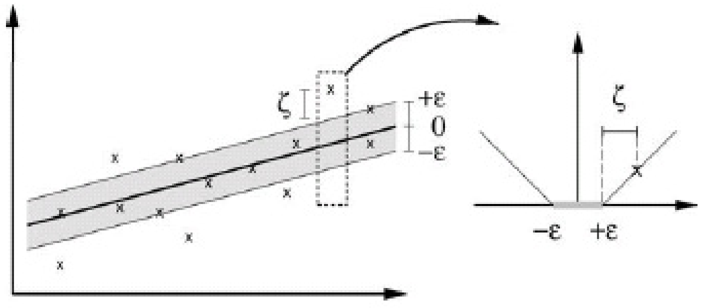
using the ϵ–**insensitive** loss

$$[z]_\epsilon = \max(0, |z| - \epsilon)$$

- Notice we penalize an error $|y^p - f(x^p, w, b)|$ only if it is $> \epsilon$

The ϵ Error Tube

- Therefore, we **do not penalize errors of predictions that fall inside an ϵ–wide tube around the true function**



SVR as a Constrained Problem

- We have $f(w,b) = \ell_\epsilon(w,b) + \frac{1}{2}\|w\|^2$
  - f is convex but $\ell_\epsilon = \sum_p \left[y^p - (w \cdot x^p + b)\right]_\epsilon$ is not smooth

  - Direct minimization of $f(w,b)$ may be difficult, so we rewrite the unconstrained SVR problem as a constrained one

- If $C = 1/\lambda$, we rewrite f as

$$f(w,b,\xi,\eta) = \frac{1}{2}\|w\|^2 + C\sum_p (\xi_p + \eta_p)$$

  with the following constraints on the errors $w \cdot x^p + b - y^p$:

$$\begin{aligned} -\xi_p - \epsilon &\le w \cdot x^p + b - y^p, \quad (y^p \text{ is above the model}) \\ \eta_p + \epsilon &\ge w \cdot x^p + b - y^p, \quad (y^p \text{ is below the model}) \\ \xi_p, \eta_p &\ge 0 \end{aligned}$$

The SVR Lagrangian

- This leads to the Lagrangian

$$\begin{aligned} L(w,b,\xi,\eta,\alpha,\beta,\gamma,\delta) &= \frac{1}{2}\|w\|^2 + C\sum_p (\xi_p + \eta_p) \\ &- \sum_p \alpha_p (w \cdot x^p + b - y^p + \xi_p + \epsilon) \\ &+ \sum_q \beta_q (w \cdot x^q + b - y^q - \eta_q - \epsilon) - \sum_p \gamma_p \xi_p - \sum_q \delta_q \eta_q \end{aligned}$$

  with $\alpha_p, \beta_q, \gamma_r, \delta_s$ all ≥ 0

- Setting $\Theta(\alpha,\beta,\gamma,\delta) = \min_{w,b,\xi,\eta} L(w,b,\xi,\eta,\alpha,\beta,\gamma,\delta)$, we have by construction

$$\Theta(\alpha,\beta,\gamma,\delta) \le L(w,b,\xi,\eta,\alpha,\beta,\gamma,\delta) \le f(w,b,\xi,\eta)$$

SVR's Dual Problem

- We derive the dual function solving the equations

$$\frac{\partial L}{\partial w_i} = 0, \ \frac{\partial L}{\partial b} = 0, \ \frac{\partial L}{\partial \xi_p} = 0, \ \frac{\partial L}{\partial \eta_p} = 0$$

- Plugging the results back in L and working things out, the minus dual function that we write again as Θ becomes

$$\begin{aligned} \Theta(\alpha,\beta,\gamma,\delta) \quad = \quad & \frac{1}{2}\sum_{p,q} (\alpha_p - \beta_p)(\alpha_q - \beta_q) x^p \cdot x^q + \\ & \epsilon \sum_p (\alpha_p + \beta_p) - \sum_p y^p (\alpha_p - \beta_p) \end{aligned}$$

- γ and δ drop out of Θ and also from the constraints, and the dual problem becomes

$$\min_{\alpha,\beta} \Theta(\alpha,\beta) \text{ subject to } 0 \le \alpha_p, \beta_q \le C, \ \sum \alpha_p = \sum \beta_q$$

Solving the SVR Dual Problem

- It can be shown that if $(w^*, b^*, \xi^*, \eta^*)$ and (α^*, β^*) are primal and dual optima respectively, then the dual gap is 0, i.e., $f(w^*, b^*, \xi^*, \eta^*) = \Theta(\alpha^*, \beta^*)$
- Things are a little bit easier if we remove the (trickier) constraint $\sum \alpha_p = \sum \beta_q$ by dropping b, i.e., assuming a homogeneous model $w \cdot x$
  - Then we only have box constraints and we can simply apply projected gradient descent
  - But risk ending in a worse model (unless we center everything)
- But the dual problem is also easy to solve, for which a simple variant of the SMO algorithm is used

KKT Conditions

- We deduce the complementary slackness KKT conditions from

$$f(w^*, b^*, \xi^*, \eta^*) = L(w^*, b^*, \xi^*, \eta^*, \alpha^*, \beta^*, \gamma^*, \delta^*) = \Theta(\alpha^*, \beta^*)$$

  namely

$$\begin{aligned} 0 &= \alpha_p^*(w^* \cdot x^p + b^* - y^p + \xi_p^* + \epsilon); \\ 0 &= \beta_q^*(w^* \cdot x^q + b^* - y^q - \eta_q^* - \epsilon); \\ 0 &= (C - \alpha_p^*)\xi_p^*; \quad 0 = (C - \beta_q^*)\eta_q^* \end{aligned}$$

  - Thus, if $0 < \alpha_p^* < C$, we have $\xi_p^* = 0$ and $y^p - (w^* \cdot x^p + b^*) = \epsilon$ (top of the tube)
  - Similarly, if $0 < \beta_q^* < C$, we have $\eta_q^* = 0$ and $y^q - (w^* \cdot x^q + b^*) = -\epsilon$ (bottom of the tube)
  - Either one can be used to derive b^* once w^* is known

Support Vectors

- The corresponding x^p, x^q are called **support vectors**
  - Now they define the ϵ–tube around the true model
- Also $\xi_p^* > 0$ implies $\alpha_p^* = C$ and $\eta_q^* > 0$ implies $\beta_q^* = C$
- The optimal w^* is

$$w^* = \sum (\alpha_p^* - \beta_p^*) x^p,$$

  with $\alpha_p^* \beta_p^* = 0$
- Notice that a given x^p can only verify one of the conditions

$$w^* \cdot x^q + b^* - y^q = \epsilon, \;\; w^* \cdot x^q + b^* - y^q = -\epsilon$$

The Kernel Trick for SVR I

- Again, stating and solving the the dual problem only requires computing dot products
- Also, the model applied to a new x is

$$f(x) = b^* + \sum (\alpha_p^* - \beta_p^*) x^p \cdot x$$

- Thus, the kernel trick can be used again to project the original patterns x into larger dimensional patterns $\Phi(x)$

The Kernel Trick for SVR II

- Again, we do not deal with the $\Phi(x)$ but just work with $\Phi(x) \cdot \Phi(x') = k(x, x')$
- The model is applied as

$$\begin{aligned} b^* + w^* \cdot \Phi(x) &= b^* + \sum (\alpha_p^* - \beta_p^*) \Phi(x^p) \cdot \Phi(x) \\ &= b^* + \sum (\alpha_p^* - \beta_p^*) k(x^p, x) \end{aligned}$$

- If we use a Gaussian kernel, the model becomes

$$f(x; w^*, b^*) = b^* + \sum (\alpha_p^* - \beta_p^*) \ e^{-\gamma \|x^p - x\|^2}$$

  i.e., a sum of Gaussians centered at the x^p

Hyperparameterizing C, γ and ϵ

- C and γ are explored as in SV classification
- In a reasonable model ϵ shouldn't be larger than σ_y
- We can try ϵ values of the form

$$2^k \sigma_y, \ \ -K \leq k \leq -1$$

- But we have to explore three parameters which is going to be quite costly
- The stopping tolerance is also somewhat tricky as it depends on gradient properties
  - The default 10^{-3} should be OK on medium size problems
- Some guidelines can be found on LIBSVM home pages

Overfitting and Underfitting

- As in SVC, large C and γ will result in overfit unless ϵ is large
- A large C forces slacks to be near 0 and thus perfect training fit
  - This is parallel to what happened in Ridge regression, since $\frac{1}{CN}$ behaves as α
- Large γ result in sharp Gaussians

- On the other hand, models with small C and γ will likely underfit
- Large ϵ models will usually underfit
  - At the extreme there will be no slacks and we are likely to end in a near constant model
- On the other hand, a very small ϵ will force 0 slacks and possible overfit
- But the joint effects of C, γ and ϵ may change the preceding observations

Other scikit-learn SVM Things

- `tol`: SVR training stops when a KKT defined value becomes smaller
  - It is not related to the value of the criterion function
- `shrinking`: tells LIBSVM to work after some point only with likely SV candidates
  - The SMO working set is reduced and iterations are faster
  - But savings may be erased by having to compute the entire gradient at some later point
- `cache_size`: size in MB of the kernel cache
  - If enough, previous kernel operations are cached and do not have to be recomputed

Takeaways on SVR I

- The primal SVR problem can be written as a regularized loss function

$$\min_{w,b} f(w,b) = \sum_p \left[y^p - (w \cdot x^p + b)\right]_\epsilon + \frac{\lambda}{2}\|w\|^2$$

- If $\mathcal{C} = \{\alpha, \beta : 0 \le \alpha_p, \beta_p \le C, \sum \alpha_p = \sum \beta_p\}$, the dual problem is now

$$\begin{aligned}\max_{\mathcal{C}} \;\; \Theta(\alpha,\beta) \;\; &= \;\; \frac{1}{2}\sum_{p,q}(\alpha_p - \beta_p)(\alpha_q - \beta_q)x^p \cdot x^q + \\ &\quad\;\; \epsilon\sum_p(\alpha_p + \beta_p) - \sum_p y^p(\alpha_p - \beta_p)\end{aligned}$$

Takeaways on SVR II

- A variant of SMO can again be used, with a cost $\Omega(N^2)$
- KKT conditions are again used to obtain w^* and b^* from α^*, β^*
- And again SVs, i.e., vectors x^p for which $\alpha_p^* > 0$ or $\beta_p^* > 0$ define the SVR model
- Using a Gaussian kernel we arrive at a final model

$$f(x; w^*, b^*) = b^* + \sum(\alpha_p^* - \beta_p^*) \;\; e^{-\gamma\|x^p - x\|^2}$$

- Two hyperparameters appear: the penalty C and the ϵ tube width
- Plus the width γ if we use a Gaussian kernel