TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

3.3 How to Solve Convex OPs and SVM

Machine Learning 1: Foundations

Marius Kloft *(TUK)*

1. Convex Optimization Problems (OPs)

2. SVM is a Convex OP

3. How to Solve Convex OPs and SVM

The Good News

Convex OPs are **easy to solve**!

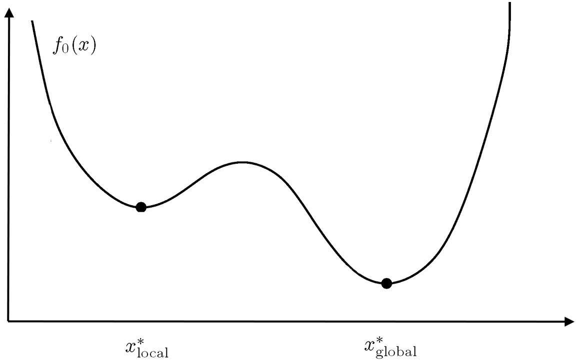
Why?

Global vs. Local Optimality

Global optimal point

$\mathbf{x}^*_{\text{global}}$ is a **globally optimal point** if

$$\mathbf{x}^*_{\text{global}} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \quad f_0(\mathbf{x}) \qquad \text{s.t.} \qquad \begin{aligned} f_i(\mathbf{x}) &\leq 0, \quad i = 1, \ldots, n \\ g_j(\mathbf{x}) &= 0, \quad j = 1, \ldots, m \end{aligned}$$

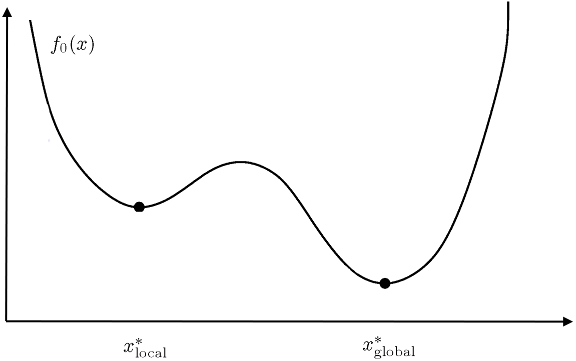


Global vs. Local Optimality

Locally optimal point

$\mathbf{x}^*_{\text{local}}$ is a **locally optimal point** if for some $R > 0$:

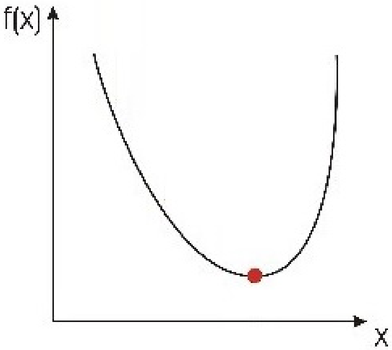
$$\mathbf{x}^*_{\text{local}} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \quad f_0(\mathbf{x}) \qquad \text{s.t.} \qquad \begin{aligned} f_i(\mathbf{x}) &\leq 0, \quad i = 1, \ldots, n \\ g_j(\mathbf{x}) &= 0, \quad j = 1, \ldots, m \\ \|\mathbf{x} - \mathbf{x}^*_{\text{local}}\| &\leq R \end{aligned}$$



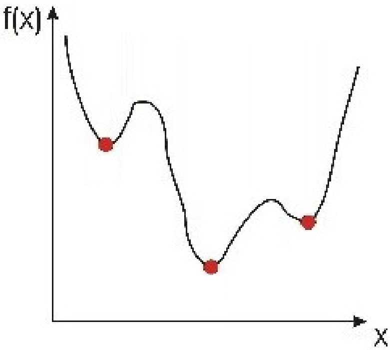
Why are Convex OPs **easy** to Solve?

Theorem

Every **locally** optimal point of a convex optimization problem is also **globally** optimal.



convex



non-convex

How can we exploit this property for solving the OP?

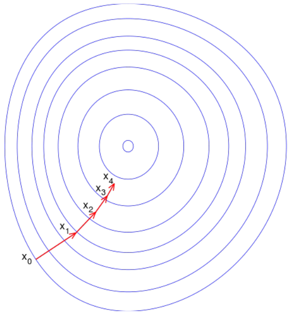
Gradient Descent

For unconstrained problems, we can iteratively move into **direction of steepest descent** (negative gradient direction) of the objective function:

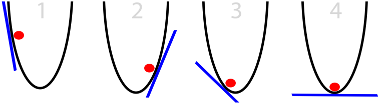
Gradient descent algorithm

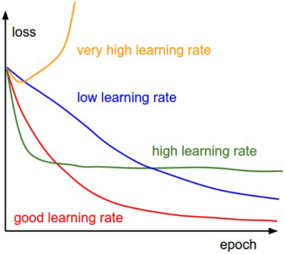
1: initialize $\mathbf{x}_0$ (e.g., randomly)
2: **for** $t = 1 : T$ **do**
3: $\quad \mathbf{x}_{t+1} := \mathbf{x}_t - \lambda_t \nabla f_0(\mathbf{x}_t)$
4: **end for**

λ_t is called **step size** or **learning rate**



An Adequate Learning Rate can be Crucial...





- A typical choice: $\lambda_t := \frac{1}{t}$

Gradient Descent Convergences

Theorem (Bertsekas, Prop. 1.2.4 & 1.3.3)

Let $f_0 : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary (possibly non-convex) objective function. Then, under some assumptions,[1] we have:

1. Gradient descent converges.[2]
2. For ideal choice of the learning rate, the convergence rate is at least as good as:

$$f_0(\mathbf{x}_t) - f_0(\mathbf{x}^*_{\text{local}}) \ \leq \ O(1/t).$$

[1] The theorem assumes Lipschitz-continues gradients with a uniformly bounded Lipschitz constant: $\exists L : \|\nabla f(\mathbf{x}) - \nabla f(\tilde{\mathbf{x}})\| \leq L \|\mathbf{x} - \tilde{\mathbf{x}}\|$. Convergence is guaranteed for all learning rate schedules satisfying $\sum_{t=1}^{\infty} \lambda_t = \infty$ and $\lambda_t \xrightarrow[t\to\infty]{} 0$, but with varying rates of convergence. The favorable $O(1/t)$ rate is achieved using the minimzation rule: $\lambda_t := \arg\min_\lambda f_0(\mathbf{x}_t - \lambda \nabla f_0(\mathbf{x}_t))$.

[2] More precisely, it converges to a stationary point (that is, a point where the gradient is zero, i.e., either a minimum, a maximum, or a sattle point). However, machine-learning practice (e.g., in deep learning) has shown that this is usually a minimum, so we are good. :)

Let's Try Applying **Gradient Descent** to the SVM OP:

$$\min_{b\in\mathbb{R},\mathbf{w}\in\mathbb{R}^d,\boldsymbol{\xi}\in\mathbb{R}^n} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \qquad \text{(SVM)}$$

$$\text{s.t.} \quad \forall i: \; 1-\xi_i - y_i(\mathbf{w}^\top\mathbf{x}_i + b) \leq 0, \quad -\xi_i \leq 0$$

What could be a problem? The constraints!

We need to get rid of the constraints...

Proposition

The linear SVM can be equivalently re-written as follows:

Unconstrained linear soft-margin SVM

$$\min_{b\in\mathbb{R},\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\right)$$

Let's Try Again Applying **Gradient Descent** to SVM

The new, unconstrained objective is :

$$f_0(\mathbf{w}, b) \;=\; \frac{1}{2}\|\mathbf{w}\|^2 \;+\; C\sum_{i=1}^{n}\max\Big(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\Big).$$

What could be another problem? f_0 is not differentiable!

We have two options to address this problem:

- Option 1: We can consider the **subgradient**,

$$\nabla \max\left(0, 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\right)$$
$$:= \begin{cases} \nabla(1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b) & \text{if} \quad y_i(\mathbf{w}^\top\mathbf{x}_i + \mathbf{b}) < 1 \\ 0 & \text{elsewise} \end{cases} ,$$

  and then use **subgradient descent**: that is, gradient descent but using the subgradient in place of the gradient.

Logistic Regression

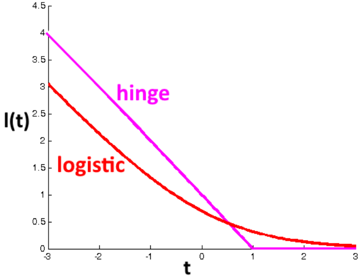
Option 2:

- Replace 'hinge' function

$$f(x) = \max(0, 1 - x)$$

appearing in SVM by its smooth approximation, the 'logistic' function:

$$l(x) = \ln(1 + \exp(-x)).$$



This results in:

Logistic Regression (LR)

$$\min_{b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d} \quad \frac{1}{2} \|\mathbf{w}\|^2 \; + \; C \sum_{i=1}^{n} \ln \Big(1 + \exp(-y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \Big)$$

LR is differentiable, so we can solve it using gradient descent.

But (Sub)gradient Descent of SVM / LR is Too Slow!

- Each evaluation of the (sub)gradient involves a FOR loop over all data points:
  FOR all $i = 1, \ldots, n$
- This is costly (n large for big data)!

But evaluating the full FOR loop can be unnecessary:

- Imagine all data points were the same (duplicates), thus:
- $\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right) = n \cdot \max\left(0, 1 - y_1(\mathbf{w}^\top \mathbf{x}_1 + b)\right)$

Real data is usually less extreme (not exact duplicates), but yet contains lots of redundancy
$\Rightarrow$ unnecessary to evaluate the full (sub)gradient in every iteration

Stochastic Gradient Descent (SGD) is Much Faster...

Stochastic subgradient descent algorithm (SVM)

1: initialize $(b, \mathbf{w})_0$ (e.g., randomly)
2: **for** $t = 1 : T$ **do**
3: Randomly select B many data points
4: Denote their indexes by $I \subset \{1, \ldots, n\}$ (i.e.,$|I| = B$)
5: Update $(b_{t+1}, \mathbf{w}_{t+1}) :=$

$$(b_t, \mathbf{w}_t) - \lambda_t \nabla \left(\frac{1}{2} \|\mathbf{w}_t\|^2 + \frac{Cn}{B} \sum_{i \in I} \max\left(0, 1 - y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t)\right) \right)$$

6: **end for**

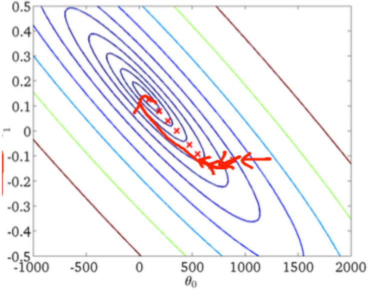
The **batch size** $B \in [1, n]$ needs to be chosen a priori.

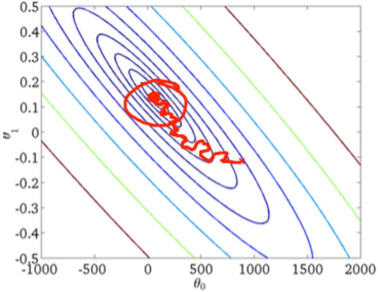
For LR simply replace in Line 5 the term

$$\max\left(0, 1 - y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t)\right) \quad \text{by} \quad \log\left(1 + \exp(-y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t))\right).$$

The classic SGD algorithm uses just a single data point per iteration ($B = 1$). Nowadays more common in ML: **mini-batch SGD**, where we use an intermediate value, such as $B = 100$.

What is the Difference Between Gradient Descent and Stochastic Gradient Descent?





SGD Convergences

Theorem (e.g., Bottou et al., 2018)

Consider SGD using the learning rate $\lambda_t := 1/t$. Then, under mild assumptions, SGD converges with high probability to a stationary point[1] with rate:

$$f_0(\mathbf{x}_t) - f_0(\mathbf{x}^*_{\text{local}}) \;\leq\; O(1/t)$$

Remark: this holds also for non-convex f_0

[1] In ML practice, this is usually a local minimum.

Software

An extremely fast implementation of SGD applied to the linear (soft-margin) SVM and logistic regression is contained in **Vowpal Wabbit**[1]

- Industry standard in ad prediction

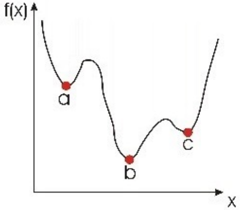
Discussion:

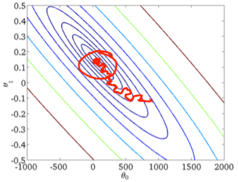
- VW trades speed for accuracy, so use it only when really necessary (=big data)
- Otherwise use LIBLINEAR, a more accurate and still relatively fast solver for SVM and LR[2]

[1] https://github.com/VowpalWabbit/vowpal_wabbit/wiki
[2] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

Conclusion

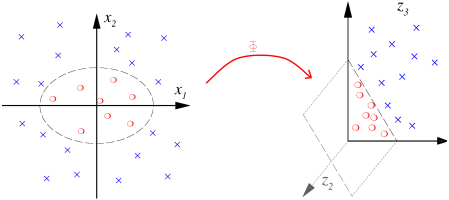
- Convex optimization problem (OP):
  - optimize convex function over convex set
- Why convexity?
  - cannot get stuck in local optimum
- SVM can be formulated as convex OP
  - unconstrained formulation of SVM
  - gradient descent slow for SVM
  - solution: stochastic gradient descent (Vowpal Wabbit)





Next Week

- Non-linear SVM
- Kernel methods



Refs I

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, Pegasos: Primal estimated sub-gradient solver for svm, *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

L. Bottou, F. E. Curtis, and J. Nocedal, Optimization methods for large-scale machine learning, *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.