

Optimization for Machine Learning

CS-439

Lecture 10: Duality, Gradient-free, and Applications

Martin Jaggi

EPFL – github.com/epfml/OptML_course

May 8, 2020

Chapter X.1

Duality

Duality

Given a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, define its **conjugate** $f^* : \mathbb{R}^d \rightarrow \mathbb{R}$ as

$$f^*(\mathbf{y}) := \max_{\mathbf{x}} \mathbf{x}^\top \mathbf{y} - f(\mathbf{x})$$

a.k.a. Legendre transform or Fenchel conjugate function.

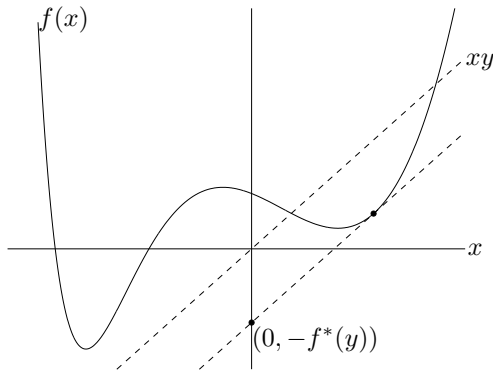


Figure: maximum gap between linear function $\mathbf{x}^\top \mathbf{y}$ and $f(\mathbf{x})$.

figure by Boyd & Vandenberghe

Properties

- ▶ f^* is always convex, even if f is not.

Proof: point-wise maximum of convex (affine) functions in \mathbf{y} .

- ▶ **Fenchel's inequality**: for any \mathbf{x}, \mathbf{y} ,

$$f(\mathbf{x}) + f^*(\mathbf{y}) \geq \mathbf{x}^\top \mathbf{y}$$

- ▶ Hence conjugate of conjugate f^{**} satisfies $f^{**} \leq f$.
- ▶ If f is closed and convex, then $f^{**} = f$.
- ▶ If f is closed and convex, then for any \mathbf{x}, \mathbf{y} ,

$$\begin{aligned} \mathbf{y} \in \partial f(\mathbf{x}) &\Leftrightarrow \mathbf{x} \in \partial f^*(\mathbf{y}) \\ &\Leftrightarrow f(\mathbf{x}) + f^*(\mathbf{y}) = \mathbf{x}^\top \mathbf{y} \end{aligned}$$

Exercise!

- ▶ Separable functions: If $f(\mathbf{u}, \mathbf{v}) = f_1(\mathbf{u}) + f_2(\mathbf{v})$, then

$$f^*(\mathbf{w}, \mathbf{z}) = f_1^*(\mathbf{w}) + f_2^*(\mathbf{z})$$

Examples

- Recall: **Indicator function** of a set $C \subseteq \mathbb{R}^d$ is

$$\iota_C(\mathbf{x}) := \begin{cases} 0 & \mathbf{x} \in C, \\ +\infty & \text{otherwise.} \end{cases}$$

If $f(\mathbf{x}) = \iota_C(\mathbf{x})$, then its conjugate is

$$f^*(\mathbf{y}) = \max_{\mathbf{x} \in C} \mathbf{y}^\top \mathbf{x}$$

called the **support function** of C .

- **Norm:** if $f(\mathbf{x}) = \|\mathbf{x}\|$, then its conjugate is

$$f^*(\mathbf{y}) = \iota_{\{\mathbf{z}: \|\mathbf{z}\|_* \leq 1\}}(\mathbf{y})$$

(i.e. indicator of the dual norm ball) Note: The **dual norm** of $\|\cdot\|$ is defined as $\|\mathbf{y}\|_* := \max_{\|\mathbf{x}\| \leq 1} \mathbf{y}^\top \mathbf{x}$. E.g. $\|\cdot\|_1 \leftrightarrow \|\cdot\|_\infty$.

Examples, cont

Generalized linear models

$$\min_{\mathbf{x} \in \mathbb{R}^d} f(A\mathbf{x}) + g(\mathbf{x})$$

reformulate

$$\min_{\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) + g(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{w} = A\mathbf{x}$$

Lagrange dual function

$$\begin{aligned} \mathcal{L}(\mathbf{u}) &:= \min_{\mathbf{x} \in \mathbb{R}^d, \mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) + g(\mathbf{x}) + \mathbf{u}^\top (\mathbf{w} - A\mathbf{x}) \\ &= -f^*(\mathbf{u}) - g^*(-A^\top \mathbf{u}) \end{aligned}$$

Dual problem

$$\max_{\mathbf{u} \in \mathbb{R}^n} [\mathcal{L}(\mathbf{u}) = -f^*(\mathbf{u}) - g^*(-A^\top \mathbf{u})].$$

Examples, cont

Lasso

$$\min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda \|\mathbf{x}\|_1$$

is an example, for $f(\mathbf{w}) := \frac{1}{2} \|\mathbf{w} - \mathbf{b}\|^2$ and $g(\mathbf{x}) := \lambda \|\mathbf{x}\|_1$.

Can compute $f^*(\mathbf{u}) = \frac{1}{2} \|\mathbf{b}\|^2 - \frac{1}{2} \|\mathbf{b} - \mathbf{u}\|^2$
and $g^*(\mathbf{v}) = \iota_{\{\mathbf{z}: \|\mathbf{z}\|_\infty \leq 1\}}(\mathbf{v}/\lambda)$,

so that the dual problem is

$$\max_{\mathbf{u} \in \mathbb{R}^n} -f^*(\mathbf{u}) - g^*(-A^\top \mathbf{u}).$$

$$\Leftrightarrow \max_{\mathbf{u} \in \mathbb{R}^n} -\frac{1}{2} \|\mathbf{b}\|^2 + \frac{1}{2} \|\mathbf{b} - \mathbf{u}\|^2 \quad \text{s.t.} \quad \|-A^\top \mathbf{u}/\lambda\|_\infty \leq 1.$$

$$\Leftrightarrow \min_{\mathbf{u} \in \mathbb{R}^n} \|\mathbf{b} - \mathbf{u}\|^2 \quad \text{s.t.} \quad \|A^\top \mathbf{u}\|_\infty \leq \lambda.$$

Why Duality?

Similarly for least squares, ridge regression, SVM, logistic regression, elastic net, etc.

Advantages:

- Duality gap gives a **certificate** of current optimization quality

$$\begin{aligned} & f(A\bar{\mathbf{x}}) + g(\bar{\mathbf{x}}) \\ & \geq \min_{\mathbf{x} \in \mathbb{R}^d} f(A\mathbf{x}) + g(\mathbf{x}) \\ & \geq \\ & \max_{\mathbf{u} \in \mathbb{R}^n} -f^*(\mathbf{u}) - g^*(-A^\top \mathbf{u}) \\ & \geq -f^*(\bar{\mathbf{u}}) - g^*(-A^\top \bar{\mathbf{u}}) \end{aligned}$$

for any $\bar{\mathbf{x}}, \bar{\mathbf{u}}$.

- Stopping criterion
- Dual can in some cases be easier to solve

Chapter X.2

Zero-Order Optimization

\Leftrightarrow **Derivative-Free** ..

\Leftrightarrow **Blackbox** ..

Look mom no gradients!

Can we optimize $\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$ if without access to gradients?

meet the newest fanciest optimization algorithm,...

Random search

pick a random direction $\mathbf{d}_t \in \mathbb{R}^d$

$\gamma := \operatorname{argmin}_{\gamma \in \mathbb{R}} f(\mathbf{x}_t + \gamma \mathbf{d}_t)$ (line-search)

$\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma \mathbf{d}_t$

Convergence rate for derivative-free random search

Converges same as gradient descent - up to a slow-down factor d .

Proof. Assume that f is a L -smooth convex, differentiable function. For any γ , by smoothness, we have:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t) + \gamma \langle \mathbf{d}_t, \nabla f(\mathbf{x}_t) \rangle + \frac{\gamma^2 L}{2} \|\mathbf{d}_t\|^2$$

Minimizing the upper bound, there is a step size $\bar{\gamma}$ for which

$$f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t) \leq f(\mathbf{x}_t) - \frac{1}{L} \left\langle \frac{\mathbf{d}_t}{\|\mathbf{d}_t\|^2}, \nabla f(\mathbf{x}_t) \right\rangle^2$$

The step size we actually took (based on f directly) can only be better:

$$f(\mathbf{x}_t + \gamma \mathbf{d}_t) \leq f(\mathbf{x}_t + \bar{\gamma} \mathbf{d}_t).$$

Taking expectations:

$$\mathbb{E}[f(\mathbf{x}_t + \gamma \mathbf{d}_t)] \leq \mathbb{E}[f(\mathbf{x}_t)] - \frac{1}{Ld} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2]$$

Convergence rate for derivative-free random search

Same as what we obtained for [gradient descent](#),
now with an **extra factor of d** . d can be huge!!!

Can do the same for different function classes, as before

- ▶ For convex functions, we get a rate of $\mathcal{O}(dL/\varepsilon)$.
- ▶ For strongly convex, you get $\mathcal{O}(dL \log(1/\varepsilon))$.

Always d times the complexity of gradient descent on the function class.

credits to Moritz Hardt

Applications for derivative-free random search

Applications

- ▶ competitive method for **Reinforcement learning**
- ▶ memory and communication advantages: never need to store a gradient
- ▶ hyperparameter optimization, and other difficult e.g. discrete optimization problems
- ▶ can be improved to learn a second-order model of the function, during optimization [Stich PhD thesis, 2014]

Reinforcement learning

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t).$$

where \mathbf{s}_t is the **state** of the system, \mathbf{a}_t is the control **action**, and \mathbf{e}_t is some random **noise**. We assume that f is fixed, but unknown.

We search for a control 'policy'

$$\mathbf{a}_t := \pi(\mathbf{a}_1, \dots, \mathbf{a}_{t-1}, \mathbf{s}_0, \dots, \mathbf{s}_t).$$

which takes a trajectory of the dynamical system and outputs a new control action.
Want to maximize overall **reward**

$$\begin{aligned} \max_{\mathbf{a}_t} \mathbb{E}_{\mathbf{e}_t} \left[\sum_{t=0}^N R_t(\mathbf{s}_t, \mathbf{a}_t) \right] \\ \text{s.t. } \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_t) \\ (\mathbf{s}_0 \text{ given}) \end{aligned}$$

Examples: Simulations, Games (e.g. Atari), Alpha Go

Chapter X.3

Adaptive & other SGD Methods

Adagrad

Adagrad is an adaptive variant of SGD

pick a stochastic gradient \mathbf{g}_t

$$\text{update } [G_t]_i := \sum_{s=0}^t ([\mathbf{g}_s]_i)^2 \quad \forall i$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[G_t]_i}} [\mathbf{g}_t]_i \quad \forall i$$

(standard choice of $\mathbf{g}_t := \nabla f_j(\mathbf{x}_t)$ for sum-structured objective functions $f = \sum_j f_j$)

- ▶ chooses an **adaptive, coordinate-wise** learning rate
- ▶ strong performance in practice
- ▶ Variants: Adadelta, Adam, RMSprop

Adam

Adam is a momentum variant of Adagrad

pick a stochastic gradient \mathbf{g}_t

$$\mathbf{m}_t := \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (\text{momentum term})$$

$$[\mathbf{v}_t]_i := \beta_2 [\mathbf{v}_{t-1}]_i + (1 - \beta_2) ([\mathbf{g}_t]_i)^2 \quad \forall i \quad (\text{2nd-order statistics})$$

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[\mathbf{v}_t]_i}} [\mathbf{m}_t]_i \quad \forall i$$

- ▶ faster forgetting of older weights
- ▶ momentum from previous gradients (see acceleration, lecture 6)
- ▶ (simplified version, without correction for initialization of $\mathbf{m}_0, \mathbf{v}_0$)
- ▶ strong performance in practice, e.g. for self-attention networks

SignSGD

Only use the sign (one bit) of each gradient entry:

SignSGD is a communication efficient variant of SGD.

pick a stochastic gradient \mathbf{g}_t

$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \gamma_t \text{sign}([\mathbf{g}_t]_i) \quad \forall i$$

(with possible rescaling of γ_t with $\|\mathbf{g}_t\|_1$)

- ▶ communication efficient for distributed training
- ▶ convergence issues