OPT V2

Silviu Filote

September 2024

Indice

Notation

Objective function	$f : dom(f) \to \mathbb{R}$
	$min\ f(x)$
Set of feasible solutions	$X \subseteq dom(f)$
General points	$\boldsymbol{x}, \boldsymbol{y} \in X \subseteq dom(f)$
Minimizer of f	$\boldsymbol{x^*}$
Lambda come cursore	λ
Distance (norma)	$\|\|v\|\| = \sqrt{v_i^2 + \ldots + v_n^2}$
Tollerance	$\varepsilon = 10^{-6}$
triangle inequality	$\|\|x + y\|\| \leq \|\|x\|\| + \|\|y\|\|$
Cauchy-Schwarz Inequality	$\|u^T \cdot v\| \leq \|\|u\|\| \cdot \|\|v\|\|$ *con* $u, v \in \mathbb{R}^d$
	$u^t \cdot v = \|\|u\|\| \cdot \|\|v\|\| \cdot cos(\alpha)$
	$-1 \leq \dfrac{u^T \cdot v}{\|\|u\|\| \cdot \|\|v\|\|} \leq 1$
	$cos(\alpha) = \dfrac{u^T \cdot v}{\|\|u\|\| \cdot \|\|v\|\|}$
Jensen's Inequality	$f\left(\sum_{i=1}^{m} \lambda_i x_i\right) \leq \sum_{i=1}^{m} \lambda_i f(x_i)$
	f *convex,* $x_i \in dom(f)$, $\lambda_i \in \mathbb{R}_+, \sum \lambda_i = 1$
Tanget hyperplane in $(\boldsymbol{x}, f(\boldsymbol{x}))$	$f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$
Critical point (stallo)	$\nabla f(\boldsymbol{x}) = 0$
M positive semidefinite	$\boldsymbol{x}^T M \boldsymbol{x} \geq 0,\ \forall \boldsymbol{x}$
M positive definite	$\boldsymbol{x}^T M \boldsymbol{x} > 0,\ \forall \boldsymbol{x} \neq 0$

It measures the straight-line distance between the points represented by vectors $\boldsymbol{y}$ and $\boldsymbol{x}$ in nn-dimensional space. The result is a non-negative scalar value.

Norm $$\|\boldsymbol{y} - \boldsymbol{x}\| = \sqrt{\sum_{i}^{n} (y_i - x_i)^2}$$

The squared norm gives you a measure of the distance without taking the square root. It is often used in optimization problems, especially in regression (e.g., mean squared error), because it emphasizes larger deviations more significantly and avoids issues with differentiability at zero.

Norm $$\|\boldsymbol{y} - \boldsymbol{x}\|^2 = \sum_{i}^{n} (y_i - x_i)^2$$

Definitions

infeasible	$min\{f(x) : x \in X\} = +\infty \Rightarrow X = \varnothing$
unbounded	$min\{f(x) : x \in X\} = -\infty$
	$\forall M > 0, \exists x \in \boldsymbol{x} : f(x) < -M$
$\boldsymbol{x} \in X$ ***is a minimizer***	$f(\boldsymbol{x}) \leq f(y), \forall y \in X$
	$\boldsymbol{x}$ *is said to be optimal solution*
	$f(\boldsymbol{x})$ *is said optimal value*
local minimum $\boldsymbol{x}$	$f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \forall y \in dom(f) : \|\|y - x\|\| < \epsilon$, *con* $\epsilon > 0$
a set C is convex	*segment between any two points of C lies in C*
	$\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in C$ *con* $\boldsymbol{x}, \boldsymbol{y} \in C$ *e* $0 \leq \lambda \leq 1$
a function f is convex	*segment lies above the graph of f*
	$f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y}), 0 \leq \lambda \leq 1$
Convergence rate $\propto 1/t$	$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq \dfrac{c}{t} = 0$ *for* $t \to \infty$
graph *of a function f*	$\{(x, f(x)) \mid x \in dom(f)\}$
epigraph *of a function f*	$epi(f) \coloneqq \{(x, \alpha) \in \mathbb{R}^{d+1} \mid x \in dom(f), \alpha \geq f(x)\}$

I order char. of convexity	*f convex iff dom(f) is convex and*
	$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$ *con* $\boldsymbol{x}, \boldsymbol{y} \in dom(f)$
II order char. of convexity	*f convex iff dom(f) is convex and*
	$\nabla^2 f(\boldsymbol{x}) \succeq 0, \forall \boldsymbol{x} \in dom(f)$
Preserve convexity	*somma of convex f, composizione* $f \circ g$
Global min, $x^* \in dom(f)$	$\nabla f(\boldsymbol{x^*}) = 0$, *f convex and differentiable*
f strictly convex	*dom(f) convex,* $\forall \boldsymbol{x} \neq \boldsymbol{y} \in dom(f)$ *and* $\forall \lambda \in (0, 1)$
	we have: $f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) < \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y})$

Active constraints g_j	$j \in I(\boldsymbol{x}_0) = \{j \mid g_j(\boldsymbol{x}_0) = 0\}$
Qualified constraints	$\nabla g_j(\boldsymbol{x}_0) : \forall j \in I(\boldsymbol{x}_0) = \{j \mid g_j(\boldsymbol{x}_0) = 0\} \to$ *linearly indip.*
KKT conditions	*for general constrianed problems*
	holds on the regularity conditions
	computationally heavy, suits best for small prob

Small values of α	*long time to converge*
Large values of α	*upredictable, overshoot an optimal value*
	I Taylor expansion is no longer a good approx
	diverge $\Rightarrow$ *terminates with a numerical overflow*
Goal $\boldsymbol{x^*}$	$f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \leq \varepsilon = 10^{-6}$
Algorithm	$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \gamma \nabla f(\boldsymbol{x}_t), \quad \gamma > 0$
Vanilla analysis	$\sum_{t=0}^{T-1} \left(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)\right) \leq \frac{\gamma}{2} \sum_{t=0}^{T-1} \|\|g_t\|\|^2 + \frac{1}{2\gamma} \|\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|\|^2$

Finite difference:	$\frac{\partial J(\bar{w})}{\partial w_i} \approx \frac{J(w_1, \ldots, w_i + \Delta, \ldots, w_d) - J(w_1, \ldots, w_i, \ldots, w_d)}{\Delta}$
Decaying α_t	$\mathbf{w} \Leftarrow \bar{\mathbf{w}} - \alpha_t \nabla J$
Exponential Decay	$\alpha_t = \alpha_0 \cdot \exp(-k \cdot t)$
Inverse Decay	$\alpha_t = \frac{\alpha_0}{1 + k \cdot t}$

Bold Driver algorithm

Ob. function improve	$\alpha_{t+1} = \alpha_t + \alpha_t * 0.05$
Ob. function worsens	*previous step undone, untill convergence*
	$\alpha_{t+1} = \alpha_t - \alpha_t * 0.5$
Noisy setting	*adjust learning rate after m steps*
	rather then each step like above
	undone in case m steps

Line search

Update	$\boldsymbol{w}_{t+1} \Leftarrow \bar{\boldsymbol{w}}_t + \alpha_t \bar{\boldsymbol{g}}_t$
Learning rate α_t	*computationally heavy*
	optimum step size choosed
	minimizing the objective function
	$\alpha_t = \arg\min_{\alpha} J\left(\bar{\boldsymbol{w}}_t + \alpha \bar{\boldsymbol{g}}_t\right)$
Optimal α_t ***for*** $\bar{w}_{t+1}$	$\bar{g}_t^\top \left[\nabla J\left(\bar{w}_t + \alpha_t \bar{g}_t\right)\right] = \bar{g}_t^\top \left[\nabla J\left(\bar{w}_{t+1}\right)\right] = 0$
Steps	*identify range* $[0, \alpha_{max}]$
	narrow range with: binary search or Armijo rule
	minimizing objective function

Bounded gradients	$\iff$	***Lipschitz continuity of f***
Smoothness	$\iff$	***Lipschitz continuity of*** ∇f ***(in the convex case)***

Lipschitz	$\|\nabla f(\boldsymbol{x})\| < B, \forall \boldsymbol{x} \in \mathbb{R}^d$	$\|f(x) - f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$
Smooth	$\|\nabla^2 f(\boldsymbol{x})\| < L, \forall \boldsymbol{x} \in \mathbb{R}^d$	$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$

- If the gradients of $f(x)$ are ***bounded***, meaning there exists a constant L then the function $f(x)$ is ***lipschitz continuous*** with constant L
- A function $f(x)$ is said to be ***smooth*** if its gradient $\nabla f(x)$ is lipschitz continuous, meaning that there exists a constant L. Thus, smoothness implies that the Hessian is bounded, meaning that the largest eigenvalue of the Hessian is upper-bounded by L

Lipschitz convex f	*bounded slope*
	$\frac{1}{T} \sum_{t=0}^{T-1} \left(f(x_t) - f(x^*)\right) \leq \frac{RB}{\sqrt{T}}, \quad \gamma := \frac{R}{B\sqrt{T}}$
Smooth f	*doesnt require convexity*
	$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|x - y\|^2, \quad \forall x, y \in X$
Sufficient decrease	$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2, \quad \forall t \geq 0$
Smooth convex f	$f(x_T) - f(x^*) \leq \frac{L}{2T}\|x_0 - x^*\|^2, \quad T > 0$

Lipschitz	$T \geq \frac{R^2 B^2}{\varepsilon^2}$	average error $\leq \frac{RB}{\sqrt{T}} \leq \varepsilon$
Smooth	$T \geq \frac{R^2 L}{2\epsilon}$	error $\leq \frac{LR^2}{2T} \leq \epsilon$

Smooth f	$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|x - y\|^2, \quad \forall x, y \in X$
Strongly convex f	$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|x - y\|^2, \quad \forall x, y \in X$
Theorem	$\|x_{t+1} - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)\|x_t - x^*\|^2, \quad t \geq 0$
	$f(x_T) - f(x^*) \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2, \quad T > 0$

Projected GD	$y_{t+1} := x_t - \gamma \nabla f(x_t)$
	$x_{t+1} := \Pi_X(y_{t+1}) := \arg\min_{x \in X} \|x - y_{t+1}\|^2$

Proximal GD (1st way) $x_{t+1} = \text{prox}_{h,\gamma}(x_t - \gamma \nabla g(x_t))$

$$\text{prox}_{h,\gamma}(z) := \arg\min_y \left\{ \frac{1}{2\gamma}\|y - z\|^2 + h(y) \right\}$$

Proximal GD (2nd way) $x_{t+1} = x_t - \gamma G_{h,\gamma}(x_t)$

$$G_{h,\gamma}(x) := \frac{1}{\gamma}\left(x - \text{prox}_{h,\gamma}(x - \gamma \nabla g(x))\right)$$

Subdifferential $\partial f(x) \subseteq \mathbb{R}^d = \{g_i\}$

$$f(y) \geq f(x) + g^\top (y - x) \quad \forall y \in \text{dom}(f)$$

Subgradient descent $x_{t+1} := x_t - \gamma_t g_t$ *con* $g_t \in \partial f(x_t)$

Differentiable in x_0 $\partial f(x) \subseteq \{\nabla f(x)\}$

f(x) convex *subgradients everywhere*

Theorem (Nesterov) $f(x_T) - f(x^*) \geq \dfrac{RB}{2(1 + \sqrt{T+1})}$

Theorem $f\left(\dfrac{2}{T(T+1)} \sum_{t=1}^{T} t \cdot \boldsymbol{x}_t\right) - f(\boldsymbol{x}^*) \leq \dfrac{2B^2}{\mu(T+1)}$

Sum structured ob. f $f(x) = \dfrac{1}{n}\sum_{i=1}^{n} f_i(x)$

$$\nabla f(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x)$$

Stocastic gradient $g_t := \nabla f_i(x_t)$

SGD (with 1 obs) *sample* $i \in [n]$ *uniformly at random*

$$x_{t+1} := x_t - \gamma_t \nabla f_i(x_t)$$

f is smooth $\|\nabla^2 f(x)\| \leq L$

Problems	**notes**
NLP	objective function or constraints is non linear
Convex Optimization	f convex and X convex set
Quadratic Programming (QP)	
Constrained Optimization	projected GD, trasform into unconstrained

Non linear programming problem (NLP):

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & x \geq 0 \end{aligned}$$

Convex Optimization Problems:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

Quadratic Programming (QP):

$$\begin{aligned} \min \quad & \tfrac{1}{2} x^\top Q x + c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Constrained Optimization Problems: *projection*

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

Remark: the objective function of the problem is a convex function of $\boldsymbol{x}$ when Q is a positive semidefinite matrix. When this condition is satisfied, the QP problem can be solved in polynomial time.

Remark:

- If the gradients of $f(x)$ are ***bounded***, meaning there exists a constant L then the function $f(x)$ is ***lipschitz continuous*** with constant L
- A function $f(x)$ is said to be ***smooth*** if its gradient $\nabla f(x)$ is lipschitz continuous, meaning that there exists a constant L. Thus, smoothness implies that the Hessian is bounded, meaning that the largest eigenvalue of the Hessian is upper-bounded by L.

Bounded gradients $\iff$ ***Lipschitz continuity of f***

Smoothness $\iff$ ***Lipschitz continuity of*** ∇f ***(in the convex case).***

Lipschitz $\|\nabla f(\boldsymbol{x})\| < B, \forall \boldsymbol{x} \in \mathbb{R}^d$ $\quad |f(x) - f(y)| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$

Smooth $\|\nabla^2 f(\boldsymbol{x})\| < L, \forall \boldsymbol{x} \in \mathbb{R}^d$ $\quad |\nabla f(x) - \nabla f(y)| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$

Same in the constrained and uncostrained optimization problems:

- **Lipschitz convex functions over** X: $O\left(1/\varepsilon^2\right)$ steps.
- **Smooth convex functions over** X: $O(1/\varepsilon)$ steps.
- **Smooth and strongly convex functions over** X: $O(\log(1/\varepsilon))$ steps.
- **Smooth convex functions - Accelerated GD:** $O(1/\sqrt{\varepsilon})$
- **Newton's method:** $O(\log\log(1/\varepsilon))$

Vanilla	$\sum_{t=0}^{T-1}\left(f(\boldsymbol{x}_t)-f(\boldsymbol{x}^*)\right) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1}\|\boldsymbol{g}_t\|^2+\frac{1}{2\gamma}\|\boldsymbol{x}_0-\boldsymbol{x}^*\|^2$
Lipschitz	$\frac{1}{T}\sum_{t=0}^{T-1}\left(f(\boldsymbol{x}_t)-f(\boldsymbol{x}^*)\right) \leq \frac{RB}{\sqrt{T}}, \quad \gamma=\frac{R}{B\sqrt{T}}$
Sufficient decrease	$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t)-\frac{1}{2L}\|\nabla f(\boldsymbol{x}_t)\|^2, \quad \forall t \geq 0$
Smooth convex f	$f(\boldsymbol{x}_T)-f(\boldsymbol{x}^*) \leq \frac{L}{2T}\|\boldsymbol{x}_0-\boldsymbol{x}^*\|^2, \quad T>0, \quad \gamma=\frac{1}{L}$
Smooth and strongly	$\|\boldsymbol{x}_{t+1}-\boldsymbol{x}^*\|^2 \leq \left(1-\frac{\mu}{L}\right)\|\boldsymbol{x}_t-\boldsymbol{x}^*\|^2, \quad t \geq 0, \quad \gamma=\frac{1}{L}$
	$f(\boldsymbol{x}_T)-f(\boldsymbol{x}^*) \leq \frac{L}{2}\left(1-\frac{\mu}{L}\right)^T\|\boldsymbol{x}_0-\boldsymbol{x}^*\|^2, \quad T>0$
Nesterov	$f(\boldsymbol{x}_T)-f(\boldsymbol{x}^*) \geq \frac{RB}{2(1+\sqrt{T+1})}$
Theorem	$f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T} t\cdot\boldsymbol{x}_t\right)-f(\boldsymbol{x}^*) \leq \frac{2B^2}{\mu(T+1)}$
	$\gamma_t=\frac{2}{\mu(t+1)}$

Lipschitz convex	$T \geq \frac{R^2B^2}{\varepsilon^2}$	average error $\leq \frac{RB}{\sqrt{T}} \leq \varepsilon$
Smooth convex	$T \geq \frac{R^2L}{2\varepsilon}$	error $\leq \frac{R^2L}{2T} \leq \varepsilon$
Strongly convex	$T \geq \frac{L}{\mu}\ln\left(\frac{R^2L}{2\varepsilon}\right)$	error $\leq \frac{R^2L}{2}\left(1-\frac{\mu}{L}\right)^T \leq \varepsilon$

Training set is linearly separable:

$$\begin{aligned} \max_{\boldsymbol{w},\theta} \quad & \rho_{\boldsymbol{w},\theta}=\min_{p=1,\ldots,P}\frac{|\boldsymbol{w}^\top\boldsymbol{x}_p+\theta|}{\|\boldsymbol{w}\|}=\frac{1}{\|\bar{\boldsymbol{w}}\|}=\bar{\boldsymbol{w}}^T(\boldsymbol{x_{p'}}-\boldsymbol{x_{p''}}) \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)\geq 1 \quad p=1,\ldots,P \end{aligned}$$

Hard Margin SVM: when the training set is not linearly separable, the problem has an empty feasible region. This means that there is no solution that satisfies all constraints. In this case, slack variables ξ_p must be introduced, leading to the new optimization problem known as the Soft Margin SVM.

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)\geq 1 \quad p=1,\ldots,P \end{aligned}$$

Non linearly separable - Soft Margin SVM: the term ξ_p represents slack variables that allow for some misclassification of the training examples. Each ξ_p corresponds to the p-th training example and indicates how much the example violates the margin constraints. The term C is a regularization parameter that controls the trade-off between maximizing the margin (the first term) and minimizing the classification error (the second term). A larger C puts more emphasis on correctly classifying all training examples, while a smaller C allows more misclassifications to achieve a wider margin.

$$\begin{aligned} \min_{w,\theta,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{p=1}^{P}\xi_p \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^T\boldsymbol{x}_p+\theta)\geq 1-\xi_p, && p=1,\ldots,P \\ & \xi_p\geq 0, && p=1,\ldots,P \end{aligned}$$

Non linearly separable - Feature space: the idea behind non-linear SVM is to map the input space into a feature space where the data points are linearly separable. It is important to recall that in linear SVM models, the dual problem can be easily constructed using the kernel function $K(\cdot,\cdot)$. Moreover, the decision function only utilizes this kernel function. Let $\mathcal{F}$ be a ***Hilbert space*** (the feature space) whose dimension is greater than n and possibly infinite, and let $\phi:\mathbb{R}^n\to\mathcal{F}$ be a mapping from $\mathbb{R}^n$ to the feature space.

$$\begin{aligned} \min_{\boldsymbol{w},\theta,\boldsymbol{\xi}} \quad & \frac{1}{2}||\boldsymbol{w}||^2+C\sum_{p=1}^{P}\xi_p \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^\top\phi(x_p)+\theta)\geq 1-\xi_p && p=1,\ldots,P \\ & \xi_p\geq 0 && p=1,\ldots,P. \end{aligned}$$

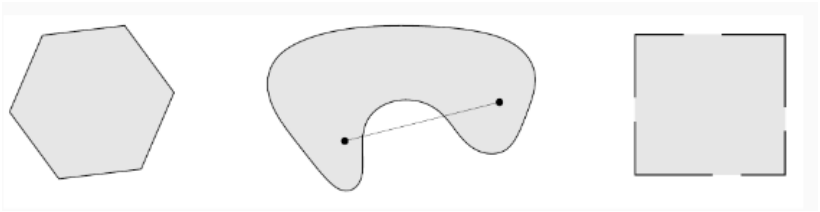
Risoluzione: lagrangian function associated to the problem $L(\boldsymbol{w},\theta,\alpha) \Rightarrow$ Wolfe dual reformulating the problem into its dual form, only lagrange multipliers $\alpha \Rightarrow$ re-write it in a compact quadratic form and resolve it getting $\alpha^* \Rightarrow \alpha^*$ into first KKT condition getting $\boldsymbol{w}^* \Rightarrow \boldsymbol{w}^*$ into the last KKT condition getting $\boldsymbol{\theta}^* \Rightarrow$ evaluating decision function $h(\boldsymbol{x}) = sign\left((\boldsymbol{w}^*)^T\boldsymbol{x}+\boldsymbol{\theta}^*\right)$ using underneath the kernel function $K(\bar{\boldsymbol{x}},\hat{\boldsymbol{x}})=\phi(\bar{\boldsymbol{x}})^\top\phi(\hat{\boldsymbol{x}})$

1 Theory

1.1 Theorems

Def: a set C is convex if the line segment between any two points of C lies in C, i.e. if $\forall \boldsymbol{x}, \boldsymbol{y} \in C$ and any λ with $0 \leq \lambda \leq 1$ (boundary included)

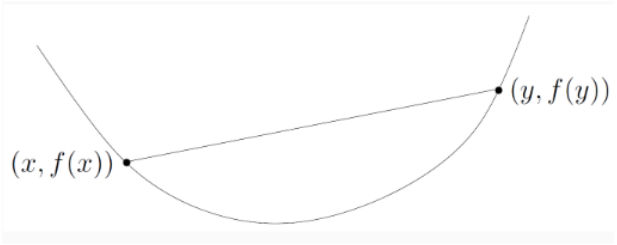
$$\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} \in C, \quad \forall \lambda \in [0,1]$$



Def: a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if:
- $dom(f)$ is a convex set
- $\forall x, y \in dom(f)$ and any λ with $0 \leq \lambda \leq 1$ we have:

$$f\left(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}\right) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$$

The line segment between $(\boldsymbol{x}; f(\boldsymbol{x})$ and $(\boldsymbol{y}; f(\boldsymbol{y}))$ lies above the graph of f



Crucial Property of Convex Optimization Problems: every local minimum is a global minimum, for convex optimization problems all algorithms (coordinate descent, gradient descent, stochastic gradient descent, projected and proximal gradient descent) do converge to the global optimum

Def: a function $f : \mathrm{dom}(f) \to \mathbb{R}$ is strictly convex if:
- $\mathrm{dom}(f)$ is convex;
- for all $\boldsymbol{x} \neq \boldsymbol{y} \in \mathrm{dom}(f)$ and all $\lambda \in (0,1)$, we have:

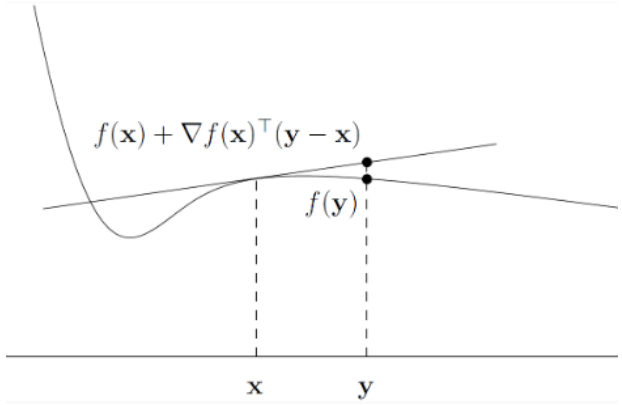
$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) < \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}).$$

1.2 First-order Characterization of Convexity

Differentiable functions: graph of the affine function $f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$ is a tangent hyperplane to the graph of f at $(\boldsymbol{x}, f(\boldsymbol{x}))$

$$\lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

Infinitesimal distance from $\boldsymbol{x}$: $\boldsymbol{y} \to \boldsymbol{x}$



Lemma: suppose that $\mathrm{dom}(f)$ is open and that f is differentiable (all over the domain); in particular, the gradient (vector of partial derivatives)

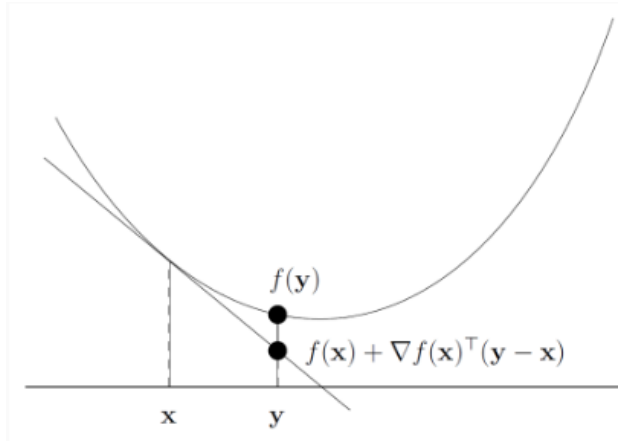
$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_d}(x) \right)$$

exists at every point $x \in \mathrm{dom}(f)$. Then f is convex if and only if $\mathrm{dom}(f)$ is convex and holds for all $x, y \in \mathrm{dom}(f)$.

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x})$$

Lemma: graph of f is above all its tanget hyperplanes

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{y} - \boldsymbol{x}) \quad \boldsymbol{x}, \boldsymbol{y} \in dom(f)$$



1.3 Second-order Characterization of Convexity

Lemma: suppose that $\text{dom}(f)$ is open and that f is twice differentiable; in particular, the Hessian (matrix of second partial derivatives)

$$\nabla^2 f(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1}(x) & \frac{\partial^2 f}{\partial x_d \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d}(x) \end{bmatrix}$$

exists at every point $x \in \text{dom}(f)$ and is symmetric. Then f is convex if and only if $\text{dom}(f)$ is convex, and for all $x \in \text{dom}(f)$, we have i.e. $\nabla^2 f(x)$ is positive semidefinite.

$$\nabla^2 f(x) \succeq 0,$$

Def: a symmetric matrix M is positive semidefinite if $\mathbf{x}^\top M\mathbf{x} \geq 0$ for all $\mathbf{x}$, and positive definite if $\mathbf{x}^\top M\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$.

Theorem: a symmetric matrix M is ***positive semidefinite*** if and only if the determinant of each principal submatrix (not only North-West) is non-negative and the determinant of M is null. A symmetric matrix M is ***positive definite*** if and only if the determinant of all the North-West submatrices is greater than 0.

1.4 Important notions

Lemma: let f be convex and suppose that $dom(f)$ is open $\Rightarrow$ then f is continuous

Lemma: $\boldsymbol{x}^*$ be a local minimum of a convex function $f : \text{dom}(f) \to \mathbb{R}$. Then $\boldsymbol{x}^*$ is a global minimum, meaning that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \text{dom}(f)$.

Lemma: suppose that f is convex and differentiable over an open domain $\text{dom}(f)$. Let $\boldsymbol{x} \in \text{dom}(f)$. If $\nabla f(\boldsymbol{x}) = 0$ (critical point), then $\boldsymbol{x}$ is a global minimum.

Lemma: let $f : \text{dom}(f) \to \mathbb{R}$ be strictly convex. Then f has at most one global min.

Lemma: suppose that $f : \text{dom}(f) \to \mathbb{R}$ is convex and differentiable over an open domain $\text{dom}(f) \subseteq \mathbb{R}^d$, and let $X \subseteq \text{dom}(f)$ be a convex set. Point $\boldsymbol{x}^* \in X$ is a minimizer of f over X if and only if

$$\nabla f(\boldsymbol{x}^*)^\top (\boldsymbol{x} - \boldsymbol{x}^*) \geq 0 \quad \forall \boldsymbol{x} \in X.$$

Questa formulazione mostra che $\nabla f(\boldsymbol{x}^*)$ agisce come una normale al piano tangente a f in $\boldsymbol{x}^*$, e il fatto che la derivata direzionale sia non negativa in ogni direzione dentro $\boldsymbol{X}$ implica che $\boldsymbol{x}^*$ sia un punto in cui f non può diminuire.

Definition: let $f : \mathbb{R}^d \to \mathbb{R}$ and $\alpha \in \mathbb{R}$. The set $f^{\leq \alpha} := \{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) \leq \alpha\}$ is the α-sublevel set of f.

Weierstrass theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, and suppose there is a nonempty and bounded sublevel set $f^{\leq \alpha}$. Then f has a global minimum.

1.5 Karush-Kuhn-Tucker conditions (KKT conditions)

Consider the NLP: where $f : \text{dom}(f) \subseteq \mathbb{R}^n \to \mathbb{R}$ and $g_i : \text{dom}(g_i) \subseteq \mathbb{R}^n \to \mathbb{R}$ are given functions and at least one of these functions (objective function or constraints) is non-linear.

$$\begin{aligned} \max \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

Problem	Necessary Conditions for Optimality	Also Sufficient if:
One-variable unconstrained	$\frac{df}{dx} = 0$	$f(x)$ concave
Multivariable unconstrained	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$	$f(\mathbf{x})$ concave
Constrained, nonnegativity constraints only	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$ (or ≤ 0 if $x_j = 0$)	$f(\mathbf{x})$ concave
General constrained problem	Karush-Kuhn-Tucker conditions	$f(\mathbf{x})$ concave and $g_i(\mathbf{x})$ convex $(i = 1, 2, \ldots, m)$

Theorem (regularity condition): if the vectors $\nabla g_j(\boldsymbol{x}_0)$ for all $j \in I(\boldsymbol{x}_0) = \{j \mid g_j(\boldsymbol{x}_0) = 0\}$ are linearly independent, then the constraints of the problem NLP are qualified.

Meaning of the Theorem:

- **Vectors** $\nabla g_j(\boldsymbol{x}_0)$: are the gradients (or partial derivatives) of the constraint functions g_j at a specific point $\boldsymbol{x}_0$. The gradient vector indicates the direction of the steepest ascent of the function g_j.
- **Index Set** $I(\boldsymbol{x}_0)$: set contains the indices of the constraints that are **active** at the point $\boldsymbol{x}_0$. A constraint $g_j(\boldsymbol{x}_0) = 0$ is considered active if it is binding at that point, meaning that the solution lies on the boundary defined by that constraint.
- **Linear Independence**: this condition is crucial because it ensures that the active constraints provide unique and non-redundant information about the feasible region.

Theorem: let $\boldsymbol{x}_0$ be a local solution of the constrained problem NLP. If the active constraints in $\boldsymbol{x}_0$ are qualified, there exists a vector $\boldsymbol{u}_0 \in \mathbb{R}^m$ such that $\boldsymbol{u}_0$ and $\boldsymbol{x}_0$ are solutions of the following system (KKT conditions):

$$\frac{\partial f(\boldsymbol{x}_0)}{\partial x_j} - \sum_{i=1}^{m} u_{0,i} \frac{\partial g_i(\boldsymbol{x}_0)}{\partial x_j} = 0, \quad j = 1, \ldots, n \tag{1}$$

$$g_j(\boldsymbol{x}_0) \leq 0, \quad j = 1, \ldots, m \tag{2}$$

$$u_{0,j} \geq 0, \quad j = 1, \ldots, m \tag{3}$$

$$u_{0,j} g_j(\boldsymbol{x}_0) = 0, \quad j = 1, \ldots, m \tag{4}$$

The u_i correspond to the dual variables in linear programming, and they have a comparable economic interpretation. However, they arise in the mathematical derivation as Lagrange multipliers.

Corollary: assume that $f(\boldsymbol{x})$ is a concave function and that $g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})$ are convex functions (i.e., this problem is a convex programming problem), where all these functions satisfy the regularity conditions. Then $\boldsymbol{x}_0$ is an optimal solution if and only if all the conditions of the previous theorem are satisfied.

Remark: the method of Lagrange multipliers and KKT conditions are powerful theoretical tools, they are not always the best choice from a practical computational perspective. They are often too difficult to apply in large-scale or complex problems due to the challenges in solving the equations and dealing with multiple critical points. However, for small, well-defined problems, these methods can still be effective.

2 Gradient descent

Remark: $\nabla f(\boldsymbol{x}) = 0$ critical analytical solution, the derivative might be difficult to solve $\Rightarrow$ a closed form solution typically does not exist, popular approach for optimizing objective functions is to use the method of gradient descent, gradient descent one starts at an initial point $\boldsymbol{x} = \boldsymbol{x}_0$ and successively updates $\boldsymbol{x}$ using the steepest descent direction, $\alpha > 0$ regulates the step size (or learning rate)

$$\boldsymbol{x} = \boldsymbol{x} - \alpha f'(\boldsymbol{x})$$

Remark: the value of $\boldsymbol{x}$ changes in each iteration by $\delta\boldsymbol{x} = -\alpha f'(\boldsymbol{x})$, note that at infinitesimally small values of the learning rate $\alpha > 0$ the above updates will always reduce $f(\boldsymbol{x}) \Rightarrow$ this is because for very small α we can use the ***first-order Taylor*** expansion (tangent line/hyperplane in the current point) to obtain the following

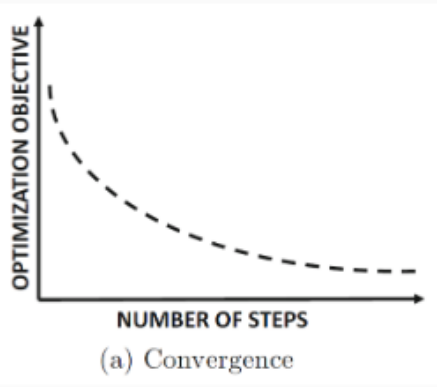
$$f(x + \delta x) \approx f(x) + \delta x f'(x) = f(x) - \alpha[f'(x)]^2 < f(x)$$

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = f(x_0) + \delta x f'(x_0)$$

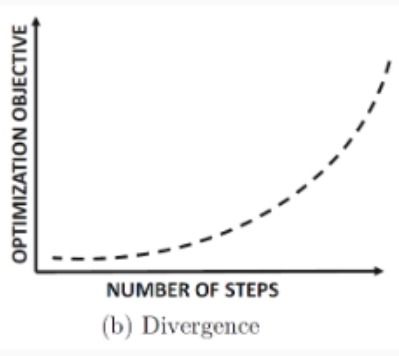
$f(\boldsymbol{x_0})$	*value of the function in* $\boldsymbol{x}_0$*, intercept*
$f'(\boldsymbol{x_0})$	*slope of the tangent line at* $\boldsymbol{x_0}$

Small values of α	*long time to converge*
Large values of α	*upredictable, overshoot an optimal value*
	I Taylor expansion is no longer a good approx
	diverge $\Rightarrow$ *terminates with a numerical overflow*

Remark: as the value of x_t nears the optimum value the derivative $f'(x_t)$ also tends to be closer and closer to zero (thereby satisfying the first-order optimality conditions) $\Rightarrow$ for termination condition the idea is to plot the current value of $f(x_t)$ with iteration index t as the algorithm progresses, the objective function value need not be monotonically decreasing over the course of the algorithm but it will tend to show ***small noisy changes*** (without significant long-term direction) after some point $\Rightarrow$ good termination point



(a) Convergence



(b) Divergence

Remark: common to reduce the learning rate over the course of the algorithm $\Rightarrow$ such an approach might not prevent divergence especially if the initial learning rate is large, ***NB:*** when the size of the parameter vector seems to increase rapidly (and the optimization objective worsens) it is a tell-tale sign of divergence

Get near to a minimum $\boldsymbol{x}^*$ / close to the optimal value $f(\boldsymbol{x}^*)$.
Assumption: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex, differentiable, and has a global min $\boldsymbol{x}^*$.

Goal: find $\boldsymbol{x}^* \in \mathbb{R}^d$ such that

$$f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \leq \varepsilon = 10^{-6}$$

Note that there can be several global minima $\boldsymbol{x}_1^* \neq \boldsymbol{x}_2^*$ with $f(\boldsymbol{x}_1^*) = f(\boldsymbol{x}_2^*)$.

Iterative Algorithm: choose $\boldsymbol{x}_0 \in \mathbb{R}^d$ for timesteps $t = 0, 1, \ldots$, and stepsize $\gamma \geq 0$.

$$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \gamma \nabla f(\boldsymbol{x}_t)$$

2.1 Vanilla analysis

In questa dimostrazione, stiamo cercando di ottenere un limite superiore per la quantità $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$, che rappresenta la differenza tra il valore della funzione obiettivo nel punto $\boldsymbol{x}_t$ e il valore della funzione obiettivo nel punto ottimo $\boldsymbol{x}^*$.

- Abbiamo definito $g_t := \nabla f(\boldsymbol{x}_t)$, che è il gradiente della funzione obiettivo f calcolato nel punto $\boldsymbol{x}_t$ all'iterazione t. La regola di aggiornamento per la discesa del gradiente può essere scritta come:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma g_t$$
$$g_t = \frac{\boldsymbol{x}_t - \boldsymbol{x}_{t+1}}{\gamma}$$

  Ora, consideriamo il prodotto scalare tra il gradiente g_t e la differenza $\boldsymbol{x}_t - \boldsymbol{x}^*$, dove $\boldsymbol{x}^*$ è l'ottimo. Questo ci aiuta a connettere il gradiente alla quantità di interesse $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$.

$$\boldsymbol{x}^* \text{ minimizer iff } \Rightarrow \nabla f(\boldsymbol{x}^*)^T(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall \boldsymbol{x} \in X$$

$$g_t{}^T(\boldsymbol{x}_t - \boldsymbol{x}^*) = \frac{1}{\gamma}(\boldsymbol{x}_t - \boldsymbol{x}_{t+1})^T(\boldsymbol{x}_t - \boldsymbol{x}^*)$$

- Apply $2\boldsymbol{v}^T\boldsymbol{w} = ||\boldsymbol{v}||^2 + ||\boldsymbol{w}||^2 - ||\boldsymbol{v} - \boldsymbol{w}||^2$

$$\cancel{2}\boldsymbol{v}^T\boldsymbol{w} = \frac{1}{2}\left(||\boldsymbol{v}||^2 + ||\boldsymbol{w}||^2 - ||\boldsymbol{v} - \boldsymbol{w}||^2\right)$$

$$\begin{aligned} g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*) &= \frac{1}{\gamma}(\boldsymbol{x}_t - \boldsymbol{x}_{t+1})^T(\boldsymbol{x}_t - \boldsymbol{x}^*) \\ &= \frac{1}{2\gamma}(||\boldsymbol{x}_t - \boldsymbol{x}_{t+1}||^2 + ||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2) \\ &= \frac{1}{2\gamma}(||\gamma g_t||^2 + ||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2) \\ &= \frac{\gamma}{2}||g_t||^2 + \frac{1}{2\gamma}(||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2) \end{aligned}$$

- Ora sommiamo entrambi i lati dell'equazione sopra per $t = 0$ fino a $T - 1$:

$$\sum_{t=0}^{T-1} g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*) = \sum_{t=0}^{T-1}\left(\frac{\gamma}{2}||g_t||^2 + \frac{1}{2\gamma}\left(||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2\right)\right)$$

Notiamo che nella somma telescopica dei termini $||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2$, quasi tutti i termini si cancellano, lasciandoci solo il primo e l'ultimo termine:

$$\sum_{t=0}^{T-1} g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*) = \frac{\gamma}{2}\sum_{t=0}^{T-1}||g_t||^2 + \frac{1}{2\gamma}(||\boldsymbol{x}_0 - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_T - \boldsymbol{x}^*||^2)$$

- Sappiamo che una funzione convessa soddisfa la seguente disuguaglianza per qualsiasi $\boldsymbol{x}$ e $\boldsymbol{y}$:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$$

Impostando $\boldsymbol{x} = \boldsymbol{x}_t$ e $\boldsymbol{y} = \boldsymbol{x}^*$, otteniamo:

$$f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t) \geq g_t^T(\boldsymbol{x}^* - \boldsymbol{x}_t)$$

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*)$$

Questo ci consente di applicare questa disuguaglianza alla somma di $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$:

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*)$$

$$\sum_{t=0}^{T-1}\left(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)\right) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1}||g_t||^2 + \frac{1}{2\gamma}||\boldsymbol{x}_0 - \boldsymbol{x}^*||^2 - \cancel{\frac{1}{2\gamma}||\boldsymbol{x}_{T+1} - \boldsymbol{x}^*||^2}$$

Oss: abbiamo dunque un limite superiore attraverso tutte le iterazioni, last iterate is not necessarily the best one, stepsize is crucial.

3 The Minutiae of Gradient Descent

Remark: in ML algorithms the gradients are computed either analytically and then hand-coded into the algorithm or computed using neural networks, in all these cases analytical or coding errors remain a real possibility ⇒ ***checking gradient correctness with finite differences***

Checking gradient correctness with finite differences: sample a few from $w_1, \ldots, w_d$ and check their partial derivatives, perturb w_i by a small amount Δ and approximate the partial derivative with respect to w_i by using the difference between the perturbed value of the objective function and the original value:

$$\frac{\partial J(\bar{w})}{\partial w_i} \approx \frac{J(w_1, \ldots, w_i + \Delta, \ldots, w_d) - J(w_1, \ldots, w_i, \ldots, w_d)}{\Delta}$$

Remark: consider a ***decaying learning rate*** α_t and the update is as follows:

Update	$\mathbf{w} \Leftarrow \bar{\mathbf{w}} - \alpha_t \nabla J$
Exponential Decay	$\alpha_t = \alpha_0 \cdot \exp(-k \cdot t)$
Inverse Decay	$\alpha_t = \frac{\alpha_0}{1 + k \cdot t}$

Bold Driver algorithm: popular approach for adjusting the learning rate depending on whether the objective function is improving or worsenin, ***learning rate is increased*** by factor of around 5% in each iteration as long as the steps improve the objective function, as soon as the ***objective function worsens*** because of a step the step is undone and an attempt is made again with the learning rate reduced by a factor of around 50% (until convergence). BD does not work in ***noisy settings*** of gradient descent ⇒ in such a case it is important to test the objective function and adjust the learning rate after m steps rather than a single step, all m steps must be undone when the objective function worsens over these steps.

Ob. function improve	$\alpha_{t+1} = \alpha_t + \alpha_t * 0.05$
Ob. function worsens	*previous step undone, untill convergence*
	$\alpha_{t+1} = \alpha_t - \alpha_t * 0.5$
Noisy setting	*adjust learning rate after m steps*
	rather then each step like above

Line search: uses the optimum step size, rarely used because it is computationally expensive, the parameter vector needs to be updated as follows

$$\boldsymbol{w}_{t+1} \Leftarrow \bar{\boldsymbol{w}}_t + \alpha_t \bar{\boldsymbol{g}}_t$$

The step-size α_t is computed as follows:

$$\alpha_t = \arg\min_{\alpha} J\left(\bar{\boldsymbol{w}}_t + \alpha \bar{\boldsymbol{g}}_t\right)$$

After performing the step the gradient is computed at $\bar{w}_{t+1}$ for the next step, the gradient at $\bar{w}_{t+1}$ will be perpendicular to the search direction $\bar{g}_t$ or else α_t will not be optimal for computing $\bar{w}_{t+1}$. Therefore, we obtain the following:

$$\bar{g}_t^\top \left[\nabla J\left(\bar{w}_t + \alpha_t \bar{g}_t\right)\right] = \bar{g}_t^\top \left[\nabla J\left(\bar{w}_{t+1}\right)\right] = 0$$

$$\boldsymbol{v} \cdot \boldsymbol{w} = v_1 w_1 + v_2 w_2 + \ldots + v_n w_n = ||\boldsymbol{v}|| \cdot ||\boldsymbol{w}|| \cdot \cos\alpha = 0 \Rightarrow \alpha = \pm\frac{\pi}{2}$$

Lemma: the gradient at the optimal point of a line search is always orthogonal to the current search direction ($\nabla J\left(\bar{w}_{t+1}\right)$ does not have a component along $\boldsymbol{g}_t$.

First step to identify a range $[0, \alpha_{max}]$ in which to perform the search, subsequently it is possible to use a variety of methods to narrow the interval such as: the ***binary-search method***, the ***golden-section search method*** and the ***Armijo rule***. The first two of these methods and exact methods instead the Armijo rule is inexact and it works even when $H(\alpha) = J(\bar{w}_t + \alpha\bar{g}_t)$ is multimodal/nonconvex in α

Binary Search: we start by initializing the binary search interval for α to

$$[a, b] = [0, \alpha_{\max}].$$

The interval $[a, b]$ is narrowed by evaluating the objective function at two closely spaced points near $\frac{a+b}{2}$. We evaluate the objective function at $\frac{a+b}{2}$ and $\frac{a+b}{2} + \varepsilon$:

$$H\left[\frac{a+b}{2}\right] \quad \text{and} \quad H\left[\frac{a+b}{2} + \varepsilon\right] \quad where \quad H(\alpha) = J(\bar{w}_t + \alpha\bar{g}_t).$$

If the function is increasing at $\frac{a+b}{2}$, the interval is narrowed to

$$\left[a, \frac{a+b}{2} + \varepsilon\right].$$

Otherwise, it is narrowed to

$$\left[\frac{(a+b)}{2}, b\right].$$

This process is repeated until an interval is reached with the required level of accuracy.

Armijo Rule: when we start at a point $\bar{w}_t$, we have a direction $\bar{g}_t$ that we want to move in to decrease the function value. At the beginning, when we haven't moved yet ($\alpha = 0$), the effectiveness of this direction can change as we go further along it. The improvement in the objective function when we first start moving in the direction $\bar{g}_t$ can be measured. This rate of improvement is represented mathematically as:

$$\left|g_t^\top \nabla F(\bar{w}_t)\right|$$

This expression tells us how much we can expect the function to improve initially (F and J different objective function). The typical improvement of the objective function at a particular value of α can be optimistically expected to be:

$$\alpha \left|g_t^\top \nabla F(\bar{w}_t)\right|$$

Armijo Rule: requires that we find a fraction $\mu \in (0, 0.5)$ of this expected improvement. A typical value of μ is around 0.25. We want to find the largest step-size α that satisfies the following condition:

$$F(\bar{w}_t) - F\left(\bar{w}_t + \alpha\bar{g}_t\right) \geq \mu\alpha \left|g_t^\top \nabla F(\bar{w}_t)\right|$$

$$F(\bar{w}_t) - F\left(\bar{w}_{t+1}\right) \geq \mu\alpha \left|g_t^\top \nabla F(\bar{w}_t)\right|$$

For very small values of α, the condition above will always be satisfied. However, to ensure faster progress, we need to find the largest possible step-size. Instead of always choosing a very small α, we aim for a larger value that still satisfies the condition.
How do we find the largest step-size? We test progressively smaller values of α and stop once the condition is satisfied. In the *backtracking line search* method, we start by testing the following values:

$$H(\alpha_{\max}), \quad H(\beta\alpha_{\max}), \quad \ldots, \quad H(\beta^r\alpha_{\max})$$

We stop once the condition is satisfied, and at that point, we use:

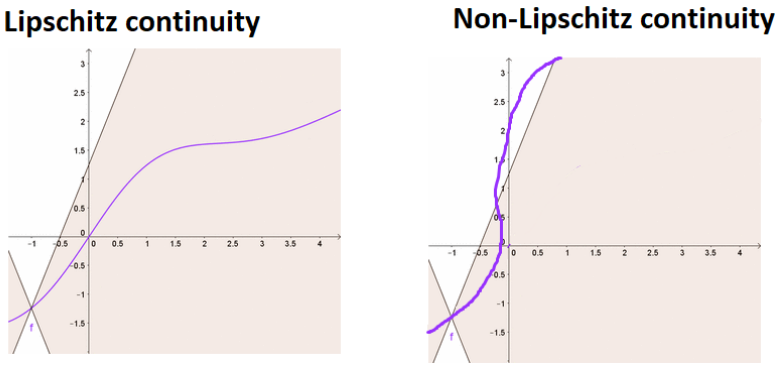
$$\alpha = \beta^r \alpha_{\max}$$

Here, β is a parameter chosen from the range $(0, 1)$, with a typical value of 0.5.

Conclusions line search: convergence is guaranteed even if the resulting solution is a local optimum, is expensive

3.1 Lipschitz Convex Functions

The Lipschitz condition (or Lipschitz continuity) ensures that the slope of the function remains under control, such that the function never escapes outside this cone.



Assume that all gradients of f are bounded in norm: equivalent to f being Lipschitz.

Def Lipschitz function: let $f : \mathbb{R}^d \to \mathbb{R}$ be a function. We say that f is *Lipschitz continuous* (or a *Lipschitz function*) with parameter $L \geq 0$ if there exists a constant $L \in \mathbb{R}^+$ such that:

$$|f(x) - f(y)| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d.$$

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, with a global minimum x^*. Furthermore, suppose that $\|x_0 - x^*\| \leq R$ and $\|\nabla f(x)\| \leq B$ for all x. By choosing the step size γ as follows:

$$\gamma := \frac{R}{B\sqrt{T}}$$

gradient descent yields:

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \leq \frac{RB}{\sqrt{T}}.$$

Steps for Lipschitz Convex Functions $O(1/\varepsilon^2)$ **Steps:** if we choose $T \geq \frac{R^2B^2}{\varepsilon^2}$

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \equiv \text{average error} \leq \frac{RB}{\sqrt{T}} \leq \varepsilon.$$

Advantages: the result is dimension-independent (no dependence on d), the bound holds for both the average error or the best iterate.

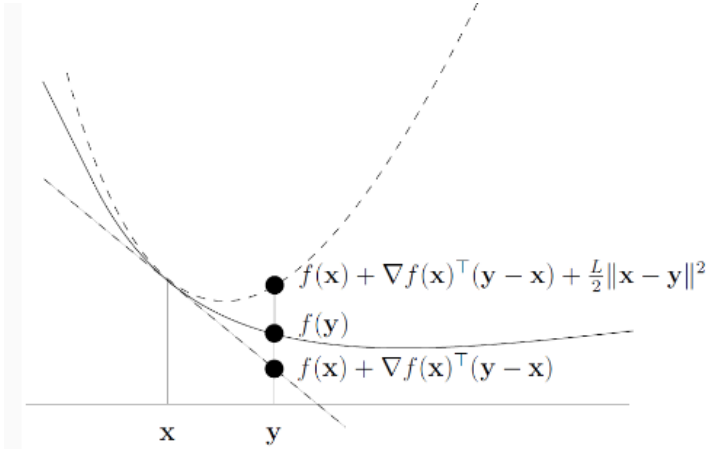
3.2 Smooth Functions

Definition: let $f : \text{dom}(f) \to \mathbb{R}$ be differentiable, and $X \subseteq \text{dom}(f)$, with $L \in \mathbb{R}^+$. The function f is called *smooth* (with parameter L) over X if:

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|x - y\|^2, \quad \forall x, y \in X.$$

The function f is *smooth* if and only if it is smooth over $\mathbb{R}^d$. ***Note:*** The definition does not require convexity.

Smoothness: for any $\boldsymbol{x}$, the graph of f is below a not too steep tangent paraboloid at $(\boldsymbol{x}, f(\boldsymbol{x}))$.



Lemma: In general quadratic functions are smooth, operations that preserve smoothness (the same that preserve convexity)

Lemma: let $f_1, \ldots, f_m$ be functions that are smooth with parameters $L_1, \ldots, L_m$, and let $\lambda_1, \ldots, \lambda_m \in \mathbb{R}^+$. Then, the function $f := \sum_{i=1}^m \lambda_i f_i$ is smooth with parameter $\sum_{i=1}^m \lambda_i L_i$. Basically we can build a f summing other smooth functions.

Lemma: let f be smooth with parameter L, and let $g(x) = Ax + b$, for $A \in \mathbb{R}^{d\times m}$ and $b \in \mathbb{R}^d$. Then the function $f \circ g$ (the composition of f and g) is smooth with parameter $L\|A\|^2$, where $\|A\|$ is the spectral norm of A.

Bounded gradients	$\iff$	***Lipschitz continuity of f***
Smoothness	$\iff$	***Lipschitz continuity of*** ∇f ***(in the convex case).***

Lipschitz	$\|\nabla f(\boldsymbol{x})\| < B, \forall \boldsymbol{x} \in \mathbb{R}^d$	$\|f(x) - f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$
Smooth	$\|\nabla^2 f(\boldsymbol{x})\| < L, \forall \boldsymbol{x} \in \mathbb{R}^d$	$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$

Lemma: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable. The following two statements are equivalent:
- f is smooth with parameter L.
- $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^d$.

Sufficient decrease lemma: let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and smooth with parameter L. With step size $\gamma := \frac{1}{L}$, gradient descent satisfies:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2, \quad \forall t \geq 0.$$

Remark: this inequality holds more specifically if f is smooth with parameter L over the line segment connecting x_t and x_{t+1}.

Proof: we begin by using the smoothness condition of the function f and the definition of the gradient descent update $x_{t+1} - x_t = -\frac{1}{L}\nabla f(x_t)$:

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)^\top (x_{t+1} - x_t) + \frac{L}{2}\|x_t - x_{t+1}\|^2.$$

Substitute the update rule $x_{t+1} - x_t = -\frac{1}{L}\nabla f(x_t)$:

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)^\top \left(-\frac{1}{L}\nabla f(x_t)\right) + \frac{L}{2}\left\|\frac{1}{L}\nabla f(x_t)\right\|^2.$$

Simplifying each term:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{L}\|\nabla f(x_t)\|^2 + \frac{1}{2L}\|\nabla f(x_t)\|^2.$$

Finally, we combine the terms:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2.$$

This shows that the function value decreases by at least $\frac{1}{2L}\|\nabla f(x_t)\|^2$ at each step, which proves the sufficient decrease in gradient descent with step size $\frac{1}{L}$.

3.3 Smooth Convex Functions

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, with a global minimum x^*. Furthermore, suppose that f is smooth with parameter L. By choosing the step size $\gamma := \frac{1}{L}$, gradient descent yields the following bound on the function value after T iterations:

$$f(x_T) - f(x^*) \leq \frac{L}{2T}\|x_0 - x^*\|^2, \quad T > 0.$$

Proof: we begin with the Vanilla Analysis inequality:

$$\sum_{t=0}^{T-1} (f(x_t) - f(x^*)) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 + \frac{1}{2\gamma}\|x_0 - x^*\|^2.$$

This time, we can bound the squared gradients using the sufficient decrease property:

$$\frac{1}{2L}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq \sum_{t=0}^{T-1} (f(x_t) - f(x_{t+1})),$$

which is equivalent to:

$$\frac{1}{2L}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq f(x_0) - f(x_T).$$

Now, substituting $\gamma = \frac{1}{L}$ into the Vanilla Analysis inequality, we have:

$$\sum_{t=0}^{T-1} (f(x_t) - f(x^*)) \leq \frac{1}{2L}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 + \frac{L}{2}\|x_0 - x^*\|^2.$$

By the sufficient decrease property, we also know that:

$$\frac{1}{2L}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq f(x_0) - f(x_T).$$

Thus, we can rewrite the above inequality as:

$$\sum_{t=0}^{T-1} (f(x_t) - f(x^*)) \leq f(x_0) - f(x_T) + \frac{L}{2}\|x_0 - x^*\|^2.$$

Rewriting this further, we have:

$$\sum_{t=1}^{T} (f(x_t) - f(x^*)) \leq \frac{L}{2}\|x_0 - x^*\|^2.$$

Since the last iterate x_T is the best (by the sufficient decrease property), we obtain:

$$f(x_T) - f(x^*) \leq \frac{1}{T}\sum_{t=1}^{T} (f(x_t) - f(x^*)) \leq \frac{L}{2T}\|x_0 - x^*\|^2.$$

We define: $R^2 := \|x_0 - x^*\|^2$. In this case, we have the following condition for the number of iterations $T \geq \frac{R^2 L}{2\varepsilon}$ required to achieve a certain error ε in $O(1/\varepsilon)$ steps:

$$\text{error} \leq \frac{LR^2}{2T} \leq \varepsilon.$$

Thus, for an error of 0.01, the number of iterations needed is: $T = 50 \cdot R^2 L$. This is significantly better than the $10,000 \cdot R^2 B^2$ iterations required in the Lipschitz case.

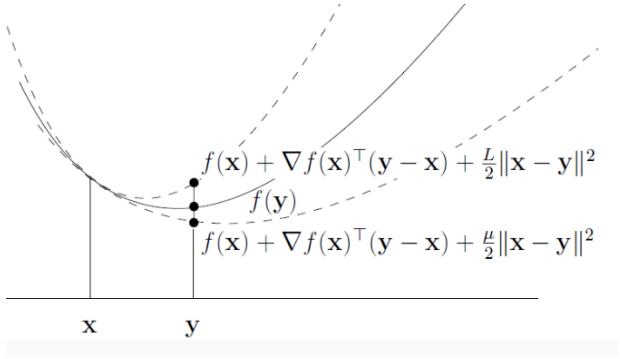
3.4 Strongly Convex Functions

Definition let $f : \text{dom}(f) \to \mathbb{R}$ be a differentiable function, $X \subseteq \text{dom}(f)$ convex and $\mu \in \mathbb{R}^+, \mu > 0$. Function f is called strongly convex (with parameter μ) over X if:

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|x - y\|^2, \quad \forall x, y \in X.$$

Lemma if f is strongly convex with parameter $\mu > 0$, then f is strictly convex and has a unique global minimum (non necessary smooth for that you must explicitly assume that the gradient is Lipschitz-continuous with a constant L)

Strong Convexity: for any x, the graph of f is above a not too flat tangent paraboloid at $(x, f(x))$.



We want to show: $\lim_{t\to\infty} x_t = x^*$. **Vanilla Analysis:**

$$\nabla f(x_t)^\top (x_t - x^*) = \frac{\gamma}{2}\|\nabla f(x_t)\|^2 + \frac{1}{2\gamma}\left(\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2\right).$$

Coming from strong convexity with $y = x^*$ and $x = x_t$ we have:

$$f(x^*) \geq f(x_t) + \nabla f(x_t)^T (x^* - x_t) + \frac{\mu}{2}\|x_t - x^*\|^2$$

$$\nabla f(x_t)^\top (x_t - x^*) \geq f(x_t) - f(x^*) + \frac{\mu}{2}\|x_t - x^*\|^2.$$

Putting it together:

$$f(x_t) - f(x^*) \leq \frac{1}{2\gamma}\left(\gamma^2\|\nabla f(x_t)\|^2 + \|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2\right) - \frac{\mu}{2}\|x_t - x^*\|^2.$$

Rewriting:

$$\|x_{t+1} - x^*\|^2 \leq 2\gamma(f(x^*) - f(x_t)) + \gamma^2\|\nabla f(x_t)\|^2 + (1 - \mu\gamma)\|x_t - x^*\|^2.$$

$2\gamma(f(x^*) - f(x_t)) + \gamma^2\|\nabla f(x_t)\|^2 =$ *costant 0 we can ignore it*

Theorem let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum x^*; suppose that f is smooth with parameter L and strongly convex with parameter $\mu > 0$. Choosing $\gamma := \frac{1}{L}$, gradient descent with arbitrary x_0 satisfies the following two properties:

- Squared distances to x^* are geometrically decreasing:

$$\|x_{t+1} - x^*\|^2 \le \left(1 - \frac{\mu}{L}\right) \|x_t - x^*\|^2, \quad t \ge 0. \tag{*}$$

- The absolute error after T iterations is exponentially small in T:

$$f(x_T) - f(x^*) \le \frac{L}{2} \left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2, \quad T > 0. \tag{**}$$

Proof of (*): bounding the noise:

$$\begin{aligned} 2\gamma(f(x^*) - f(x_t)) + \gamma^2 \|\nabla f(x_t)\|^2 &= \frac{2}{L}(f(x^*) - f(x_t)) + \frac{1}{L^2}\|\nabla f(x_t)\|^2 \\ \textit{strong convexity} &\le \frac{2}{L}(f(x_{t+1}) - f(x_t)) + \frac{1}{L^2}\|\nabla f(x_t)\|^2 \\ \textit{sufficient decrease} &\le -\frac{1}{L^2}\|\nabla f(x_t)\|^2 + \frac{1}{L^2}\|\nabla f(x_t)\|^2 = 0 \end{aligned}$$

Hence, the noise is non-positive, and we get:

$$\|x_{t+1} - x^*\|^2 \le (1 - \mu\gamma)\|x_t - x^*\|^2 = \left(1 - \frac{\mu}{L}\right) \|x_t - x^*\|^2.$$

Proof of (**): from (*) we get:

$$\|x_T - x^*\|^2 \le \left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2.$$

Using smoothness together with $\nabla f(x^*) = 0$:

$$\begin{aligned} f(x_T) - f(x^*) &\le \nabla f(x^*)^\top (x_T - x^*) + \frac{L}{2}\|x_T - x^*\|^2 \\ &= 0 + \frac{L}{2}\|x_T - x^*\|^2 \end{aligned}$$

Putting it together:

$$f(x_T) - f(x^*) \le \frac{L}{2}\|x_T - x^*\|^2 \le \frac{L}{2} \left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2.$$

Conclusion: having $R^2 := \|x_0 - x^*\|^2$ and $T \ge \frac{L}{\mu} \ln\left(\frac{R^2 L}{2\epsilon}\right)$:

$$\text{error} \le \frac{L}{2} \left(1 - \frac{\mu}{L}\right)^T R^2 \le \varepsilon$$

To reach absolute error at most ε, we only need $O(\log(1/\varepsilon))$. Getting $\frac{L}{\mu} \ln(50 \cdot R^2 L)$ iterations for error 0.01 as opposed to $50 \cdot R^2 L$ in the smooth case.

4 Projected Gradient Descent

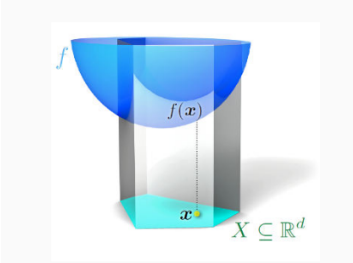
Constrained Optimization Problem

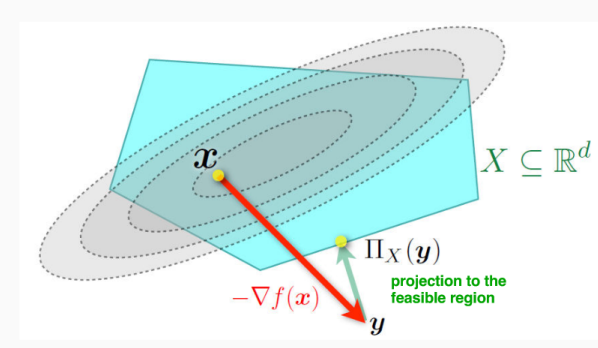
$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X \end{aligned}$$

Solving Constrained Optimization Problems

- (A) Projected Gradient Descent.
- (B) Transform it into an unconstrained problem.

Idea: project onto X after every step $\Pi_X(y) := \arg\min_{x \in X} \|x - y\|$ (find the closest solution in the feasible region).





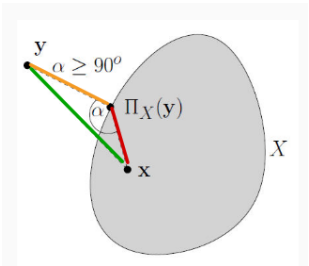
Projected Gradient Descent:

$$\begin{aligned} y_{t+1} &:= x_t - \gamma \nabla f(x_t), \\ x_{t+1} &:= \Pi_X(y_{t+1}) := \arg\min_{x \in X} \|x - y_{t+1}\|^2, \end{aligned}$$

for timesteps $t = 0, 1, \ldots,$ and stepsize $\gamma \ge 0$.

Fact: let $X \subseteq \mathbb{R}^d$ be closed and convex, $x \in X$, and $y \in \mathbb{R}^d$. Then:

- (a) $(x - \Pi_X(y))^\top (y - \Pi_X(y)) \le 0$. (NB: $\cos\alpha < 0$)
- (b) $\|x - \Pi_X(y)\|^2 + \|y - \Pi_X(y)\|^2 \le \|x - y\|^2$.



The same number of steps as gradient over $\mathbb{R}^d$: the previous proofs has to be adapted and each step involves a projection onto X may be not efficient

- **Lipschitz convex functions over X:** $O\left(1/\varepsilon^2\right)$ steps.
- **Smooth convex functions over X:** $O\left(1/\varepsilon\right)$ steps.
- **Smooth and strongly convex functions over X:** $O(\log(1/\varepsilon))$ steps.

5 Composite Optimization Problems

Consider objective functions composed as:

$$f(x) := g(x) + h(x)$$

where g is a nice function, whereas h is a simple additional term, which, however, doesn't satisfy the assumptions of niceness that we used in the convergence analysis so far. In particular, an important case is when h **is not differentiable**.

The classical gradient step for minimizing g:

$$x_{t+1} = \arg\min_y \left[g(x_t) + \nabla g(x_t)^\top (y - x_t) + \frac{1}{2\gamma}\|y - x_t\|^2 \right].$$

For the stepsize $\gamma = \frac{1}{L}$, it exactly **minimizes the local quadratic model of** g at our current iterate x_t, **formed by the smoothness property** with parameter L. Now for $f = g + h$, keep the same for g, and add h unmodified:

$$x_{t+1} := \arg\min_y \left[g(x_t) + \nabla g(x_t)^\top (y - x_t) + \frac{1}{2\gamma}\|y - x_t\|^2 + h(y) \right]$$

Since $g(x_t)$ is independent of y, we can drop it from the minimization, which leaves:

$$x_{t+1} := \arg\min_y \left[\nabla g(x_t)^\top (y - x_t) + \frac{1}{2\gamma}\|y - x_t\|^2 + h(y) \right].$$

It can be rewritten as:

$$\begin{aligned}
&\frac{1}{2\gamma}\|y - x_t\|^2 + \nabla g(x_t)^\top (y - x_t) \\
&= \frac{1}{2\gamma}(y - x_t)^\top (y - x_t) + \nabla g(x_t)^\top (y - x_t) \\
&= \frac{1}{2\gamma}\left(y^\top y - 2y^\top x_t + x_t^\top x_t\right) + \nabla g(x_t)^\top (y - x_t) \\
&= \frac{1}{2\gamma} y^\top y - \frac{1}{2\gamma} \cdot 2y^\top x_t + \frac{1}{2\gamma} x_t^\top x_t + \nabla g(x_t)^\top y - \nabla g(x_t)^\top x_t \\
&= \frac{1}{2\gamma} y^\top y - \frac{1}{\gamma} y^\top x_t + \frac{1}{2\gamma} x_t^\top x_t + \nabla g(x_t)^\top y - \nabla g(x_t)^\top x_t \\
&= \frac{1}{2\gamma} y^\top y + \nabla g(x_t)^\top y - \frac{1}{\gamma} y^\top x_t + \text{constant terms} \\
&= \frac{1}{2\gamma} y^\top y + \nabla g(x_t)^\top y - \frac{1}{\gamma} x_t^\top y + \text{constant terms} \\
&= \frac{1}{2\gamma} y^\top y + \left(\nabla g(x_t) - \frac{1}{\gamma} x_t\right)^\top y + \text{constant terms} \\
&= \frac{1}{2\gamma} y^\top y + \frac{1}{\gamma}\left(\gamma \nabla g(x_t) - x_t\right)^\top y + \text{constant terms} \\
&= \frac{1}{2\gamma} y^\top y - \frac{1}{\gamma} y^\top \left(x_t - \gamma \nabla g(x_t)\right) + \text{constant terms} \\
&= \frac{1}{2\gamma}\left(y - (x_t - \gamma \nabla g(x_t))\right)^\top \left(y - (x_t - \gamma \nabla g(x_t))\right) + \text{constant terms} \\
&= \frac{1}{2\gamma}\|y - (x_t - \gamma \nabla g(x_t))\|^2 + \text{constant terms}.
\end{aligned}$$

To complete the square, we start with the expression:

$$\begin{aligned}
&\frac{1}{2\gamma} y^\top y - \frac{1}{\gamma} y^\top (x_t - \gamma \nabla g(x_t)) + \text{constant terms} \\
&= \frac{1}{2\gamma} y^\top y - \frac{2}{2\gamma} y^\top (x_t - \gamma \nabla g(x_t)) + \text{constant terms} \\
&= \frac{1}{2\gamma}\left(y^\top y - 2y^\top (x_t - \gamma \nabla g(x_t))\right) + \text{constant terms} \\
&= \frac{1}{2\gamma}\left(y - (x_t - \gamma \nabla g(x_t))\right)^\top \left(y - (x_t - \gamma \nabla g(x_t))\right) + \text{constant terms}
\end{aligned}$$

The **proximal gradient descent** update:

$$x_{t+1} := \arg\min_y \left[g(x_t) + \nabla g(x_t)^\top (y - x_t) + \frac{1}{2\gamma}\|y - x_t\|^2 + h(y) \right]$$

$$= \arg\min_y \left[\frac{1}{2\gamma}\|y - (x_t - \gamma \nabla g(x_t))\|^2 + h(y) \right]$$

5.1 The Proximal Gradient Descent Algorithm

An iteration of proximal gradient descent is defined as:

$$x_{t+1} = \text{prox}_{h,\gamma}(x_t - \gamma \nabla g(x_t)),$$

where the proximal mapping for a given function h, and parameter $\gamma > 0$ is defined as

$$\text{prox}_{h,\gamma}(z) := \arg\min_y \left\{ \frac{1}{2\gamma}\|y - z\|^2 + h(y) \right\}.$$

Thus with h and γ fixed we have:

$$\begin{aligned}
x_{t+1} &= \text{prox}_{h,\gamma}(x_t - \gamma \nabla g(x_t)) \\
&:= \arg\min_y \left\{ \frac{1}{2\gamma}\|y - (x_t - \gamma \nabla g(x_t))\|^2 + h(y) \right\}
\end{aligned}$$

The update step can be equivalently written as

$$x_{t+1} = x_t - \gamma G_{h,\gamma}(x_t),$$

for

$$G_{h,\gamma}(x) := \frac{1}{\gamma}\left(x - \text{prox}_{h,\gamma}(x - \gamma \nabla g(x))\right),$$

being the so-called **generalized gradient of** f.

5.2 A Generalization of Gradient Descent

- When $h \equiv 0$ recovers standard gradient descent
- When $h \equiv \iota_X$ recovers projected gradient descent

Given a closed convex set X, the **indicator function** of the set X is defined as the convex function

$$\iota_X : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$$

$$x \mapsto \iota_X(x) := \begin{cases} 0 & \text{if } x \in X \\ +\infty & \text{otherwise.} \end{cases}$$

The proximal mapping becomes

$$\text{prox}_{h,\gamma}(z) := \arg\min_y \left\{ \frac{1}{2\gamma}\|y - z\|^2 + \iota_X(y) \right\} = \arg\min_{y \in X} \|y - z\|^2.$$

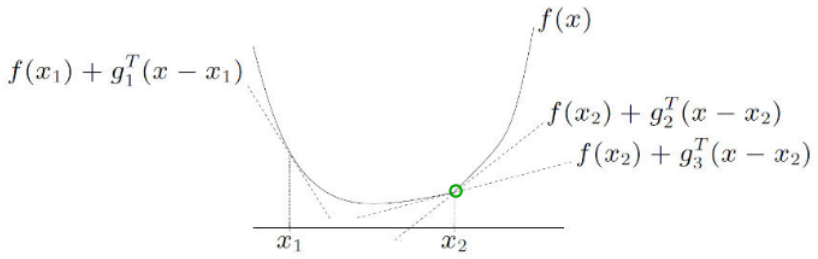
5.3 Subgradients

What if f is not differentiable?

Definition: A vector $g \in \mathbb{R}^d$ is a subgradient of a function f at a point x if:

$$f(y) \geq f(x) + g^\top (y - x) \quad \forall y \in \text{dom}(f).$$

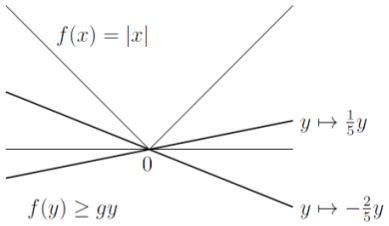
The set of all subgradients of f at x is denoted by $\partial f(x) \subseteq \mathbb{R}^d$ and is called the **subdifferential**.



Subgradient Condition at $x = 0$: A vector $g \in \mathbb{R}^d$ is a subgradient of a function f at $x = 0$ if:

$$f(y) \geq f(0) + g(y - 0) = gy \quad \forall y \in \text{dom}(f).$$

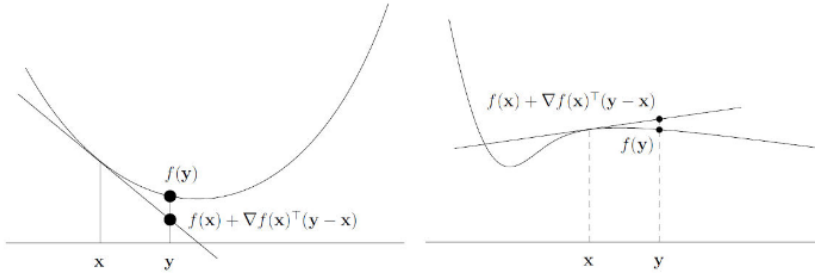
The subdifferential at $x = 0$ is given by $\partial f(0) = [-1, 1]$.



Lemma: if $f : \text{dom}(f) \to \mathbb{R}$ is differentiable at $x \in \text{dom}(f)$, then

$$\partial f(x) \subseteq \{\nabla f(x)\}.$$

This means that there is either exactly one subgradient $\nabla f(x)$ or no subgradient at all.



Lemma: a function $f : \text{dom}(f) \to \mathbb{R}$ is convex if and only if $\text{dom}(f)$ is convex and $\partial f(x) \neq \emptyset$ for all $x \in \text{dom}(f)$. ***convex = subgradients everywhere***

Lemma (convex and lipschitz): let $f : \text{dom}(f) \to \mathbb{R}$ be convex, $\text{dom}(f)$ open, and $B \in \mathbb{R}^+$. Then the following two statements are equivalent:

1. $\|g\| \leq B$ for all $x \in \text{dom}(f)$ and all $g \in \partial f(x) \subseteq \mathbb{R}^d$
2. $|f(x) - f(y)| \leq B\|x - y\|$ for all $x, y \in \text{dom}(f)$

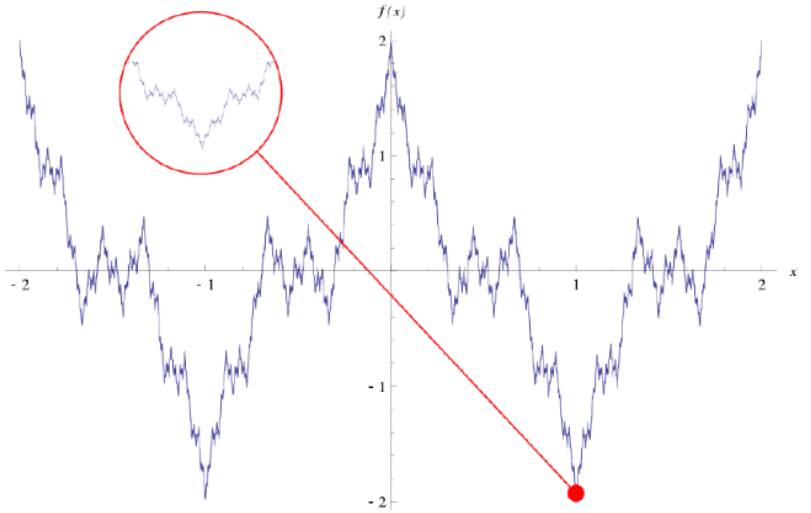
Lemma: suppose that $f : \text{dom}(f) \to \mathbb{R}$ and $x \in \text{dom}(f)$. If $0 \in \partial f(x)$, then x is a global minimum.

Proof: by the definition of subgradients, $g = 0 \in \partial f(x)$ gives

$$f(y) \geq f(x) + g^\top (y - x) = f(x)$$

for all $y \in \text{dom}(f)$. Thus, x is a global minimum.

How wild can a non-differentiable convex function be? **Weierstrass function:** a function that is continuous everywhere but differentiable nowhere



Theorem: a convex function $f : \mathrm{dom}(f) \to \mathbb{R}$ is differentiable almost everywhere. In other words:
- the set of points where f is non-differentiable has measure 0 (no volume)
- $\forall x \in \mathrm{dom}(f)$ and $\forall \varepsilon > 0$, there exists a point x' such that

$$\|x - x'\| \leq \varepsilon \quad \text{and} \quad f \text{ is differentiable at } x'.$$

5.4 Subgradient performances

Subgradient Descent: Choose $x_0 \in \mathbb{R}^d$ and let $g_t \in \partial f(x_t)$, the update rule is:

$$x_{t+1} := x_t - \gamma_t g_t$$

for times $t = 0, 1, \ldots$, and stepsizes $\gamma_t \geq 0$. The stepsize can vary with time. This variability is also possible in (projected) gradient descent.

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and B-Lipschitz continuous with a global minimum x^*; furthermore, suppose that $\|x_0 - x^*\| \leq R$. Choosing the constant stepsize:

$$\gamma := \frac{R}{B\sqrt{T}},$$

subgradient descent yields:

$$\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) - f(x^*) \leq \frac{RB}{\sqrt{T}}$$

Proof: the proof is identical to the one proposed in the theorem for convex and differentiable f with a global minimum x^*, $\|x_0 - x^*\| \leq R$, and $\|\nabla f(x)\| \leq B$ for all x. The only differences are:
- In the vanilla analysis, now use $g_t \in \partial f(x_t)$ instead of $g_t = \nabla f(x_t)$.
- The inequality $f(x_t) - f(x^*) \leq g_t^\top (x_t - x^*)$ now follows from the subgradient property instead of the first-order characterization of convexity.

Theorem (Nesterov): for any $T \leq d - 1$ and starting point x_0, there exists a function f in the problem class of B-Lipschitz functions over $\mathbb{R}^d$, such that any (sub)gradient method has an objective error at least

$$f(x_T) - f(x^*) \geq \frac{RB}{2(1 + \sqrt{T+1})}.$$

Definition: let $f : \mathrm{dom}(f) \to \mathbb{R}$ be convex, $\mu \in \mathbb{R}^+$, $\mu > 0$. Function f is called strongly convex (with parameter μ) if:

$$f(y) \geq f(x) + g^\top (y - x) + \frac{\mu}{2}\|x - y\|^2, \quad \forall x, y \in \mathrm{dom}(f), \forall g \in \partial f(x).$$

Lemma: let $f : \mathrm{dom}(f) \to \mathbb{R}$ be convex, $\mathrm{dom}(f)$ open, $\mu \in \mathbb{R}^+$, $\mu > 0$. Function f is strongly convex with parameter μ if and only if the function $f_\mu : \mathrm{dom}(f) \to \mathbb{R}$ defined by:

$$f_\mu(x) = f(x) - \frac{\mu}{2}\|x\|^2, \quad x \in \mathrm{dom}(f) \quad \textit{is convex}$$

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be strongly convex with parameter $\mu > 0$ and let x^* be the unique global minimum of f. With decreasing stepsize

$$\gamma_t = \frac{2}{\mu(t+1)}, \quad t > 0,$$

subgradient descent yields:

$$f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T} t \cdot \boldsymbol{x}_t\right) - f(\boldsymbol{x}^*) \leq \frac{2B^2}{\mu(T+1)}$$

where $B = \max_{1=1}^{T} \|g_t\|$.

Discussion:
- The weighted average of iterates achieves the bound, with later iterates receiving more weight.
- This bound is independent of the initial distance $\|x_0 - x^*\|$, but not entirely: B typically depends on $\|x_0 - x^*\|$ (for example, $B = O(\|x_0 - x^*\|)$ for quadratic functions).
- We can only hope that B is small, which can be checked while running the algorithm.
- What if we don't know the parameter μ of strong convexity?
  - In practice, try several values of μ and select the best solution obtained.

Lipschitz convex f:	$O(1/\varepsilon^2)$
Strong Convexity	$O(1/\varepsilon)$

6 Stochastic Gradient Descent

Many objective functions are **sum-structured**:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

where $f_i(x)$ is the cost function of the i-th observation, taken from a training set of n observations. Evaluating $\nabla f(x)$ for a sum-structured function is computationally expensive, as it involves the sum of n gradients:

$$\nabla f(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x), \quad x \textit{ fixed}, \quad \textit{on all } n \textit{ data}$$

Example: Each $f_i(x)$ could represent the cost of the i-th data point in a machine learning model, making the evaluation costly when the dataset is large.

Stochastic Gradient Descent: choose an initial point x_0. At each iteration, sample $i \in [n]$ uniformly at random and update for times $t = 0, 1, 2, \ldots$, and stepsizes $\gamma_t \geq 0$:

$$x_{t+1} := x_t - \gamma_t \nabla f_i(x_t), \quad \textit{sensitive to noise}$$

Key Insight: only update using the gradient of f_i instead of the full gradient of a random observation i. This makes each iteration n times cheaper than in full gradient descent. The vector $g_t := \nabla f_i(x_t)$ is called a ***stochastic gradient***. While g_t is a vector of d random variables, we also simply refer to it as a random variable.

Unbiasedness of Stochastic Gradients: in stochastic gradient descent, we cannot directly use convexity to apply the inequality

$$f(x_t) - f(x^*) \leq g_t^\top (x_t - x^*),$$

which holds for full gradient methods. This is because the inequality may or may not hold depending on the specific realization of the stochastic gradient g_t. Instead, we rely on the fact that this inequality holds *in expectation*. The key property we exploit is that g_t is an *unbiased estimate* of the full gradient $\nabla f(x_t)$. Formally, we have:

$$\mathbb{E}[g_t \mid x_t = x] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x), \quad x \in \mathbb{R}^d.$$

This means that, on average, the stochastic gradient g_t provides a correct approximation of the true gradient $\nabla f(x_t)$, which allows us to carry out convergence analysis based on expectations.

The Inequality $f(x_t) - f(x^*) \leq g_t^\top (x_t - x^*)$ holds in Expectation: for any fixed x, the linearity of conditional expectations gives:

$$\mathbb{E}\left[g_t^\top (x - x^*) \mid x_t = x\right] = \mathbb{E}\left[g_t \mid x_t = x\right]^\top (x - x^*) = \nabla f(x)^\top (x - x^*).$$

The event $\{x_t = x\}$ (current iterate equal to x) can occur only for x in some finite set X, since x_t is determined by the choice of indices in all previous iterations.

Partition Theorem:

$$\mathbb{E}\left[g_t^\top (x_t - x^*)\right] = \sum_{x \in X} \mathbb{E}\left[g_t^\top (x - x^*) \mid x_t = x\right] \text{prob}(x_t = x)$$

$$= \sum_{x \in X} \nabla f(x)^\top (x - x^*) \text{prob}(x_t = x) = \mathbb{E}\left[\nabla f(x_t)^\top (x_t - x^*)\right].$$

Hence,

$$\mathbb{E}\left[g_t^\top (x_t - x^*)\right] = \mathbb{E}\left[\nabla f(x_t)^\top (x_t - x^*)\right] \geq \mathbb{E}\left[f(x_t) - f(x^*)\right].$$

- **Expectation Decomposition:** The expectation $\mathbb{E}[g_t^\top (x_t - x^*)]$ is split over all possible values $x \in X$, which are the outcomes for x_t. For each possible x, the conditional expectation $\mathbb{E}[g_t^\top (x - x^*) \mid x_t = x]$ is computed, weighted by the probability of $x_t = x$.
- **Unbiased Gradient:** Since g_t is an unbiased estimator of the gradient $\nabla f(x)$, the conditional expectation simplifies to $\nabla f(x)^\top (x - x^*)$. Thus, the sum becomes the expectation of the true gradient:

$$\mathbb{E}[g_t^\top (x_t - x^*)] = \sum_{x \in X} \nabla f(x)^\top (x - x^*) \text{prob}(x_t = x) = \mathbb{E}[\nabla f(x_t)^\top (x_t - x^*)].$$

- **Main Result:** This shows that the expectation of the stochastic gradient behaves like the expectation of the true gradient:

$$\mathbb{E}[g_t^\top (x_t - x^*)] = \mathbb{E}[\nabla f(x_t)^\top (x_t - x^*)].$$

- **Convexity Property:** Finally, by the convexity of f, the gradient satisfies:

$$\mathbb{E}[\nabla f(x_t)^\top (x_t - x^*)] \geq \mathbb{E}[f(x_t) - f(x^*)],$$

  meaning that, in expectation, the stochastic gradient method still respects the fundamental properties of convex functions.

6.1 Bounded Stochastic Gradients

Theorem: Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, x^* a global minimum; furthermore, suppose that $\|x_0 - x^*\| \leq R$ and that $\mathbb{E}[\|g_t\|^2] \leq B^2$ for all t. Choosing the constant stepsize:

$$\gamma := \frac{R}{B\sqrt{T}}$$

stochastic gradient descent yields:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[f(x_t)] - f(x^*) \leq \frac{RB}{\sqrt{T}}.$$

Proof: using vanilla analysis (with g_t as the stochastic gradient), we start with the following inequality:

$$\sum_{t=0}^{T-1} g_t^\top (x_t - x^*) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1} \|g_t\|^2 + \frac{1}{2\gamma}\|x_0 - x^*\|^2.$$

Next, we take expectations and apply convexity in expectation:

$$\sum_{t=0}^{T-1} E[f(x_t) - f(x^*)] \leq \sum_{t=0}^{T-1} E[g_t^\top (x_t - x^*)].$$

Substituting the previous inequality into this gives us:

$$\sum_{t=0}^{T-1} E[f(x_t) - f(x^*)] \leq \frac{\gamma}{2}\sum_{t=0}^{T-1} E[\|g_t\|^2] + \frac{1}{2\gamma}\|x_0 - x^*\|^2.$$

Now, since we know that $E[\|g_t\|^2] \leq B^2$ for all t, we can simplify further:

$$\sum_{t=0}^{T-1} E[f(x_t) - f(x^*)] \leq \frac{\gamma}{2}B^2 T + \frac{1}{2\gamma}R^2.$$

The result follows by optimizing γ.

6.2 Convergence Rate Comparison: SGD vs GD

Classic GD: for vanilla analysis, we assumed that $\|\nabla f(x)\|^2 \leq B_{GD}^2$ for all $x \in \mathbb{R}^d$, where B_{GD} was a constant. Thus, for a sum-objective:

$$\left\| \frac{1}{n} \sum_i \nabla f_i(x) \right\|^2 \leq B_{GD}^2 \quad \forall x.$$

SGD: assuming the same for the expected squared norms of our stochastic gradients, now called B_{SGD}^2:

$$\frac{1}{n} \sum_i \|\nabla f_i(x)\|^2 \leq B_{SGD}^2 \quad \forall x.$$

By Jensen's inequality for $\|\cdot\|_2$:

$$B_{GD}^2 \approx \left\| \frac{1}{n} \sum_i \nabla f_i(x) \right\|^2 \leq \frac{1}{n} \sum_i \|\nabla f_i(x)\|^2 \approx B_{SGD}^2.$$

Therefore, B_{GD}^2 can be smaller than B_{SGD}^2, but they are often comparable. The analysis is very similar if larger mini-batches are used.

6.3 Tame Strong Convexity

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and strongly convex with parameter $\mu > 0$; let x^* be the unique global minimum of f. With decreasing stepsize

$$\gamma_t := \frac{2}{\mu(t+1)},$$

stochastic gradient descent yields:

$$\mathbb{E}\left[f\left(\frac{2}{T(T+1)} \sum_{t=1}^{T} t \cdot x_t \right) - f(x^*) \right] \leq \frac{2B^2}{\mu(T+1)},$$

where $B^2 := \max_{t=1}^{T} \mathbb{E}[\|g_t\|^2]$. This result is almost the same as that for subgradient descent, but it is in expectation.

Proof: take expectations over the vanilla analysis before summing up (with varying stepsize γ_t):

$$\mathbb{E}\left[g_t^\top (x_t - x^*) \right] = \frac{\gamma_t}{2} \mathbb{E}[\|g_t\|^2] + \frac{1}{2\gamma_t} \left(\mathbb{E}[\|x_t - x^*\|^2] - \mathbb{E}[\|x_{t+1} - x^*\|^2] \right).$$

Using strong convexity in expectation:

$$\mathbb{E}\left[g_t^\top (x_t - x^*) \right] = \mathbb{E}\left[\nabla f(x_t)^\top (x_t - x^*) \right] \geq \mathbb{E}[f(x_t) - f(x^*)] + \frac{\mu}{2} \mathbb{E}[\|x_t - x^*\|^2].$$

Putting it all together (with $\mathbb{E}[\|g_t\|^2] \leq B^2$):

$$\mathbb{E}[f(x_t) - f(x^*)] \leq \frac{B^2 \gamma_t}{2} + \frac{1}{\gamma_t} \cdot \frac{\mu}{2} \mathbb{E}[\|x_t - x^*\|^2] - \frac{1}{2\gamma_t} \mathbb{E}[\|x_{t+1} - x^*\|^2].$$

The proof continues as for subgradient descent, this time with expectations.

6.4 Mini-Batch SGD

Instead of using a single element f_i, we consider the average of several of them:

$$\tilde{g}_t := \frac{1}{m} \sum_{j=1}^{m} g_t^j.$$

Extreme cases:

$$m = 1 \iff \text{SGD as originally defined;}$$
$$m = n \iff \text{full gradient descent.}$$

Benefit: gradient computation can be naively parallelized. **Variance intuition:** taking an average of many independent random variables reduces the variance. Thus, for a larger size of the mini-batch m, $\tilde{g}_t$ will be closer to the true gradient, in expectation:

$$\begin{aligned} \mathbb{E}[\|\tilde{g}_t - \nabla f(x_t)\|^2] &= \mathbb{E}\left[\left\| \frac{1}{m} \sum_{j=1}^{m} g_t^j - \nabla f(x_t) \right\|^2 \right] \\ &= \frac{1}{m} \mathbb{E}\left[\|g_t^1 - \nabla f(x_t)\|^2 \right] \\ &= \frac{1}{m} \mathbb{E}\left[\|g_t^1\|^2 \right] - \frac{1}{m} \|\nabla f(x_t)\|^2 \leq \frac{B^2}{m} \end{aligned}$$

Using a modification of the SGD analysis, we can relate this quantity to the convergence rate of full gradient descent.

6.5 Gradient Descent Variants

Batch Gradient Descent: choose $x_0 \in \mathbb{R}^d$. For $t = 0, 1, \ldots$:

$$x_{t+1} := x_t - \gamma \nabla f(x_t) = x_t - \gamma \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_t),$$

with learning rate $\gamma \geq 0$.

Stochastic gradient descent: choose $x_0 \in \mathbb{R}^d$. For $t = 0, 1, \ldots$: sample $i \in \{1, \ldots, N\}$ uniformly at random:

$$x_{t+1} := x_t - \gamma_t \nabla f_i(x_t),$$

with learning rate $\gamma_t \geq 0$.

Stochastic gradient descent with Mini-Batch: choose $x_0 \in \mathbb{R}^d$. For $t = 0, 1, \ldots$: choose a random subset $\{h_1, \ldots, h_m\} \subseteq \{1, \ldots, N\}$:

$$x_{t+1} := x_t - \gamma_t \frac{1}{m} \sum_{j=1}^{m} \nabla f_{h_j}(x_t),$$

with learning rate $\gamma_t \geq 0$.

For problems that are not necessarily differentiable, we modify SGD to use a subgradient of f_i in each iteration. The update of stochastic subgradient descent is given by:

$$\textit{Sample } i \in [n] \textit{ uniformly at random, let } g_t \in \partial f_i(x_t)$$

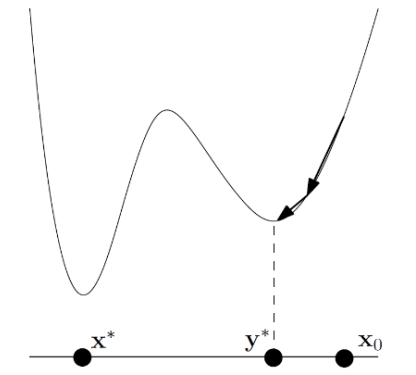
$$x_{t+1} := x_t - \gamma_t g_t.$$

In other words, we are using an unbiased estimate of a subgradient at each step, $\mathbb{E}[g_t | x_t] \in \partial f(x_t)$. Convergence occurs in $O(1/\varepsilon^2)$, by using the subgradient property at the beginning of the proof, where convexity was applied.

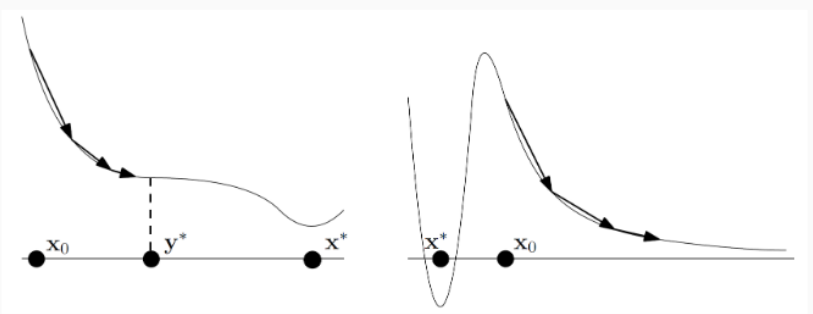
For constrained optimization, our theorem for the SGD convergence in $O(1/\varepsilon^2)$ steps directly extends to constrained problems as well. After every step of SGD, a projection back to X is applied as usual. The resulting algorithm is called **projected SGD.**

6.6 Gradient Descent in the Non-Convex World

May get stuck in a **local minimum** and miss the global minimum.



Even if there is a unique local minimum (equal to the global min) we: may get stuck in a saddle point, run off to infinity, possibly encounter other bad behaviors.

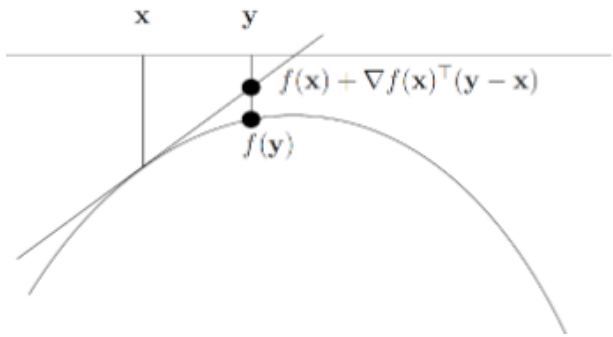


Under favorable conditions, we sometimes can say something useful about the behavior of gradient descent, even on non-convex functions.

Recall: smoth function doesnt require convex functions.

Concave functions a function f is called concave if $-f$ is convex. For all x, the graph of a differentiable concave function is below the tangent hyperplane at x. This implies that concave functions are smooth with $L = 0 \ldots$ but they are considered boring from an optimization point of view (no global minimum), as gradient descent may run off to infinity.

$$\begin{aligned} f(y) &\leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|x - y\|^2 \\ &\leq f(x) + \nabla f(x)^\top (y - x) \end{aligned}$$



Lemma: let $f : \mathrm{dom}(f) \to \mathbb{R}$ be twice differentiable, with $X \subseteq \mathrm{dom}(f)$ a convex set, and $\|\nabla^2 f(x)\| \leq L$ for all $x \in X$, where $\|\cdot\|$ is the spectral norm. Then f is smooth with parameter L over X. ***Bounded Hessians*** $\Rightarrow$ ***Smooth***

Examples: all quadratic functions $f(x) = x^T Ax + b^T x + c$, $f(x) = \sin(x)$ (many global minima).

6.7 Convergence of Gradients

we will prove that

$$\|\nabla f(x_t)\|_2 \to 0 \quad \text{as} \quad t \to \infty$$

at the same rate as

$$f(x_t) - f(x^*) \to 0$$

in the convex case. However, in the non-convex case, $f(x_t) - f(x^*)$ itself may not converge to 0. It may or may not mean that we converge to a critical point, where

$$\nabla f(y^*) = 0.$$

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum x^*; furthermore, suppose that f is smooth with parameter L. Choosing stepsize

$$\gamma := \frac{1}{L},$$

gradient descent yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq \frac{2L}{T}(f(x_0) - f(x^*)), \quad T > 0.$$

In particular,

$$\|\nabla f(x_t)\|^2 \leq \frac{2L}{T}(f(x_0) - f(x^*))$$

for some $t \in \{0, \ldots, T-1\}$. Also,

$$\lim_{t\to\infty} \|\nabla f(x_t)\|^2 = 0.$$

Proof: sufficient decrease does not require convexity:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2, \quad t \geq 0.$$

Rewriting this, we get:

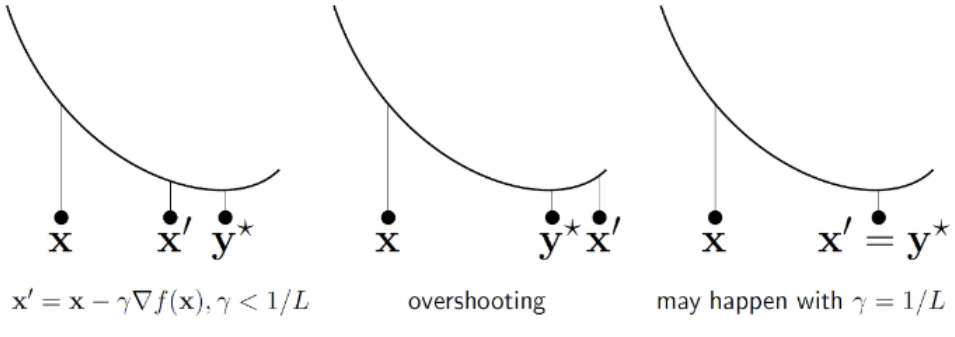
$$\|\nabla f(x_t)\|^2 \leq 2L(f(x_t) - f(x_{t+1})).$$

Now, consider the telescoping sum:

$$\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq 2L(f(x_0) - f(x_T)) \leq 2L(f(x_0) - f(x^*)).$$

The statement follows by dividing by T.

No Overshooting: in the smooth setting, and with stepsize $\frac{1}{L}$, gradient descent cannot overshoot, i.e., pass a critical point.



6.8 SGD in conclusion

In **Stochastic Gradient Descent (SGD)**, the update to the current iterate x_t is performed using the gradient of just one randomly chosen observation i from the dataset, rather than the full gradient computed over all observations. Here's how this relates to the various concepts discussed:

1. **Single Observation**: the stochastic gradient g_t is calculated using the cost function f_i corresponding to a randomly selected observation i from the dataset. This means that instead of computing the gradient of the overall objective function (which averages over all observations), SGD updates the parameters based on the gradient of just one observation.
2. **Current Iterate**: the gradient is evaluated at the current iterate x_t. This means that you use the parameter values from the previous step x_t when calculating the gradient for the selected observation.
3. **Sampling**: The observation i is sampled uniformly at random from the set of observations during each iteration t. This randomness is what makes the gradient "stochastic," as it introduces variability into the updates based on which observation is selected.
4. **Sum-Structured Objective Functions**: many optimization problems involve objective functions that are structured as the average of individual cost functions for each observation. This means calculating the full gradient involves a potentially expensive computation involving all n observations.
5. **Cost-Efficiency of SGD**: by sampling one observation at a time, SGD significantly reduces the computational cost per iteration. Instead of evaluating $\nabla f(x)$ as the sum of all gradients, it uses $g_t = \nabla f_i(x_t)$, where i is a randomly chosen index. This leads to each iteration being n times cheaper than full gradient descent.
6. **Unbiasedness of Stochastic Gradients**: the expectation of the stochastic gradient provides a valid estimate of the true gradient. Even though g_t can vary because it depends on the specific observation i chosen, its expected value still converges to the true gradient $\nabla f(x_t)$. This is crucial because it allows the analysis to rely on expected values, rather than specific realizations.
7. **Inequality in Expectation**: because of the unbiased nature of g_t, we can use it in place of the true gradient in the convergence analysis, leading to results that reflect the expected behavior of the optimization process. The inequalities you derived show that even with the randomness introduced by sampling, the algorithm maintains convergence properties typical of deterministic methods.

Bounded Stochastic Gradients	$O(1/\varepsilon^2)$
Tame Strong Convexity	$O(1/\varepsilon)$

7 Accelerated Gradient Descent

Is Gradient Descent the Best Possible Algorithm? maybe gradient descent is not the best possible algorithm? After all, it is just some algorithm that uses gradient information.

First-Order Method: a first-order method is an algorithm that gains access to f only via an oracle that is able to return values of f and ∇f at arbitrary points. Gradient descent is a specific first-order method.

Optimal First-Order Method for Smooth Convex Functions: according to Nemirovski and Yudin (1979), every first-order method needs, in the worst case, $\Omega(1/\sqrt{\varepsilon})$ steps (gradient evaluations) in order to achieve an additive error of ε on smooth functions. There is a gap between $O(1/\varepsilon)$ (gradient descent) and the lower bound.

Nesterov's Accelerated Gradient Descent: Nesterov (1983) showed that there is a first-order method that needs only $O(1/\sqrt{\varepsilon})$ steps on smooth convex functions. By the lower bound of Nemirovski and Yudin, this is the best possible algorithm. The algorithm is known as ***Nesterov's accelerated gradient descent***.

7.1 Nesterov's Accelerated Gradient Descent

Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex, differentiable, and smooth with parameter L. Choose $\boldsymbol{z}_0 = \boldsymbol{y}_0 = \boldsymbol{x}_0$ arbitrarily. For $t \geq 0$, set

$$y_{t+1} := x_t - \frac{1}{L}\nabla f(x_t)$$

$$z_{t+1} := z_t - \frac{t+1}{2L}\nabla f(x_t)$$

$$x_{t+1} := \frac{t+1}{t+3}y_{t+1} + \frac{2}{t+3}z_{t+1}.$$

Steps: perform a ***smooth step*** from x_t to y_{t+1}, perform a ***more aggressive step*** from z_t to z_{t+1}, the next iterate x_{t+1} is a weighted average of y_{t+1} and z_{t+1} where we compensate for the more aggressive step by giving z_{t+1} a relatively low weight.

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex, differentiable with a global minimum x^*; furthermore, suppose that f is smooth with parameter L. Accelerated gradient descent yields

$$f(y_T) - f(x^*) \leq \frac{2L||z_0 - x^*||^2}{T(T+1)}, \quad T > 0.$$

To reach an error at most ε, accelerated gradient descent therefore only needs $O(1/\sqrt{\varepsilon})$ steps instead of $O(1/\varepsilon)$. Recall the bound for ***gradient descent***:

$$f(x_T) - f(x^*) \leq \frac{L}{2T}||x_0 - x^*||^2, \quad T > 0.$$

7.2 1-Dimensional Case: Newton-Raphson Method

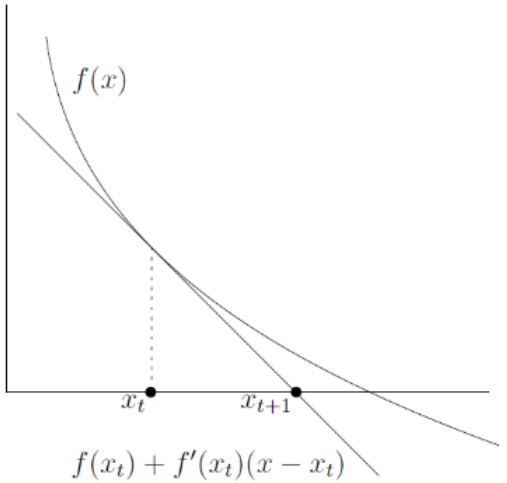
The Newton-Raphson method is a root-finding algorithm that iteratively improves approximations to the ***roots (or zeroes)*** of a differentiable, real-valued function. Given a function $f : \mathbb{R} \to \mathbb{R}$, ***the goal*** is to find a point x where $f(x) = 0$, i.e., a root of the function.

$$f'(x_t) = \frac{f(x_{t+1}) - f(x_t)}{x_{t+1} - x_t}$$

$$f(x_{t+1}) = 0$$

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}, \quad t \geq 0$$

$$x_{t+1} \text{ solves: } \quad f(x_t) + f'(x_t)(x - x_t) = 0$$



Remark: at each iteration, Newton's method seeks to solve the linear approximation of f at x_t. Using the first-order Taylor expansion of f around x_t, we approximate:

$$f(x) \approx f(x_t) + f'(x_t)(x - x_t).$$

Setting this approximation to zero to find the root of the linearized function, we obtain:

$$f(x_t) + f'(x_t)(x_{t+1} - x_t) = 0.$$

Solving for x_{t+1}, we derive the update rule:

$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}.$$

So the method constructs a tangent line to the function f at x_t, and the next iterate x_{t+1} is the point where this tangent line crosses the x-axis. The process continues until the sequence $\{x_t\}$ converges to a root of $f(x)$.

1-dimensional case: find a global minimum x^* of a differentiable convex function $f : \mathbb{R} \to \mathbb{R}$. This can equivalently be stated as searching for a zero of the derivative f': apply the Newton-Raphson method to f'. Update step:

$$f'(x_t) + f''(x_t)(x_{t+1} - x_t) = 0.$$

$$x_{t+1} := x_t - \frac{f'(x_t)}{f''(x_t)} = x_t - f''(x_t)^{-1} f'(x_t) \quad \text{(needs } f \text{ to be twice differentiable).}$$

d-dimensional case: Newton's method for min a convex function $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_{t+1} := x_t - \nabla^2 f(x_t)^{-1} \nabla f(x_t).$$

7.3 Newton's Method = Adaptive Gradient Descent

General update scheme: where $H(x) \in \mathbb{R}^{d \times d}$ is some matrix.

$$x_{t+1} = x_t - H(x_t) \nabla f(x_t),$$

Newton's method:	$H = \nabla^2 f(x_t)^{-1}$
Gradient descent:	$H = \gamma I$

7.4 Convergence in One Step on Quadratic Functions

A nondegenerate quadratic function is a function of the form

$$f(x) = \frac{1}{2} x^\top M x - q^\top x + c,$$

where $M \in \mathbb{R}^{d \times d}$ is an invertible symmetric matrix, $q \in \mathbb{R}^d$, and $c \in \mathbb{R}$. Let $x^* = M^{-1} q$ be the unique solution of $\nabla f(x) = 0 \Rightarrow x^*$ is the unique global minimum if f is convex.

Lemma: on nondegenerate quadratic functions, with any starting point $x_0 \in \mathbb{R}^d$, Newton's method yields $x_1 = x^*$.

Proof: we have $\nabla f(x) = Mx - q$ (this implies $x^* = M^{-1} q$) and $\nabla^2 f(x) = M$. Hence:

$$x_1 = x_0 - \nabla^2 f(x_0)^{-1} \nabla f(x_0) = x_0 - M^{-1}(M x_0 - q) = M^{-1} q = x^*.$$

7.5 Minimizing the Second-Order Taylor Approximation

Alternative interpretation of Newton's Method: each step of Newton's method minimizes the local second-order Taylor approximation.

Lemma: let f be convex and twice differentiable at $x_t \in \mathrm{dom}(f)$, with $\nabla^2 f(x_t) \succ 0$ being invertible. The vector x_{t+1} resulting from the Newton step satisfies:

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^d} \left(f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2} (x - x_t)^\top \nabla^2 f(x_t)(x - x_t) \right).$$

Under suitable conditions, and starting close to the global minimum, Newton's method will reach a distance of at most ε to the minimum within $\log\log(1/\varepsilon)$ steps.

Remark: this is much faster than anything we have seen so far, however we need to start close to the minimum already $\Rightarrow$ this is a local convergence result. Global convergence results that hold for every starting point were unknown for Newton's method until very recently.

Theorem: let $f : \mathrm{dom}(f) \to \mathbb{R}$ be convex with a unique global minimum x^*. Suppose there is a ball $X \subseteq \mathrm{dom}(f)$ with center x^*, such that:

1. **Bounded inverse Hessians:** there exists a real number $\mu > 0$ such that:

$$\|\nabla^2 f(x)^{-1}\| \leq \frac{1}{\mu}, \quad \forall x \in X.$$

2. **Lipschitz continuous Hessians:** there exists a real number $B > 0$ such that:

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq B \|x - y\|, \quad \forall x, y \in X.$$

Then, for $x_t \in X$ and x_{t+1} resulting from the Newton step, we have:

$$\|x_{t+1} - x^*\| \leq \frac{B}{2\mu} \|x_t - x^*\|^2.$$

Corollary: with the assumptions and terminology of the convergence theorem, and if $\|x_0 - x^*\| \leq \frac{\mu}{B}$, then Newton's method yields:

$$\|x_T - x^*\| \leq \frac{\mu}{B} \left(\frac{1}{2}\right)^{2T-1}, \quad T \geq 0.$$

Starting close to the global minimum, we will reach a distance of at most ε to the minimum within $O(\log\log(1/\varepsilon))$ steps. Almost constant Hessians close to optimality, so f behaves almost like a quadratic function, which has truly constant Hessians and allows Newton's method to converge in one step.

7.6 Strong Convexity $\Rightarrow$ Bounded inverse Hessians

One way to ensure bounded inverse Hessians is to require strong convexity over X.

Lemma: let $f : \mathrm{dom}(f) \to \mathbb{R}$ be twice differentiable and strongly convex with parameter μ over an open convex subset $X \subseteq \mathrm{dom}(f)$, meaning that

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2} \|x - y\|^2, \quad \forall x, y \in X.$$

Then $\nabla^2 f(x)$ is invertible, and $\|\nabla^2 f(x)^{-1}\| \leq \frac{1}{\mu}$ for all $x \in X$, where $\|\cdot\|$ is the spectral norm.

7.7 Downside of Newton's Method

A significant computational bottleneck occurs in each step of Newton's method:
- Compute and invert the Hessian matrix, or
- Solve the linear system $\nabla^2 f(x_t) \Delta x = -\nabla f(x_t)$ for the next step Δx.

The matrix/system has size $d \times d$, taking up to $O(d^3)$ time to invert or solve. In many applications, d is large, making the method computationally expensive.

7.8 The Secant Method

Another iterative method for finding zeros in dimension 1: we start from the Newton-Raphson step

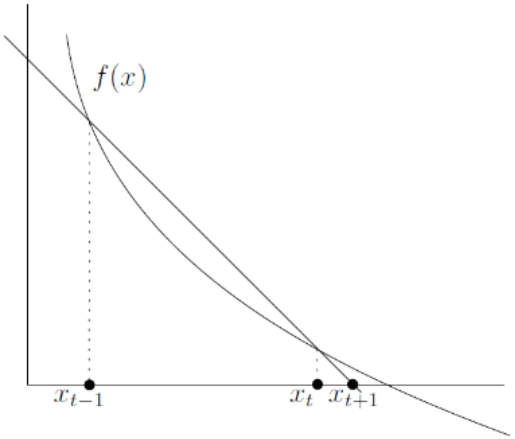
$$x_{t+1} := x_t - \frac{f(x_t)}{f'(x_t)}.$$

Now, using a finite difference approximation[1] of $f'(x_t)$:

$$f'(x_t) \approx \frac{f(x_t) - f(x_{t-1})}{x_t - x_{t-1}},$$

we obtain the **Secant Method**:

$$x_{t+1} := x_t - \frac{f(x_t)(x_t - x_{t-1})}{f(x_t) - f(x_{t-1})}.$$



Secant method for root-finding and optimization: the secant method constructs a line through two points, $(x_{t-1}, f(x_{t-1}))$ and $(x_t, f(x_t))$, and finds the next iterate, x_{t+1}, where this line intersects the x-axis. This gives us a derivative-free version of the Newton-Raphson method:

$$x_{t+1} := x_t - \frac{f(x_t)(x_t - x_{t-1})}{f(x_t) - f(x_{t-1})}.$$

Secant method for optimization: we can also optimize a differentiable univariate function f by applying the secant method to f':

$$x_{t+1} := x_t - \frac{f'(x_t)(x_t - x_{t-1})}{f'(x_t) - f'(x_{t-1})}.$$

This is a second-derivative-free version of Newton's method for optimization. Can we generalize this to higher dimensions to obtain a Hessian-free version of Newton's method in $\mathbb{R}^d$?

[1] For small $|x_t - x_{t-1}|$

The secant condition: we apply a finite difference approximation to f'' in one dimension:

$$H_t := \frac{f'(x_t) - f'(x_{t-1})}{x_t - x_{t-1}} \approx f''(x_t) \quad \iff \quad f'(x_t) - f'(x_{t-1}) = H_t(x_t - x_{t-1}).$$

Newton's method	$x_{t+1} := x_t - f''(x_t)^{-1} f'(x_t)$
Secant method	$x_{t+1} := x_t - H_t^{-1} f'(x_t)$

Generalizing to higher dimensions: in higher dimensions, let $H_t \in \mathbb{R}^{d \times d}$ be a symmetric matrix that satisfies the ***d-dimensional secant condition***:

$$\nabla f(x_t) - \nabla f(x_{t-1}) = H_t(x_t - x_{t-1}).$$

The secant method step then becomes:

$$x_{t+1} := x_t - H_t^{-1} \nabla f(x_t).$$

7.9 Quasi-Newton Methods

Newton's method	$x_{t+1} := x_t - \nabla^2 f(x_t)^{-1} \nabla f(x_t)$
Secant method	$x_{t+1} := x_t - H_t^{-1} \nabla f(x_t)$
Secant condition	$\nabla f(x_t) - \nabla f(x_{t-1}) = H_t(x_t - x_{t-1})$

If f is twice differentiable, the secant condition and a first-order approximation of $\nabla f(x)$ at x_t yield:

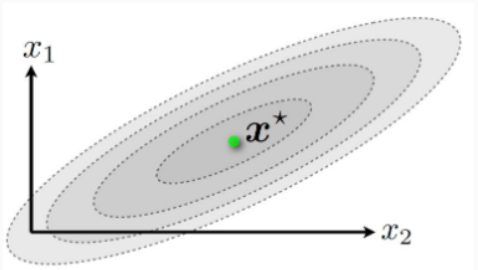
$$\nabla f(x_t) - \nabla f(x_{t-1}) = H_t(x_t - x_{t-1}) \approx \nabla^2 f(x_t)(x_t - x_{t-1}).$$

Thus, we might hope that $H_t \approx \nabla^2 f(x_t)$, meaning that the secant method approximates Newton's method. In the one-dimensional case ($d = 1$), there is a unique number H_t that satisfies the secant condition. In higher dimensions ($d > 1$), the secant condition has infinitely many symmetric solutions for H_t, as it represents an underdetermined linear system.

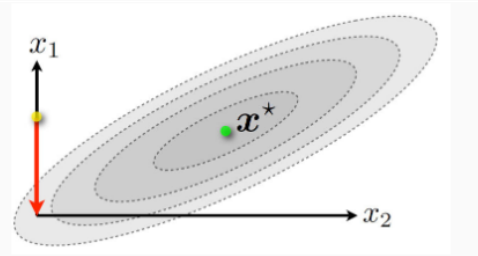
Quasi-newton methods: any scheme that, in each step of the secant method, chooses a symmetric H_t that satisfies the secant condition defines a Quasi-Newton method. These methods approximate the Hessian matrix $\nabla^2 f(x_t)$ in order to avoid the computational cost of directly computing or inverting the Hessian at each step.

8 Coordinate Descent

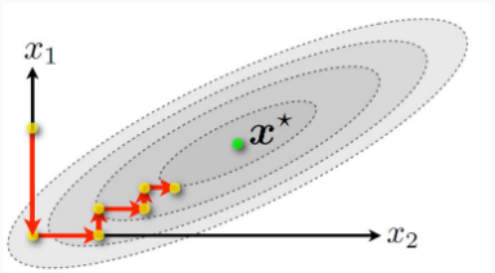
Goal: find $\boldsymbol{x^*} \in \mathbb{R}^d$ minimizing $f(\boldsymbol{x})$, updating one coordinate at a time, while keeping others fixed.



Example: $d = 2$.



Example: $d = 2$.



Example: $d = 2$.

Coordinate descent modifies only one coordinate of the current iterate x_t at each step. The process selects a coordinate $i_t \in \{1, \ldots, d\}$ uniformly at random and updates it using the following rule:

$$x_{t+1} := x_t + \gamma e_{i_t},$$

where e_{i_t} is the standard basis vector corresponding to the i_t-th coordinate. There are two main variants of this method:

1. **Gradient-based step-size**:

$$x_{t+1} := x_t - \frac{1}{L}\nabla_{i_t} f(x_t) e_{i_t}.$$

In this variant, the step-size is determined using the gradient of the function in the direction of the selected coordinate. To proceed, we assume coordinate-wise smoothness, which is expressed as:

$$f(x + \gamma e_i) \leq f(x) + \gamma \nabla_i f(x) + \frac{L}{2}\gamma^2 \quad \forall x \in \mathbb{R}^d,\ \forall \gamma \in \mathbb{R},\ \forall i.$$

This condition is equivalent to a coordinate-wise Lipschitz gradient:

$$|\nabla_i f(x + \gamma e_i) - \nabla_i f(x)| \leq L|\gamma| \quad \forall x \in \mathbb{R}^d,\ \forall \gamma \in \mathbb{R},\ \forall i.$$

Additionally, we assume strong convexity for the function f, which ensures better convergence properties of the method.

2. **Exact coordinate minimization**: This variant involves solving the single-variable minimization problem:

$$\arg\min_{\gamma \in \mathbb{R}} f(x_t + \gamma e_{i_t})$$

in closed form, which directly minimizes the function along the chosen coordinate.

Convergence theorem: let f be coordinate-wise smooth with constant L, and strongly convex with parameter $\mu > 0$. Then, coordinate descent with a step-size of $\frac{1}{L}$, following the update rule

$$x_{t+1} := x_t - \frac{1}{L}\nabla_{i_t} f(x_t) e_{i_t},$$

where the active coordinate i_t is chosen uniformly at random, has an expected linear convergence rate given by:

$$\mathbb{E}[f(x_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(x_0) - f^*].$$

Proof: plugging the update rule into the smoothness condition, we have:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}|\nabla_{i_t} f(x_t)|^2.$$

Taking expectation with respect to i_t, we obtain:

$$\mathbb{E}[f(x_{t+1})] \leq f(x_t) - \frac{1}{2L}\mathbb{E}[|\nabla_{i_t} f(x_t)|^2].$$

Since i_t is chosen uniformly at random from $\{1, \ldots, d\}$, we can express the expectation as:

$$\mathbb{E}[|\nabla_{i_t} f(x_t)|^2] = \frac{1}{d}\sum_{i=1}^{d} |\nabla_i f(x_t)|^2.$$

Thus, we have:

$$\mathbb{E}[f(x_{t+1})] \leq f(x_t) - \frac{1}{2dL}||\nabla f(x_t)||^2.$$

Using the lemma that strongly convex functions satisfy the Polyak-Lojasiewicz (PL) inequality:

$$\frac{1}{2}||\nabla f(x)||^2 \geq \mu(f(x) - f^*) \quad \forall x,$$

we can substitute this into our inequality. We obtain:

$$\mathbb{E}[f(x_{t+1})] - f^* \leq f(x_t) - f^* - \frac{\mu}{dL}(f(x_t) - f^*).$$

This simplifies to:

$$\mathbb{E}[f(x_{t+1}) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)(f(x_t) - f^*).$$

This completes the proof.

Definition: a function f satisfies the Polyak-Lojasiewicz Inequality (PL) if there exists some $\mu > 0$ such that:

$$\frac{1}{2}||\nabla f(x)||^2 \geq \mu(f(x) - f^*), \quad \forall x.$$

Lemma (Strong Convexity $\Rightarrow$ PL): let f be strongly convex with parameter $\mu > 0$. Then f satisfies the PL inequality for the same μ.

Corollary: For minimization of a function f that is: coordinate-wise smooth with constant L, satisfies the Polyak-Lojasiewicz (PL) inequality and has a non-empty solution set X^*. The expected convergence rate for random coordinate descent with a step-size of $\frac{1}{L}$ is given by:

$$E[f(x_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(x_0) - f^*].$$

Remark:

- **Random coordinate selection**: while random selection of coordinates works well, it is not always the most efficient method.
- **Individual smoothness constants**: for each coordinate i, we can define a smoothness constant L_i:

$$f(x + \gamma e_i) \leq f(x) + \gamma \nabla_i f(x) + \frac{L_i}{2}\gamma^2.$$

- **Modified selection probabilities**: instead of uniformly selecting coordinates, you can use modified probabilities:

$$P[i_t = i] = \frac{L_i}{\sum_i L_i}.$$

- **Faster convergence rate**: if you use a step-size of $\frac{1}{L_i}$ based on the individual smoothness constants, the expected convergence rate improves to:

$$E[f(x_t) - f^*] \leq \left(1 - \frac{\mu}{d\bar{L}}\right)^t [f(x_0) - f^*],$$

where $\bar{L} = \frac{1}{d}\sum_{i=1}^d L_i$ and $\bar{L} \ll L = \max_i L_i$.

Steepest Coordinate descent: a coordinate selection rule that maximizes the gradient magnitude:

$$i_t := \arg\max_{i \in [d]} |\nabla_i f(x_t)|$$

Deterministic vs random: steepest coordinate descent has the same convergence rate as random coordinate descent. To show this, we can use the inequality:

$$\max_i |\nabla_i f(x)|^2 \geq \frac{1}{d}\sum_i |\nabla_i f(x)|^2.$$

Since the algorithm is deterministic, there is no need to take expectations in the proof.

Corollary: Steepest coordinate descent with a step-size of $\frac{1}{L}$ has the linear convergence rate given by:

$$E[f(x_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(x_0) - f^*].$$

9 Optimization

9.1 No gradients

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

Random search:

Pick a random direction $\boldsymbol{d}_t \in \mathbb{R}^d$.

Perform line-search to find:

$$\gamma := \arg\min_{\gamma \in \mathbb{R}} f(\boldsymbol{x}_t + \gamma \boldsymbol{d}_t)$$

Update the iterate:

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

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

Same as what we obtained for gradient descent, now with an **extra factor of** d (d can be huge). We can apply the same analysis for different function classes, as before:

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

In all cases, we see the complexity is d times the complexity of gradient descent for the same function class.

9.2 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.

9.3 Reinforcement learning

The system dynamics are described by (we assume that f is fixed, but unknown):

$$s_{t+1} = f(s_t, a_t, e_t),$$

where s_t is the ***state*** of the system, a_t is the ***control action***, and e_t is some ***random noise***. The ***control policy*** $a_t := \pi(a_1, \dots, a_{t-1}, s_0, \dots, s_t)$ maps the history of actions and states to the current action a_t. The optimization aims to find the best sequence of these actions.

$$a_t := \pi(a_1, \dots, a_{t-1}, s_0, \dots, s_t),$$

Goal: we are searching for a ***control policy*** that maximizes the **expected total reward** over the time horizon 0 to N, considering the uncertainty from e_t:

$$\max_{a_t} \mathbb{E}_{e_t} \left[\sum_{t=0}^{N} R_t(s_t, a_t) \right]$$
$$s.t. \quad s_{t+1} = f(s_t, a_t, e_t)$$
$$s_0 \text{ given}$$

Adagrad

Adagrad (Adaptive Gradient Algorithm) is an adaptive variant of Stochastic Gradient Descent (SGD). It adjusts the learning rate for each parameter based on the historical gradients, allowing for more effective training.

$$\text{pick a stochastic gradient } \mathbf{g}_t$$
$$\text{update } [G_t]_i := \sum_{s=0}^{t} ([\mathbf{g}_s]_i)^2 \ \forall \ i$$
$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[G_t]_i}} [\mathbf{g}_t]_i \ \forall \ i$$

A standard choice of g_t is (where f_j is the objective function evaluated on a mini-batch or a single training example, allowing for quicker updates to the model compared to using the full dataset):

$$g_t := \nabla f_j(x_t) \quad \text{for sum-structured objective functions } f = \sum_j f_j.$$

Adaptive learning rates: adagrad's update rule allows it to choose an ***adaptive, coordinate-wise learning rate***. This means that features that have frequent updates will have smaller learning rates over time, whereas features that are less frequently updated will have larger learning rates.

Remarks: robustness to feature scaling, faster convergence, learning rate decay (learning rate becames too small over time due to the accumulated squared gradients), variants of Adagrad (Adadelta, Adam, RMSprop).

Explanation:

- $\boldsymbol{[G_t]_i}$ **:** this update rule calculates the sum of the squares of each component of the stochastic gradients over all previous iterations s from 0 to t, by accumulating this information Adagrad adapts the learning rate for each parameter based on the historical information of its updates.
- $\boldsymbol{[x_{t+1}]_i}$: we are updating the parameters for the next iteration denoted by x_{t+1}, the term $\frac{[g_t]_i}{\sqrt{[G_t]_i}}$ computes the effective learning rate for the i-th parameter: the stochastic gradient $[g_t]_i$ for the i-th parameter indicates the direction to update the parameter, $\sqrt{[G_t]_i}$ normalizes the gradient by the accumulated squared gradient for the i-th parameter. As the accumulated gradient grows, the effective learning rate for that parameter decreases, which helps prevent overfitting and allows for a more stable convergence

Adam

Adam (Adaptive Moment Estimation) computes adaptive learning rates for each parameter from estimates of first and second moments of the gradients, allowing for more effective and efficient training.

$$\text{pick a stochastic gradient } \mathbf{g}_t$$
$$\mathbf{m}_t := \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t \quad \text{(momentum term)}$$
$$[\mathbf{v}_t]_i := \beta_2 [\mathbf{v}_{t-1}]_i + (1 - \beta_2)([\mathbf{g}_s]_i)^2 \ \forall \ i \quad \text{(2nd-order statistics)}$$
$$[\mathbf{x}_{t+1}]_i := [\mathbf{x}_t]_i - \frac{\gamma}{\sqrt{[\mathbf{v}_t]_i}} [\mathbf{m}_t]_i \ \forall \ i.$$

Adaptive learning rates: Adam's update rule allows it to choose an ***adaptive, coordinate-wise learning rate*** based on both the first moment (mean) and the second moment (variance) of the gradients. This means that parameters that receive consistent gradients will have their learning rates adjusted accordingly.

Remarks: Adam is robust to feature scaling, converges faster than standard SGD, and is effective in handling sparse gradients. It also includes mechanisms to prevent the learning rates from becoming too small over time.

Explanation:

- $\boldsymbol{m_t}$ **:** this is the first moment estimate (the momentum term) calculated as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

  where β_1 is the decay rate for the first moment (typically set to 0.9). This term accumulates the exponentially decaying average of past gradients, giving more weight to recent gradients.
- $\boldsymbol{v_t}$ **:** this is the second moment estimate (the variance term) calculated as:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

  where β_2 is the decay rate for the second moment (commonly set to 0.999). This term accumulates the exponentially decaying average of past squared gradients, allowing the algorithm to adapt the learning rate based on the variance of the gradients.

- $[\boldsymbol{x_{t+1}}]_{\boldsymbol{i}}$ **:** the parameter update for the next iteration is given by:

$$[x_{t+1}]_i = [x_t]_i - \frac{\gamma}{\sqrt{v_t} + \varepsilon} m_t$$

where γ is the base learning rate and ε is a small constant (typically 10^{-8}) added for numerical stability. The term $\frac{m_t}{\sqrt{v_t}+\varepsilon}$ provides a normalized update that adapts the step size based on the past gradients and their variances.

SignSGD

SignSGD is a communication-efficient variant of Stochastic Gradient Descent (SGD) that utilizes only the sign of each gradient entry, making it particularly suitable for distributed training scenarios. A standard update rule in SignSGD is given by:

pick a stochastic gradient $\mathbf{g}_t$

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

where g_t is the stochastic gradient at iteration t, and γ_t is the learning rate, which may be adjusted based on the L_1 norm of the gradient, $||g_t||_1$.

9.4 Non Linear Programming (Recall)

Definition: consider a very general optimization problem of the form:

$$\begin{aligned} \min_x \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_j(\boldsymbol{x}) = b_j, \quad j = 1, \dots, m \\ & h_i(\boldsymbol{x}) \le d_i, \quad i = 1, \dots, p \end{aligned}$$

or the equivalent more concise form ***(1)***:

$$\begin{aligned} \min_x \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{b} \\ & \boldsymbol{h}(\boldsymbol{x}) \le \boldsymbol{d} \end{aligned}$$

where $f, g_j, h_i : \mathbb{R}^n \to \mathbb{R}$. In the special case when all functions f, g_j, h_i are linear, problem is a linear program (LP). When some of the functions f, g_j, h_i are nonlinear, problem is a Nonlinear Program (NLP).

Assumption: we consider the class of NLPs where the functions f, g_i, h_j are once or twice continuously differentiable.

Remark: the optimality conditions for linear and convex quadratic programs extend to this more general context. In particular, for a general NLP, the theory described below applies to ***local minima***.

Theorem First-order necessary conditions: suppose f, g_i, h_j are continuously differentiable. If a point $x^* \in X$ is a local minimum of (1) and satisfies the linear independence constraint qualification, then there exist some Lagrange multipliers $y \in \mathbb{R}^m$ and $s \in \mathbb{R}^p$ such that:

$$\nabla f(x^*) + \sum_{j=1}^{m} y_j \nabla g_j(x^*) + \sum_{i=1}^{p} s_i \nabla h_i(x^*) = 0$$

$$s \ge 0$$

$$s_i(h_i(x^*) - d_i) = 0 \quad \text{for } i = 1, \dots, p$$

9.5 Algorithms

We next describe three main algorithmic approaches to solving a constrained optimization problem, namely:

1. **the Generalized Reduced Gradient**
2. **the Sequential Quadratic Programming**
3. **the Interior-Point Methods**

Generalized reduced gradient

The main idea behind the generalized reduced gradient method is to reduce a constrained problem to a sequence of unconstrained problems in a space of lower dimension.

- Consider the special case when the equality constraints are linear:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \quad (3)$$

  for some $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.
- Without loss of generality, we may assume that $\boldsymbol{A}$ has ***full row rank***, as otherwise either some constraints are redundant or the problem is infeasible.
- Since $\boldsymbol{A}$ has full rank, we can partition both $\boldsymbol{A}$ and $\boldsymbol{x}$ as follows:

$$\boldsymbol{A} = [\boldsymbol{A}_B \, \boldsymbol{A}_N] \quad \text{and} \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_B \\ \boldsymbol{x}_N \end{bmatrix}^\top$$

  for some subset $B \subseteq \{1, \dots, n\}$ such that $\boldsymbol{A}_B$ is non-singular. Therefore,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \Leftrightarrow \quad \boldsymbol{A}_B\boldsymbol{x}_B + \boldsymbol{A}_N\boldsymbol{x}_N = \boldsymbol{b} \quad \Leftrightarrow \quad \boldsymbol{x}_B = \boldsymbol{A}_B^{-1}(\boldsymbol{b} - \boldsymbol{A}_N\boldsymbol{x}_N).$$

- Consequently, problem (3) is equivalent to the following reduced space unconstrained minimization problem:

$$\min_{\boldsymbol{x}_N} \hat{f}(\boldsymbol{x}_N)$$

  where $\hat{f}(\boldsymbol{x}_N) = f(\boldsymbol{A}_B^{-1}(\boldsymbol{b} - \boldsymbol{A}_N\boldsymbol{x}_N), \boldsymbol{x}_N)$.
- Now, consider a more general program with nonlinear equality constraints:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{b} \end{aligned} \quad (4)$$

- We can extend the above approach by approximating the nonlinear equality constraints with their first-order Taylor approximation. More precisely, suppose the current point is $\boldsymbol{x}^k$. Consider the modification of (4) obtained by replacing $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{b}$ with its first-order Taylor approximation:

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{x}^k) + \nabla \boldsymbol{g}(\boldsymbol{x}^k)^\top (\boldsymbol{x} - \boldsymbol{x}^k) = \boldsymbol{b}. \quad (G) \end{aligned}$$

Observe that (G) is of the form (3) and is thus amenable to the type of reduced space approach described above. The following algorithm describes a template for a generalized reduced gradient approach to problem (4).

Algorithm Generalized reduced gradient

1: choose $\mathbf{x}^0$
2: **for** $k = 0, 1, \ldots$ **do**
3: solve the linearized constraints problem (G) to find a search direction $\Delta \mathbf{x}^k$
4: choose a step length $\alpha > 0$ and set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \Delta \mathbf{x}^k$
5: **end for**

Sequential Quadratic Programming

The central idea of sequential quadratic programming (SQP) is to capitalize on algorithms for quadratic programming to solve more general nonlinear programming problems of the form (1).

Given a current iterate $\boldsymbol{x}^k$, problem (1) can be approximated with the following quadratic program:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}^k) + \nabla f(\boldsymbol{x}^k)^\top (\boldsymbol{x} - \boldsymbol{x}^k) + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}^k)^\top \boldsymbol{B}^k (\boldsymbol{x} - \boldsymbol{x}^k) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}^k) + \nabla \boldsymbol{g}(\boldsymbol{x}^k)^\top (\boldsymbol{x} - \boldsymbol{x}^k) = \boldsymbol{b} \\ & \boldsymbol{h}(\boldsymbol{x}^k) + \nabla \boldsymbol{h}(\boldsymbol{x}^k)^\top (\boldsymbol{x} - \boldsymbol{x}^k) \leq \boldsymbol{d} \quad (S) \end{aligned}$$

where $\boldsymbol{B}^k = \nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{s}^k)$ is the Hessian of the Lagrangian function with respect to the $\boldsymbol{x}$ variables, and $(\boldsymbol{y}^k, \boldsymbol{s}^k)$ is the current estimate of the vector of Lagrange multipliers.

The following algorithm describes a template for a sequential quadratic programming approach to problem (4). At each iteration, the step length α is typically chosen to balance both the goals of reducing the objective function and satisfying the constraints.

Algorithm Sequential quadratic programming

1: choose $\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0$
2: **for** $k = 0, 1, \ldots$ **do**
3: solve the quadratic program (S) to find a search direction $(\Delta \mathbf{x}^k, \Delta \mathbf{y}^k, \Delta \mathbf{s}^k)$
4: choose a step length $\alpha > 0$ and set $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k) + \alpha(\Delta \mathbf{x}^k, \Delta \mathbf{y}^k, \Delta \mathbf{s}^k)$
5: **end for**

Interior-Point Methods

Interior-point methods generate a sequence of iterates that satisfy some inequalities strictly, and each iteration of the algorithm aims to make progress towards satisfying the optimality conditions (2).

- The algorithm becomes more elaborate for nonlinear programs due to the nonlinearities in the constraints.
- Given a vector $\boldsymbol{s} \in \mathbb{R}^p$, let $\boldsymbol{S} \in \mathbb{R}^{p \times p}$ denote the diagonal matrix defined by $\boldsymbol{S}_{ii} = s_i$ for $i = 1, \ldots, n$, and let $\mathbf{1} \in \mathbb{R}^p$ denote the vector whose components are all 1s.
- The optimality conditions (2) can be restated as:

$$\begin{bmatrix} \nabla f(\boldsymbol{x}) + \nabla \boldsymbol{g}(\boldsymbol{x}) \boldsymbol{y} + \nabla \boldsymbol{h}(\boldsymbol{x}) \boldsymbol{s} \\ \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{b} \\ \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{z} - \boldsymbol{d} \\ \boldsymbol{SZ1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{s}, \boldsymbol{z} \geq 0.$$

- Given $\mu > 0$, let $(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu), \boldsymbol{s}(\mu))$ be the solution to the following perturbed version of the above optimality conditions:

$$\begin{bmatrix} \nabla f(\boldsymbol{x}) + \nabla \boldsymbol{g}(\boldsymbol{x}) \boldsymbol{y} + \nabla \boldsymbol{h}(\boldsymbol{x}) \boldsymbol{s} \\ \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{b} \\ \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{z} - \boldsymbol{d} \\ \boldsymbol{SZ1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mu \mathbf{1} \end{bmatrix}, \quad \boldsymbol{s}, \boldsymbol{z} \geq 0.$$

- The first condition above can be written as $\boldsymbol{r}_\mu(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s}) = 0$ for the residual vector:

$$\boldsymbol{r}_\mu(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s}) = \begin{bmatrix} \nabla f(\boldsymbol{x}) + \nabla \boldsymbol{g}(\boldsymbol{x}) \boldsymbol{y} + \nabla \boldsymbol{h}(\boldsymbol{x}) \boldsymbol{s} \\ \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{b} \\ \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{z} - \boldsymbol{d} \\ \boldsymbol{SZ1} - \mu \mathbf{1} \end{bmatrix}.$$

- The central path is the set $\{(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu), \boldsymbol{s}(\mu)) : \mu > 0\}$.
- Under suitable assumptions, $(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu), \boldsymbol{s}(\mu))$ converges to a local optimal solution to (2).
- This suggests the following algorithmic strategy: Suppose $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s})$ is near $(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu), \boldsymbol{s}(\mu))$ for some $\mu > 0$. Use $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s})$ to move to a better point $(\boldsymbol{x}^+, \boldsymbol{y}^+, \boldsymbol{z}^+, \boldsymbol{s}^+)$ near $(\boldsymbol{x}(\mu^+), \boldsymbol{y}(\mu^+), \boldsymbol{z}(\mu^+), \boldsymbol{s}(\mu^+))$ for some $\mu^+ < \mu$.
- It can be shown that if a point $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s})$ is on the central path, then the corresponding value of μ satisfies $\boldsymbol{z}^\top \boldsymbol{s} = p\mu$.
- Likewise, given $\boldsymbol{z}, \boldsymbol{s} > 0$, define:

$$\mu(\boldsymbol{z}, \boldsymbol{s}) := \frac{\boldsymbol{z}^\top \boldsymbol{s}}{p}.$$

- To move from a current point $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s})$ to a new point, we use the Newton step for the nonlinear system of equations $\boldsymbol{r}_\mu(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s}) = 0$:

$$(\Delta \boldsymbol{x}, \Delta \boldsymbol{y}, \Delta \boldsymbol{z}, \Delta \boldsymbol{s}) = -\boldsymbol{r}'_\mu(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s})^{-1} \boldsymbol{r}_\mu(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{s}). \quad (N)$$

The following algorithm presents a template for an interior-point method.

Algorithm Interior-point method for nonlinear programming

1: choose $\mathbf{x}^0, \mathbf{y}^0$ and $\mathbf{z}^0, \mathbf{s}^0 > 0$
2: **for** $k = 0, 1, \ldots$ **do**
3: solve the Newton system (N) for $(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{s}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k, \mathbf{s}^k)$ and $\mu := 0.1\mu(\mathbf{z}^k, \mathbf{s}^k)$
4: choose a step length $\alpha \in (0, 1]$ and set $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{z}^{k+1}, \mathbf{s}^{k+1}) = (\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k, \mathbf{s}^k) + \alpha(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{z}, \Delta\mathbf{s})$
5: **end for**

The following algorithm presents a template for an interior-point method. The step length α in step 4 should be chosen via a backtracking procedure so that $\boldsymbol{z}^{k+1}, \boldsymbol{s}^{k+1} > 0$ and the size of $\boldsymbol{r}_\mu(\boldsymbol{x}^{k+1}, \boldsymbol{y}^{k+1}, \boldsymbol{z}^{k+1}, \boldsymbol{s}^{k+1})$ is sufficiently smaller than $\boldsymbol{r}_\mu(\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k, \boldsymbol{s}^k)$.

9.6 Summary algorithms

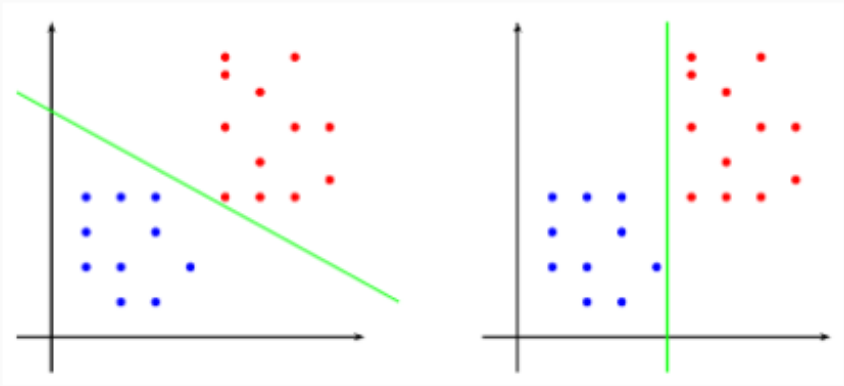
Each method refines the solution by exploiting different approaches to handle nonlinearity and constraints: SQP uses quadratic approximations, interior-point methods focus on maintaining feasibility within constraints, and GRG reduces problem complexity by separating variables.

- **Generalized Reduced Gradient (GRG):** tackles constrained optimization by dividing variables into dependent and independent sets. The dependent variables are expressed as functions of the independent ones, reducing the dimensionality of the problem. The gradient of the objective function is then computed in the reduced space, and adjustments are made to the independent variables. The method iterates by updating both sets of variables and adjusting the constraints until an optimal solution is found.
- **Sequential Quadratic Programming (SQP):** solves nonlinear optimization problems by iteratively approximating the original problem as a quadratic program. At each step, the objective function and constraints are linearized around the current point, forming a quadratic subproblem. The Hessian matrix of the Lagrangian is used to update the quadratic approximation, and a step length is chosen to ensure both objective function reduction and constraint satisfaction. The process is repeated, refining the solution with each iteration until convergence.
- **Interior-Point Methods:** solve constrained optimization problems by generating a sequence of points within the feasible region (interior) of the problem. Each iteration minimizes a perturbed version of the optimality conditions using Newton's method. The algorithm enforces strict inequalities, ensuring that the variables remain within the interior of the feasible region. A central path is followed, defined by a parameter μ, which decreases with each iteration, leading the solution towards optimality. The step length is chosen carefully to reduce residuals while maintaining feasibility.

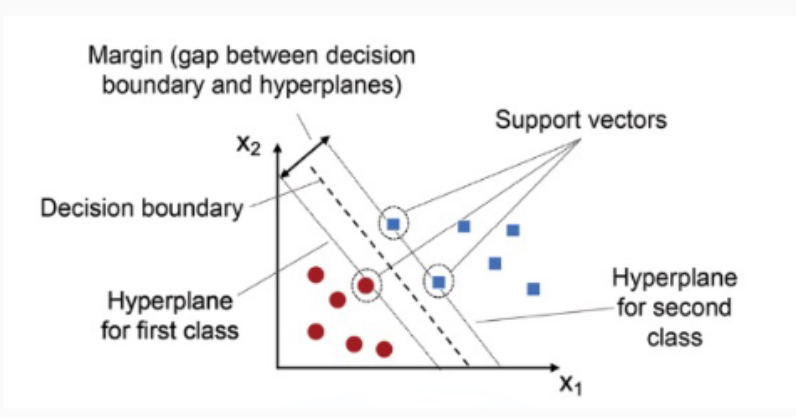
10 Support Vector Machine

Objective: support Vector Machine (SVM) is a recent and very effective class of supervised Machine Learning (ML) models and algorithms for classification and regression.

The figure below shows ***two different hyperplanes*** (in this case, two lines) that both separate the red and blue points. In terms of the training set, the two hyperplanes are equivalent, but the classification of new points is highly different.



Goal: for classification problems, the fundamental idea in SVM is to determine, instead of a single separating hyperplane, a pair of parallel hyperplanes for which the margin is maximum.



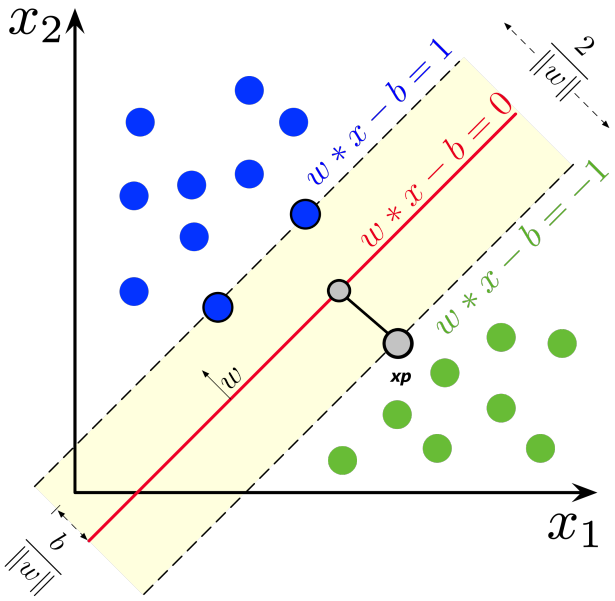
10.1 Linear Separable SVM Model

Let $(\boldsymbol{x}_p, y_p)$ with $p = 1, \ldots, P$ be the training set where: $\boldsymbol{x}_p \in \mathbb{R}^N$ represents an input observation and $y_p \in \{-1, +1\}$ is its associated label. The training set is linearly separable if and only if there exists a hyperplane $H_{\boldsymbol{w},\theta}$, where $\boldsymbol{w} \in \mathbb{R}^N$ and $\theta \in \mathbb{R}$, such that for all $p = 1, \ldots, P$:

$$\begin{cases} \boldsymbol{w}^\top \boldsymbol{x}_p + \theta > 0, & \text{if } y_p = +1, \\ \boldsymbol{w}^\top \boldsymbol{x}_p + \theta < 0, & \text{if } y_p = -1, \end{cases}$$

Remark: this equation represents a linear function that computes the dot product between the weight vector $\boldsymbol{w}$ and the observation $\boldsymbol{x}_p$, and then adds the bias θ. The result defines the position of the observation relative to the hyperplane. The parameters $\boldsymbol{w}$ and θ must be chosen such that: all observations with label $+1$ lie on one side of the hyperplane where the linear function is positive, all observations with label -1 lie on the other side of the hyperplane, where the linear function is negative.

Definition: a hyperplane in SVM is a linear decision boundary that separates the data points of two classes (e.g., class $y_p = +1$ and class $y_p = -1$) in a high-dimensional space. Mathematically, a hyperplane is defined by the equation: $\boldsymbol{w}^\top \boldsymbol{x}_p + \theta = 0$, where: $\boldsymbol{w}$ is the weight vector (normal to the hyperplane), $\boldsymbol{x}_p$ is the feature vector (input observation), θ is the bias term (shifts the hyperplane from the origin).



For a linearly separable training set, there always exists a separating hyperplane $H_{\bar{\boldsymbol{w}},\bar{\theta}}$ such that for all $p = 1, \ldots, P$:

$$\begin{cases} \bar{\boldsymbol{w}}^\top \boldsymbol{x}_p + \bar{\theta} \geq +1, & \text{if } y_p = +1, \\ \bar{\boldsymbol{w}}^\top \boldsymbol{x}_p + \bar{\theta} \leq -1, & \text{if } y_p = -1. \end{cases}$$

In fact, let $H_{\boldsymbol{w},\theta}$ be a separating hyperplane and define:

$$\alpha = \min_{p=1,\ldots,P,\, y_p=+1} (\boldsymbol{w}^\top \boldsymbol{x}_p + \theta), \quad \alpha > 0$$

$$\beta = \max_{p=1,\ldots,P,\, y_p=-1} (\boldsymbol{w}^\top \boldsymbol{x}_p + \theta), \quad \beta < 0$$

Remarks:

- We define two values, α and β, which represent how close the data points of the respective classes $+1$ and -1 are to the separating hyperplane, in terms of the linear function used to describe the hyperplane.
- α represents the point in class $+1$ that is closest to the hyperplane (on the positive side), instead β maximizes the value of the linear function for points belonging to class -1, since points from class -1 lie on the other side of the hyperplane (where the linear function is negative).
- The values α and β are not direct Euclidean distances but represent the output of the linear function $\boldsymbol{w}^\top \boldsymbol{x}_p + \theta$. However, they are **proportional** to the distances of the points to the hyperplane. The true distance from a point $\boldsymbol{x}_p$ to the separating hyperplane is given by:

$$\frac{\boldsymbol{w}^\top \boldsymbol{x}_p + \theta}{\|\boldsymbol{w}\|}$$

This represents the perpendicular (Euclidean) distance from the point $\boldsymbol{x}_p$ to the hyperplane $H_{\boldsymbol{w},\theta}$, regardless of whether it belongs to class $+1$ or -1.

On fixed $\boldsymbol{x}_p$:

$$\boldsymbol{w}^T \boldsymbol{x}_p + \theta \propto \frac{\boldsymbol{w}^T \boldsymbol{x}_p + \theta}{\|w\|}$$

Moreover, let $\gamma = \min\{\alpha, -\beta\}$ (the smallest distance), $\overline{\boldsymbol{w}} = \frac{\boldsymbol{w}}{\gamma}$ and $\overline{\theta} = \frac{\theta}{\gamma}$. Now, for each $p = 1, \ldots, P$ if $y_p = +1$, then:

$$\frac{1}{\gamma} \left(\boldsymbol{w}^\top \boldsymbol{x}_p + \theta \right) \geq \frac{1}{\gamma} \cdot \gamma \quad \text{and hence} \quad \overline{\boldsymbol{w}}^\top \boldsymbol{x}_p + \overline{\theta} \geq 1,$$

While if $y_p = -1$ then:

$$\boldsymbol{w}^\top \boldsymbol{x}_p + \theta \leq -\gamma \quad \text{and hence} \quad \overline{\boldsymbol{w}}^\top \boldsymbol{x}_p + \overline{\theta} \leq -1.$$

Therefore, a ***training set is linearly separable*** if there exists a separating hyperplane $H_{\boldsymbol{w},\theta}$ such that for all $p = 1, \ldots, P$:

$$\begin{cases} \boldsymbol{w}^\top \boldsymbol{x}_p + \theta \geq +1 & \text{if } y_p = +1, \\ \boldsymbol{w}^\top \boldsymbol{x}_p + \theta \leq -1 & \text{if } y_p = -1. \end{cases}$$

Note that the conditions above can be rewritten as:

$$y_p(\boldsymbol{w}^\top \boldsymbol{x}_p + \theta) \geq 1 \quad \forall p = 1, \ldots, P.$$

Definition (hyperplane margin): let the training set be linearly separable. The margin $\rho_{\boldsymbol{w},\theta}$ of $H_{\boldsymbol{w},\theta}$ is the minimum distance between points in the training set of both classes and the hyperplane:

$$\rho_{\boldsymbol{w},\theta} = \min_{p=1,\ldots,P} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_p + \theta|}{\|\boldsymbol{w}\|}$$

where $\| \cdot \|$ is the Euclidean norm. SVM will choose as the separating hyperplane $H_{\boldsymbol{w}^*,\theta^*}$ the one for which the margin is maximum. This can be achieved by solving the following optimization problem:

$$\begin{aligned} \max \quad & \rho_{\boldsymbol{w},\theta} \\ s.t. \quad & y_p(\boldsymbol{w}^\top \boldsymbol{x}_p + \theta) \geq 1, \quad p = 1, \ldots, P \end{aligned}$$

Notice that since for all $p = 1, \dots, P$:

$$y_p(\boldsymbol{w}^\top \boldsymbol{x}_p + \theta) \geq 1 \iff |\boldsymbol{w}^\top \boldsymbol{x}_p + \theta| \geq 1$$

then from the definition of $\rho_{\boldsymbol{w},\theta}$, we have:

$$\rho_{\boldsymbol{w},\theta} = \min_{p=1,\dots,P} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_p + \theta|}{\|\boldsymbol{w}\|} \geq \frac{1}{\|\boldsymbol{w}\|}.$$

Moreover, for each separating hyperplane $H_{\boldsymbol{w},\theta}$, there exists a separating hyperplane $H_{\overline{\boldsymbol{w}},\overline{\theta}}$ that maximize the margin by minimizing $\|w\|$ (we maximize the margin). As we minimize $\|w\|$, we maximize the margin $\frac{1}{\|w\|}$. Hence, $\frac{1}{\|w\|}$ becomes the largest margin achievable by the SVM formulation because it directly results from the optimization problem.

$$\rho_{\boldsymbol{w},\theta} \leq \rho_{\overline{\boldsymbol{w}},\overline{\theta}} = \frac{1}{\|\overline{\boldsymbol{w}}\|}$$

and $p', p'' \in \{1, \dots, P\}$ are ***support vectors*** because they lie exactly on the margin hyperplanes satisfying:

$$y_{p'} = +1 \quad \text{and} \quad \overline{\boldsymbol{w}}^\top \boldsymbol{x}_{p'} + \overline{\theta} = +1,$$

$$y_{p''} = -1 \quad \text{and} \quad \overline{\boldsymbol{w}}^\top \boldsymbol{x}_{p''} + \overline{\theta} = -1.$$

In fact, for the separating hyperplane $H_{\boldsymbol{w},\theta}$, let $p', p'' \in \{1, \dots, P\}$ such that $y_{p'} = +1$, $y_{p''} = -1$ and:

$$\alpha = \min_{p=1,\dots,P, y_p=+1} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_p + \theta|}{\|\boldsymbol{w}\|} = \frac{|\boldsymbol{w}^\top \boldsymbol{x}_{p'} + \theta|}{\|\boldsymbol{w}\|},$$

$$\beta = \min_{p=1,\dots,P, y_p=-1} \frac{|\boldsymbol{w}^\top \boldsymbol{x}_p + \theta|}{\|\boldsymbol{w}\|} = \frac{|\boldsymbol{w}^\top \boldsymbol{x}_{p''} + \theta|}{\|\boldsymbol{w}\|}.$$

Remark: the inequality $\rho_{\boldsymbol{w},\theta} \leq \min\{\alpha, \beta\}$ simply indicates that the margin cannot exceed the closest distances to the hyperplane for either class since the margin is determined by the closest points, the inequality $\rho_{\boldsymbol{w}} \leq \frac{\alpha+\beta}{2}$ sugggest if α and β are exactly the same (symmetric case) then the margin would be exactly $\frac{\alpha+\beta}{2}$, $\rho_{\boldsymbol{w},\theta} = \frac{\boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''})}{\|\boldsymbol{w}\|}$ where the term $\boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''})$ represents the difference between the support vectors from the two classes along the direction of $\boldsymbol{w}$ $\Rightarrow$ so the margin is this difference normalized by the magnitude of $\boldsymbol{w}$ which gives the perpendicular distance between the two support vectors.

$$\rho_{\boldsymbol{w},\theta} \leq \min\{\alpha, \beta\} \leq \frac{\alpha + \beta}{2} = \boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''}).$$

We define a scaling factor γ and a shifted bias term δ such that the hyperplane becomes normalized and we choose them such that the support vectors satisfy:

$$\gamma \boldsymbol{w}^\top \boldsymbol{x}_{p'} + \delta = +1,$$

$$\gamma \boldsymbol{w}^\top \boldsymbol{x}_{p''} + \delta = -1,$$

To derive γ and δ, we can manipulate the equations. From the two equations, we can express γ as follows:

$$\gamma = \frac{2}{\boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''})} \quad \text{and} \quad \delta = \frac{\boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''})}{\boldsymbol{w}^\top(\boldsymbol{x}_{p'} + \boldsymbol{x}_{p''})}.$$

Now:

$$\boldsymbol{w}^\top \boldsymbol{x}_{p'} + \theta \geq 1 \geq 0 \quad \text{and} \quad \boldsymbol{w}^\top \boldsymbol{x}_{p''} + \theta \leq -1,$$

hence:

$$-(\boldsymbol{w}^\top \boldsymbol{x}_{p''} + \theta) \geq 1 \geq 0.$$

Therefore:

$$\boldsymbol{w}^\top \boldsymbol{x}_{p'} + \theta - (\boldsymbol{w}^\top \boldsymbol{x}_{p''} + \theta) = \boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''}) \geq 2 > 0.$$

So, $0 < \gamma \leq 2$. We now define a new weight vector $\overline{\boldsymbol{w}} = \gamma \boldsymbol{w}$ and a new bias term $\overline{\theta} = \delta$. This transformation gives us a new hyperplane $H_{\overline{\boldsymbol{w}},\overline{\theta}}$ that is normalized and satisfies the conditions:

$$\boldsymbol{w}^\top \boldsymbol{x}_p \geq \boldsymbol{w}^\top \boldsymbol{x}_{p'} \quad \forall p \in \{1, \dots, P : y_p = +1\},$$
$$\boldsymbol{w}^\top \boldsymbol{x}_p \leq \boldsymbol{w}^\top \boldsymbol{x}_{p''} \quad \forall p \in \{1, \dots, P : y_p = -1\},$$

then:

$$\gamma \boldsymbol{w}^\top \boldsymbol{x}_p + \overline{\theta} \geq \gamma \boldsymbol{w}^\top \boldsymbol{x}_{p'} + \overline{\theta} = +1 \quad \forall p \in \{1, \dots, P : y_p = +1\},$$
$$\gamma \boldsymbol{w}^\top \boldsymbol{x}_p + \overline{\theta} \leq \gamma \boldsymbol{w}^\top \boldsymbol{x}_{p''} + \overline{\theta} = -1 \quad \forall p \in \{1, \dots, P : y_p = -1\}.$$

Hence:

$$\overline{\boldsymbol{w}}^\top \boldsymbol{x}_p + \overline{\theta} \geq +1 \quad \forall p \in \{1, \dots, P : y_p = +1\},$$
$$\overline{\boldsymbol{w}}^\top \boldsymbol{x}_p + \overline{\theta} \leq -1 \quad \forall p \in \{1, \dots, P : y_p = -1\}.$$

Moreover for the rescaled and not rescaled hyperplane the margin is:

$$\rho_{\overline{\boldsymbol{w}},\overline{\theta}} = \min_{p=1,\dots,P} \frac{|\overline{\boldsymbol{w}}^\top \boldsymbol{x}_p + \overline{\theta}|}{\|\overline{\boldsymbol{w}}\|} = \frac{1}{\|\overline{\boldsymbol{w}}\|} = \frac{1}{\|\gamma \boldsymbol{w}\|} = \boldsymbol{w}^\top(\boldsymbol{x}_{p'} - \boldsymbol{x}_{p''}) = \rho_{\boldsymbol{w},\theta},$$

and:

$$\overline{\boldsymbol{w}}^\top \boldsymbol{x}_{p'} + \overline{\theta} = +1, \quad \overline{\boldsymbol{w}}^\top \boldsymbol{x}_{p''} + \overline{\theta} = -1.$$

Therefore the problem:

$$\begin{aligned} \max_{\boldsymbol{w},\theta} \quad & \rho_{\boldsymbol{w},\theta} \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^\top \boldsymbol{x}_p + \theta) \geq 1 \quad p = 1, \dots, P \end{aligned}$$

is equivalent to solving the following convex quadratic optimization problem:

Hard Margin SVM (*):

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^\top \boldsymbol{x}_p + \theta) \geq 1 \quad p = 1, \dots, P \end{aligned}$$

Notice that even though the objective function of problem is not strictly convex (in $\boldsymbol{w}$ and θ), it admits a unique solution.

The Lagrangian function associated to problem (*) is:

$$L(\boldsymbol{w},\theta,\alpha) = \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} - \sum_{p=1}^{P}\alpha_p\left(y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)-1\right)$$
$$= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{p=1}^{P}\alpha_p\left(y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)-1\right)$$

Remark: the term $\frac{1}{2}\|\boldsymbol{w}\|^2$ represents the objective function we want to minimize and by minimizing $\|\boldsymbol{w}\|^2$ we maximize the margin between the two parallel hyperplanes, the term $y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)$ ensures that the training samples are correctly classified with a margin of at least 1, the α_p values are the Lagrange multipliers which correspond to the constraints in the optimization problem. Each multiplier α_p indicates the importance or weight of the corresponding constraint in the solution. When a constraint is active its corresponding Lagrange multiplier will be greater than zero. If a constraint is not active, the multiplier will be zero. Therefore, the Lagrange multipliers help in identifying which data points (support vectors) are most crucial for defining the decision boundary.

and the **Karush–Kuhn–Tucker (KKT) optimality conditions** are:

$$\boldsymbol{w} = \sum_{p=1}^{P}\alpha_p y_p \boldsymbol{x}_p$$
$$\sum_{p=1}^{P}\alpha_p y_p = 0$$
$$\alpha_p \geq 0 \qquad p=1,\dots,P$$
$$y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)-1 \geq 0 \qquad p=1,\dots,P$$
$$\alpha_p\left[y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)-1\right] = 0 \qquad p=1,\dots,P$$

Remark: the first KKT condition indicates that the optimal $\boldsymbol{w}$ is a linear combination (with coefficients $y_p\alpha_p$) of the training set elements $\boldsymbol{x}_p$. The last set of complementarity conditions implies that the only vectors entering in the linear combination are those for which $y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)=1$. These vectors are called **support vectors**.

The ***Wolfe dual*** is a reformulation of the original optimization problem into its dual form, which allows for more efficient computation in some cases. In the context of Support Vector Machines (SVM), solving the dual problem is advantageous because it transforms the problem from a constrained optimization problem over the primal variables $(\boldsymbol{w},\theta)$ to a problem involving only the Lagrange multipliers α. The Wolfe dual is given by:

$$\max_{\boldsymbol{w},\theta,\alpha} \quad L(\boldsymbol{w},\theta,\alpha)$$
$$\text{s.t.} \quad \nabla_{\boldsymbol{w}}L(\boldsymbol{w},\theta,\alpha) = 0$$
$$\nabla_\theta L(\boldsymbol{w},\theta,\alpha) = 0$$
$$\alpha \geq 0$$

Where the expaned equations are the following:

$$\max_{\boldsymbol{w},\theta,\boldsymbol{\alpha}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{p=1}^{P}\alpha_p\left[y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)-1\right]$$
$$\text{s.t.} \quad \boldsymbol{w} = \sum_{p=1}^{P}\alpha_p y_p \boldsymbol{x}_p$$
$$\sum_{p=1}^{P}\alpha_p y_p = 0$$
$$\alpha_p \geq 0 \quad \forall p=1,\dots,P$$

Now:

$$L(\boldsymbol{w},\theta,\alpha) = \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} - \sum_{p=1}^{P}\alpha_p\left(y_p(\boldsymbol{w}^\top\boldsymbol{x}_p+\theta)-1\right)$$
$$= \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} - \sum_{p=1}^{P}\left(\alpha_p y_p \boldsymbol{w}^T\boldsymbol{x}_p + \alpha_p y_p\theta - \alpha_p\right)$$
$$= \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} - \sum_{p=1}^{P}\alpha_p y_p \boldsymbol{w}^T\boldsymbol{x}_p - \sum_{p=1}^{P}\alpha_p y_p\theta + \sum_{p=1}^{P}\alpha_p$$
$$= -\frac{1}{2}\boldsymbol{w}^\top\boldsymbol{w} + \boldsymbol{w}^\top\left(\boldsymbol{w} - \sum_{p=1}^{P}\alpha_p y_p\boldsymbol{x}_p\right) - \theta\sum_{p=1}^{P}\alpha_p y_p + \sum_{p=1}^{P}\alpha_p.$$

Therefore, after substituting for $\boldsymbol{w}$, the dual problem can be written as:

$$\min_{\alpha} \frac{1}{2}\left(\sum_{p=1}^{P}\alpha_p y_p\boldsymbol{x}_p\right)^\top\left(\sum_{q=1}^{P}\alpha_q y_q\boldsymbol{x}_q\right) - \sum_{p=1}^{P}\alpha_p$$
$$\text{s.t.} \quad \sum_{p=1}^{P}\alpha_p y_p = 0, \quad \alpha_p \geq 0 \quad p=1,\dots,P.$$

Let $K:\mathbb{R}^N\times\mathbb{R}^N\to\mathbb{R}$ be a ***kernel function*** defined as:

$$K(\bar{\boldsymbol{x}},\hat{\boldsymbol{x}}) = \bar{\boldsymbol{x}}^\top\hat{\boldsymbol{x}}$$

This means that K takes two vectors from $\mathbb{R}^N$, denoted by $\bar{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}$, and maps them to a real number through their dot product (or inner product). This kernel is particularly useful when using linear SVM models because it directly computes the dot product in the input space, which corresponds to the decision boundary in the same space $\Rightarrow$ ***kernel trick***. Then, the minimization problem above can be re-written as:

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \boldsymbol{e}^\top\boldsymbol{\alpha}$$
$$\text{s.t.} \quad \boldsymbol{y}^\top\boldsymbol{\alpha} = 0,$$
$$\boldsymbol{\alpha} \geq 0$$

where $\boldsymbol{e}$ is a vector with all components equal to 1, $Q \in \mathbb{R}^{P\times P}$ and:

$$Q_{pq} = y_p y_q K(\boldsymbol{x}_p, \boldsymbol{x}_q).$$

Solve the problem

$$\begin{aligned}
&\alpha^* \to \min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \boldsymbol{e}^\top\boldsymbol{\alpha} && \textit{Wolfe dual quadratic form}\\
&\boldsymbol{w}^* = \sum_{p=1}^{P} \alpha_p^* y_p \boldsymbol{x}_p && \textit{see first KKT condition}\\
&\theta^* \to y_{p'}\left((\boldsymbol{w}^*)^\top \boldsymbol{x}_{p'} + \theta^*\right) - 1 = 0 && \textit{see last KKT conditions}
\end{aligned}$$

Once the dual problem is solved (let α^* be the optimal solution), the separating hyperplane $(\boldsymbol{w}^*)^\top \boldsymbol{x} + \theta^* = 0$ is obtained as follows:

$$1) \qquad \boldsymbol{w}^* = \sum_{p=1}^{P} \alpha_p^* y_p \boldsymbol{x}_p \quad \text{(see first KKT condition)}$$

In this way we get $\boldsymbol{w^*}$. Also if p' is a support vector (that is, $\alpha_{p'}^* > 0$), then:

$$2) \qquad y_{p'}\left((\boldsymbol{w}^*)^\top \boldsymbol{x}_{p'} + \theta^*\right) - 1 = 0 \quad \text{(see last KKT conditions)}$$

from which θ^* can be easily computed. Then, the ***decision function*** $h(\cdot)$ for linear SVM is:

$$h(\boldsymbol{x}) = \text{sign}\left((\boldsymbol{w}^*)^\top \boldsymbol{x} + \theta^*\right) = \text{sign}\left(\sum_{p=1}^{P} \alpha_p^* y_p K(\boldsymbol{x}_p, \boldsymbol{x}) + \theta^*\right)$$

where $\boldsymbol{x} \in \mathbb{R}^N$.

10.2 Non linearly separable SVM model - Soft margins

When the training set is not linearly separable, the problem

$$\begin{aligned}
&\min_{w} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 && (1)\\
&\text{s.t.} \quad y_p(\boldsymbol{w}^T\boldsymbol{x}_p + \theta) \geq 1, \quad p = 1,\dots,P && (2)
\end{aligned}$$

has an empty feasible region. This means that there is no solution that satisfies all constraints. In this case, slack variables ξ_p must be introduced, leading to the new optimization problem known as the Soft Margin SVM.

Soft Margin SVM

$$\begin{aligned}
&\min_{w,\theta,\xi} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{p=1}^{P}\xi_p && (3)\\
&\text{s.t.} \quad y_p(\boldsymbol{w}^T\boldsymbol{x}_p + \theta) \geq 1 - \xi_p, \quad p = 1,\dots,P && (4)\\
&\qquad \xi_p \geq 0, \quad p = 1,\dots,P && (5)
\end{aligned}$$

Remark: the term ξ_p represents slack variables that allow for some misclassification of the training examples. Each ξ_p corresponds to the p-th training example and indicates how much the example violates the margin constraints. The term C is a regularization parameter that controls the trade-off between maximizing the margin (the first term) and minimizing the classification error (the second term). A larger C puts more emphasis on correctly classifying all training examples, while a smaller C allows more misclassifications to achieve a wider margin.

For the hyperplane $H_{\boldsymbol{w},\theta}$, if a point $\boldsymbol{x}_p$ is not correctly classified, then the corresponding quantity ξ_p is greater than 1. In fact:

$$\begin{aligned}
&\text{If } y_p = +1, \quad \boldsymbol{w}^T\boldsymbol{x}_p - \theta \leq 0 \implies \xi_p = 1 - \boldsymbol{w}^T\boldsymbol{x}_p + \theta \geq 1,\\
&\text{If } y_p = -1, \quad \boldsymbol{w}^T\boldsymbol{x}_p + \theta \geq 0 \implies \xi_p = 1 + \boldsymbol{w}^T\boldsymbol{x}_p + \theta \geq 1.
\end{aligned}$$

Moreover, values of ξ_p in the interval $(0, 1)$ correspond to points correctly classified by the hyperplane but within the margin. The objective function comprises two terms: 1) the first term corresponds to margin maximization: $\frac{1}{2}\|\boldsymbol{w}\|^2$. 2) The second term provides an upper bound on the number of incorrectly classified training points: $\sum \xi_p$.

The Lagrangian function associated is:

$$L(\boldsymbol{w}, \theta, \xi, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{p=1}^{P}\xi_p - \sum_{p=1}^{P}\alpha_p\left(y_p(\boldsymbol{w}^T\boldsymbol{x}_p + \theta) - 1 + \xi_p\right)$$

The Karush–Kuhn–Tucker optimality conditions are:

$$\begin{aligned}
&\boldsymbol{w} = \sum_{p=1}^{P}\alpha_p y_p \boldsymbol{x}_p\\
&\sum_{p=1}^{P}\alpha_p y_p = 0\\
&0 \leq \alpha_p \leq C && p = 1,\dots,P\\
&\xi_p \geq 0 && p = 1,\dots,P\\
&\xi_p(C - \alpha_p) = 0 && p = 1,\dots,P\\
&y_p(\boldsymbol{w}^T\boldsymbol{x}_p + \theta) - 1 + \xi_p \geq 0 && p = 1,\dots,P\\
&\alpha_p\left[y_p(\boldsymbol{w}^T\boldsymbol{x}_p + \theta) - 1 + \xi_p\right] = 0 && p = 1,\dots,P
\end{aligned}$$

The dual problem is now formulated as:

$$\begin{aligned}
&\min_{\alpha} \quad \frac{1}{2}\left(\sum_{p=1}^{P}\alpha_p y_p \boldsymbol{x}_p\right)^T\left(\sum_{q=1}^{P}\alpha_q y_q \boldsymbol{x}_q\right) - \sum_{p=1}^{P}\alpha_p\\
&\text{s.t.} \quad \sum_{p=1}^{P}\alpha_p y_p = 0\\
&\qquad 0 \leq \alpha_p \leq C, \quad p = 1,\dots,P
\end{aligned}$$

or, in more compact form:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \text{s.t.} \quad & y^T\alpha = 0 \\ & 0 \le \alpha \le Ce \end{aligned}$$

where e is a vector with all components equal to 1, $Q \in \mathbb{R}^{P\times P}$ and:

$$Q_{pq} = y_p y_q K(\boldsymbol{x}_p, \boldsymbol{x}_q)$$

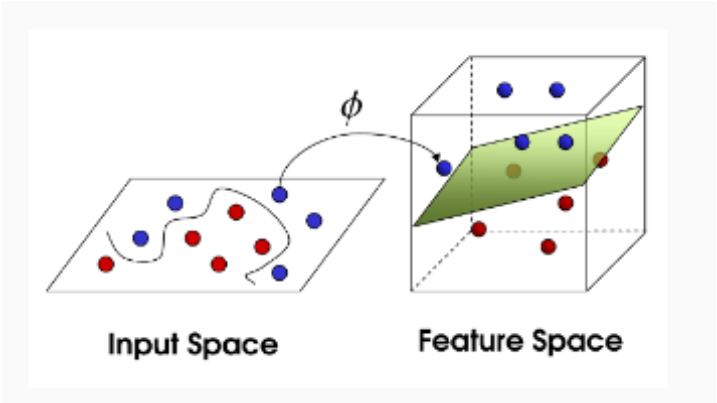
Remark: The support vectors can be categorized into two types: - Free support vectors, corresponding to $0 < \alpha_p^* < C$. - Bounded support vectors, corresponding to $\alpha_p^* = C$. The vector $\boldsymbol{w}^*$ is a linear combination of support vectors, while the scalar θ^* can be calculated using any free support vector.

Difference between the vectors:

- **Free support vectors** $0 < \alpha_p^* < C$**:** are vectors lie exactly on the margin (the boundary defined by the support vector hyperplanes) and are not misclassified. Since their multipliers are non-zero but less than C, they contribute to defining the optimal separating hyperplane while still allowing for some margin.
- **Bounded support vectors** $\alpha_p^* = C$**:** are typically located on the incorrect side of the margin, meaning they are either misclassified or very close to the wrong side of the hyperplane. The multiplier $\alpha_p^* = C$ means that these points contribute maximally to the formulation of the hyperplane. They do not move away from their position even if the margin is increased.

10.3 Non linearly separable SVM model - Feature space

The idea behind non-linear SVM is to map the input space into a feature space where the data points are linearly separable. It is important to recall that in linear SVM models, the dual problem can be easily constructed using the kernel function $K(\cdot,\cdot)$. Moreover, the decision function only utilizes this kernel function. Let $\mathcal{F}$ be a ***Hilbert space*** (the feature space) whose dimension is greater than n and possibly infinite, and let $\phi : \mathbb{R}^n \to \mathcal{F}$ be a mapping from $\mathbb{R}^n$ to the feature space.



For the non-linear SVM model, the optimization problem is now defined as follows:

$$\begin{aligned} \min_{\boldsymbol{w},\theta,\boldsymbol{\xi}} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{p=1}^{P}\xi_p & \\ \text{s.t.} \quad & y_p(\boldsymbol{w}^\top\phi(x_p)+\theta) \ge 1-\xi_p & p=1,\dots,P \\ & \xi_p \ge 0 & p=1,\dots,P. \end{aligned}$$

The KKT Conditions for the Non-Linear SVM are given by:

$$\begin{aligned} & \boldsymbol{w} = \sum_{p=1}^{P}\alpha_p y_p \phi(x_p) & \\ & \sum_{p=1}^{P}\alpha_p y_p = 0 & \\ & 0 \le \alpha_p \le C & p=1,\dots,P \\ & \xi_p \ge 0 & p=1,\dots,P \\ & \xi_p(C-\alpha_p) = 0 & p=1,\dots,P \\ & y_p(\boldsymbol{w}^\top\phi(\boldsymbol{x_p})+\theta) - 1 + \xi_p \ge 0 & p=1,\dots,P \\ & \alpha_p[y_p(\boldsymbol{w}^\top\phi(\boldsymbol{x_p})+\theta) - 1 + \xi_p] = 0 & p=1,\dots,P. \end{aligned}$$

The dual problem is formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - e^\top\boldsymbol{\alpha} \\ \text{s.t.} \quad & y^\top\boldsymbol{\alpha} = 0 \\ & 0 \le \boldsymbol{\alpha} \le C\boldsymbol{e}, \end{aligned}$$

Let $\boldsymbol{e}$ be a vector with all components equal to 1, $\boldsymbol{Q} \in \mathbb{R}^{P\times P}$, and

$$\boldsymbol{Q}_{pq} = y_p y_q K(\boldsymbol{x}_p, \boldsymbol{x}_q) \quad \text{with} \quad K(\boldsymbol{x}_p, \boldsymbol{x}_q) = \phi(\boldsymbol{x}_p)^\top\phi(\boldsymbol{x}_q).$$

Remark: the function $K : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is called the Kernel function. Once the dual problem is solved (let $\boldsymbol{\alpha}^*$ be its optimal solution), the optimal hyperplane $H^{\boldsymbol{w}^*,\theta^*}$ can be obtained by:

$$\boldsymbol{w}^* = \sum_{p=1}^{P}\boldsymbol{\alpha}_p^* y_p \phi(\boldsymbol{x}_p) \quad \text{(see first KKT condition);}$$

If p' is a support vector (i.e., $0 < \boldsymbol{\alpha}_{p'}^* < C$), then:

$$y_{p'}\left(\boldsymbol{w}^{*\top}\phi(\boldsymbol{x}_{p'}) + \theta^*\right) - 1 = y_{p'}\left(\sum_{p=1}^{P}\boldsymbol{\alpha}_p^* y_p K(\boldsymbol{x}_p, \boldsymbol{x}_{p'}) + \theta^*\right) - 1 = 0,$$

from which θ^* can be easily computed. Then, the decision function $h(\cdot)$ for non-linear SVM is:

$$h(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^{*\top}\phi(\boldsymbol{x}) + \theta^*\right) = \text{sign}\left(\sum_{p=1}^{P}\boldsymbol{\alpha}_p^* y_p K(\boldsymbol{x}_p, \boldsymbol{x}) + \theta^*\right).$$

Kernels Remark: the difficulty in explicitly using the mapping $\phi(\cdot)$ to construct the feature space is that the resulting space can be extremely high-dimensional. However, it is essential to notice that both in calculating θ^* and in evaluating the decision function on a new point, the actual form of the function $\phi(\cdot)$ is not necessary; it suffices to calculate the Kernel function $K(\cdot,\cdot)$.

Therefore, a fundamental question arises: under which hypothesis is a function $K : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ a Kernel function, that is,

$$K(\bar{\boldsymbol{x}}, \hat{\boldsymbol{x}}) = \phi(\bar{\boldsymbol{x}})^\top \phi(\hat{\boldsymbol{x}})$$

for some function $\phi : \mathbb{R}^N \to \mathcal{F}$? A related question is: for which mappings $\phi(\cdot)$ is it easy to evaluate the scalar product $\phi(\bar{\boldsymbol{x}})^\top \phi(\hat{\boldsymbol{x}})$?

Clearly, due to the properties of the inner product, the matrix $\boldsymbol{K} \in \mathbb{R}^{P\times P}$ with $\boldsymbol{K}_{pq} = K(\boldsymbol{x}_p, \boldsymbol{x}_q)$ must be symmetric. In the finite-dimensional case, the results can be simplified as follows.

Theorem: Let $K : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ be a symmetric function. The function $K(\cdot,\cdot)$ is a Kernel function if and only if the matrix $\boldsymbol{K} \in \mathbb{R}^{P\times P}$ with $\boldsymbol{K}_{pq} = K(\boldsymbol{x}_p, \boldsymbol{x}_q)$ is positive semidefinite for each training set composed of P vectors.

The most common Kernels are:

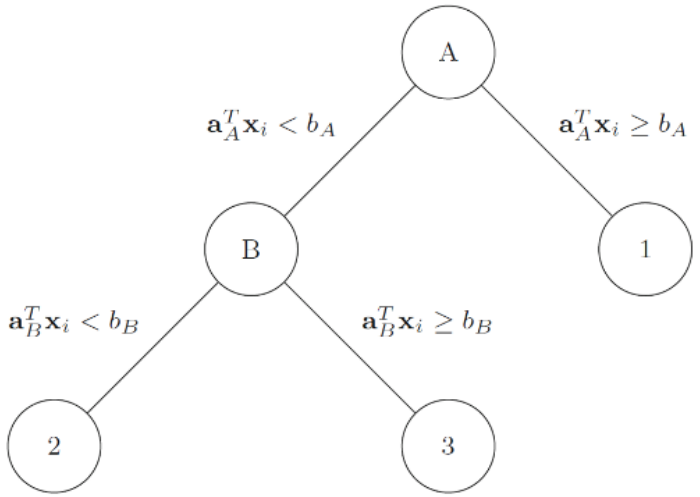
Linear Kernel:	$K(\bar{\boldsymbol{x}}, \hat{\boldsymbol{x}}) = \bar{\boldsymbol{x}}^\top \hat{\boldsymbol{x}}$
Polynomial Kernel:	$K(\bar{\boldsymbol{x}}, \hat{\boldsymbol{x}}) = (\bar{\boldsymbol{x}}^\top \hat{\boldsymbol{x}})^d$
Gaussian Kernel:	$K(\bar{\boldsymbol{x}}, \hat{\boldsymbol{x}}) = e^{-\frac{\|\|\bar{\boldsymbol{x}}-\hat{\boldsymbol{x}}\|\|^2}{2\sigma^2}}$
Sigmoidal Kernel:	$K(\bar{\boldsymbol{x}}, \hat{\boldsymbol{x}}) = \tanh(\rho_1 \bar{\boldsymbol{x}}^\top \hat{\boldsymbol{x}} + \rho_2)$

11 Decision Trees

We are given the training data $(\boldsymbol{X}, \boldsymbol{y})$, containing n observations $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, each with p features $\boldsymbol{x}_i \in \mathbb{R}^p$ and a class label $y_i \in \{1, \ldots, K\}$, indicating which of K possible labels is assigned to this point. We assume without loss of generality that the values for each dimension across the training data are normalized to the $0-1$ interval, meaning each $\boldsymbol{x}_i \in [0,1]^p$.

Objective: the Decision Tree (DT) method seeks to recursively partition $[0,1]^p$ to yield a number of hierarchical, disjoint regions that represent a classification tree.

Example: the DT as two branch nodes A and B and three leaf nodes 1, 2, and 3.



The final tree is comprised of branch nodes and leaf nodes:

- **Branch nodes** apply a split with parameters $\boldsymbol{a}$ and b. For a given point i, if $\boldsymbol{a}^\top \boldsymbol{x}_i < b$, the point will follow the left branch from the node; otherwise, it takes the right branch. A subset of methods, including CART, produce univariate or axis-aligned decision trees, which restrict the split to a single dimension, i.e., a single component of $\boldsymbol{a}$ will be 1 and all others will be 0.
- **Leaf nodes** are assigned a class that will determine the prediction for all data points that fall into the leaf node. The assigned class is almost always taken to be the class that occurs most often among points contained in the leaf node.

Remark: classical DT methods (e.g., CART, ID3, and C4.5) take a ***top-down approach*** to building the tree. At each step of the partitioning process, they seek to find a split that will partition the current region in such a way as to maximize a so-called ***splitting criterion***. This criterion is often based on the ***label impurity*** of the data points contained in the resulting regions, instead of minimizing the resulting misclassification error. The algorithm proceeds to recursively partition the two new regions that are created by the hyperplane split. The partitioning terminates once any one of a number of stopping criteria are met.

11.1 CART method

The criteria for CART[1] are as follows:

- A split cannot be created if either side of the partition contains fewer than a specified minimum number of nodes,, $N_{\min}$.
- All points in the candidate node share the same class.

Once the splitting process is complete, a class label $1, \ldots, K$ is assigned to each region. This class will be used to predict the class of any points contained inside the region. As mentioned earlier, this assigned class will typically be the most common class among the points in the region. The final step in the process is ***pruning the tree*** in an attempt to avoid overfitting. The pruning process works upwards through the partition nodes from the bottom of the tree. The decision of whether to prune a

node is controlled by the so-called ***complexity parameter*** α, which balances the additional complexity of adding the split at the node against the increase in predictive accuracy that it offers.

Remark: a higher complexity parameter leads to more and more nodes being pruned off, resulting in smaller trees.

Using the details of the CART procedure, we can state the problem that CART attempts to solve as a formal optimization problem. There are two parameters in this problem:

- The trade-off between accuracy and complexity of the tree: α;
- The minimum number of points we require in any leaf node: $N_{\min}$.

Optimal Tree Problem: Given these parameters and the training data $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, we seek a tree $\mathbb{T}$ that solves the problem:

$$\begin{aligned} &\min R(\mathbb{T}) + \alpha|\mathbb{T}| \\ &\text{s.t.} \quad N(l) \geq N_{\min} \quad \forall l \in \text{leaves}(\mathbb{T}) \end{aligned} \tag{P}$$

where $R(\mathbb{T})$ is the misclassification error of the tree $\mathbb{T}$ on the training data, $|\mathbb{T}|$ is the number of branch nodes in tree $\mathbb{T}$, and $N(l)$ is the number of training points contained in leaf node l.

Remark: notice that if we can solve this problem in a single step, we eliminate the need to use an impurity measure when growing the tree and also remove the need to prune the tree after creation, as we have already accounted for the complexity penalty while growing the tree.

Remark: we briefly note that our choice to use CART to define the optimal tree problem was arbitrary, and one could similarly define this problem based on another method like C4.5. We simply use this problem to demonstrate the advantages of taking a problem that is traditionally solved by a heuristic and instead solving it to optimality.

Remark: additionally, we note that the literature found that CART and C4.5 did not differ significantly in any measure of tree quality, including out-of-sample accuracy. Hence, we do not believe our choice of CART over C4.5 to be an important one.

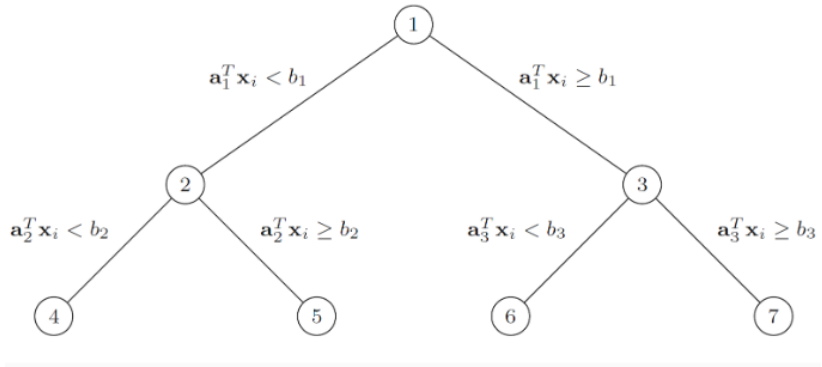
Mixed Integer Optimization (MIO) Approach

As mentioned previously, the ***top-down, greedy nature*** of state-of-the-art DT creation algorithms can lead to solutions that are only ***locally optimal***. The natural way to pose the task of creating the globally optimal DT is as a ***Mixed Integer Optimization (MIO) problem***. Indeed, at every step in tree creation, we are required to make a number of discrete decisions:

1. At every new node, we must choose to either branch or stop.
2. After choosing to stop branching at a node, we must choose a label to assign to this new leaf node.
3. After choosing to branch, we must choose which of the variables to branch on.
4. When classifying the training points according to the tree under construction, we must choose to which leaf node a point will be assigned such that the structure of the tree is respected.

Formulating this problem using MIO allows us to model all of these discrete decisions in a single problem. Modeling the construction process in this way allows us to consider the full impact of the decisions being made at the top of the tree, rather than simply making a series of locally optimal decisions, also avoiding the need for pruning and impurity measures. We will next formulate the optimal tree creation problem (P) as a MIO problem. Consider the problem of trying to construct an optimal DT with a maximum depth of D. Given this depth, we can construct the maximal tree of this depth, which has $T = 2^{(D+1)} - 1$ nodes, which we index by $t = 1, \ldots, T$.

Example: tree of depth D = 2, brach nodes: 1-2-3, leaf nodes: 4,5,6,7.



We use the following notation:

Parent node of node t	$p(t)$
Anchestors of node t	$A(t) = L(t) \cup R(t)$
Achestors from root on left till t	$L(t)$
Achestors from root on right till t	$R(t)$

Note: so, $L(t)$ specifically includes only those ancestors of node t for which you took the left branch at that ancestor node when moving from the root to t.For example, in the tree of the previous slide: $L(5) = \{1\}$, $R(5) = \{2\}$, $A(5) = \{1, 2\}$. Considering 5 I took the left branch from 2 so $L(5) = 2$.

We divide the nodes in the tree into two sets:

1. **Branch nodes:** nodes $t \in T_B = \{1, \ldots, \lfloor T/2 \rfloor\}$ apply a split of the form $\boldsymbol{a}^\top \boldsymbol{x} < b$. Points that satisfy this split follow the left branch in the tree and those that do not follow the right branch.
2. **Leaf nodes:** nodes $t \in T_L = \{\lfloor T/2 \rfloor + 1, \ldots, T\}$ make a class prediction for each point that falls into the leaf node.

We track the split applied at node $t \in T_B$ with variables $\boldsymbol{a}_t \in \mathbb{R}^p$ and $b_t \in \mathbb{R}$.

Remark: we restrict our model to univariate decision trees (like CART), and so the hyperplane split at each node should only involve a single variable. This is enforced by setting the elements of $\boldsymbol{a}_t$ to be binary variables that sum to 1.

We want to allow the option of not splitting at a branch node. We use the binary variables d_t to track which branch nodes apply splits:

$$d_t = \begin{cases} 1 & \text{if node } t \text{ applies a split} \\ 0 & \text{otherwise.} \end{cases}$$

If a branch node does not apply a split, then we model this by setting $\boldsymbol{a}_t = 0$ and $b_t = 0$. This has the effect of forcing all points to follow the right split at this node, since the condition for the left split is $0 < 0$ which is never satisfied.
We enforce this with the following constraints:

$$\sum_{j=1}^{p} a_{jt} = d_t \quad \forall t \in T_B \tag{1}$$

$$0 \leq b_t \leq d_t \quad \forall t \in T_B \tag{2}$$

$$a_{jt} \in \{0,1\} \quad \forall j = 1,\dots,p,\ t \in T_B \tag{3}$$

The second inequality is valid for b_t since we have assumed that each $\boldsymbol{x}_i \in [0,1]^p$, and we know that $\boldsymbol{a}_t$ has one element that is 1 if and only if $d_t = 1$, with the remainder being 0. Therefore, it is always true that

$$0 \leq \boldsymbol{a}_t^\top \boldsymbol{x}_i \leq d_t$$

for any i and t, and we need only consider values for b_t in this same range. Next, we will enforce the hierarchical structure of the tree. We restrict a branch node from applying a split if its parent does not also apply a split:

$$d_t \leq d_{p(t)} \quad \forall t \in T_B \setminus \{1\}, \tag{4}$$

where no such constraint is required for the root node. We have constructed the variables that allow us to model the tree structure using MIO; we now need to track the allocation of points to leaves and the associated errors that are induced by this structure. We introduce the binary variables z_{it} to track the points assigned to each leaf node, where:

$$z_{it} = \begin{cases} 1 & \text{if point } i \text{ is in node } t \\ 0 & \text{otherwise.} \end{cases}$$

We also introduce binary variables l_t, where:

$$l_t = \begin{cases} 1 & \text{if leaf } t \text{ contains any point} \\ 0 & \text{otherwise.} \end{cases}$$

We use these binary variables together to enforce a minimum number of points at each leaf, given by $N_{\min}$:

$$z_{it} \leq l_t \quad \forall t \in T_L \tag{5}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{\min} l_t \quad \forall t \in T_L. \tag{6}$$

We also force each point to be assigned to exactly one leaf:

$$\sum_{t \in T_L} z_{it} = 1 \quad \forall i = 1,\dots,n. \tag{7}$$

Finally, we apply constraints enforcing the splits that are required by the structure of the tree when assigning points to leaves:

$$a_m^\top x_i < b_m + M_1(1 - z_{it}) \quad \forall i = 1,\dots,n,\ t \in T_L,\ m \in L(t) \tag{*}$$

$$a_m^\top x_i \geq b_m - M_2(1 - z_{it}) \quad \forall i = 1,\dots,n,\ t \in T_L,\ m \in R(t)$$

where M_1, M_2 are sufficiently large constants such that the constraints are always satisfied when $z_{it} = 0$. **Note:** that the constraints $(*)$ use a strict inequality which is not supported by MIO solvers, so this must be converted into a form that does not use a strict inequality. To do this we can add a small constant ϵ to the left-hand side of $(*)$ and change the inequality to be non-strict:

$$a_m^\top x_i + \epsilon \leq b_m + M_1(1 - z_{it}) \quad \forall i = 1,\dots,n,\ t \in T_L,\ m \in L(t).$$

However, if ϵ is too small, this could cause numerical instabilities in the MIO solver, so we seek to make ϵ as big as possible without affecting the feasibility of any valid solution to the problem. We can achieve this by specifying a different ϵ_j for each feature j. The largest valid value is the smallest non-zero distance between adjacent values of this feature. To find this, we sort the values of the j-th feature and take:

$$\epsilon_j = \min\left\{x_{(i+1),j} - x_{(i),j} \;\middle|\; x_{(i+1),j} \neq x_{(i),j},\ i = 1,\dots,n-1\right\}.$$

We can then use these values for ϵ in the constraint, where the value of ϵ_j that is used is selected according to the feature we are using for this split:

$$a_m^\top (x_i + \epsilon - \epsilon_{\min}) + \epsilon_{\min} \leq b_m + M_1(1 - z_{it}) \quad \forall i = 1,\dots,n,\ t \in T_L,\ m \in L(t)$$

and where $\epsilon_{\min} = \min_j\{\epsilon_j\}$.We must also specify values for the big-M constants M_1 and M_2. As mentioned previously, we know that both $a_t^\top x_i \in [0,1]$ and $b_t \in [0,1]$, and so the largest possible value of $a_t^\top(x_i + \epsilon) - b_t$ is $1 + \epsilon_{\max}$, where $\epsilon_{\max} = \max_j\{\epsilon_j\}$. We can therefore set:

$$M_1 = 1 + \epsilon_{\max}.$$

Similarly, we have the largest possible value of $b_t - a_t^\top x_i$ is 1, so we can set:

$$M_2 = 1.$$

This gives the following final constraints that will enforce the splits in the tree:

$$a_m^\top (x_i + \epsilon - \epsilon_{\min}) + \epsilon_{\min} \leq b_m + (1 + \epsilon_{\max})(1 - z_{it}) \quad \forall i = 1,\dots,n,\ t \in T_L,\ m \in L(t) \tag{8}$$

$$a_m^\top x_i \geq b_m - (1 - z_{it}) \quad \forall i = 1,\dots,n,\ t \in T_L,\ m \in R(t). \tag{9}$$

The objective is to minimize the misclassification error, so an incorrect label prediction has cost 1, and a correct label prediction has cost 0. We set N_{kt} to be the number of points of label k in node t, and N_t to be the total number of points in node t:

$$N_{kt} = \sum_{i : y_i = k} z_{it} \quad \forall t \in T_L,\ k = 1,\dots,K \tag{10}$$

$$N_t = \sum_{i=1}^{n} z_{it} \quad \forall t \in T_L. \tag{11}$$

We need to assign a label to each leaf node t in the tree, which we denote with $c_t \in \{1, \ldots, K\}$. It is clear that the optimal label to predict is the most common of the labels among all points assigned to the node:

$$c_t = \arg \max_{k=1,\ldots,K} \{N_{kt}\}. \quad (\diamond)$$

We will use binary variables c_{kt} to track the prediction of each node, where:

$$c_{kt} = \begin{cases} 1 & \text{if } c_t = k \\ 0 & \text{otherwise.} \end{cases}$$

We must make a single class prediction at each leaf node that contains points:

$$\sum_{k=1}^{K} c_{kt} = l_t \quad \forall t \in T_L. \quad (12)$$

Since we know how to make the optimal prediction at each leaf t using $(\diamond)$, the optimal misclassification loss in each node, denoted L_t, is going to be equal to the number of points in the node less the number of points of the most common label:

$$L_t = N_t - \max_{k=1,\ldots,K} \{N_{kt}\} = \min_{k=1,\ldots,K} \{N_t - N_{kt}\},$$

which can be linearized to give:

$$L_t \geq N_t - N_{kt} - M(1 - c_{kt}) \quad \forall t \in T_L,\, k = 1, \ldots, K \quad (13)$$

$$L_t \leq N_t - N_{kt} + M c_{kt} \quad \forall t \in T_L,\, k = 1, \ldots, K \quad (14)$$

$$L_t \geq 0 \quad \forall t \in T_L \quad (15)$$

where again M is a sufficiently large constant that makes the constraint inactive depending on the value of c_{kt}. Here, we can take $M = n$ as a valid value. The total misclassification cost is therefore:

$$\sum_{t \in T_L} L_t$$

and the complexity C of the tree is the number of splits included in the tree, given by:

$$C = \sum_{t \in T_B} d_t. \quad (16)$$

Following CART, we normalize the misclassification against the baseline accuracy, $\hat{L}$, obtained by simply predicting the most popular class for the entire dataset. This makes the effect of α independent of the dataset size. This means the objective from problem (P) can be written:

$$\min \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \cdot C.$$

Putting all of this together gives the following MIO formulation for (P).

11.2 The Optimal Classification Trees (OCT) Model

$$\begin{aligned} \min \quad & \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \cdot C \\ \text{s.t.} \quad & (1) \text{ to } (16) \\ & z_{it}, l_t, c_{kt} \in \{0, 1\} \quad \forall i = 1, \ldots, n,\, k = 1, \ldots, K,\, t \in T_L \\ & d_t \in \{0, 1\} \quad \forall j = 1, \ldots, p,\, t \in T_B. \end{aligned}$$

Remark: this model as presented is in a form that can be directly solved by any MIO solver. The difficulty of the model is primarily determined by the number of binary variables z_{it}, which is $n \cdot 2^D$.
Empirically:

- We observe that we can find high-quality solutions in minutes for depths up to 4 for datasets with thousands of points.
- Beyond this depth or dataset size, the rate of finding solutions is slower, and more time is required.

11.3 OCT with Hyperplane Splits

So far, we have only considered decision trees that use a single variable in their splits at each node, known as univariate decision trees. Now, we show that it is simple to extend our MIO formulation for univariate trees to yield a problem for determining the ***optimal multivariate decision tree (DT)***. This shows the flexibility and power of modeling the problem using MIO. In a multivariate DT, we are no longer restricted to choosing a single variable upon which to split, and instead can choose a general ***hyperplane split at each node***. The variables a_t will be used to model the split at each node as before, except we relax (3) and instead choose $a_t \in [-1, 1]^p$ at each branch node t. We must modify (1) to account for the possibility these elements are negative by dealing with the absolute values instead:

$$\sum_{j=1}^{p} |a_{jt}| \leq d_t \quad \forall t \in T_B.$$

which can be linearized using auxiliary variables to track the value of $|a_{jt}|$:

$$\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t \quad \forall t \in T_B \quad (17)$$

$$\hat{a}_{jt} \geq a_{jt} \quad \forall t \in T_B,\, j = 1, \ldots, p \quad (18)$$

$$\hat{a}_{jt} \geq -a_{jt} \quad \forall t \in T_B. \quad (19)$$

These constraints force the split to be all zeros if $d_t = 0$ and no split is applied, otherwise imposing no restriction on a_t.
We now have that $a_t^\top x_i \in [-1, 1]$, so we replace (2) with:

$$-d_t \leq b_t \leq d_t \quad \forall t \in T_B. \quad (20)$$

Now we consider the split constraints $(*)$ and (9). Previously we had that the range of $(a_t^\top x_i - b_t)$ was $[-1, 1]$, whereas it is now $[-2, 2]$. This means we need $M = 2$ to ensure that the constraint is trivially satisfied when $z_{it} = 0$. The constraints therefore become:

$$a_m^\top x_i < b_m + 2(1 - z_{it}) \quad \forall i = 1, \ldots, n,\, t \in T_L,\, m \in L(t) \quad (*)$$

$$a_m^\top x_i \geq b_m - 2(1 - z_{it}) \quad \forall i = 1, \ldots, n,\, t \in T_L,\, m \in R(t). \tag{21}$$

As before, we need to convert the strict inequality in $(*)$ to a non-strict version. We do this by introducing a sufficiently small constant μ:

$$a_m^\top x_i + \mu \leq b_m + (2 + \mu)(1 - z_{it}) \quad \forall i = 1, \ldots, n,\, t \in T_L,\, m \in L(t). \tag{22}$$

Note that we need to include μ in the rightmost term to ensure the constraint is always satisfied when $z_{ik} = 0$.

Remark:

- Unlike in the univariate case, we cannot choose a value for μ in an intelligent manner, and instead need to choose a small constant.
- We must take care when choosing the value of μ. A value too small can lead to numerical issues in the MIO solver, while too large reduces the size of the feasible region, potentially reducing the quality of the optimal solution. We take $\mu = 0.005$ as a compromise between these extremes.

In the univariate case, we controlled the complexity of the tree by penalizing the number of splits. In the multivariate regime, a single split may use multiple variables, and it seems natural to treat splits with a greater number of variables as more complex than those with fewer. To achieve this, we can instead penalize the total number of variables used in the splits of the tree, which we note in the univariate case is exactly the number of splits in the tree.

To achieve this, we introduce binary variables s_{jt} to track if the j-th feature is used in the t-th split:

$$-s_{jt} \leq a_{jt} \leq s_{jt} \quad \forall t \in T_B,\, j = 1, \ldots, p. \tag{23}$$

We must also make sure that the values of s_{jt} and d_t are compatible. The following constraints ensure that $d_t = 1$ if and only if any variable is used in the split:

$$s_{jt} \leq d_t \quad \forall t \in T_B,\, j = 1, \ldots, p \tag{24}$$

$$\sum_{j=1}^{p} s_{jt} \geq d_t \quad \forall t \in T_B. \tag{25}$$

Finally, we modify the definition of complexity C to penalize the number of variables used across the splits in the tree rather than simply the number of splits:

$$C = \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt}. \tag{26}$$

Combining all of these changes yields:

The Optimal Classification Trees with Hyperplanes (OCT-H) Model

$$\begin{aligned}
\min \quad & \frac{1}{\hat{L}} \sum_{t \in T_L} L_t + \alpha \cdot C \\
\text{s.t.} \quad & \text{(4) to (7)} \\
& \text{(10) to (15)} \\
& \text{(17) to (26)} \\
& z_{it}, l_t, c_{kt} \in \{0, 1\} \quad \forall i = 1, \ldots, n,\, k = 1, \ldots, K,\, t \in T_L \\
& d_t, s_{jt} \in \{0, 1\} \quad \forall j = 1, \ldots, p,\, t \in T_B.
\end{aligned}$$

Remark: from this formulation, we can easily obtain the OCT problem as a special case by restoring the integrality constraints on a_t. The close relationship between the univariate and multivariate problems reinforces the notion that the MIO formulation is the natural way to view the decision tree problem.

11.4 Notation

Left branch of nodo for point i	$\boldsymbol{a}^T\boldsymbol{x}_i < b$
Classical DT methods	*top down greedy approach*
Splitting criterion	*label impurity*
Min number nodes per region ssplit	N_{min}
Per each region assigned a label	$1, \dots, K$
Pruning the tree	*prevent overfitting*
	complexity parameter α
Small tree	*high* α
Tree	$\mathbb{T}$
Misclassification error	$R(\mathbb{T})$
N° of branch nodes in tree	$\lvert\mathbb{T}\rvert$
Left node	l
N° of training points left points	$N(l)$
DT in MIO	*from local to global opt*
Depth	D
Parent node of node t	$p(t)$
Anchestors of node t	$A(t) = L(t) \cup R(t)$
Achestors from root on left till t	$L(t)$
Achestors from root on right till t	$R(t)$
Total nodes	$2^{D+1} - 1$
Branch nodes	$t \in T_B = \{1, \dots, \lfloor T/2 \rfloor\}$
Leaf nodes	$t \in T_L = \{\lfloor T/2 \rfloor + 1, \dots, T\}$
Node splitted	d_t *binary var*
Variable selected per split (single)	$\boldsymbol{a}_t = \sum a_{it} = 1$ *var binary*
Point i in node t	z_{it} *binary var*
Leaf node t contains points	l_t *binary var*
Large costants	M_1, M_2
N° of points label k in node t	N_{kt}
Label to each leaf node t	c_t
Prediction each node t	c_{kt}

Node t no split $\Rightarrow d_t = 0 \Rightarrow a_t = 0 \Rightarrow b_t = 0$.

$$\boldsymbol{a}^T\boldsymbol{x}_i < b \;\Rightarrow\; 0\boldsymbol{x}_i < b \;\Rightarrow\; \textit{never satisfied}$$

12 Proofs

Lemma: $\boldsymbol{x}^*$ be a local minimum of a convex function $f : \text{dom}(f) \to \mathbb{R}$. Then $\boldsymbol{x}^*$ is a global minimum, meaning that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \text{dom}(f)$.

Proof [(1)]: suppose $\boldsymbol{x}^*$ is the minimizer and there exists $\boldsymbol{y} \in \text{dom}(f)$ such that $f(\boldsymbol{y}) < f(\boldsymbol{x}^*)$. Define $\boldsymbol{y}' := \lambda \boldsymbol{x}^* + (1-\lambda)\boldsymbol{y}$ for $\lambda \in (0,1)$. From convexity, we get that $f(\boldsymbol{y}') < f(\boldsymbol{x}^*)$. Choosing λ so close to 1 that $\|\boldsymbol{y}' - \boldsymbol{x}^*\| < \varepsilon$ yields a contradiction to $\boldsymbol{x}^*$ being a local minimum.

12.1 Vanilla analysis

In questa dimostrazione, stiamo cercando di ottenere un limite superiore per la quantità $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$, che rappresenta la differenza tra il valore della funzione obiettivo nel punto $\boldsymbol{x}_t$ e il valore della funzione obiettivo nel punto ottimo $\boldsymbol{x}^*$.

- Abbiamo definito $g_t := \nabla f(\boldsymbol{x}_t)$, che è il gradiente della funzione obiettivo f calcolato nel punto $\boldsymbol{x}_t$ all'iterazione t. La regola di aggiornamento per la discesa del gradiente può essere scritta come:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma g_t$$
$$g_t = \frac{\boldsymbol{x}_t - \boldsymbol{x}_{t+1}}{\gamma}$$

  Ora, consideriamo il prodotto scalare tra il gradiente g_t e la differenza $\boldsymbol{x}_t - \boldsymbol{x}^*$, dove $\boldsymbol{x}^*$ è l'ottimo. Questo ci aiuta a connettere il gradiente alla quantità di interesse $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$.

$$\boldsymbol{x}^* \textit{ minimizer iff } \Rightarrow \nabla f(\boldsymbol{x}^*)^T(\boldsymbol{x} - \boldsymbol{x}^*) \geq 0, \quad \forall \boldsymbol{x} \in X$$

$$g_t{}^T(\boldsymbol{x}_t - \boldsymbol{x}^*) = \frac{1}{\gamma}(\boldsymbol{x}_t - \boldsymbol{x}_{t+1})^T(\boldsymbol{x}_t - \boldsymbol{x}^*)$$

- Apply $2\boldsymbol{v}^T\boldsymbol{w} = ||\boldsymbol{v}||^2 + ||\boldsymbol{w}||^2 - ||\boldsymbol{v} - \boldsymbol{w}||^2$

$$2\boldsymbol{v}^T\boldsymbol{w} = \frac{1}{2}\left(||\boldsymbol{v}||^2 + ||\boldsymbol{w}||^2 - ||\boldsymbol{v} - \boldsymbol{w}||^2\right)$$

$$\begin{aligned} g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*) &= \frac{1}{\gamma}(\boldsymbol{x}_t - \boldsymbol{x}_{t+1})^T(\boldsymbol{x}_t - \boldsymbol{x}^*) \\ &= \frac{1}{2\gamma}(||\boldsymbol{x}_t - \boldsymbol{x}_{t+1}||^2 + ||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2) \\ &= \frac{1}{2\gamma}(||\gamma g_t||^2 + ||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2) \\ &= \frac{\gamma}{2}||g_t||^2 + \frac{1}{2\gamma}(||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2) \end{aligned}$$

- Ora sommiamo entrambi i lati dell'equazione sopra per $t = 0$ fino a $T-1$:

$$\sum_{t=0}^{T-1} g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*) = \sum_{t=0}^{T-1}\left(\frac{\gamma}{2}||g_t||^2 + \frac{1}{2\gamma}\left(||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2\right)\right)$$

  Notiamo che nella somma telescopica dei termini $||\boldsymbol{x}_t - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_{t+1} - \boldsymbol{x}^*||^2$, quasi tutti i termini si cancellano, lasciandoci solo il primo e l'ultimo termine:

$$\sum_{t=0}^{T-1} g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*) = \frac{\gamma}{2}\sum_{t=0}^{T-1}||g_t||^2 + \frac{1}{2\gamma}(||\boldsymbol{x}_0 - \boldsymbol{x}^*||^2 - ||\boldsymbol{x}_T - \boldsymbol{x}^*||^2)$$

- Sappiamo che una funzione convessa soddisfa la seguente disuguaglianza per qualsiasi $\boldsymbol{x}$ e $\boldsymbol{y}$:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$$

  Impostando $\boldsymbol{x} = \boldsymbol{x}_t$ e $\boldsymbol{y} = \boldsymbol{x}^*$, otteniamo:

$$f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t) \geq g_t^T(\boldsymbol{x}^* - \boldsymbol{x}_t)$$

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*)$$

  Questo ci consente di applicare questa disuguaglianza alla somma di $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$:

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq g_t^T(\boldsymbol{x}_t - \boldsymbol{x}^*)$$
$$\sum_{t=0}^{T-1}\left(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)\right) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1}||g_t||^2 + \frac{1}{2\gamma}||\boldsymbol{x}_0 - \boldsymbol{x}^*||^2 - \cancel{\frac{1}{2\gamma}||\boldsymbol{x}_{T+1} - \boldsymbol{x}^*||^2}$$

Oss: abbiamo dunque un limite superiore attraverso tutte le iterazioni, last iterate is not necessarily the best one, stepsize is crucial.

12.2 Lipschitz Convex Functions

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, with a global minimum x^*. Furthermore, suppose that $\|x_0 - x^*\| \leq R$ and $\|\nabla f(x)\| \leq B$ for all x. By choosing the step size γ as follows:

$$\gamma := \frac{R}{B\sqrt{T}}$$

gradient descent yields:

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \leq \frac{RB}{\sqrt{T}}.$$

Proof [(2)]: we start by plugging $\|x_0 - x^*\| \leq R$ and $\|\nabla f(x)\| \leq B$ into the Vanilla Analysis:

$$\begin{aligned} \sum_{t=0}^{T-1}(f(x_t) - f(x^*)) &\leq \frac{\gamma}{2}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 + \frac{1}{2\gamma}\|x_0 - x^*\|^2. \\ &\leq \frac{\gamma}{2}B^2T + \frac{1}{2\gamma}R^2 \end{aligned}$$

Next, we choose γ to minimize the following quadratic function:

$$q(\gamma) = \frac{\gamma}{2}B^2T + \frac{R^2}{2\gamma}.$$

To find the minimum of $q(\gamma)$, we solve $q'(\gamma) = 0$:

$$q'(\gamma) = \frac{B^2T}{2} - \frac{R^2}{2\gamma^2} = 0.$$

Solving for γ yields:

$$\gamma = \frac{R}{B\sqrt{T}}.$$

Substituting $\gamma = \frac{R}{B\sqrt{T}}$ into $q(\gamma)$ gives:

$$q\left(\frac{R}{B\sqrt{T}}\right) = \frac{RB}{\sqrt{T}}.$$

Finally, dividing by T gives the result:

$$\frac{1}{T}\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \leq \frac{RB}{\sqrt{T}}.$$

12.3 Sufficient decrease lemma

Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and smooth with parameter L. With step size $\gamma := \frac{1}{L}$, gradient descent satisfies:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2, \quad \forall t \geq 0.$$

Proof [(3)]**:** we begin by using the smoothness condition of the function f and the definition of the gradient descent update $x_{t+1} - x_t = -\frac{1}{L}\nabla f(x_t)$:

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)^\top (x_{t+1} - x_t) + \frac{L}{2}\|x_t - x_{t+1}\|^2.$$

Substitute the update rule $x_{t+1} - x_t = -\frac{1}{L}\nabla f(x_t)$:

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)^\top \left(-\frac{1}{L}\nabla f(x_t)\right) + \frac{L}{2}\left\|-\frac{1}{L}\nabla f(x_t)\right\|^2.$$

Simplifying each term:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{L}\|\nabla f(x_t)\|^2 + \frac{1}{2L}\|\nabla f(x_t)\|^2.$$

Finally, we combine the terms:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2.$$

This shows that the function value decreases by at least $\frac{1}{2L}\|\nabla f(x_t)\|^2$ at each step, which proves the sufficient decrease in gradient descent with step size $\frac{1}{L}$.

12.4 Smooth Convex Functions

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, with a global minimum x^*. Furthermore, suppose that f is smooth with parameter L. By choosing the step size $\gamma := \frac{1}{L}$, gradient descent yields the following bound on the function value after T iterations:

$$f(x_T) - f(x^*) \leq \frac{L}{2T}\|x_0 - x^*\|^2, \quad T > 0.$$

Proof: we begin with the Vanilla Analysis inequality:

$$\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 + \frac{1}{2\gamma}\|x_0 - x^*\|^2.$$

This time, we can bound the squared gradients using the sufficient decrease property:

$$\frac{1}{2L}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 \leq \sum_{t=0}^{T-1}(f(x_t) - f(x_{t+1})),$$

which is equivalent to:

$$\frac{1}{2L}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 \leq f(x_0) - f(x_T).$$

Now, substituting $\gamma = \frac{1}{L}$ into the Vanilla Analysis inequality, we have:

$$\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \leq \frac{1}{2L}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 + \frac{L}{2}\|x_0 - x^*\|^2.$$

By the sufficient decrease property, we also know that:

$$\frac{1}{2L}\sum_{t=0}^{T-1}\|\nabla f(x_t)\|^2 \leq f(x_0) - f(x_T).$$

Thus, we can rewrite the above inequality as:

$$\sum_{t=0}^{T-1}(f(x_t) - f(x^*)) \leq f(x_0) - f(x_T) + \frac{L}{2}\|x_0 - x^*\|^2.$$

Rewriting this further, we have:

$$\sum_{t=1}^{T}(f(x_t) - f(x^*)) \leq \frac{L}{2}\|x_0 - x^*\|^2.$$

Since the last iterate x_T is the best (by the sufficient decrease property), we obtain:

$$f(x_T) - f(x^*) \leq \frac{1}{T}\sum_{t=1}^{T}(f(x_t) - f(x^*)) \leq \frac{L}{2T}\|x_0 - x^*\|^2.$$

12.5 Strongly convex

Theorem let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum x^*; suppose that f is smooth with parameter L and strongly convex with parameter $\mu > 0$. Choosing $\gamma := \frac{1}{L}$, gradient descent with arbitrary x_0 satisfies the following two properties:

- Squared distances to x^* are geometrically decreasing:

$$\|x_{t+1} - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right) \|x_t - x^*\|^2, \quad t \geq 0. \tag{*}$$

- The absolute error after T iterations is exponentially small in T:

$$f(x_T) - f(x^*) \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2, \quad T > 0. \tag{**}$$

Proof of (*): bounding the noise:

$$\begin{aligned} 2\gamma(f(x^*) - f(x_t)) + \gamma^2\|\nabla f(x_t)\|^2 &= \frac{2}{L}(f(x^*) - f(x_t)) + \frac{1}{L^2}\|\nabla f(x_t)\|^2 \\ \textit{strong convexity} &\leq \frac{2}{L}(f(x_{t+1}) - f(x_t)) + \frac{1}{L^2}\|\nabla f(x_t)\|^2 \\ \textit{sufficient decrease} &\leq -\frac{1}{L^2}\|\nabla f(x_t)\|^2 + \frac{1}{L^2}\|\nabla f(x_t)\|^2 = 0 \end{aligned}$$

Hence, the noise is non-positive, and we get:

$$\|x_{t+1} - x^*\|^2 \leq (1 - \mu\gamma)\|x_t - x^*\|^2 = \left(1 - \frac{\mu}{L}\right)\|x_t - x^*\|^2.$$

Proof of (**): from (*) we get:

$$\|x_T - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2.$$

Using smoothness together with $\nabla f(x^*) = 0$:

$$\begin{aligned} f(x_T) - f(x^*) &\leq \nabla f(x^*)^\top (x_T - x^*) + \frac{L}{2}\|x_T - x^*\|^2 \\ &= 0 + \frac{L}{2}\|x_T - x^*\|^2 \end{aligned}$$

Putting it together:

$$f(x_T) - f(x^*) \leq \frac{L}{2}\|x_T - x^*\|^2 \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^T \|x_0 - x^*\|^2.$$

Conclusion: having $R^2 := \|x_0 - x^*\|^2$ and $T \geq \frac{L}{\mu}\ln\left(\frac{R^2 L}{2\epsilon}\right)$:

$$\text{error} \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^T R^2 \leq \varepsilon$$

12.6 Bounded Stochastic Gradients

Theorem: Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable, x^* a global minimum; furthermore, suppose that $\|x_0 - x^*\| \leq R$ and that $\mathbb{E}[\|g_t\|^2] \leq B^2$ for all t. Choosing the constant stepsize:

$$\gamma := \frac{R}{B\sqrt{T}}$$

stochastic gradient descent yields:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[f(x_t)] - f(x^*) \leq \frac{RB}{\sqrt{T}}.$$

Proof [4]**:** using vanilla analysis (with g_t as the stochastic gradient), we start with the following inequality:

$$\sum_{t=0}^{T-1} g_t^\top (x_t - x^*) \leq \frac{\gamma}{2}\sum_{t=0}^{T-1}\|g_t\|^2 + \frac{1}{2\gamma}\|x_0 - x^*\|^2.$$

Next, we take expectations and apply convexity in expectation:

$$\sum_{t=0}^{T-1} E[f(x_t) - f(x^*)] \leq \sum_{t=0}^{T-1} E[g_t^\top (x_t - x^*)].$$

Substituting the previous inequality into this gives us:

$$\sum_{t=0}^{T-1} E[f(x_t) - f(x^*)] \leq \frac{\gamma}{2}\sum_{t=0}^{T-1} E[\|g_t\|^2] + \frac{1}{2\gamma}\|x_0 - x^*\|^2.$$

Now, since we know that $E[\|g_t\|^2] \leq B^2$ for all t, we can simplify further:

$$\sum_{t=0}^{T-1} E[f(x_t) - f(x^*)] \leq \frac{\gamma}{2}B^2 T + \frac{1}{2\gamma}R^2.$$

The result follows by optimizing γ.

12.7 ~~Tame Strong Convexity~~

Theorem: Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable and strongly convex with parameter $\mu > 0$; let x^* be the unique global minimum of f. With decreasing stepsize

$$\gamma_t := \frac{2}{\mu(t+1)},$$

stochastic gradient descent yields:

$$\mathbb{E}\left[f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T} t \cdot x_t\right) - f(x^*)\right] \leq \frac{2B^2}{\mu(T+1)},$$

where $B^2 := \max_{t=1}^{T} \mathbb{E}[\|g_t\|^2]$. This result is almost the same as that for subgradient descent, but it is in expectation.

Proof [(5)]**:** take expectations over the vanilla analysis before summing up (with varying stepsize γ_t):

$$\mathbb{E}\left[g_t^\top (x_t - x^*)\right] = \frac{\gamma_t}{2}\mathbb{E}[\|g_t\|^2] + \frac{1}{2\gamma_t}\left(\mathbb{E}[\|x_t - x^*\|^2] - \mathbb{E}[\|x_{t+1} - x^*\|^2]\right).$$

Using strong convexity in expectation:

$$\mathbb{E}\left[g_t^\top (x_t - x^*)\right] = \mathbb{E}\left[\nabla f(x_t)^\top (x_t - x^*)\right] \geq \mathbb{E}[f(x_t) - f(x^*)] + \frac{\mu}{2}\mathbb{E}[\|x_t - x^*\|^2].$$

Putting it all together (with $\mathbb{E}[\|g_t\|^2] \leq B^2$):

$$\mathbb{E}[f(x_t) - f(x^*)] \leq \frac{B^2\gamma_t}{2} + \frac{1}{\gamma_t} \cdot \frac{\mu}{2}\mathbb{E}[\|x_t - x^*\|^2] - \frac{1}{2\gamma_t}\mathbb{E}[\|x_{t+1} - x^*\|^2].$$

The proof continues as for subgradient descent, this time with expectations.

12.8 Convergence of Gradients

we will prove that

$$\|\nabla f(x_t)\|_2 \to 0 \quad \text{as} \quad t \to \infty$$

at the same rate as

$$f(x_t) - f(x^*) \to 0$$

in the convex case. However, in the non-convex case, $f(x_t) - f(x^*)$ itself may not converge to 0. It may or may not mean that we converge to a critical point, where

$$\nabla f(y^*) = 0.$$

Theorem: let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with a global minimum x^*; furthermore, suppose that f is smooth with parameter L. Choosing stepsize

$$\gamma := \frac{1}{L},$$

gradient descent yields

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq \frac{2L}{T}(f(x_0) - f(x^*)), \quad T > 0.$$

In particular,

$$\|\nabla f(x_t)\|^2 \leq \frac{2L}{T}(f(x_0) - f(x^*))$$

for some $t \in \{0, \ldots, T-1\}$. Also,

$$\lim_{t\to\infty} \|\nabla f(x_t)\|^2 = 0.$$

Proof: sufficient decrease does not require convexity:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2, \quad t \geq 0.$$

Rewriting this, we get:

$$\|\nabla f(x_t)\|^2 \leq 2L(f(x_t) - f(x_{t+1})).$$

Now, consider the telescoping sum:

$$\sum_{t=0}^{T-1} \|\nabla f(x_t)\|^2 \leq 2L(f(x_0) - f(x_T)) \leq 2L(f(x_0) - f(x^*)).$$

The statement follows by dividing by T.

No Overshooting: in the smooth setting, and with stepsize $\frac{1}{L}$, gradient descent cannot overshoot, i.e., pass a critical point.

12.9 Convergence in One Step on Quadratic Functions

Lemma: on nondegenerate quadratic functions, with any starting point $x_0 \in \mathbb{R}^d$, Newton's method yields $x_1 = x^*$.

Proof [(6)]**:** we have $\nabla f(x) = Mx - q$ (this implies $x^* = M^{-1}q$) and $\nabla^2 f(x) = M$. Hence:

$$x_1 = x_0 - \nabla^2 f(x_0)^{-1}\nabla f(x_0) = x_0 - M^{-1}(Mx_0 - q) = M^{-1}q = x^*.$$

12.10 ~~Coordinate descent~~

Convergence theorem: let f be coordinate-wise smooth with constant L, and strongly convex with parameter $\mu > 0$. Then, coordinate descent with a step-size of $\frac{1}{L}$, following the update rule

$$x_{t+1} := x_t - \frac{1}{L}\nabla_{i_t} f(x_t) e_{i_t},$$

where the active coordinate i_t is chosen uniformly at random, has an expected linear convergence rate given by:

$$\mathbb{E}[f(x_t) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)^t [f(x_0) - f^*].$$

Proof [(7)]**:** plugging the update rule into the smoothness condition, we have:

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}|\nabla_{i_t} f(x_t)|^2.$$

Taking expectation with respect to i_t, we obtain:

$$\mathbb{E}[f(x_{t+1})] \leq f(x_t) - \frac{1}{2L}\mathbb{E}[|\nabla_{i_t} f(x_t)|^2].$$

Since i_t is chosen uniformly at random from $\{1, \ldots, d\}$, we can express the expectation as:

$$\mathbb{E}[|\nabla_{i_t} f(x_t)|^2] = \frac{1}{d}\sum_{i=1}^{d} |\nabla_i f(x_t)|^2.$$

Thus, we have:

$$\mathbb{E}[f(x_{t+1})] \leq f(x_t) - \frac{1}{2dL}||\nabla f(x_t)||^2.$$

Using the lemma that strongly convex functions satisfy the Polyak-Lojasiewicz (PL) inequality:

$$\frac{1}{2}||\nabla f(x)||^2 \geq \mu(f(x) - f^*) \quad \forall x,$$

we can substitute this into our inequality. We obtain:

$$\mathbb{E}[f(x_{t+1})] - f^* \leq f(x_t) - f^* - \frac{\mu}{dL}(f(x_t) - f^*).$$

This simplifies to:

$$\mathbb{E}[f(x_{t+1}) - f^*] \leq \left(1 - \frac{\mu}{dL}\right)(f(x_t) - f^*).$$

This completes the proof.

13 Recap

Algorthims and theorems:

GD: *convex*	$\sum_{t=0}^{T-1}\left(f(\boldsymbol{x}_t)-f(\boldsymbol{x}^*)\right)\leq\frac{\gamma}{2}\sum_{t=0}^{T-1}\|\|g_t\|\|^2+\frac{1}{2\gamma}\|\|\boldsymbol{x}_0-\boldsymbol{x}^*\|\|^2$				
GD: *convex, B-Lipschitz*	$\frac{1}{T}\sum_{t=0}^{T-1}\left(f(x_t)-f(x^*)\right)\leq\frac{RB}{\sqrt{T}}$	$\gamma=\frac{R}{B\sqrt{T}}$	$\frac{RB}{\sqrt{T}}\leq\varepsilon$	$T=\frac{R^2B^2}{\varepsilon^2}$	$O(1/\varepsilon^2)$
Suff. decrese: *L-Smooth*	$f(x_{t+1})\leq f(x_t)-\frac{1}{2L}\|\nabla f(x_t)\|^2,\quad\forall t\geq 0$	$\gamma=\frac{1}{L}$			
GD: *convex, L-Smooth*	$f(x_T)-f(x^*)\leq\frac{L}{2T}\|x_0-x^*\|^2,\quad T>0$	$\gamma=\frac{1}{L}$	$\frac{LR^2}{2T}\leq\varepsilon$	$T=\frac{LR^2}{2\varepsilon}$	$O(1/\varepsilon)$
GD: *L-Smooth, Strongly conv*	$\|x_{t+1}-x^*\|^2\leq\left(1-\frac{\mu}{L}\right)\|x_t-x^*\|^2,\quad t\geq 0$	$\gamma=\frac{1}{L}$	$\frac{LR^2}{2}\left(1-\frac{\mu}{L}\right)^T\leq\varepsilon$	$T=\frac{L}{\mu}\ln\left(\frac{R^2L}{2\varepsilon}\right)$	$O(1/\log\varepsilon)$
	$f(x_T)-f(x^*)\leq\frac{L}{2}\left(1-\frac{\mu}{L}\right)^T\|x_0-x^*\|^2,\quad T>0$				
Nevestrov: *B-Lipschitz*	$f(x_T)-f(x^*)\geq\frac{RB}{2(1+\sqrt{T+1})}\quad T\leq d-1$				
Lemma: *strongly conv*	$f_\mu(x)=f(x)-\frac{\mu}{2}\|x\|^2,\quad x\in\mathrm{dom}(f)\quad$ *is convex*				
SubGD: *strongly conv*	$f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T}t\cdot\boldsymbol{x}_t\right)-f(\boldsymbol{x}^*)\leq\frac{2B^2}{\mu(T+1)}$	$\gamma_t=\frac{2}{\mu(t+1)},\quad t>0$			
SGD: *strongly conv*	$\mathbb{E}\left[f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T}t\cdot x_t\right)-f(x^*)\right]\leq\frac{2B^2}{\mu(T+1)}$	$\gamma_t:=\frac{2}{\mu(t+1)}$			
GD: *L-Smooth*	$\|\nabla f(x_t)\|^2\leq\frac{2L}{T}(f(x_0)-f(x^*))$	$\gamma=\frac{1}{L}$			
AGD: *convex, L-Smooth*	$f(y_T)-f(x^*)\leq\frac{2L\|\|z_0-x^*\|\|^2}{T(T+1)},\quad T>0$	$\gamma=\frac{1}{L}$	$O(1/\sqrt{\varepsilon})$		
Newton's: *II diff, convex*	$\|\nabla^2 f(x)^{-1}\|\leq\frac{1}{\mu},\quad\forall x\in X$	$O(\log\log(1/\varepsilon))$			
	$\|\nabla^2 f(x)-\nabla^2 f(y)\|\leq B\|x-y\|,\quad\forall x,y\in X$				
	$\|x_{t+1}-x^*\|\leq\frac{B}{2\mu}\|x_t-x^*\|^2$				
	$\|x_T-x^*\|\leq\frac{\mu}{B}\left(\frac{1}{2}\right)^{2T-1},\quad T\geq 0\quad\|x_0-x^*\|\leq\frac{\mu}{B}$				
PL inequality: *strongly*	$\frac{1}{2}\|\|\nabla f(x)\|\|^2\geq\mu(f(x)-f^*),\quad\forall x$				
Coordinate D: *L-smooth, PL*	$E[f(x_t)-f^*]\leq\left(1-\frac{\mu}{dL}\right)^t[f(x_0)-f^*]$				

Algorithm	**update**	**Iterations**
GD	$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \gamma \nabla f(\boldsymbol{x}_t) \quad con\ \gamma \geq 0$	
GD lipschitz	$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \frac{R}{B\sqrt{T}} \nabla f(\boldsymbol{x}_t)$	$10.000 \cdot R^2 B^2$
GD smooth convex	$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \frac{1}{L} \nabla f(\boldsymbol{x}_t) \quad con\ L \in \mathbb{R}^+$	$50 \cdot R^2 L$
GD strongly convex	$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \frac{1}{L} \nabla f(\boldsymbol{x}_t) \quad con\ L \in \mathbb{R}^+$	$\frac{L}{\mu} \ln(50 \cdot R^2 L)$
Projected GD	$x_{t+1} = \prod_{x \in X} [x_t - \gamma \nabla f(x_t)]$	
Proximal GD	$x_{t+1} = \text{prox}_{h,\gamma}(x_t - \gamma \nabla g(x_t))$	
	$\text{prox}_{h,\gamma}(z) := \arg\min_y \left\{ \frac{1}{2\gamma} \|y - z\|^2 + h(y) \right\}$	
Subgradient descent	$x_{t+1} := x_t - \gamma_t g_t \quad con \quad g_t \in \partial f(x_t)$	
SGD (with 1 obs)	*sample* $i \in [n]$ *uniformly at random*	
	$x_{t+1} := x_t - \gamma_t \nabla f_i(x_t)$	
Batch SGD	$x_{t+1} := x_t - \gamma_t \frac{1}{m} \sum_{j=1}^{m} \nabla f_{h_j}(x_t)$	
Accelerated GD	$y_{t+1} := x_t - \frac{1}{L} \nabla f(x_t)$	
	$z_{t+1} := z_t - \frac{t+1}{2L} \nabla f(x_t)$	
	$x_{t+1} := \frac{t+1}{t+3} y_{t+1} + \frac{2}{t+3} z_{t+1}.$	$\boldsymbol{x}_0 = \boldsymbol{z}_0 = \boldsymbol{y}_0$
Newton's Method	$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$	I taylor
	$x_{t+1} := x_t - \frac{f'(x_t)}{f''(x_t)} = x_t - f''(x_t)^{-1} f'(x_t)$	
	$x_{t+1} := x_t - \nabla^2 f(x_t)^{-1} \nabla f(x_t)$	II taylor
Secant method	$x_{t+1} := x_t - \frac{f(x_t)(x_t - x_{t-1})}{f(x_t) - f(x_{t-1})}$	finite diff
	$x_{t+1} := x_t - \frac{f'(x_t)(x_t - x_{t-1})}{f'(x_t) - f'(x_{t-1})}$	
Coordinate descent	$x_{t+1} := x_t - \frac{1}{L} \nabla_{i_t} f(x_t) e_{i_t}$	
	$\arg\min_{\gamma \in \mathbb{R}} f(x_t + \gamma e_{i_t})$	

Notations:

- $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| = R$
- se smooth richiede sempre la differenziabilitá
- strongly convex $\Rightarrow$ strinctly convex $\Rightarrow$ unico minimo globale
- PL $\Rightarrow$ strongly convex, same μ

Remark: the objective function $f(\boldsymbol{x})$ that has to be minimized or maximized can be a paraboloid based on the strong convexity or smoothness property.

$$\textbf{Smooth:} \quad f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|x - y\|^2, \quad \forall x, y \in X.$$

$$\textbf{Strongly conv:} \quad f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}\|x - y\|^2, \quad \forall x, y \in X.$$

$$x_{t+1} = \arg\min_y \left[g(x_t) + \nabla g(x_t)^\top (y - x_t) + \frac{1}{2\gamma}\|y - x_t\|^2 \right], \quad \gamma = \frac{1}{L}$$

14 Project

$$a = \sum_{j=0}^{d-1} \varphi_j \cdot \theta_j = \boldsymbol{\varphi}^T \cdot \boldsymbol{\theta} \in \mathbb{R} \tag{6}$$

$$s(a) = \frac{1}{1+e^{-a}} = \frac{e^a}{a+e^a} = \begin{cases} a \gg 0 & \Rightarrow \quad s(a) \approx 1 \\ a \ll 0 & \Rightarrow \quad s(a) \approx 0 \end{cases} \tag{7}$$

$$P(y=1|\boldsymbol{\varphi}) = s(a) = s(\boldsymbol{\varphi}^T \cdot \boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\varphi}^T \cdot \boldsymbol{\theta}}} \equiv \pi \tag{8}$$

$$y \sim Bernoulli(\pi) = \pi^y \cdot (1-\pi)^{1-y} \tag{9}$$

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \prod_{i=1}^{N} P\big(y(i)|\boldsymbol{\varphi(i)}, \boldsymbol{\theta}\big) = \prod_{i=1}^{N} \pi(i)^{y(i)} \cdot \big(1-\pi(i)\big)^{1-y(i)} \tag{10}$$

$$\begin{aligned}
-\ln\big[\mathcal{L}(\pi|Y)\big] &= -\ln\left[\prod_{i=1}^{N} \pi(i)^{y(i)} \cdot \big(1-\pi(i)\big)^{1-y(i)}\right] \\
&= -\sum_{i=1}^{N} \ln\left[\pi(i)^{y(i)} \cdot \big(1-\pi(i)\big)^{1-y(i)}\right] \\
&= -\sum_{i=1}^{N} \left(\ln\big[\pi(i)^{y(i)}\big] + \ln\big[\big(1-\pi(i)\big)^{1-y(i)}\big]\right) \\
&= -\sum_{i=1}^{N} \Big(y(i) \cdot \ln\big[\pi(i)\big] + \big(1-y(i)\big) \cdot \ln\big[1-\pi(i)\big]\Big) \\
&= \sum_{i=1}^{N} \left(y(i) \cdot \ln\left[\frac{1}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right] + \big(1-y(i)\big) \cdot \ln\left[\frac{e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right]\right) \\
&= \sum_{i=1}^{N} \left[y(i) \cdot \left(\ln\left(\frac{1}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right) - \ln\left(\frac{e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right)\right) + \ln\left(\frac{e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right)\right] \\
&= \sum_{i=1}^{N} \left[y(i) \cdot \left(\ln\left(\frac{1}{e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right)\right) + \ln\left(\frac{e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right)\right] \\
&= \sum_{i=1}^{N} \left[y(i) \cdot \boldsymbol{\varphi}_i^\top \boldsymbol{\theta} + \ln\left(\frac{e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right)\right] \\
&= \sum_{i=1}^{N} \left[y(i) \cdot \boldsymbol{\varphi}_i^\top \boldsymbol{\theta} + \ln\left(\frac{1}{1+e^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}\right)\right] \\
&= \sum_{i=1}^{N} \left(y(i) \cdot \boldsymbol{\varphi}_i^\top \boldsymbol{\theta} - \ln\left[1+e^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}\right]\right) \equiv J(\boldsymbol{\theta})
\end{aligned} \tag{11}$$

$$\pi(i) \equiv P\big(y(i)=1|\boldsymbol{\varphi}(i)\big) = s(\boldsymbol{\varphi}(i)^T \cdot \boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\varphi}(i)^T \cdot \boldsymbol{\theta}}} \tag{12}$$

$$\hat{\theta} = \arg\min_{\theta} J(\theta) = \arg\min_{\theta} \sum_{i=1}^{N} \left[\ln\left(1 + e^{\boldsymbol{x}_i^\top \theta}\right) - y(i) \cdot \boldsymbol{x}_i^\top \theta \right] \tag{13}$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ln(1 + e^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}) = \frac{e^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}}{1 + e^{\boldsymbol{\varphi}_i^\top \boldsymbol{\theta}}} \cdot \boldsymbol{\varphi}_i = \pi(i) \cdot \boldsymbol{\varphi}_i \tag{14}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{N} (\pi(i) \cdot \boldsymbol{\varphi}_i - y(i) \cdot \boldsymbol{\varphi}_i) \tag{15}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{N} (\pi(i) - y(i)) \cdot \boldsymbol{\varphi}_i \tag{16}$$

$$\begin{aligned} f_Y(y(1), y(2), \ldots, y(N)|\mu, \sigma^2) = f_Y(Y|\mu, \sigma^2) = \mathcal{L}(\mu, \sigma^2|Y) &= \prod_{i=1}^{N} \mathcal{N}(y(i)|\mu, \sigma^2) \\ &= f_y(y(1)|\mu, \sigma^2) \cdot \ldots \cdot f_y(y(N)|\mu, \sigma^2) \end{aligned} \tag{17}$$

$$\underset{2\times 1}{\widehat{\boldsymbol{\theta}}_{ML}} = \begin{bmatrix} \widehat{\mu} \\ \widehat{\sigma}^2 \end{bmatrix} = arg \ \max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}|Y) = arg \ \max_{\boldsymbol{\theta}} \prod_{i=1}^{N} \mathcal{N}\big(y(i)|\mu, \sigma^2\big) \tag{18}$$

Asymptotically correct:	$\lim_{N\to+\infty} \mathbb{E}\big[\widehat{\boldsymbol{\theta}}_{ML}\big] = \boldsymbol{\theta}^0$
Consistent:	the larger the N, the more precise the estimate
Asymptotically efficient:	$\lim_{N\to+\infty} \mathrm{Var}\big[\widehat{\boldsymbol{\theta}}_{ML}\big] = M^{-1}$
Asymptotically normal:	$\widehat{\boldsymbol{\theta}}_{ML} \sim \mathcal{N}(\boldsymbol{\theta}^0, M^{-1})$ as $N \to +\infty$

$$a = \sum_{j=0}^{d-1} x_j \cdot \theta_j = \boldsymbol{x}^\top \cdot \boldsymbol{\theta} \in \mathbb{R} \tag{1}$$

$$s(a) = \frac{1}{1+e^{-a}} = \frac{e^a}{a+e^a} = \begin{cases} a \gg 0 & \Rightarrow \quad s(a) \approx 1 \\ a \ll 0 & \Rightarrow \quad s(a) \approx 0 \end{cases} \tag{2}$$

$$P(y=1|\boldsymbol{x}) = s(a) = s(\boldsymbol{x}^T \cdot \boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{x}^T \cdot \boldsymbol{\theta}}} \equiv \pi \tag{3}$$

$$y \sim \text{Bernoulli}(\pi) = \pi^y \cdot (1-\pi)^{1-y} \tag{4}$$

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \prod_{i=1}^{N} P\big(y(i)|\boldsymbol{x}(\boldsymbol{i}), \boldsymbol{\theta}\big) = \prod_{i=1}^{N} \pi(i)^{y(i)} \cdot \big(1-\pi(i)\big)^{1-y(i)} \tag{5}$$

$$\begin{aligned}
-\ln\left[\mathcal{L}(\pi|Y)\right] &= -\ln\left(\prod_{i=1}^{N} \pi(i)^{y(i)} \cdot (1-\pi(i))^{1-y(i)}\right) \\
&= -\sum_{i=1}^{N} \ln\left(\pi(i)^{y(i)} \cdot (1-\pi(i))^{1-y(i)}\right) \\
&= -\sum_{i=1}^{N} \left(\ln\left[\pi(i)^{y(i)}\right] + \ln\left[(1-\pi(i))^{1-y(i)}\right]\right) \\
&= -\sum_{i=1}^{N} \left(y(i) \cdot \ln\left[\pi(i)\right] + (1-y(i)) \cdot \ln\left[1-\pi(i)\right]\right) \\
&= -\sum_{i=1}^{N} \left(y(i) \cdot \ln\left[\frac{1}{1+e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}\right] + (1-y(i)) \cdot \ln\left[\frac{e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}\right]\right) \\
&= -\sum_{i=1}^{N} \left(y(i) \cdot \ln\left[\frac{1}{e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}\right] + \ln\left[\frac{e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}\right]\right) \\
&= -\sum_{i=1}^{N} \left(y(i) \cdot \boldsymbol{x}_i^\top \boldsymbol{\theta} + \ln\left[\frac{e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}{1+e^{-\boldsymbol{x}_i^\top \boldsymbol{\theta}}}\right]\right) \\
&= -\sum_{i=1}^{N} \left(y(i) \cdot \boldsymbol{x}_i^\top \boldsymbol{\theta} + \ln\left[\frac{1}{1+e^{\boldsymbol{x}_i^\top \boldsymbol{\theta}}}\right]\right) \\
&= -\sum_{i=1}^{N} \left(y(i) \cdot \boldsymbol{x}_i^\top \boldsymbol{\theta} - \ln\left[1+e^{\boldsymbol{x}_i^\top \boldsymbol{\theta}}\right]\right) \equiv J(\boldsymbol{\theta})
\end{aligned} \tag{6}$$

$$\pi(i) \equiv P\big(y(i)=1|\boldsymbol{x}(i)\big) = s(\boldsymbol{x}(i)^T \cdot \boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{x}(i)^T \cdot \boldsymbol{\theta}}}$$

$$\hat{\theta} = \arg\min_{\theta} J(\theta) = \arg\min_{\theta} \left\{ -\sum_{i=1}^{N} \left[\ln\left(1+e^{\boldsymbol{x}_i^\top \theta}\right) - y(i) \cdot \boldsymbol{x}_i^\top \theta\right] \right\} \tag{7}$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ln(1+e^{\boldsymbol{x}_i^\top \boldsymbol{\theta}}) = \frac{e^{\boldsymbol{x}_i^\top \boldsymbol{\theta}}}{1+e^{\boldsymbol{x}_i^\top \boldsymbol{\theta}}} \cdot \boldsymbol{x}_i = \pi(i) \cdot \boldsymbol{x}_i \tag{8}$$

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= -\sum_{i=1}^{N} (y(i) \cdot \boldsymbol{x}_i - \pi(i) \cdot \boldsymbol{x}_i) \\ &= -\sum_{i=1}^{N} (y(i) - \pi(i)) \cdot \boldsymbol{x}_i \end{aligned} \tag{9}$$

$$\begin{aligned} \nabla_{\theta} J(\theta) &= -\mathbf{X}^\top (\boldsymbol{y} - \boldsymbol{\pi}) \\ &= -\mathbf{X}^\top (\boldsymbol{y} - \widehat{y}) \end{aligned} \tag{10}$$

$$\nabla_{\boldsymbol{\theta}} \pi(i) = \pi(i)(1-\pi(i)) \cdot \boldsymbol{x}_i \tag{11}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \sum_{i=1}^{N} (\pi(i) - y(i)) \cdot \boldsymbol{x}_i$$

$$\begin{aligned} H(\boldsymbol{\theta}) \equiv \nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \left(\pi(i) - y(i)\right) \cdot \boldsymbol{x}_i \\ &= \sum_{i=1}^{N} \pi(i)(1-\pi(i)) \cdot \boldsymbol{x}_i \cdot \boldsymbol{x}_i^\top \end{aligned} \tag{12}$$

$$W = \begin{bmatrix} \pi(1)(1-\pi(1)) & 0 & 0 & \cdots & 0 \\ 0 & \pi(2)(1-\pi(2)) & 0 & \cdots & 0 \\ 0 & 0 & \pi(3)(1-\pi(3)) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \pi(N)(1-\pi(N)) \end{bmatrix}$$

$$H(\boldsymbol{\theta}) = X^\top W X \tag{13}$$

Tabella 1: Algorithms executed, all of them has a $t_{max} = 300000$ step. The tolerance conditions used as termination coditions is the following: $f(\boldsymbol{x}) - f(\boldsymbol{x^*}) \leq \varepsilon = 10^{-6}$. In order to not repeat the same concept will be filled with dots just look the first filled above row

Algorithm	**Iterative step**	**Notes**
Gradient descent (GD0)	$x_{t+1} := x_t - \gamma \nabla f(x_t)$	$x_0 \in \mathbb{R}^d$ *starting point, stepsize* $\gamma \geq 0$
Gradient descent (GD1)	...	Bold driver algorithm (update after m steps)
Gradient descent (GD2)	...	Armijo rule
Gradient descent (GD3)	...	Binary search
Gradient descent smooth	$x_{t+1} := x_t - \frac{1}{L} \nabla f(x_t)$	$\gamma = \frac{1}{L}$, where $L = \frac{1}{4}\|X^T X\| \geq 0$
SGD1	$x_{t+1} := x_t - \gamma_t \nabla f_i(x_t)$,	$t = 0, 1, \ldots$: sample $i \in \{1, \ldots, N\}$ uniformly at random
SGD2	$x_{t+1} := x_t - \gamma_t \frac{1}{m} \sum_{j=1}^{m} \nabla f_{h_j}(x_t)$	$t = 0, 1, \ldots$: choose a random subset $\{h_1, \ldots, h_m\} \subseteq \{1, \ldots, N\}$
Accelerated GD		
Newton's method		
Quasi-Newton Methods		

Tabella 2: Performance comparison of various gradient descent algorithms. The initial $\boldsymbol{\theta_0}$ values were randomized to low values to ensure an unbiased starting point for optimization. The table includes the number of steps taken and the execution time for each algorithm, highlighting the efficiency and effectiveness of each approach in minimizing the logistic loss function.

Algorithm	**Steps**	**Execution Time (s)**
Gradient descent (GD0)	5680	4.7516
Gradient descent (GD1)	0	0.0103
Gradient descent (GD2)	150	0.3100
Gradient descent (GD3)	150	4.8402
Gradient descent smooth	2	0.0049

Estimated $\hat{\boldsymbol{\theta}}$ vectors:

$$\begin{bmatrix} -0.218 \\ 1.249 \\ 1.701 \\ 1.246 \\ 0.158 \\ 1.012 \\ 1.148 \\ -0.181 \end{bmatrix} \quad \begin{bmatrix} -0.024 \\ 0.224 \\ 0.250 \\ 0.229 \\ -0.003 \\ 0.243 \\ 0.226 \\ -0.025 \end{bmatrix} \quad \begin{bmatrix} -0.269 \\ 1.385 \\ 1.955 \\ 1.359 \\ 0.174 \\ 1.102 \\ 1.281 \\ -0.231 \end{bmatrix} \quad \begin{bmatrix} -0.269 \\ 1.385 \\ 1.955 \\ 1.359 \\ 0.174 \\ 1.102 \\ 1.281 \\ -0.231 \end{bmatrix} \quad \begin{bmatrix} -0.006 \\ 0.011 \\ 0.005 \\ -0.004 \\ -0.002 \\ -0.006 \\ -0.004 \\ 0.008 \end{bmatrix}$$