CIS3333: Mathematics of Machine Learning Fall 2025

Lecture: Constrained Optimization and Duality

Date: November 24th, 2025 *Author: Surbhi Goel*

In the past few lectures, we studied unconstrained optimization:

$$\min_{x \in \mathbb{R}^n} f(x).$$

We developed gradient descent and SGD, proved convergence rates, and saw how to choose step sizes. But many real-world problems have *constraints*. For example:

- In PCA, we maximize $u^\top \Sigma u$ subject to $\|u\|_2 = 1$. Without the constraint, we could scale u arbitrarily large.
- In linear regression, we sometimes constrain the parameter norm: $\|\theta\|_2 \leq R$. This prevents overfitting.
- For probability distributions p over n items, we require $p_i \geq 0$ and $\sum_i p_i = 1$. These are definitional constraints.

Today we'll see how to extend gradient descent to handle constraints. The general form is:

$$\min_{x \in \mathcal{C}} f(x),$$

where $\mathcal{C} \subseteq \mathbb{R}^n$ is a constraint set.

1 Projected Gradient Descent

Let us focus on the case of $\mathcal{C} \subseteq \mathbb{R}^n$ being a *convex* set (e.g., $\mathcal{C} = \{x : \|x\|_2 \leq R\}$ or $\mathcal{C} = \{x : x \geq 0\}$). We will modify gradient descent to account for the constraint.

The idea is simple: take a gradient step (which might leave $\mathcal{C}$), then project back onto the constraint set. The projection operator is defined as follows:

Definition 1 (Euclidean Projection)**.** *The projection of a point y onto a closed convex set $\mathcal{C}$ is:*

$$\Pi_{\mathcal{C}}(y) = \arg\min_{x \in \mathcal{C}} \|x - y\|_2.$$

In words, $\Pi_{\mathcal{C}}(y)$ is the closest point in $\mathcal{C}$ to y. This should look familiar from the linear algebra module: when $\mathcal{C}$ is a subspace (which is a convex set, check!), this is the orthogonal projection. Here we have generalized it to any convex set.

The **Projected Gradient Descent (PGD)** update rule is:

$$y_{t+1} = x_t - \eta \nabla f(x_t) \qquad \text{(same as GD/SGD)}$$
$$x_{t+1} = \Pi_{\mathcal{C}}(y_{t+1}) \qquad \text{(project back to } \mathcal{C}\text{)}$$

At each iteration, we move in the direction of steepest descent, then project back to the constraint set if we've left it.

1.1 Why does PGD work?

In the last couple of lectures, we proved that GD and SGD converge: the function value $f(x_T)$ approaches the optimum $f(x^*)$. The key technique in both proofs was to track the distance to the optimum, $\|x_t - x^*\|_2^2$, as a *potential function*. For PGD, we take an unconstrained gradient step (which gives y_{t+1}) and then project (which gives x_{t+1}). *Will this extra projection step break the convergence proof?*

The answer is no, thanks to a geometric property of projections. For any *convex* set $\mathcal{C}$, the projection operator is *non-expansive*: projecting a point y onto $\mathcal{C}$ cannot move it farther away from any point $x \in \mathcal{C}$. This means that the projection step can only *decrease* the distance to the optimum ($x^* \in \mathcal{C}$), so the same convergence proof from GD carries over[1].

Proposition 2 (Non-expansiveness of projection)**.** *For any closed convex set $\mathcal{C}$, any $y \in \mathbb{R}^n$ and any $x \in \mathcal{C}$,*

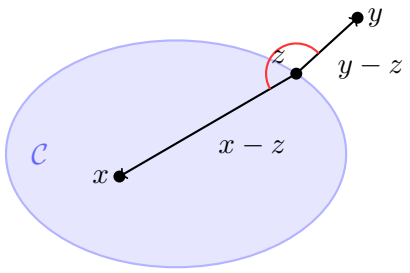
$$\|\Pi_{\mathcal{C}}(y) - x\|_2 \leq \|y - x\|_2.$$

To prove this, we will use the following lemma:

Lemma 3 (Obtuse angle property of projections)**.** *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be closed[2] and convex, $y \in \mathbb{R}^n$, and $z = \Pi_{\mathcal{C}}(y)$ its Euclidean projection. Then for every $x \in \mathcal{C}$,*

$$\langle y - z,\ x - z \rangle \leq 0.$$

Geometric intuition. The lemma says that the angle between the vectors $(y - z)$ and $(x - z)$ is obtuse ($\geq 90°$). The following figure illustrates this:



Now let's formally prove the lemma.

Proof of Lemma 3. The key idea is to use the fact that z is the *closest* point in $\mathcal{C}$ to y.

Fix any $x \in \mathcal{C}$. Because $\mathcal{C}$ is convex, the line segment between z and x lies entirely in $\mathcal{C}$:

$$x_t := (1 - t)z + tx \in \mathcal{C}, \qquad \text{for all } t \in [0, 1].$$

[1] We omit the full proof here, but it is a good exercise to prove it for yourself using the non-expansiveness property of projection.

[2] A set is closed if it contains all its limit points, that is, if every convergent sequence in the set converges to a point in the set.

Since z is the closest point in $\mathcal{C}$ to y, we have

$$\|y - z\|_2 \leq \|y - x_t\|_2 \quad \text{for all } t \in [0, 1].$$

Squaring both sides and rearranging:

$$0 \leq \|y - x_t\|_2^2 - \|y - z\|_2^2 \quad \text{for all } t \in [0, 1].$$

Now expand $\|y - x_t\|_2^2$. Write $x_t = z + t(x - z)$, so $y - x_t = (y - z) - t(x - z)$. Then

$$\|y - x_t\|_2^2 = \|(y - z) - t(x - z)\|_2^2 = \|y - z\|_2^2 - 2t\langle y - z, x - z\rangle + t^2\|x - z\|_2^2.$$

Subtracting $\|y - z\|_2^2$ from both sides:

$$0 \leq -2t\langle y - z, x - z\rangle + t^2\|x - z\|_2^2 \quad \text{for all } t \in [0, 1].$$

Let $t > 0$ and divide both sides by t:

$$0 \leq -2\langle y - z, x - z\rangle + t\|x - z\|_2^2.$$

Now take the limit as $t \to 0^+$. The term $t\|x - z\|_2^2$ vanishes, leaving:

$$0 \leq -2\langle y - z, x - z\rangle \quad \Rightarrow \quad \langle y - z, x - z\rangle \leq 0.$$

This completes the proof. □

Now we can prove the non-expansiveness of projection:

Proposition 4 (Non-expansiveness of projection)**.** *For any closed convex set $\mathcal{C}$, any $y \in \mathbb{R}^n$ and any $x \in \mathcal{C}$,*

$$\|\Pi_{\mathcal{C}}(y) - x\|_2 \leq \|y - x\|_2.$$

Proof. Let $z = \Pi_{\mathcal{C}}(y)$. For any $x \in \mathcal{C}$, expand:

$$\|y - x\|_2^2 = \|(y - z) + (z - x)\|_2^2 = \|y - z\|_2^2 + 2\langle y - z, z - x\rangle + \|z - x\|_2^2.$$

By Lemma 3, $\langle y - z, x - z\rangle \leq 0$, which is equivalent to $\langle y - z, z - x\rangle \geq 0$. Thus

$$\|y - x\|_2^2 \;\geq\; \|y - z\|_2^2 + \|z - x\|_2^2 \;\geq\; \|z - x\|_2^2.$$

Taking square roots gives $\|z - x\|_2 \leq \|y - x\|_2$, as desired. □

What goes wrong for non-convex sets? Convexity is essential. If $\mathcal{C}$ is not convex, the line segment between z and x may leave $\mathcal{C}$, so the proof above breaks down.

Example (1D). Let $\mathcal{C} = \{-1, 1\} \subset \mathbb{R}$ (two isolated points). Take $x = -1 \in \mathcal{C}$ and $y = 0.1$.

- The distance from y to x is $|y - x| = |0.1 - (-1)| = 1.1$.
- The projection of y onto $\mathcal{C}$ is $z = 1$, because $|0.1 - 1| = 0.9$, which is smaller than $|0.1 - (-1)| = 1.1$.
- Now check the distance from the projection z to x: $|z - x| = |1 - (-1)| = 2$.

We see that $|z - x| = 2 > 1.1 = |y - x|$. The projection step moved us *farther* from x!

1.2 Examples of Projections

For PGD to be practical, we need to be able to compute $\Pi_{\mathcal{C}}(y)$ efficiently. Fortunately, for many common constraint sets, projection has a simple closed form or can be computed quickly.

- **Non-negative orthant ($x \geq 0$):** $\Pi(y)_i = \max(0, y_i)$ for each coordinate i. This is just the ReLU function applied coordinate-wise! This can be computed in $O(n)$ time.
- **Box constraints ($l \leq x \leq u$):** Clip each coordinate to the interval $[l, u]$:

$$\Pi(y)_i = \begin{cases} l & \text{if } y_i < l \\ y_i & \text{if } l \leq y_i \leq u \\ u & \text{if } y_i > u \end{cases}$$

  This can be computed in $O(n)$ time.
- **ℓ_2-Ball ($\|x\|_2 \leq R$):** If $\|y\|_2 \leq R$, then y is already in the ball, so $\Pi(y) = y$. If $\|y\|_2 > R$, scale y down to the boundary:

$$\Pi(y) = R \cdot \frac{y}{\|y\|_2}.$$

  This can be computed in $O(n)$ time.
- **Probability simplex ($p_i \geq 0, \sum_i p_i = 1$):** Given $y \in \mathbb{R}^n$, there is an $O(n \log n)$ algorithm (sort, then shift-and-clip) to project onto the simplex. We won't cover the details here; see Chen and Ye (2011) for the algorithm.

2 Lagrangian Duality

Projected GD works well when we can easily project onto $\mathcal{C}$. But what if the constraint set is complex? For example, projecting onto the intersection of 100 halfspaces is not straightforward. An alternative approach is to convert the constraints into *penalties* in the objective function.

Consider a general constrained problem:

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

For example, $g_i(x) = \|x\|_2^2 - R^2$ encodes the constraint $\|x\|_2 \leq R$.

The Lagrangian. We introduce nonnegative multipliers $\lambda_i \geq 0$ (one per constraint) and define the *Lagrangian*:

$$\mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x).$$

Think of λ_i as a *cost per unit violation* of constraint i. For any fixed $\lambda \geq 0$, $\mathcal{L}(x, \lambda)$ is an *unconstrained* objective, and $\lambda_i g_i(x)$ is the penalty for violating constraint i.

The primal problem. We can rewrite the original constrained problem as:

$$p^* = \min_x \max_{\lambda \geq 0} \mathcal{L}(x, \lambda).$$

Why? Let's look at the inner maximization for a fixed x:

- If x is feasible (all $g_i(x) \leq 0$), then $\lambda_i g_i(x) \leq 0$. To maximize the sum, we should set $\lambda_i = 0$ (or anything if $g_i(x) = 0$). The max value is exactly $f(x)$.
- If x is infeasible (some $g_k(x) > 0$), we can let $\lambda_k \to \infty$, so the max value is $+\infty$.

So the inner max acts as an "indicator function": it returns $f(x)$ if feasible and $+\infty$ if not. Thus, minimizing this over x is exactly the original constrained problem! This is called the *primal* formulation.

2.1 The Dual Problem and Weak Duality

Swapping the order. What if we swap the order of min and max? Define the *dual function*:

$$D(\lambda) := \min_x \mathcal{L}(x, \lambda) = \min_x \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \right).$$

For each choice of penalty weights $\lambda \geq 0$, $D(\lambda)$ is the best value we can achieve by minimizing the penalized objective. The *dual problem* is:

$$d^* = \max_{\lambda \geq 0} D(\lambda).$$

Interpretation: over all choices of penalty weights $\lambda \geq 0$, which one gives the best (tightest) lower bound on the constrained optimum?

Weak Duality. The dual always provides a lower bound on the primal. To see this, fix any $\lambda \geq 0$ and any feasible x (i.e., $g_i(x) \leq 0$ for all i). Then:

$$D(\lambda) = \min_{x'} \mathcal{L}(x', \lambda) \leq \mathcal{L}(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \leq f(x),$$

where the last inequality uses $\lambda_i \geq 0$ and $g_i(x) \leq 0$. Taking $\max_{\lambda \geq 0}$ on the left and min over feasible x on the right:

$$d^* \ \leq \ p^*.$$

This is called *weak duality*: the dual optimum is always a lower bound on the primal optimum.

Strong duality. Under additional conditions (e.g., if f and g_i are convex and Slater's condition holds), we have $d^* = p^*$ (*strong duality*). We won't prove this here, but it's a powerful result: it means we can solve the dual problem instead of the primal, which is sometimes easier.

2.2 Example: Constrained vs. Regularized Regression

Let's revisit the linear regression example from the beginning of the lecture. Recall that we often add an ℓ_2 penalty to prevent overfitting. There are two equivalent ways to do this:

Constrained form (hard constraint).

$$\min_{\theta} \|Y - X\theta\|_2^2 \quad \text{s.t.} \quad \|\theta\|_2^2 \leq R^2.$$

This says: minimize the squared error, but don't let the parameter norm exceed R.

Regularized form (soft penalty).

$$\min_{\theta} \|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2.$$

This is ridge regression: we add a penalty $\lambda\|\theta\|_2^2$ to the objective. Large λ encourages small $\|\theta\|$.

Connection via Lagrangian duality. The constrained form has one constraint: $g(\theta) = \|\theta\|_2^2 - R^2 \leq 0$. The Lagrangian is:

$$\mathcal{L}(\theta, \lambda) = \|Y - X\theta\|_2^2 + \lambda(\|\theta\|_2^2 - R^2).$$

For any fixed $\lambda \geq 0$, minimizing over θ gives:

$$\theta_\lambda = \arg\min_{\theta} \Big(\|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2 \Big).$$

This is exactly the ridge objective! So for each λ, the Lagrangian gives us a ridge problem.

The dual function is:

$$D(\lambda) = \min_{\theta} \mathcal{L}(\theta, \lambda) = \min_{\theta} \Big(\|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2 \Big) - \lambda R^2.$$

Maximizing $D(\lambda)$ over $\lambda \geq 0$ (the dual problem) chooses the penalty weight λ so that the optimal ridge solution θ_λ satisfies $\|\theta_\lambda\|_2^2 \approx R^2$.

Lagrangian duality shows that for the right choice of λ (found by solving the dual), the two give the same solution. This is why we often use ridge regression in practice: it's easier to optimize (unconstrained), and duality guarantees it's equivalent to the constrained form for some radius R.

3 Course Wrap-Up: Connecting the Threads

Let's go back to the first lecture when we defined the machine learning problem. We have training data $(x_1, y_1), \ldots, (x_N, y_N)$ drawn from some distribution $\mathcal{D}$, a hypothesis class $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$ of candidate predictors, and a loss function ℓ that measures prediction quality. We said that the training objective is to find the function f that minimizes the empirical risk:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}(f), \quad \text{where } \hat{R}(f) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i).$$

While we minimize the empirical risk $\hat{R}(f)$ on the training data, we actually care about the true risk $R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$ on new data.

This raised three fundamental questions, which organized the three modules of this course.

3.1 Generalization: From Probability to Uniform Convergence

The question: Why does minimizing empirical risk $\hat{R}(f)$ on training data lead to low true risk $R(f)$ on new data?

The tools: We modeled data as draws from a distribution and used concentration inequalities (Markov, Chebyshev, Chernoff, Hoeffding) to show that empirical averages concentrate around their expectations.

The result: For finite hypothesis classes with bounded loss, the gap $|R(f) - \hat{R}(f)|$ is at most $O\left(\sqrt{\frac{\log|\mathcal{F}|+\log(1/\delta)}{N}}\right)$ with high probability. This bound holds uniformly over all $f \in \mathcal{F}$.

The techniques: The central tools are concentration inequalities and union bounds. Many generalization proofs follow the same pattern: first bound the chance that one function's empirical average is far from its true expectation, then use a union bound to make the guarantee hold for all $f \in \mathcal{F}$ at once.

3.2 Representation: From Linear Algebra to Kernels and RKHS

The question: What functions can we represent with finite data? Can we work in infinite-dimensional feature spaces without infinite parameters?

The tools: We formalized data as vectors and models as linear maps, studying vector spaces, projections, eigenvalues, and PCA. We then extended these concepts from $\mathbb{R}^n$ to infinite-dimensional Hilbert spaces and Reproducing Kernel Hilbert Spaces (RKHS), where inner products measure similarity and kernels $K(x, x')$ implicitly define feature maps.

The result: The Representer Theorem shows that for regularized ERM over an RKHS $\mathcal{H}$, the optimal solution has the finite form $\hat{f}(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x)$, despite $\mathcal{H}$ being infinite-dimensional.

The techniques: The key tools are inner products, orthogonality, and kernels. Many representation arguments follow a common pattern: rewrite predictors as linear combinations of basis functions or kernel evaluations $K(x_i, x)$, then use orthogonality or positive semidefiniteness of the kernel matrix to control norms, errors, and complexity.

3.3 Optimization: From Gradients to SGD, Conditioning, and Constraints

The question: How do we actually minimize the empirical risk? How fast can we find good parameters?

The tools: We studied gradients and Hessians as local linear and quadratic approximations. These let us quantify smoothness and curvature, which control how quickly optimization algorithms converge.

The results: For smooth convex functions, gradient descent converges at rate $O(1/T)$: error after T steps is $\lesssim \frac{\text{dist}^2}{T}$. SGD (using single-sample gradients) converges at rate $O(1/\sqrt{T})$, trading speed per iteration for slower overall convergence. Conditioning (ratio $\lambda_{\max}/\lambda_{\min}$ of the Hessian) determines how step sizes must be chosen. We also saw practical fixes—momentum, adaptive methods (Adam)—and how to handle constraints via projection (PGD) or penalties (Lagrangian duality).

The techniques: The workhorses are gradients, Hessians, and basic inequalities using smoothness and convexity. Many convergence proofs follow the same template: use smoothness to upper bound $f(x_{t+1}) - f(x_t)$ under the GD update, telescope the resulting inequality over t, and then use convexity (and conditioning) to turn these bounds into explicit rates in terms of step sizes and curvature.

3.4 Putting It Together

Almost any ML method can be described by four design choices: the **function class**, the **loss**, the **regularizer** (or constraint), and the **optimization algorithm**. Roughly speaking, the function class controls what patterns we can represent, the loss encodes what we care about, the regularizer controls complexity, and the optimization algorithm searches for good parameters. The mathematical foundations from this course (probability, linear algebra, and calculus) are what let us analyze each of these choices: probability tells us when a given loss and regularizer will generalize, linear algebra tells us what the function class can represent, and calculus/optimization tells us how our algorithm will behave. These ideas are the building blocks for much of modern ML, statistics, and optimization, and you will see them in more sophisticated forms in many later courses.