Introduction to Computational Learning Theory **Spring 2021**

Lecture 8: Learning with Convex Functions

Lecturer: Roi Livni *Scribe:*

Hardness of learning binary problems is often attributed to the *non–convexity* of the problem. This has led to the construction of what is referred to as *convex relaxations* for classification tasks. As we will see in this next part of the course, convex functions are extremely attractive as there are multiple efficient algorithms for optimizing over this class of functions (under mild assumptions). In general, the problem in (binary) learning is to minimize an objective function of the form:

$$\underset{h\in\mathcal{H}}{\text{minimize}}\ \text{err}_S(h) = \frac{1}{m}\sum_{i=1}^{m}\ell_{0,1}(h).$$

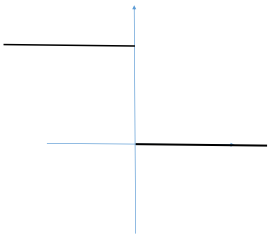
A straightforward and elegant method to solve minimization problems is through what is called *local search methods* where we start at some initial solutions, and iteratively move to improve it.

However, binary classification problems are not always smooth– namely, there is no local structure in the problem, and the problem is mostly combinatorial. Thus one solution might be to "smooth" the loss function. In other words to replace the zero-one loss with a smoother version where the gradient of the function gives us local information. However, merely smoothing might still lead to a hard-to-optimize problem. and without certain assumptions, we cannot always relate the loss of the zero one function to the loss of the new optimization problem. Convex functions, on the other hand, are amenable to local search methods, and as we will discuss can be efficiently optimized.

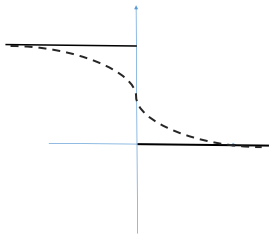
8.0.1 Convex functions and convex sets

We next define what is a convex problem, we begin by defining convex functions:

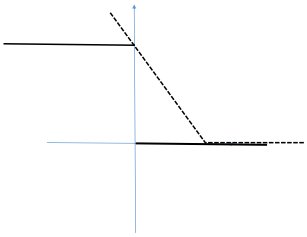
Definition 8.1. *A function $f : X \to \mathbb{R}$ over a domain $X \subseteq \mathbb{R}^d$ is called convex if for every $0 \leq \lambda \leq 1$ and*



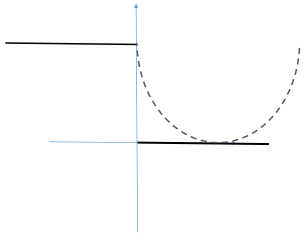
(a) zero-one loss.



(b) A smoothed sigmoidal version of the zero one loss



(c) Hinge-loss, one possibility of a convex surrogate function



(d) Square-loss, yet another convex surrogate

Figure 8.1

$x, y \in X$:

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

If $-f$ is a convex function, then f is called concave.

If f is a differentiable function at x and convex, then the following bound always hold:

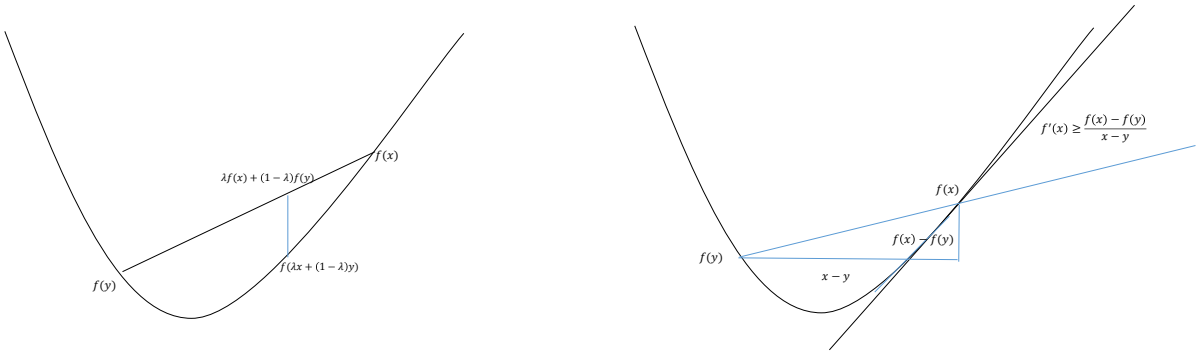
$$f(y) - f(x) \geq \nabla f(x) \cdot (y - x) \tag{8.1}$$

When f is not differentiable at x, we define the subdifferential of f at x denoted

$$\partial f(x) = \{v : f(y) - f(x) \geq v \cdot (y - x),\ \forall y\}.$$

For a convex function the set $\partial f(x)$ is always non empty, and x is the a minimizer of f iff $0 \in \partial f(x)$.

Example 8.1. *The following functions are convex*



(a) $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ (b) $f'(x) \geq \frac{f(x)-f(y)}{x-y}$

Figure 8.2: Convex Function: illustration of two characterizations

1. ℓ_2 ***norm squared:*** *The function* $\|\mathbf{x}\|^2 = \sum_{i=1}^n x_i^2$ *is a convex function.*

2. ℓ_1 ***norm:*** *The function* $\|\mathbf{x}\|_1 = \sum |\mathbf{x}_i|$ *is known to be convex, in particular the absolute value function is convex.*

3. ℓ_p ***norm:*** *The function* $\|\mathbf{x}\|_p = \sqrt[p]{\sum x_i^p}$ *is always convex.*

4. ***General Norms:*** *A function* $\|\cdot\|$ *is called a norm if*

   - $\|x\| \geq 0$ *and* $\|x\| = 0$ *iff* $x = 0$.
   - $\|\lambda x = |\lambda| \|x\|$.
   - $\|x + y\| \leq \|x\| + \|y\|$.

   Then any norm is a convex function.

5. ***Logistic loss*** *the function* $f_\alpha(x) = \ln(1 + \exp(\alpha x))$ *is convex for any* α.

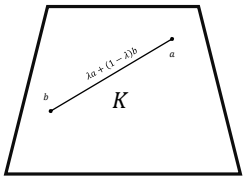
The following facts are usefull for the construction of new convex functions

Fact 8.1.

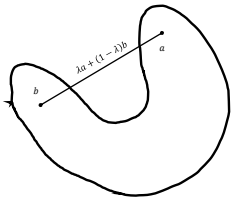
If f, g *are convex then so is* $f + g$.

If f *is convex and* $\alpha \geq 0$ *then* $\alpha \cdot f$ *is convex.*

If f, g *are convex then* $\max(f, g)$ *is convex.*



(a) A convex set



(b) Not a convex set

If f is convex and A is a linear operator then $f(A \cdot \mathbf{x})$ is also convex.

Next, we define a convex set.

Definition 8.2. *A set $K \subseteq \mathbb{R}^n$ is called convex if for every $x, y \in K$ and $0 \leq \lambda \leq 1$:*

$$\lambda x + (1-\lambda)y \in K.$$

Example 8.2. *The following are convex sets*

1. *The positive cone: $\{\mathbf{x} : \mathbf{x}_i \geq 0 i = 1, \ldots, n\} \subseteq \mathbb{R}^n$ is convex.*
2. *The Euclidean ball: $\{\mathbf{x} : \|\mathbf{x}\| \leq 1\}$ is a convex set.*
3. *The unit–cube $\{\mathbf{x} : \max\{\mathbf{x}_i\} \leq 1\}$.*
4. *In general, if f is convex then the epigraph set $\{(\mathbf{x}, \alpha) : f(\mathbf{x}) \leq a\}$ is a convex set.*

Definition 8.3. *A problem of the form*

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in K \end{aligned}$$

is called a convex program if f is convex and K is a convex set.

Equivalently, a concave program is a problem of the form

$$\begin{aligned} \text{maximize} \quad & g(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in K \end{aligned}$$

when g is concave and K is convex.

8.1 Convex learning problems

A convex learning problem is described by a triplet $(\mathcal{Z}, K, \ell)$, where K is identified with some convex set in Euclidean space $\mathbb{R}^d$ and $\ell(\mathbf{w}, \mathbf{z})$ is a convex function over $\mathbf{w}$ for every fixed $\mathbf{z} \in \mathcal{Z}$.

We will focus here on the case that $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is some set in Euclidean space and $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = [-1, 1]$.

We generally assume that the learner observes IID samples from $\mathcal{Z}$, drawn according to some unknown distribution D, and her objective is to minimize the following loss w.r.t $\mathbf{w} \in K$:

$$\mathcal{L}(\mathbf{w}) = \mathbf{E}_{\mathbf{z} \sim D}[\ell(\mathbf{w}, z)]$$

Example 8.3 (Linear Regression)**.** *Sometimes convex learning problems are interesting of their own merit. Such is the example of linear regression. In linear regression we observe samples $(\mathbf{x}, y) \subseteq \mathbb{R}^n \times \mathbb{R}$ and we aim to find a* linear regressor *i.e. a function $f_{\mathbf{w}}(\mathbf{x}) = y$ that would best fit the distribution:*

$$\underset{w}{\text{minimize}} \quad \mathop{\mathbb{E}}_{(x,y) \sim D}[(\mathbf{w} \cdot \mathbf{x} - y)^2]$$

The empirical loss, given sample S, is given by:

$$\underset{w}{\text{minimize}} \quad \frac{1}{m} \sum_{i=1}^{m} [(\mathbf{w} \cdot \mathbf{x}^{(i)} - y_i)^2]$$

Reduction of polynomial interpolation to linear regression We can reduce various problems to the task of linear regression. For example, consider, the following problem: we observe data $(x, y) \sim D$ where $x \in \mathbb{R}$. We want to find a polynomial p of degree d such that $p(x) = y$. (if $d = 1$, this is exactly linear regression).

We can solve this problem by mapping each vector $x \to \mathbf{x}$, where

$$\mathbf{x} = (1, x^1, x^2, \ldots, x^d).$$

.

For a sample point x_j we will denote its embedding in this monomial features space as $\mathbf{x}^{(j)} := (1, x^1, x^2, \ldots, x^d)$.

Now performing linear regression, we can find $\mathbf{w}$ such that

$$p_{\mathbf{w}}(x_j) = \sum_{i=1}^{d+1} w_i x_j^{i-1} = \mathbf{w} \cdot \mathbf{x}^{(j)}.$$

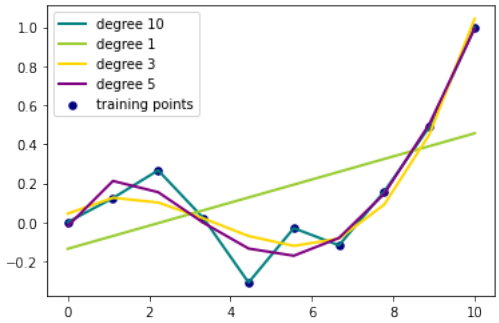
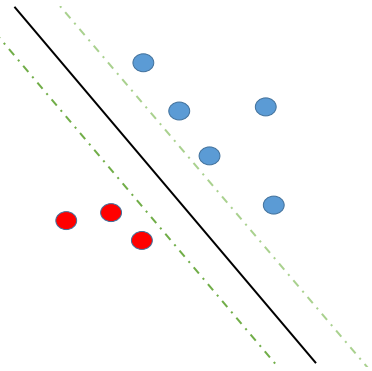


Figure 8.4: Different polynomial interpolation by embedding point $x_j \to \mathbf{x}^{(j)}$, and solving the problem $\sum_{i=1}^{10}(\mathbf{w} \cdot \mathbf{x}^{(j)} - y_j)^2$.

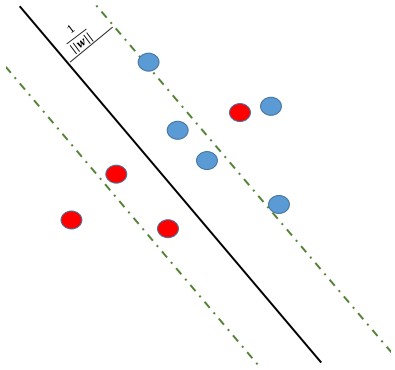
Example 8.4 (SVM)**.** *The support vector machine algorithm is concerened with finding the linear classifier with largest margin, as we will see this can be formulized as minimizing a surrogate loss function.*

For the realizable case, the objective of Hard-SVM may be written as

$$\begin{array}{ll} \text{minimize} & \|\mathbf{w}\|^2 \\ \textit{subject to} & y_i \mathbf{w} \cdot \mathbf{x}_i \geq 1. \forall i = 1, \ldots m. \end{array}$$



(a) The Hard-SVM setup when points are sepearable. The Hard SVM algorithm aims at finding a classifier that separates the norm with at least $\mathbf{w} \cdot \mathbf{x} \geq 1$. The dashed line represent that domain where $\mathbf{w} \cdot \mathbf{x} = 1$



(b) The Soft-SVM aims at finding a solution that minimizes the hinge loss with additional norm regulerization. The solution may include points inside the margin (where the hinge loss penalizes even correctly classified points) for minimizing the loss on misclassified points.

By choosing sufficiently small λ the hard SVM can be relaxed to:

$$\begin{aligned} \underset{\mathbf{w},\xi}{\text{minimize}} \quad & \lambda\|\mathbf{w}\|^2 + \frac{1}{m}\sum \xi_i \\ \text{subject to} \quad & \xi_i \geq 0 \\ & \xi_i \geq 1 - y_i \mathbf{w} \cdot \mathbf{x}^{(i)} \end{aligned}$$

Solving the equation for ξ_i we get:

$$\text{minimize} \quad \lambda\|\mathbf{w}\|^2 + \frac{1}{m}\sum \max(0, 1 - y_i \mathbf{w} \cdot \mathbf{x}^{(i)})$$

Let us denote $\ell_{\text{hinge}}(a, y) = \max(0, 1 - y \cdot a)$. Then we obtain the regulerized SVM formulation

$$\text{minimize} \quad \lambda\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \ell_{\text{hinge}}(\mathbf{w} \cdot \mathbf{x}^{(i)}, y_i)$$

Note that the above problem is well defined even for large λ (even though it will not necessarily converge to a realizable solution, even if one exists). As we will discuss, the effect of λ is in fact desirable to control the *complexity* of $\mathbf{w}$.

For example, again consider the case where $\mathbf{x} = (1, x, x^2, x^3, \ldots, x^n)$. and assume $n \gg m$. We know that for every sample $S = \{x^{(i)}, y_i\}_{i=1}^m$, we can regress some n-degree polynomial $\mathbf{w}_0 + \mathbf{w}_1 \cdot x_i + \mathbf{w}_2 x_i^2 + \cdots \mathbf{w}_n \mathbf{x}_i^n = y_i$. Hence, with such small data set we may easily *overfit*.

Therefore it is customery to consider two versions of convex loss problems that inhibit the complexity of the model

Constrained convex problem:

$$\begin{aligned} &\text{minimize } \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{w}; \mathbf{z}_i) \\ &\text{subject to } \quad \|\mathbf{w}\| < B \end{aligned}$$

Here we control the complexity of the model by selecting B. As we allow B to grow we allow more expressive power to the model, but by increasing B we also increase the complexity of the problem.

Another (and in fact equivalent) formulation is the regulerized version, which we will write for ℓ_2–regulerized loss:

$$\text{minimize } \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{w}; \mathbf{z}_i)$$

Beyond ℓ_2 We can also choose a different *regularization* term, for example replace ℓ_2 norm with ℓ_1 norm

$$\text{minimize } \lambda\|\mathbf{w}\|_1 + \frac{1}{m} \sum \ell(\mathbf{w}; \mathbf{z}_i)$$

To motivate this: Suppose that the true linear classifier is given by a single feature (more generally a *sparse* vector, but for concretness we will assume a special case where there is a single feature that determines the true labelling). In other words, there is a vector $_j = (0, 0, \ldots, \underbrace{1}_{j}, \ldots, 0)$ such that $y = \text{sign}(_j \cdot \mathbf{x} + \sigma n)$. where n is a Gaussian random variable (that introduces noise to the label). Given sampled data $S = \{(x_i, y_i)\}_{i=1}^m$ the objective so far tradeoffs between the ℓ_2 norm of the classifier and the empirical risk. It may prefer a

non-sparse solution (which will exhibit smaller ℓ_2 norm then the classifier $_j$). Intuitively then, the ℓ_1 solution encourages sparse solutions and is a good choice of regularization of we have reasons to assume that there is a good sparse solution to the problem.

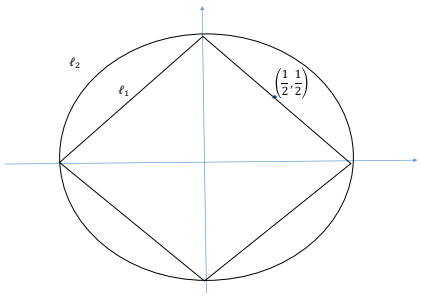


Figure 8.6: ℓ_1 vs. ℓ_2 balls. The ℓ_2 balls penalizes solutions with large ℓ_2 norm, thus for example it would prefer the solution $(\frac{1}{2}, \frac{1}{2})$ over the solutions $(1, 0)$ or $(0, 1)$. ℓ_1 ball, in contrast is less sensitive to the distribution of the weight over the vector, and thus if there is a good sparse solution, it will not penalize it as ℓ_2.

In the following example we can see the effect of different regularization schemes in the task of regression. In this example data was generated according to the following process: each example $\mathbf{x}$ is a vector in $\mathbb{R}^{10}$, where

$$\mathbf{x}^{(j)} = (1, j, j^2, \ldots, j^9).$$

The label y is given by $y = p(x) + 0.2 \cdot n$ where n is Guassian noise and p is the polynomial $p(x) = (x/10)^9$. The noiseless data can be seen in fig. 8.7c

Note that learning a linear classifier over $\mathbf{x}$ is equivalent to learning a polynomial of degree 9 over $\mathbb{R}$: In other words, for any $\mathbf{w} \in \mathbb{R}^{10}$,

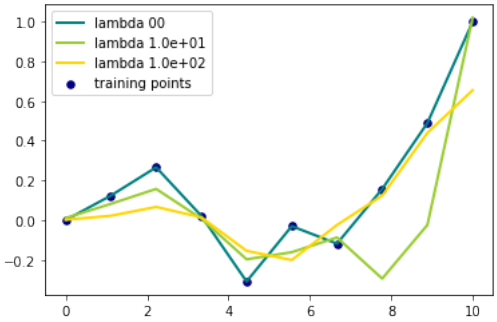
$$\mathbf{w} \cdot \mathbf{x}^{(j)} = \sum_{i=1}^{9} w_i j^i.$$

Next, in fig. 8.7b we consider the learning problem

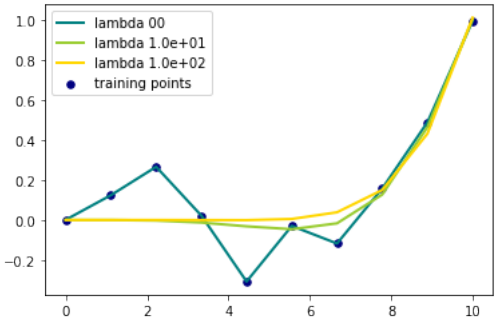
$$\lambda \|\mathbf{w}\|_1 + \frac{1}{m} \sum_{i=1}^{m} (\mathbf{w} \cdot \mathbf{x}^{(i)} - y_i)^2,$$

for $\lambda = 0, 1, 10.$ and similarly in fig. 8.7a we consider the learning problem m

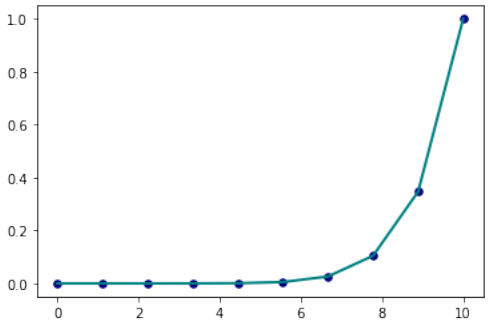
$$\lambda\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}^{(i)} - y_i)^2.$$



(a) Regression of noisy data using Ridge: $\lambda\|\mathbf{w}\|_2^2 + \sum_{i=1}^{10}(\mathbf{w}\cdot\mathbf{x}^{(i)} - y_i)^2$



(b) Regression of noisy data using Lasso: $\lambda\|\mathbf{w}\|_1^2 + \sum_{i=1}^{10}(\mathbf{w}\cdot\mathbf{x}^{(i)} - y_i)^2$



(c) noiseless data $y = (x/10)^9$

More generally, we would want to consider problems of the form:

$$\text{minimize} \quad \underbrace{\lambda R(\mathbf{w})}_{\text{Regularization term}} + \underbrace{\frac{1}{m}\sum_{i=1}^{m}\ell(\mathbf{w}, \mathbf{z}_i)}_{\text{empirical risk}}$$

The first term is called *regularization* - it can be thought of as a measure of "complexity" of the solution. Large regularization loss means high complexity.

The second term is the empirical risk – it is exactly as before, trying to minimize the empirical loss, but not necessarily with respect to the $0-1$ loss, and we might want to consider *surrogate losses*.

Finally, λ is a hyperparameter that controls the tradeoff between "complexity" and "fit". As λ is smaller, we prefer solutions with small empirical loss even at the expanse of complexity.

We will discuss in the next few lectures how this framework allows us to circumvent hardness when certain assumptions are made on the problem. For example, even if agnostic learning is in general hard we will show how under certain assumptions we can still find a good classifier efficiently.

We will also see how, (again, under certain assumptions as VC theory says that this is impossible in the general case) we can search for good linear classifiers in high dimension (i.e. in the regime where $m \ll d$).

References