Empirical Risk Minimization

Cornell CS 4/5780 — Spring 2022

SVM with soft constraints

If the data is low dimensional it is often the case that there is no separating hyperplane between the two classes. In this case, there is no solution to the optimization problems stated above. We can fix this by allowing the constraints to be violated ever so slight with the introduction of slack variables:

$$\begin{gathered}\min_{\mathbf{w},b} \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^n \xi_i \\ s.t.\ \forall i\ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \\ \forall i\ \xi_i \geq 0\end{gathered}$$

The slack variable ξ_i allows the input $\mathbf{x}_i$ to be closer to the hyperplane (or even be on the wrong side), but there is a penalty in the objective function for such "slack". If C is very large, the SVM becomes very strict and tries to get all points to be on the right side of the hyperplane. If C is very small, the SVM becomes very loose and may "sacrifice" some points to obtain a simpler (i.e. lower $\|\mathbf{w}\|_2^2$) solution.

Unconstrained Formulation:

Let us consider the value of ξ_i for the case of $C \neq 0$. Because the objective will always try to minimize ξ_i as much as possible, the equation must hold as an *equality* and we have:

$$\xi_i = \begin{cases} 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b) & \text{if } y_i(\mathbf{w}^T\mathbf{x}_i + b) < 1 \\ 0 & \text{if } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{cases}$$

This is equivalent to the following closed form:

$$\xi_i = \max(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b), 0).$$

If we plug this closed form into the objective of our SVM optimization problem, we obtain the following *unconstrained* version as loss function and regularizer:

$$\min_{\mathbf{w},b} \underbrace{\mathbf{w}^T\mathbf{w}}_{l_2-regularizer} + C\sum_{i=1}^n \underbrace{\max\left[1 - y_i(\mathbf{w}^T\mathbf{x} + b), 0\right]}_{hinge-loss}$$

This formulation allows us to optimize the SVM paramters ($\mathbf{w}, b$) just like logistic regression (e.g. through gradient descent). The only difference is that we have the **hinge-loss** instead of the **logistic loss**.

The hinge loss is the SVM's error function of choice, whereas the l_2-regularizer reflects the complexity of the solution, and penalizes complex solutions.

This is an example of empirical risk minimization with a loss function ℓ and a regularizer r,

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^n \underbrace{l(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{Loss} + \underbrace{\lambda r(w)}_{Regularizer},$$

where the loss function is a continuous function which penalizes training error, and the regularizer is a continuous function which penalizes classifier complexity.[1]

Commonly Used Binary Classification Loss Functions

Different Machine Learning algorithms use different loss functions; Table 4.1 shows just a few:

Loss $\ell(h_{\mathbf{w}}(\mathbf{x}_i, y_i))$	**Usage**	**Comments**
Hinge-Loss $\max\left[1 - h_{\mathbf{w}}(\mathbf{x}_i)y_i, 0\right]^p$	• Standard SVM($p = 1$) • (Differentiable) Squared Hingeless SVM ($p = 2$)	When used for Standard SVM, the loss function denotes the size of the margin between linear separator and its closest points in either class. Only differentiable everywhere with $p = 2$.
Log-Loss $\log(1 + e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i})$	Logistic Regression	One of the most popular loss functions in Machine Learning, since its outputs are well-calibrated probabilities.
Exponential Loss $e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i}$	AdaBoost	This function is very aggressive. The loss of a mis-prediction increases *exponentially* with the value of $-h_{\mathbf{w}}(\mathbf{x}_i)y_i$. This can lead to nice convergence results, for example in the case of Adaboost, but it can also cause problems with noisy data.
Zero-One Loss $\delta(\text{sign}(h_{\mathbf{w}}(\mathbf{x}_i)) \neq y_i)$	Actual Classification Loss	Non-continuous and thus impractical to optimize.

Table 4.1: Loss Functions With Classification $y \in \{-1, +1\}$

Quiz: What do all these loss functions look like with respect to $z = yh(\mathbf{x})$?

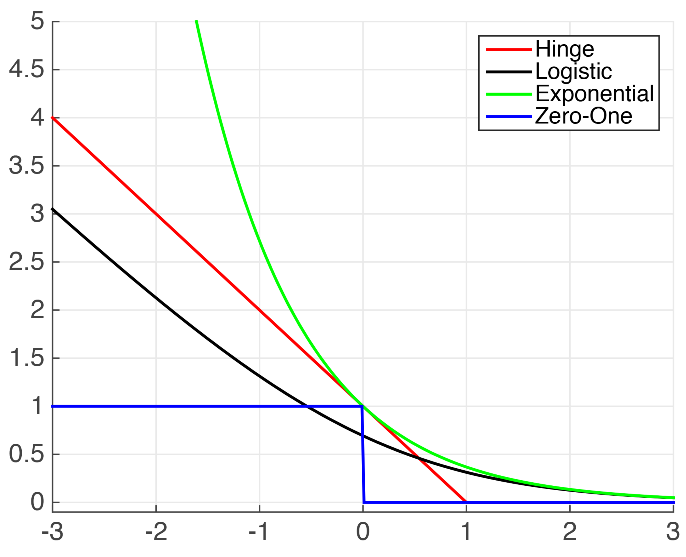

Figure 4.1: Plots of Common Classification Loss Functions - x-axis: $h(\mathbf{x}_i)y_i$, or "correctness" of prediction; y-axis: loss value

Some questions about the loss functions:

1. Which functions are strict upper bounds on the 0/1-loss?
2. What can you say about the hinge-loss and the log-loss as $z \to -\infty$?

Commonly Used Regression Loss Functions

Regression algorithms (where a prediction can lie anywhere on the real-number line) also have their own host of loss functions:

Loss $\ell(h_{\mathbf{w}}(\mathbf{x}_i, y_i))$	**Comments**
Squared Loss $(h(\mathbf{x}_i) - y_i)^2$	○ Most popular regression loss function ○ Estimates Mean Label ○ ADVANTAGE: Differentiable everywhere ○ DISADVANTAGE: Somewhat sensitive to outliers/noise ○ Also known as Ordinary Least Squares (OLS)
Absolute Loss $\|h(\mathbf{x}_i) - y_i\|$	○ Also a very popular loss function ○ Estimates Median Label ○ ADVANTAGE: Less sensitive to noise ○ DISADVANTAGE: Not differentiable at 0
Huber Loss ○ $\frac{1}{2}(h(\mathbf{x}_i) - y_i)^2$ if $\|h(\mathbf{x}_i) - y_i\| < \delta$, ○ otherwise $\delta(\|h(\mathbf{x}_i) - y_i\| - \frac{\delta}{2})$	○ Also known as Smooth Absolute Loss ○ ADVANTAGE: "Best of Both Worlds" of Squared and Absolute Loss ○ Once-differentiable ○ Takes on behavior of Squared-Loss when loss is small, and Absolute Loss when loss is large.
Log-Cosh Loss $log(cosh(h(\mathbf{x}_i) - y_i))$, $cosh(x) = \frac{e^x + e^{-x}}{2}$	ADVANTAGE: Similar to Huber Loss, but twice differentiable everywhere

Table 4.2: Loss Functions With Regression, i.e. $y \in \mathbb{R}$

Quiz: What do the loss functions in Table 4.2 look like with respect to $z = h(\mathbf{x}_i) - y_i$?

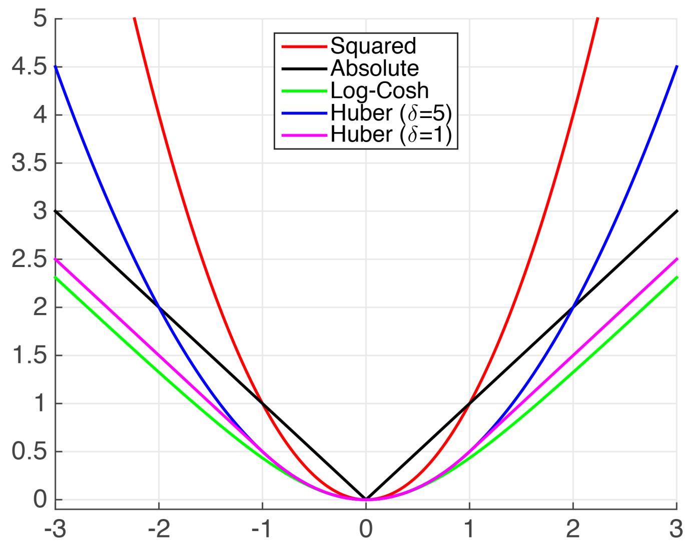

Figure 4.2: Plots of Common Regression Loss Functions - x-axis: $h(\mathbf{x}_i)y_i$, or "error" of prediction; y-axis: loss value

Regularizers

When we look at regularizers it helps to change the formulation of the optimization problem to obtain a better geometric intuition:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}), y_i) + \lambda r(\mathbf{w}) \Leftrightarrow \min_{\mathbf{w},b} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(\mathbf{x}), y_i) \text{ subject to: } r(\mathbf{w}) \leq B$$

For each $\lambda \geq 0$, there exists $B \geq 0$ such that the two formulations in (4.1) are equivalent, and vice versa. In previous sections, l_2-regularizer has been introduced as the component in SVM that reflects the complexity of solutions. Besides the l_2-regularizer, other types of useful regularizers and their properties are listed in Table 4.3.

Regularizer $r(\mathbf{w})$	**Properties**
l_2-**Regularization** $r(\mathbf{w}) = \mathbf{w}^\top \mathbf{w} = \|\mathbf{w}\|_2^2$	◦ ADVANTAGE: Strictly Convex ◦ ADVANTAGE: Differentiable ◦ DISADVANTAGE: Uses weights on all features, i.e. relies on all features to some degree (ideally we would like to avoid this) - these are known as Dense Solutions.
l_1-**Regularization** $r(\mathbf{w}) = \|\mathbf{w}\|_1$	◦ Convex (but not strictly) ◦ DISADVANTAGE: Not differentiable at 0 (the point which minimization is intended to bring us to ◦ Effect: Sparse (i.e. not Dense) Solutions
l_p-**Norm** $\|\mathbf{w}\|_p = (\sum_{i=1}^{d} v_i^p)^{1/p}$	Some additional notes on the Special Cases: 1. Ridge Regression is very fast if data isn't too high dimensional. 2. Ridge Regression is just 1 line of Julia / Python. 3. There is an interesting connection between Ordinary Least Squares and the first principal component of PCA (Principal Component Analysis). PCA also minimizes square loss, but looks at perpendicular loss (the horizontal distance between each point and the regression line) instead. ◦ (often $0 < p \leq 1$) ◦ DISADVANTAGE: Non-convex ◦ ADVANTAGE: Very sparse solutions ◦ Initialization dependent ◦ DISADVANTAGE: Not differentiable

Table 4.3: Types of Regularizers

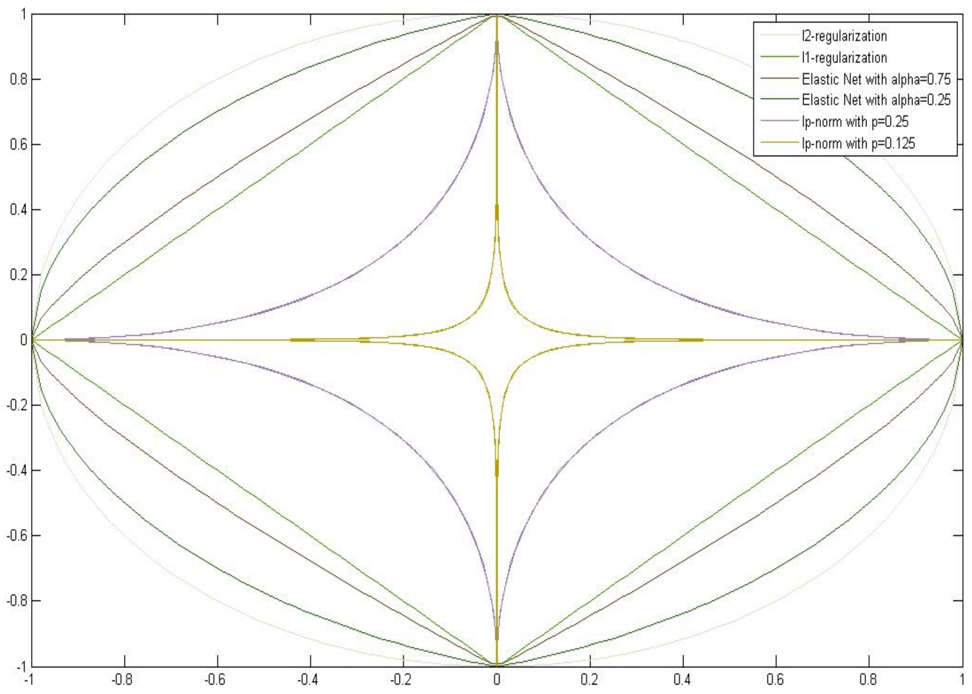

Figure 4.3: Plots of Common Regularizers

[1] In Bayesian Machine Learning, it is common to optimize λ, but for the purposes of this class, it is assumed to be fixed.

Famous Special Cases

This section includes several special cases that deal with risk minimization, such as Ordinary Least Squares, Ridge Regression, Lasso, and Logistic Regression. Table 4.4 provides information on their loss functions, regularizers, as well as solutions.

Loss and Regularizer	**Comments**
Ordinary Least Squares $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2$	∘ Squared Loss ∘ No Regularization ∘ Closed form solution: ∘ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$ ∘ $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ ∘ $\mathbf{y} = [y_1, \dots, y_n]$
Ridge Regression $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2 + \lambda\|w\|_2^2$	∘ Squared Loss ∘ l_2-Regularization ∘ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbb{I})^{-1}\mathbf{X}\mathbf{y}^\top$
Lasso $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|_1$	∘ + sparsity inducing (good for feature selection) ∘ + Convex ∘ - Not strictly convex (no unique solution) ∘ - Not differentiable (at 0) ∘ Solve with (sub)-gradient descent or SVEN
Elastic Net $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2 \; \alpha \in [0, 1)$	∘ ADVANTAGE: Strictly convex (i.e. unique solution) ∘ + sparsity inducing (good for feature selection) ∘ + Dual of squared-loss SVM, see SVEN ∘ DISADVANTAGE: - Non-differentiable
Logistic Regression $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}\log\left(1 + e^{-y_i(\mathbf{w}^\top \mathbf{x}_i + b)}\right)$	∘ Often l_1 or l_2 Regularized ∘ Solve with gradient descent. ∘ $\Pr(y\|x) = \frac{1}{1+e^{-y(\mathbf{w}^\top x + b)}}$
Linear Support Vector Machine $\min_{\mathbf{w},b} C\sum_{i=1}^{n}\max[1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0] + \|\mathbf{w}\|_2^2$	∘ Typically l_2 regularized (sometimes l_1). ∘ Quadratic program. ∘ When kernelized leads to **sparse** solutions. ∘ Kernelized version can be solved very efficiently with specialized algorithms (e.g. SMO)

Table 4.4: Special Cases