Empirical Risk Minimization

Recap

Recall the unconstrained SVM formulation

$$\min_{w} C \sum_{i=1}^{n} \max\left[1 - y_i(\vec{w}^T \vec{x}_i + b), 0\right] + \|w\|_2^2$$

hinge loss $\qquad$ L_2-regularizer

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,

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} \underbrace{l(h_w(\vec{x}_i), y_i)}_{\text{loss}} + \underbrace{\lambda r(\vec{w})}_{\text{regularizer}} \; ; \; \lambda = \frac{1}{C}$$

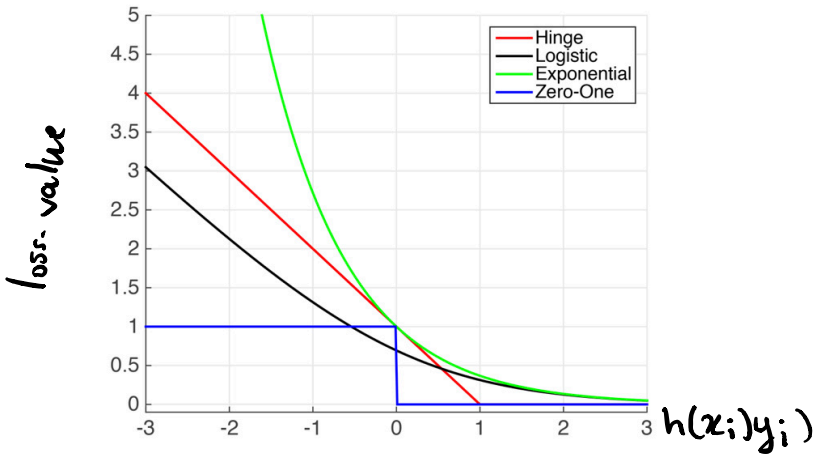
where the loss function is a continuous function which penalizes training error, and a regularizer is a continuous function which penalizes complexity.

Commonly Used Binary Classification Loss Functions

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 $y \in \{-1, +1\}$

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



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 $\lvert h(\mathbf{x}_i) - y_i \rvert$	◦ 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 $\lvert h(\mathbf{x}_i) - y_i \rvert < \delta$, ◦ otherwise $\delta(\lvert h(\mathbf{x}_i) - y_i \rvert - \frac{\delta}{2})$	◦ 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. $y \in \mathbb{R}$

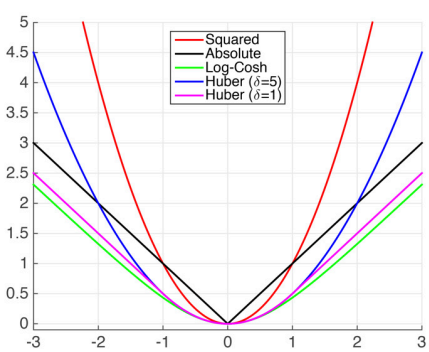


Figure 4.2: Plots of Common Regression Loss Functions - x-axis: $h(\mathbf{x}_i)y_i$, or "error" of prediction; y-axis: loss value

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

$$\min_{w,b} \sum_{i=1}^{n} \ell(h_w(x_i), y_i) + \lambda r(\vec{w}) \iff \min_{w,b} \sum_{i=1}^{n} \ell(h_w(x_i), y_i)$$

Subject to: $r(\vec{w}) \leq B$

$\forall \lambda \geq 0$, $\exists B \geq 0$ such that the two formulations in (4.1) are equivalint, and vice versa.

Besides the l_2-regularizer, other types of useful regularizers are listed below.

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 $\|\mathbf{w}\|_p = (\sum_{i=1}^{d} v_i^p)^{1/p}$	◦ (often $0 < p \leq 1$) ◦ DISADVANTAGE: Non-convex ◦ ADVANTAGE: Very sparse solutions ◦ Initialization dependent ◦ DISADVANTAGE: Not differentiable

Table 4.3: Types of Regularizers

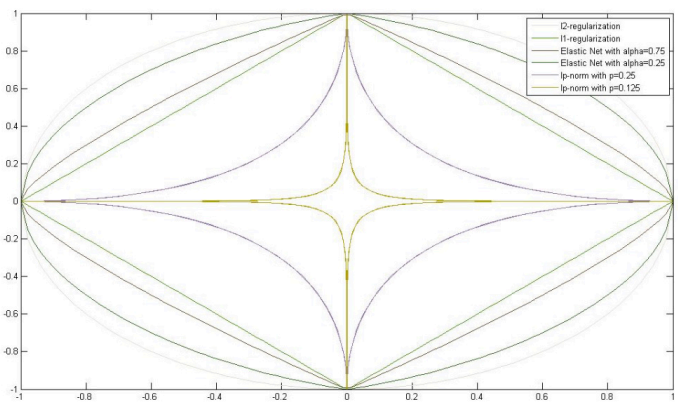


Figure 4.3: Plots of Common Regularizers

Famous Special Cases

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: ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$ ◦ $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ ◦ $\mathbf{y} = [y_1, \ldots, y_n]$
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)

Notes on special cases:

① Ridge regression is very fast if data isn't too high dimensional
↳ 1 line of Julia/Python

② There's an interesting connection between OLS and the first principal component of PCA.

- PCA also minimizes square loss, but looks at perpendicular loss (horizontal distance between each point and the regression line) instead

Note: In Bayesian Machine Learning, it is common to optimize λ, but for this class, it is assumed to be fixed.