

Regularization

Nishanth Nair

15 Nov, 2019

Overview

- If we look at the optimization problem: find a linear separator that is not too complicated. We will see that finding an “optimal” separator is computationally prohibitive, and so will need to “relax” the optimality requirement. This leads us to a convex objective that combines a loss function (how well are we doing on the training data) and a regularizer (how complicated is our learned model).
- This learning framework is known as both **Tikhonov regularization** and **structural risk minimization**.
- In this framework, we get a regularized objective function of the form:

$$J(\theta) = \sum_i L(m_i(\theta)) + \lambda.R(\theta)$$

$$m_i = y_i.f(x_i), \quad y_i \in [-1, 1]$$

$$f(x_i) = \theta^T.x_i$$

- Here, we are trying to optimize a trade-off between a solution that gives low training error (the first term) and a solution that is simple (the second term). Here, R is a form of regularization for hyperplanes. In this formulation, λ becomes a hyperparameter for the optimization.

Convex Surrogate Loss Functions

The **Gold Standard** loss function also called “0-1” loss or L_{01} is implicitly used to evaluate classifiers - i.e count number of mistakes made. This is given by :

$$L_{01}(y) = \begin{cases} 0, & \text{if } y \geq 0 \\ 1, & \text{if } y < 0 \end{cases}$$

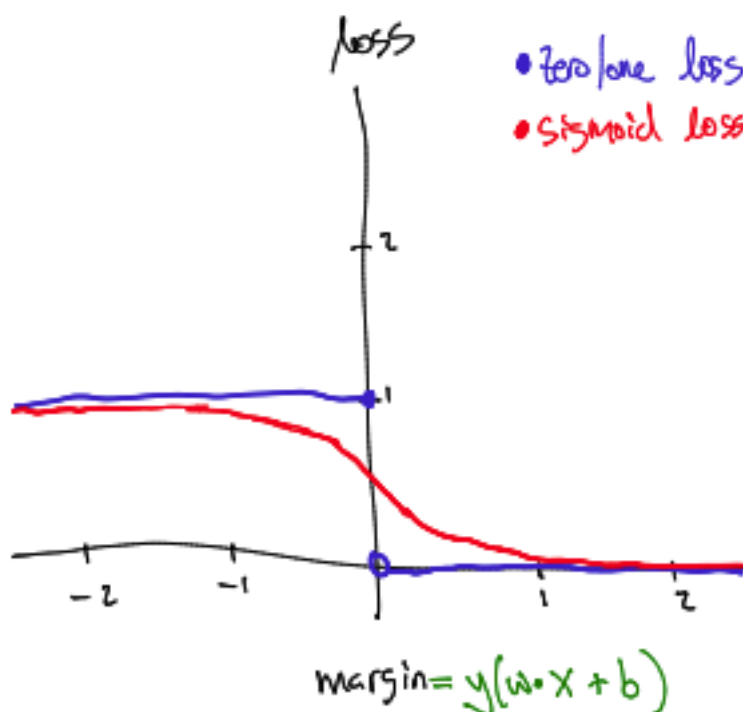


Figure 1: Zero-One Loss and smooth version

Optimizing zero-one loss is hard because small changes to the parameters can have a large impact on the value of the objective function. For instance, if there is a positive training example with $w \cdot x + b = -0.0000001$, then adjusting b upwards by 0.00000011 will decrease the error rate by 1. But adjusting it upwards by 0.00000009 will have no effect. This makes it really difficult to figure out good ways to adjust the parameters. One way to address this problem is to replace the non-smooth zero-one loss with a smooth approximation. For example, we could use an “S” shaped sigmoid function. The benefit of using such an S-function is that it is smooth, and potentially easier to optimize. The difficulty is that it is **not convex**.

Since zero-one loss is hard to optimize, we want to optimize something else, instead. Since convex functions are easy to optimize, we want to approximate zero-one loss with a convex function. This approximating function will be called a **surrogate loss**. The surrogate losses will always be **upper bounds on the true loss function**: this guarantees that if you minimize the surrogate loss, you are also pushing down the real loss.

Loss functions for binary classification

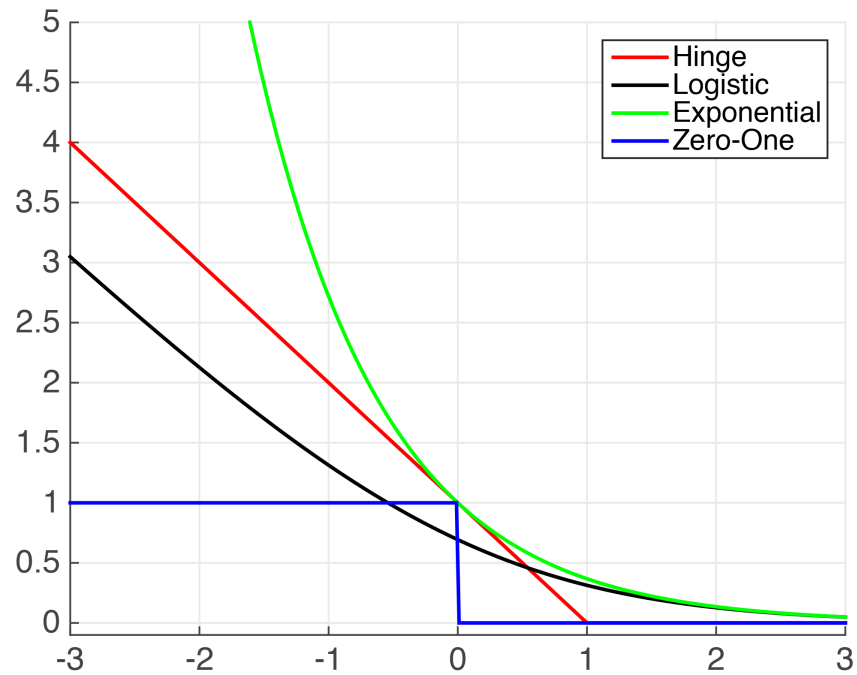


Figure 2: Common Loss functions for classification

Hinge Loss

$$L_{hinge} = \max[0, 1 - y \cdot \hat{y}]^p$$

- Standard SVM, $p=1$
- Squared hingless SVM (differentiable), $p=2$
- 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

$$L_{log} = \log(1 + e^{-y \cdot \hat{y}}) = y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}), \text{ (since } \hat{y} \text{ is sigmoid of } x \text{)}$$

- Used in Logistic regression

Exponential Loss

$$L_{exp} = e^{-y \cdot \hat{y}}$$

- Used in AdaBoost
- This function is very aggressive. The loss of a mis-prediction increases exponentially with the value of $-y.\hat{y}$. This can lead to nice convergence results, for example in the case of Adaboost, but it can also cause problems with noisy data.

Loss functions for Regression

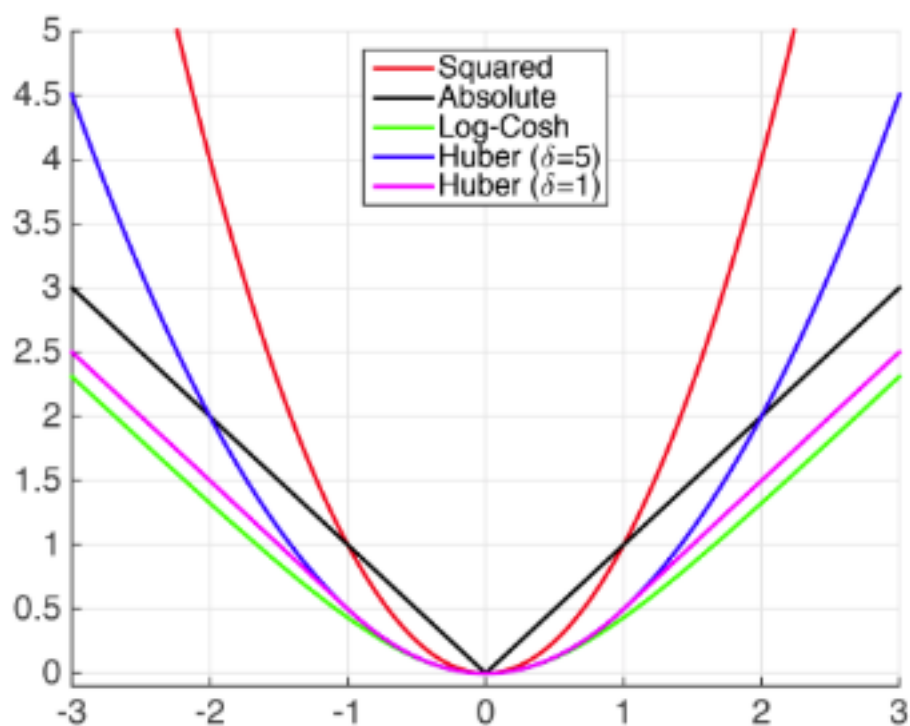


Figure 3: Common Loss functions for Regression

Squared Loss

$$L_{sq} = (y - \hat{y})^2$$

- 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

$$L_{abs} = |y - \hat{y}|$$

- Estimates Median Label
- ADVANTAGE: Less sensitive to noise
- DISADVANTAGE: Not differentiable at 0

Huber Loss

$$L_{huber} = \begin{cases} \frac{1}{2} \cdot (y - \hat{y})^2, & \text{if } |y - \hat{y}| < \delta \\ \delta \cdot (|y - \hat{y}| - \frac{\delta}{2}), & \text{otherwise} \end{cases}$$

- 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.

Regularization

The goal of adding a regularizer is to prevent overfitting. We need to ensure the regularizer is convex so that optimization is possible. Another goal is to ensure the components of the weight vector are small (close to zero). This is a form of **inductive bias**.

- So, why do small values of the weight vector correspond to “simple” functions?
 - One interpretation: Assume, $y = w \cdot x + b$
 - * Suppose that we have an example x with label $+1$.
 - * We might believe that other examples, x' that are near x should also have label $+1$.
 - * For example, if we obtain x' by changing x by some small value ϵ and leave the rest the same, we should assume that the classification would be the same.
 - * If you do this, the difference between y and y' will be exactly $\epsilon \cdot w$.
 - i.e. $x' = x + \epsilon \implies y' = w(x + \epsilon) + b \implies y' = y + \epsilon \cdot w$
 - * So if w is reasonably small, this is unlikely to have much of an effect on the classification decision.
 - * On the other hand, if w is large, this could have a large effect.
 - Second Interpretation: Assume, $y = w_1 \cdot x_1 + \dots + b$
 - * The partial derivative w.r.t $x_1 = \frac{\partial}{\partial w_1}(w_1 \cdot x_1 + \dots + b) = x_1$
 - * From the derivative, we can see that the rate of change of the prediction function is proportional to the individual weights.
 - * So if you want the function to change slowly, you want to ensure that the weights stay small.

Common Regularization functions

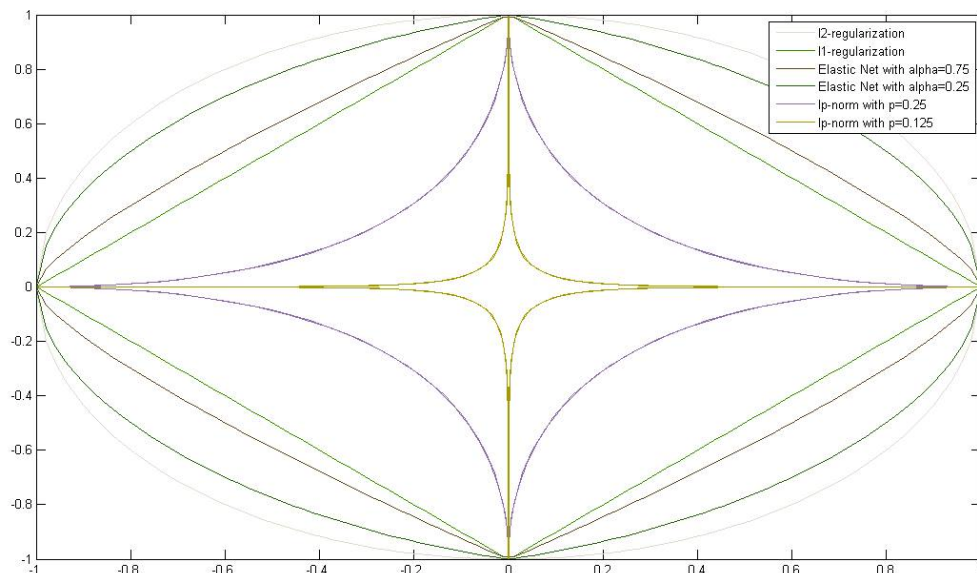


Figure 4: Common Regularization functions

P-Norms, L_p

$$L_p = ||\theta||_p = \left(\sum_i |\theta_i|^p \right)^{\frac{1}{p}}$$

- smaller values of p “prefer” sparser vectors.
- It is for this reason that small p -norms tend to yield weight vectors with many zero entries (aka sparse weight vectors).
- Unfortunately, for $p < 1$ the norm becomes non-convex.
- By changing the value of p , you can interpolate between a square (the so-called “max norm”, when $p \rightarrow \infty$), down to a circle (2-norm), diamond (1-norm) and pointy star shape ($p < 1$).
- **L_1 Regularization** (Lasso):
 - $p=1$
 - Represented as L_1 or $||\theta||_1$
 - Convex (but not strictly)
 - Also called “Absolute” Norm
 - DISADVANTAGE: Not differentiable at 0 (the point which minimization is intended to bring us to)
 - Effect: Sparse (i.e. not Dense) Solutions
 - Used for feature selection, since it drives down weight to zero for irrelevant features.

- **L_2 Regularization:**

- $p=2$
- Represented as L_2 or $||\theta||_2$
- In practice, typically used with square term and ignoring the square root. Thus, also called Euclidean norm
- 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.

Elastic Net

$$\text{Elastic Net} = \alpha \cdot ||\theta||_1 + (1 - \alpha) \cdot ||\theta||_2^2, \quad \text{where } \alpha \in [0, 1]$$

- Elastic Net first emerged as a result of critique on (L_1) , whose variable selection can be too dependent on data and thus unstable.
- The solution is to combine the penalties of (L_1) and (L_2) to get the best of both worlds.
- α is the mixing parameter between L_1 ($\alpha = 1$) and L_2 ($\alpha = 0$)
- ADVANTAGE: Strictly convex (i.e. unique solution)
- DISADVANTAGE: Non-differentiable

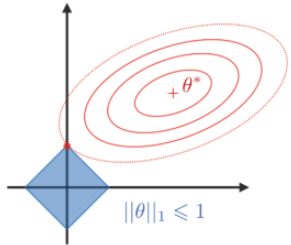
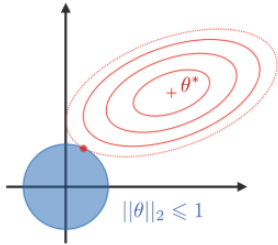
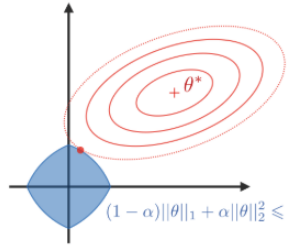
LASSO	Ridge	Elastic Net
<ul style="list-style-type: none"> • Shrinks coefficients to 0 • Good for variable selection 	Makes coefficients smaller	Tradeoff between variable selection and small coefficients
 <p>$\theta _1 \leq 1$</p>	 <p>$\theta _2 \leq 1$</p>	 <p>$(1 - \alpha) \theta _1 + \alpha \theta _2^2 \leq 1$</p>
$\dots + \lambda \theta _1$ $\lambda \in \mathbb{R}$	$\dots + \lambda \theta _2^2$ $\lambda \in \mathbb{R}$	$\dots + \lambda [(1 - \alpha) \theta _1 + \alpha \theta _2^2]$ $\lambda \in \mathbb{R}, \alpha \in [0, 1]$

Figure 5: Summary

Common Optimzation functions:

Ordinary Least Squares

- Model: $y = \theta^T .x$
- Objective Function: Minimize $\frac{1}{n} \sum_{i=1}^n (\theta^T .x_i - y_i)^2$
- Solution Options:
 - Closed Form:

$$\theta = (X.X^T)^{-1} . (X.Y^T)$$

- Gradient Descent update rule:

$$\theta_j = \theta_j + \alpha . \sum_{i=1}^n (y^i - \hat{y}_j^i) . x_j^i$$

- Notes:
 - Squared Loss
 - No regularization

Ridge Regression

- Model: $y = \theta^T .x$
- Objective Function: Minimize $\frac{1}{n} \sum_{i=1}^n (\theta^T .x_i - y_i)^2 + \lambda ||\theta||_2^2$
- Solution Options:
 - Closed Form:

$$\theta = (X.X^T + \lambda . I)^{-1} . (X.Y^T)$$

- Gradient Descent:

$$\theta_j = (1 - 2 . \alpha . \lambda) . \theta_j + 2 . \alpha \sum_{i=1}^n (y^i - \hat{y}_j^i) . x_j^i$$

- Notes:
 - Squared Loss
 - L2 regularization

Lasso Regression

- Model: $y = \theta^T .x$
- Objective Function: Minimize $\frac{1}{n} \sum_{i=1}^n (\theta^T .x_i - y_i)^2 + \lambda ||\theta||_1$
- Solution Options:
 - Closed Form: NA
 - (sub) Gradient Descent or coordinate descent:

- * For each feature, j first compute ρ_j : The difference between actual outcome and the predicted outcome considering all EXCEPT the j th variable.

$$\rho_j = \sum_{i=1}^n (y^i - \hat{y}_j^i + \theta_j \cdot x_j^i) \cdot x_j^i$$

- * Now update θ as:

$$\theta_j = \begin{cases} \rho_j + \lambda, & \text{if } \rho_j < -\lambda \\ 0, & \text{if } -\lambda \leq \rho_j \leq \lambda \\ \rho_j - \lambda, & \text{if } \rho_j > \lambda \end{cases}$$

- Notes:
 - Sparse solution
 - Good for feature selection
 - Convex
 - Not differentiable at 0
 - L1 regularization

Elasticnet Regression

Logistic Regression

- Model: $P(Y_i = 1|x, \theta) = \frac{1}{1+e^{-(\theta^T \cdot x)}}$
- Objective Function: Minimize $\frac{1}{n} \sum_{i=1}^n [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$
- Solution Options:
 - Closed Form: NA
 - Gradient Descent:

$$\theta_j = \theta_j + \alpha \cdot \frac{1}{N} \cdot \sum_{i=1}^N (y^i - \hat{y}_j^i) \cdot x_j^i$$

- Objective Function (L2): Minimize $\frac{1}{n} \sum_{i=1}^n [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)] + \frac{\lambda}{2} \cdot \|\theta\|_2^2$
- Solution Options:
 - Closed Form: NA
 - Gradient Descent:

$$\theta_j = \theta_j + \alpha.[(\sum_{i=1}^N (y^i - \hat{y}_j^i).x_j^i) + \lambda.\theta_j]$$