ERM, Regularization and Model Selection (CS3780/5780, Sp26)

1 Empirical Risk Minimization and Regularization

Regularized Empirical Risk Minimizer: When $\lambda = 0$ just called ERM

$$\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \ell(h_{\mathbf{w}}(x_i), y_i) + \lambda r(\mathbf{w})$$

loss term regularizer

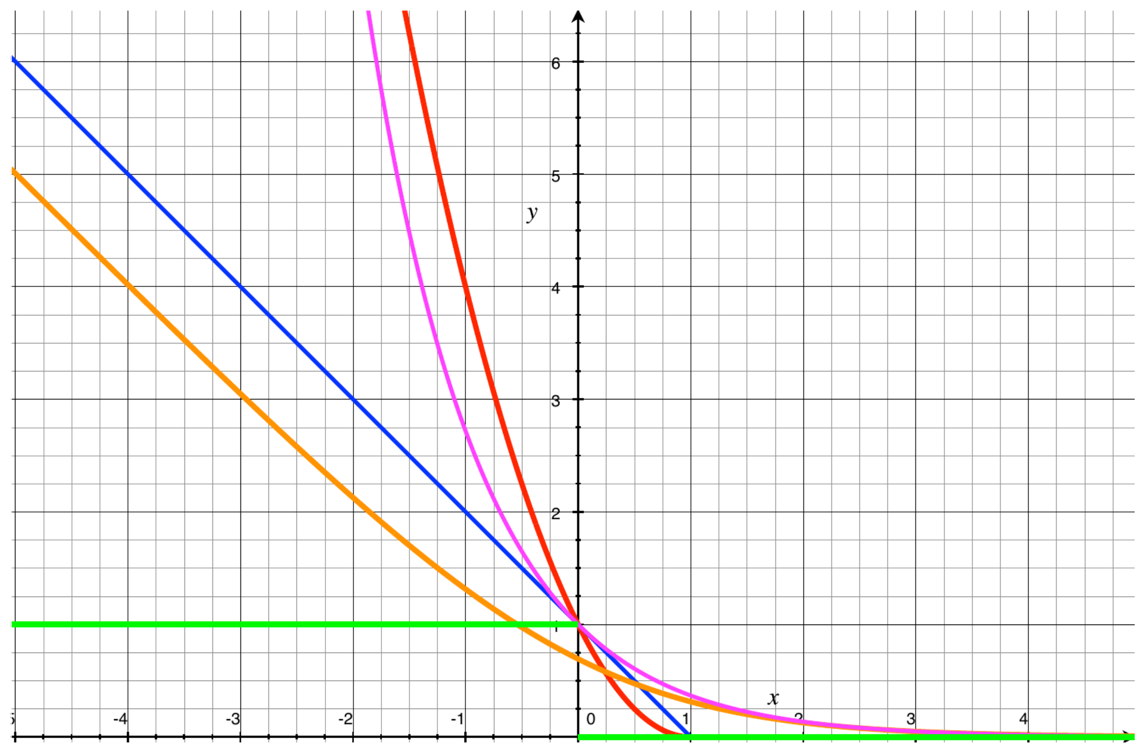
2 Commonly Used Binary Classification Loss Functions

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

Loss $\text{loss}(h_{\mathbf{w}}(\mathbf{x}_i), y_i)$	**Usage**	**Comments**
Hinge-Loss $\max\left[1 - h_{\mathbf{w}}(\mathbf{x}_i)y_i, 0\right]^p$	Standard SVM ($p = 1$), (Differentiable) squared hinge loss 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 $p = 2$.
Logistic $\log\left(1 + e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i}\right)$	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 misprediction 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 $\delta\left(\text{sign}(h_{\mathbf{w}}(\mathbf{x}_i)) \neq y_i\right)$	Actual Classification Loss	Non-continuous and thus impractical to optimize.

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

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 $z \to -\infty$?
3. As $z \to -\infty$, the log-loss and the hinge loss become increasingly parallel.



3 Commonly Used Regression Loss Functions

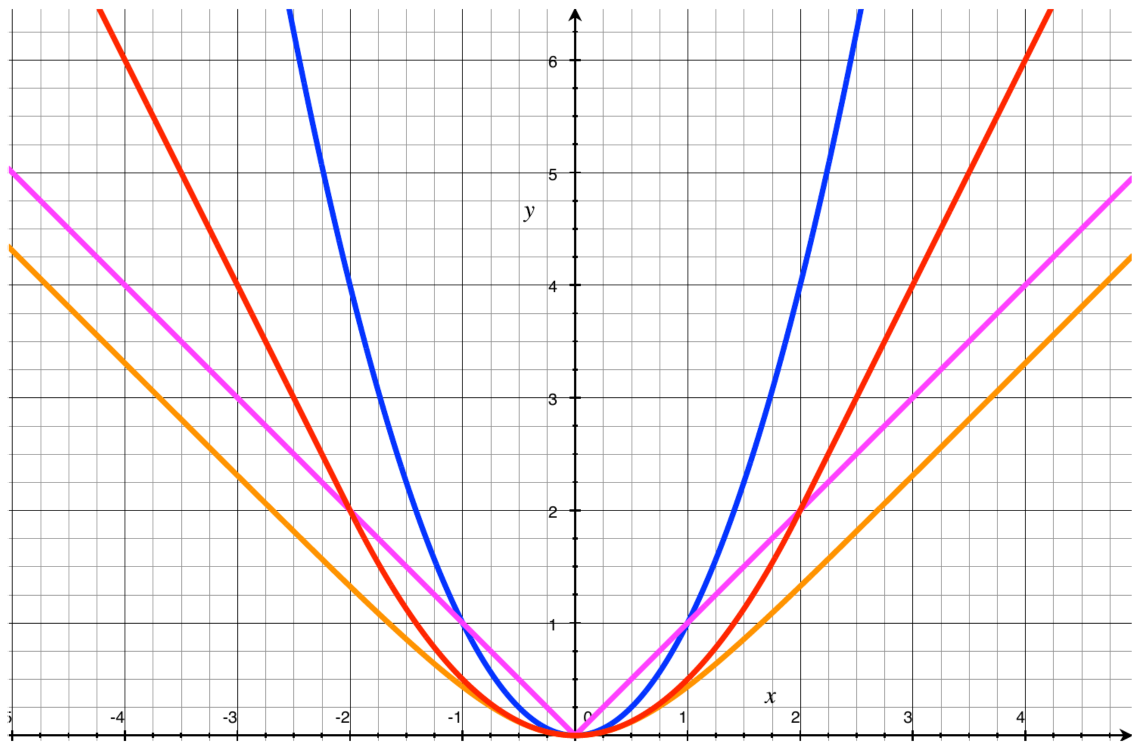
Regression algorithms (where a prediction can lie anywhere on the real-number line) also have their own host of loss functions:

Table 2: Loss Functions With Regression, i.e. $y \in \mathbb{R}$

Loss $\text{loss}(h_{\mathbf{w}}(\mathbf{x}_i), y_i)$	**Comments**
Squared Loss $(h(\mathbf{x}_i) - y_i)^2$	• Most popular regression loss function • Estimates Mean label • Also known as Ordinary Least Squares (OLS) • ✓ Differentiable everywhere • ✗ Somewhat sensitive to outliers/noise
Absolute Loss $\lvert h(\mathbf{x}_i) - y_i \rvert$	• Also a very popular loss function • Estimates Median label • ✓ Less sensitive to noise • ✗ Not differentiable at 0

Loss $\text{loss}(h_{\mathbf{w}}(\mathbf{x}_i), y_i)$	**Comments**
Huber Loss $\frac{1}{2}\big(h(\mathbf{x}_i) - y_i\big)^2$ if $\lvert h(\mathbf{x}_i) - y_i \rvert < \delta$, $\delta\big(\lvert h(\mathbf{x}_i) - y_i\rvert - \frac{\delta}{2}\big)$ otherwise.	• Also known as Smooth Absolute Loss • "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 $\log\big(\cosh(h(\mathbf{x}_i) - y_i)\big)$, $\cosh(x) = \dfrac{e^x + e^{-x}}{2}$	• ✓ Similar to Huber Loss, but twice differentiable everywhere • ✗ More expensive to compute

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



4 Regularizers

When we investigate regularizers it helps to change the formulation of the optimization problem from an unconstrained to a constraint formulation, to obtain a better geometric intuition:

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

For each $\lambda \geq 0$, there exists $B \geq 0$ such that the two formulations above are equivalent, and vice versa. In previous sections, we have already seen the l_2- regularizer in the context of SVMs, Ridge Regression, or Logistic Regression. Besides the l_2-regularizer, other types of useful regularizers and their properties are listed in Table 4.3.

Table 3: Most popular Regularizers

Regularizer $r(\mathbf{w})$	**Properties**
l_2**-Regularization** $r(\mathbf{w}) = \mathbf{w}^\top \mathbf{w} = \lVert\mathbf{w}\rVert_2^2$	• ✓ Strictly convex • ✓ Differentiable • ✗ 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** $r(\mathbf{w}) = \lVert\mathbf{w}\rVert_1$	• Convex (but not strictly) • ✗ Not differentiable at 0 (the point which minimization is intended to bring us to) • Effect: Sparse (i.e. not Dense) solutions
l_p**-Norm** $\lVert\mathbf{w}\rVert_p = \left(\sum_{i=1}^{d} \lvert w_i\rvert^p\right)^{1/p}$	• ✗ Non-convex • ✓ Very sparse solutions (if $0 < p < 1$) • ✗ Not differentiable, initialization dependent

5 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.

Table 4: Special Cases

Loss and Regularizer	Comments
Ordinary Least Squares $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2$	• 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 $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2 + \lambda\|\mathbf{w}\|_2^2$	• Squared Loss • l_2-Regularization • $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y}^\top$
Lasso $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2 + \lambda\|\mathbf{w}\|_1$	• ✓ 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 $\min_{\mathbf{w},b} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2 + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2$ $\alpha \in (0,1)$	• ✓ Strictly convex (i.e. unique solution) • ✓ sparsity inducing (good for feature selection) • ✓ Dual of squared-loss SVM, see SVEN • ✗ 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. • $\Pr(y \mid x) = \frac{1}{1 + e^{-y(\mathbf{w}^\top x + b)}}$
Linear Support Vector Machine $\min_{\mathbf{w},b} C\sum_{i=1}^{n}\max\left[1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0\right] + \|\mathbf{w}\|_2^2$	• Quadratic program. • Kernelized version can be solved efficiently with specialized algorithms (e.g. SMO).

6 Model Selection

When we train a classifier, two failure modes can arise:

Underfitting: the model class is not expressive enough to capture the main patterns in the sample. In this case, both training error and test error are typically large.

Overfitting: the learned classifier becomes too tailored to the peculiarities of the training set. Training error may keep decreasing, but test error eventually increases because the classifier starts relying on patterns that do not generalize.

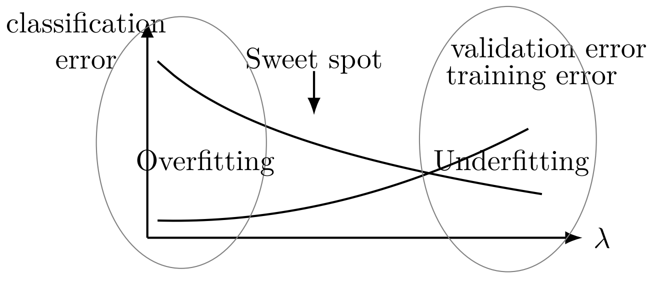


Figure 1: overfitting and underfitting

Identify the Sweet Spot

Split the available sample into a training portion and a validation portion. Fit the algorithm on the training split and evaluate on the validation split for several values of λ (for example, $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, \ldots$).

k-fold cross validation

Partition the training data into k folds. For each round, train on $k-1$ folds and keep one fold out for validation. Repeat this so that every fold serves as the held-out fold exactly once, then average the validation errors across the k runs. This gives a more reliable estimate of validation performance, and one can also examine variability across folds. In the limiting case $k = n$, only a single point is held out each time; this is LOOCV (Leave-One-Out Cross Validation).

LOOCV is especially useful when the data set is small and we cannot afford large validation block.

Telescopic search

Use a coarse-to-fine strategy. First, search across orders of magnitude for λ; then zoom in around the best region found so far. For example, one may first try $\lambda \in \{0.01, 0.1, 1, 10, 100\}$, and suppose 10 performs best. Then refine locally with a denser search such a $\lambda \in \{5, 10, 15, 20, 25, \ldots, 95\}$.

Grid and Random search

When there are multiple hyper-parameters—for example λ together with a kernel width σ—one simple approach is *grid search*: choose a discrete set of candidate values for each hyper-parameter and evaluate every combination. The cost of this method grows exponentially with the number of hyper-parameters. Moreover, if the model is relatively insensitive to one coordinate, a substantial amount of computation may be spent varying that coordinate without much benefit.

A common alternative is *random search*, where candidate settings are sampled randomly from specified ranges instead of being placed on a rigid grid. This can be more efficient when performance depends strongly on only some of the hyper-parameters, because it explores many more distinct values along each important coordinate.

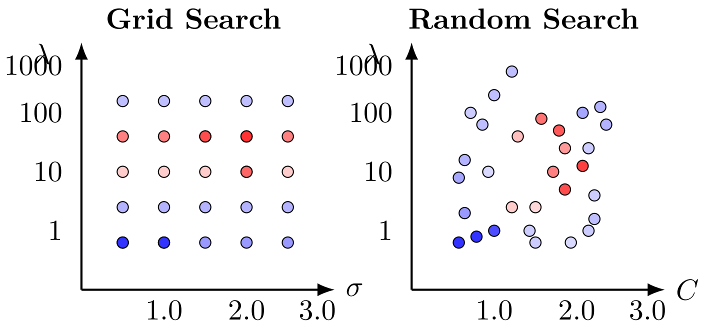
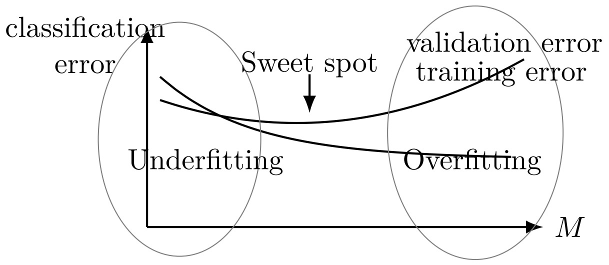


Figure 2: Grid Search vs Random search. Red marks lower loss and blue marks higher loss; random search explores more distinct values along each coordinate.

Early Stopping

Stop the optimizer after M (with $M \geq 0$) gradient steps, (even if the not fully converged).


classification error
Sweet spot
validation error
training error
Underfitting
Overfitting
M