UNIVERSITY AT ALBANY
STATE UNIVERSITY OF NEW YORK

CSI 436/536 (Spring 2025)

Machine Learning

Lecture 8: Loss and Gradient Descent

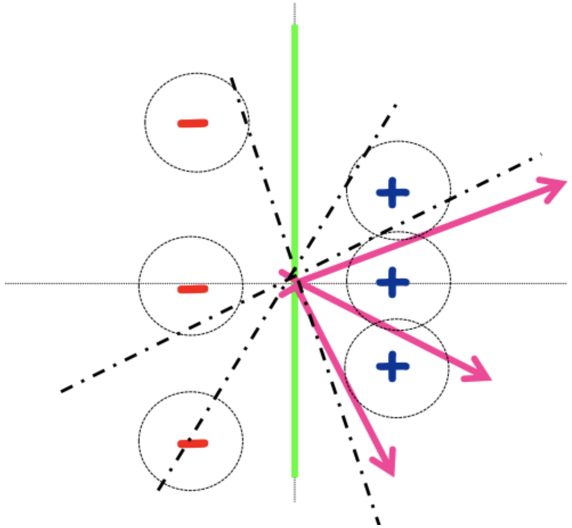
Chong Liu

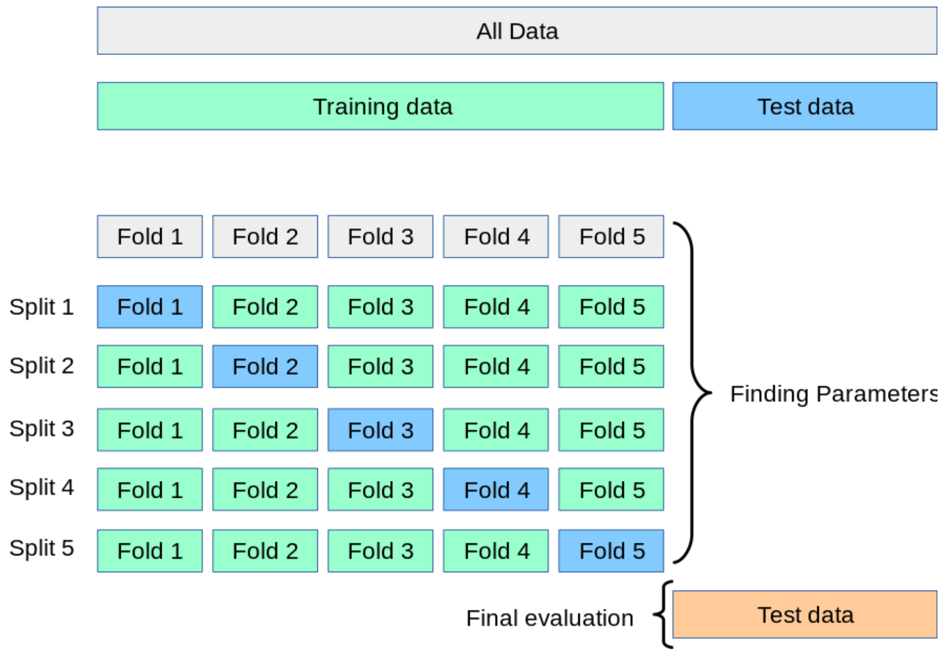
Department of Computer Science

Feb 19, 2025

Recap: linear classifier

- Problem of overfitting
    - Too dependent on training data
    - Bad on test data
- Data splitting methods:
    - Holdout
    - Cross validation
- Perceptron algorithm





Today

- Learn how to train a machine learning classifier!
- Surrogate loss
- Continuous optimization
- Gradient Descent (GD)

Recap: Linear classifier

- Take input feature vector
  - $\text{Score}(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$
  - $x_1 = 1(\text{has hyperlinks})$
  - $x_2 = 1(\text{on contact list})$
  - $x_3 = \text{proportion of misspelling}$
  - $x_4 = \text{length}$
- Let label space be $\{-1,1\}$
- Linear classifier:
  - $h_w(x) = \begin{cases} 1, \text{if Score}(x) \geq 0 \\ -1, \text{if Score}(x) < 0 \end{cases}$

Key question: How to train linear classifier (find w)?

$$\min_{w \in \mathbb{R}^d} \text{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(h_w(x_i) \neq y_i)$$

Discussion:

- 0-1 loss:

$$\mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i)$$

- Training problem:

$$\min_{w \in \mathbb{R}^d} \mathrm{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i)$$

- How can you minimize 0-1 loss?

0-1 loss is unfortunately very hard to optimize

- 0-1 loss:

$$\mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i)$$

- Training problem:

$$\min_{w \in \mathbb{R}^d} \mathrm{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i)$$

- Given n data points, the learner needs to check 2^n different configurations.
  - Why 2? Prediction matches / doesn't match label y.
  - It is known as NP-hard.
  - Highly inefficient when n is large.

Just “relax”: relaxing a hard problem into an easier one

$$\min_{w \in \mathbb{R}^d} \mathrm{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i)$$

$$\Downarrow$$

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(w^T x_i, y_i).$$

New loss function is called "surrogate loss"

$$\min_{w\in\mathbb{R}^d} \mathrm{Error}(w) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i)$$

$$\Downarrow$$

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(w^T x_i, y_i).$$

Key point: Choice of surrogate loss must satisfy

$$\mathbf{1}(\mathrm{sign}(w^T x_i) \neq y_i) \leq \ell(w^T x_i, y_i)$$

- Discussion: why?

Loss functions

- 0-1 loss:

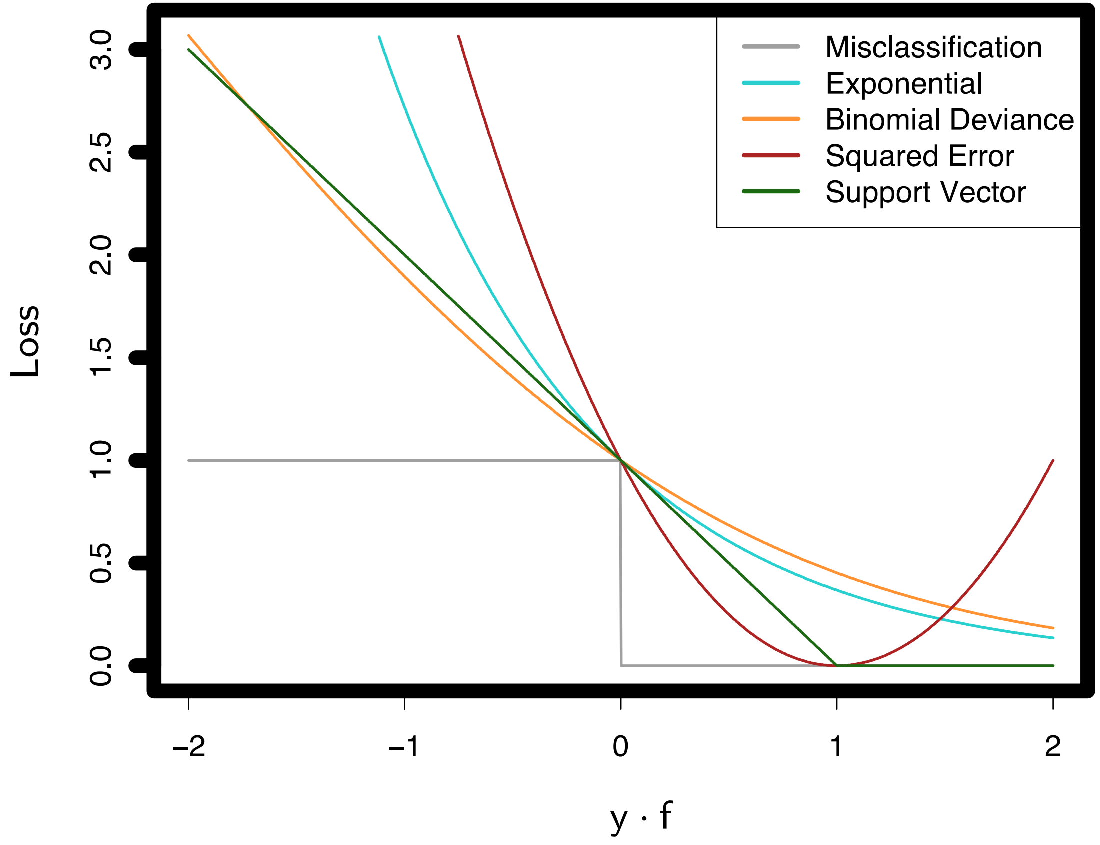
$$\mathbf{1}(h_w(x) \neq y) = \mathbf{1}(\text{sign}(S_w(x)) \neq y)$$

- Surrogate losses:
  - Logistic loss (binomial deviance):

$$\log_2(1 + \exp(-y \cdot S_w(x)))$$

  - Hinge loss (support vector):

$$\max(0, 1 - y \cdot S_w(x))$$



(Figure from Section 10.4 of "Elements of Statistical Learning")

In-class exercise: Intuition of the logistic loss

$$\log_2(1 + \exp(-y \cdot S_w(x)))$$

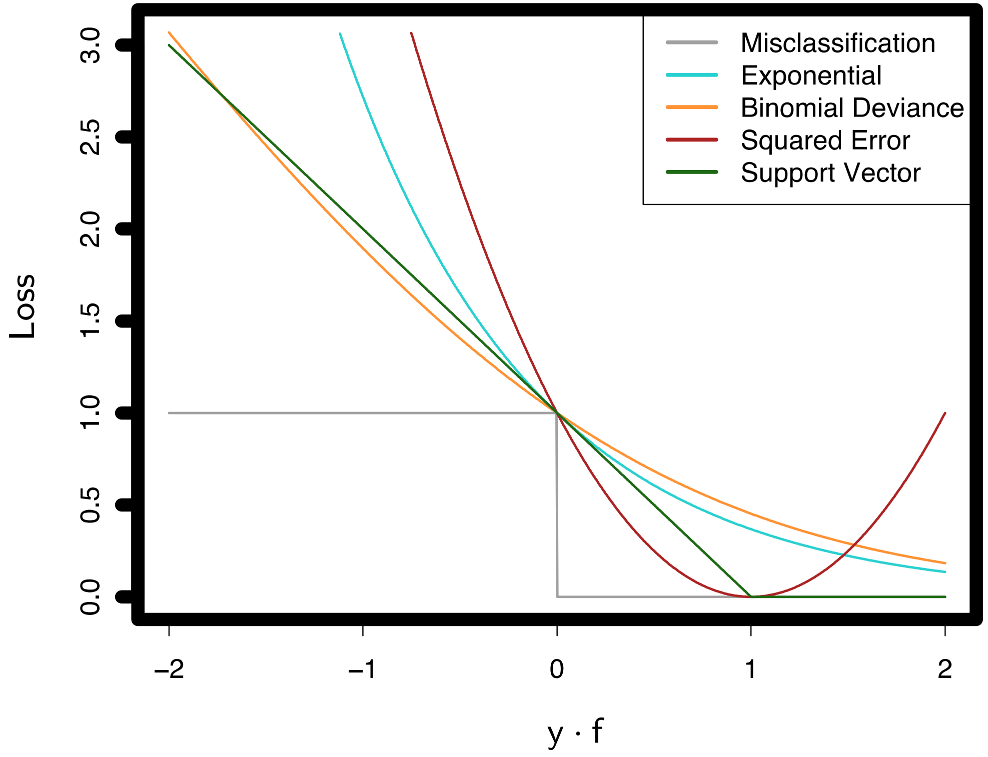
Try plotting the logistic loss as a function of $y \cdot S_w(x)$

1. What happens when the classifier predicts correctly?

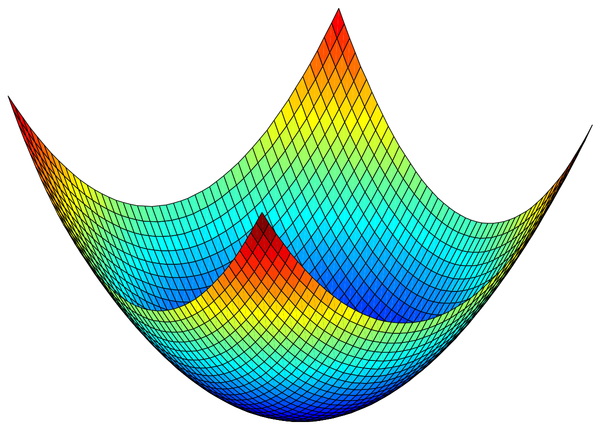
2. What happens when the classifier predicts incorrectly?

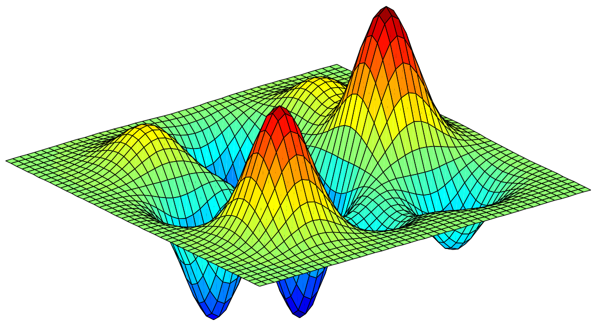
Which surrogate loss is easier to minimize?

- Continuous
- Differentiable
  - Except hinge loss, i.e., loss used in "support vector machine (SVM)"
- Convex



Convex vs Nonconvex optimization





- Unique optimum: global/ local.

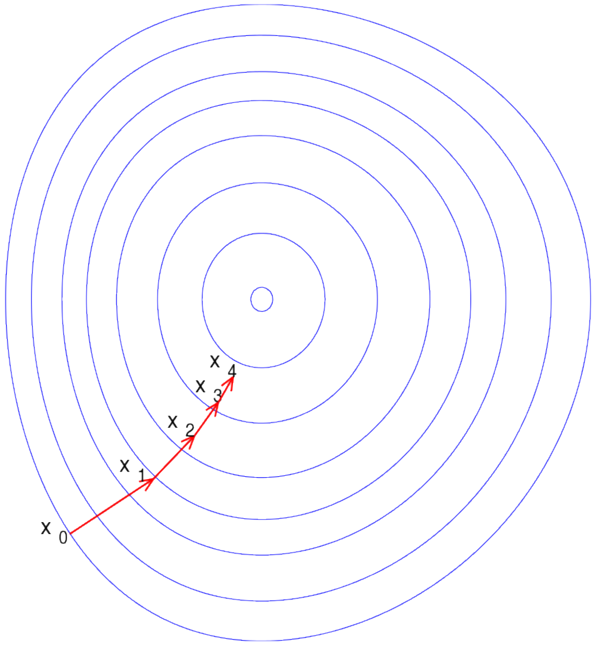
- Multiple local optima
- In high dimensions possibly exponential local optima

* Be careful: The surrogate loss being convex does not imply all ML problems using surrogate losses are convex. Linear classifiers are, but non-linear classifiers are usually not.

How do we optimize a continuously differentiable function in general?

- The problem: $$\min_{\theta} f(\theta)$$
- Gradient descent in iterations

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$



In-class exercise: gradient descent

- $\min f(x) = x^2$

1. Find x_4 given $x_0 = 2, \eta = 0.1$
2. Find x_4 given $x_0 = 2, \eta = 0.4$
3. Find x_4 given $x_0 = 4, \eta = 0.4$
4. Find x_4 given $x_0 = 2, \eta = 1.5$

Gradient Descent Demo in 2-D

- An excellent demo tool:
  - https://github.com/lilipads/gradient_descent_viz