Loss function + SGD

Razvan Marinescu

What is a loss?

- Recall least square

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{3}(\mathbf{x}^{(i)} \cdot \theta - y^{(i)})^2 = \frac{1}{2}\sum_{i=1}^{3}({\mathbf{x}^{(i)}}^\top \theta - y^{(i)})^2$$

- Logistic regression

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{m} y^{(i)} \log(g(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - g(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})$$

Linear regression – review

- Another way of writing this

$$L(\theta) := \sum_{i \in \text{ data}} \ell(\theta; \mathbf{x}^{(i)}, y^{(i)})$$

- Empirical risk minimization

- How do we optimize it?

Loss functions

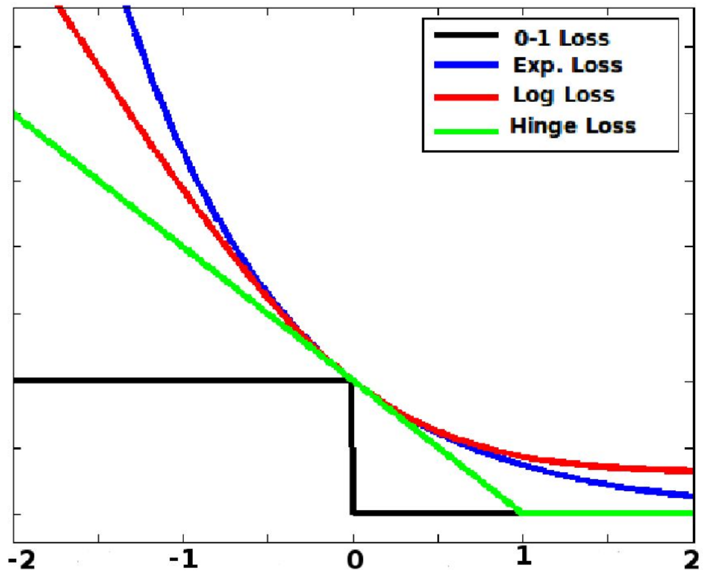
a = model prediction
y = label from training/test data

0/1: $l^{(0/1)}(y,a) = 1[ya \leq 0]$

Hinge: $l^{(hin)}(y,a) = max\{0, 1 - ya\}$

Logistic: $l^{(log)}(y,a) = \frac{1}{log2} log(1 + exp[-ya])$

Exponential: $l^{(exp)}(y,a) = exp[-ya]$

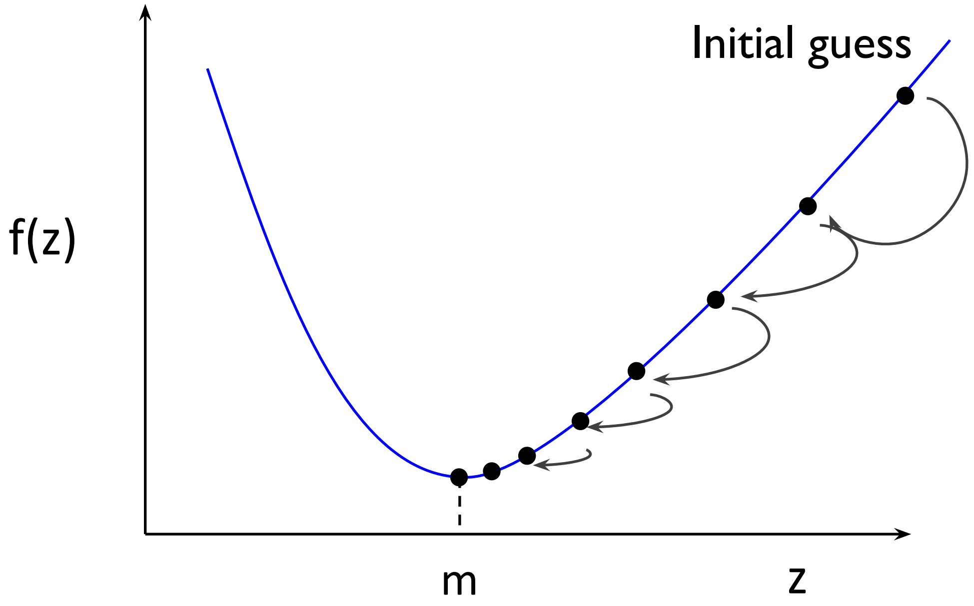


Different loss function can result in different classifiers

These are all convex and so can be minimized using algorithms like Gradient Descent

Refresher: Gradient Descent

- Used for finding the minima of an optimization function
- Idea: Start at a random initial guess on the function and iteratively move towards its minima



How do we know in which direction is the minima?

Direction of negative gradient

***Refresher: gradient of a function points in the direction of the greatest rate of increase of the function, and its magnitude is the slope of the graph in that direction.

Stochastic gradient descent

a single data-point pair

- Randomly order data $\mathbf{x}^{(i)}, y^{(i)}$;
- Compute $\partial_\theta \ell(\theta_i; \mathbf{x}^{(i)}, y^{(i)})$;
- Update $\theta_{i+1} := \theta_i - \eta \cdot \partial_\theta \ell(\theta_i; \mathbf{x}^{(i)}, y^{(i)})$.

Repeat until convergence

- Consider SGD for example $\mathbf{x}^{(1)} = (1,2)$, $y^{(1)} = 5$, and θ initially $(1,1)$.
- The squared-error on this example is $(1 * \theta_0 + 2 * \theta_1 - 5)^2 = 4$, and its contribution to $J(\theta)$ is 2 (half the squared error).

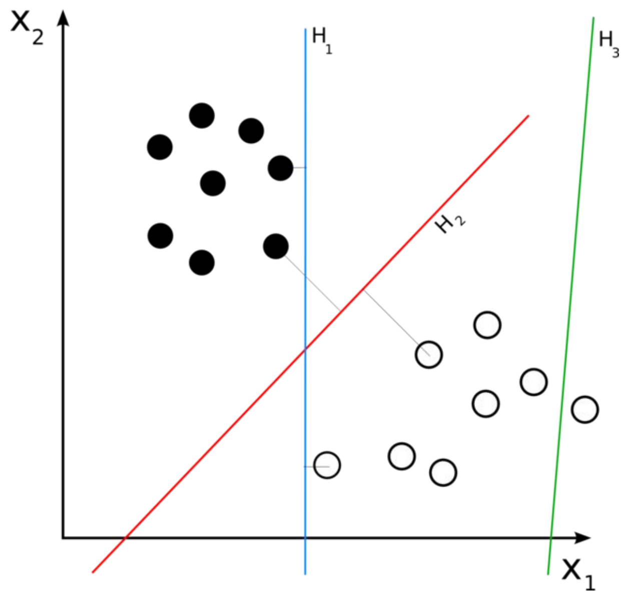
  At $\theta = (1,1)$:

$$\frac{\partial \text{contribution}}{\partial \theta_0} = \frac{2}{2}(1 * \theta_0 + 2 * \theta_1 - 5) * 1 = -2$$
$$\frac{\partial \text{contribution}}{\partial \theta_1} = \frac{2}{2}(1 * \theta_0 + 2 * \theta_1 - 5) * 2 = -4$$

- with step size $\frac{1}{20}$, update θ to $\theta - \frac{1}{20}\nabla_\theta(\text{contribution})$.
- New $\theta = (1,1) - (\frac{-2}{20}, \frac{-4}{20}) = (1.1, 1.2)$

One ML model: Linear classification

- Outputs a classification (one of N possible classes) based on the value of a linear combination of the characteristics (features)



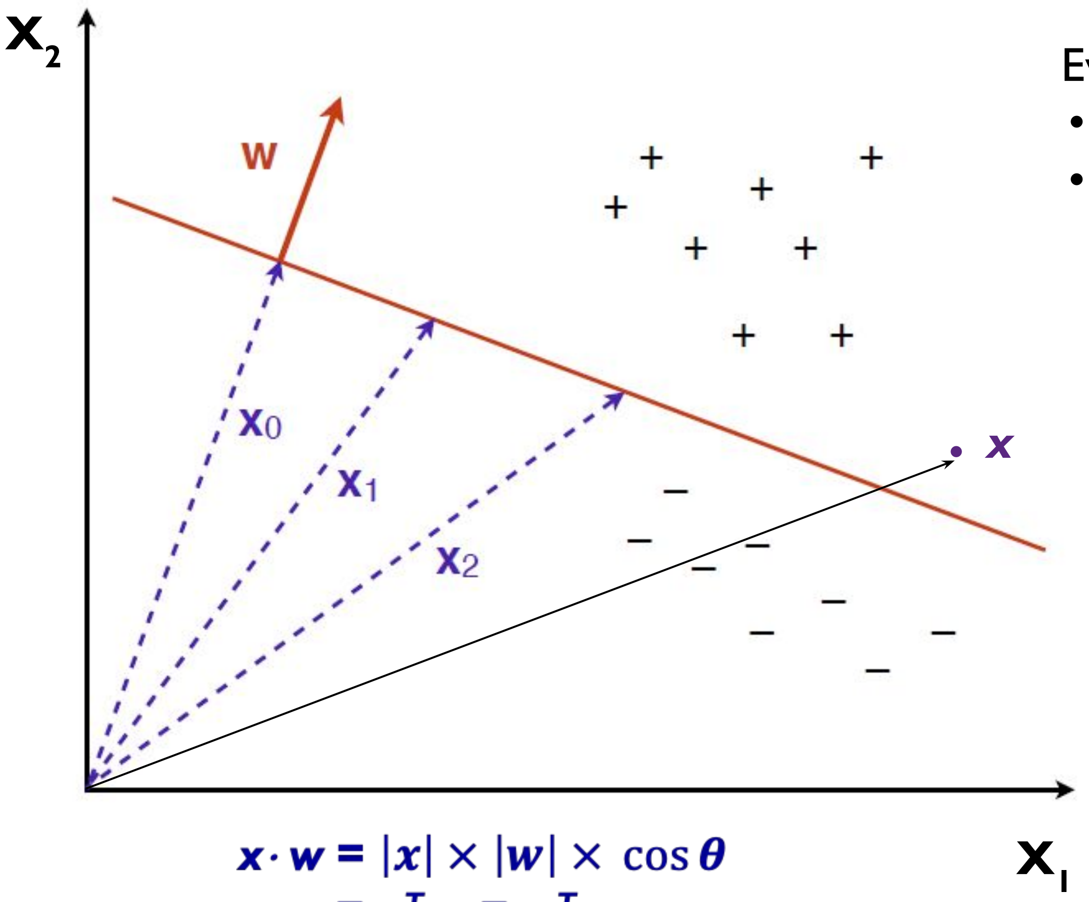
Example with 2 features – 2D feature vector (x_1, x_2):

- Linear classifiers H_1 and H_2 successfully partition the two classes of dots
- H3 does not

Which is better, H_1 or H_2? Why?

Goal: Find a linear decision boundary

Linear classification



$x \cdot w = |x| \times |w| \times \cos\theta$

$x \cdot w = x^T w = w^T x$

$x_0 \cdot w = x_1 \cdot w = x_2 \cdot w = b$

How to determine if a feature vector ***x*** is on the **+** or **–** side of the line?

Evaluate the dot product of ***x*** and ***w***:

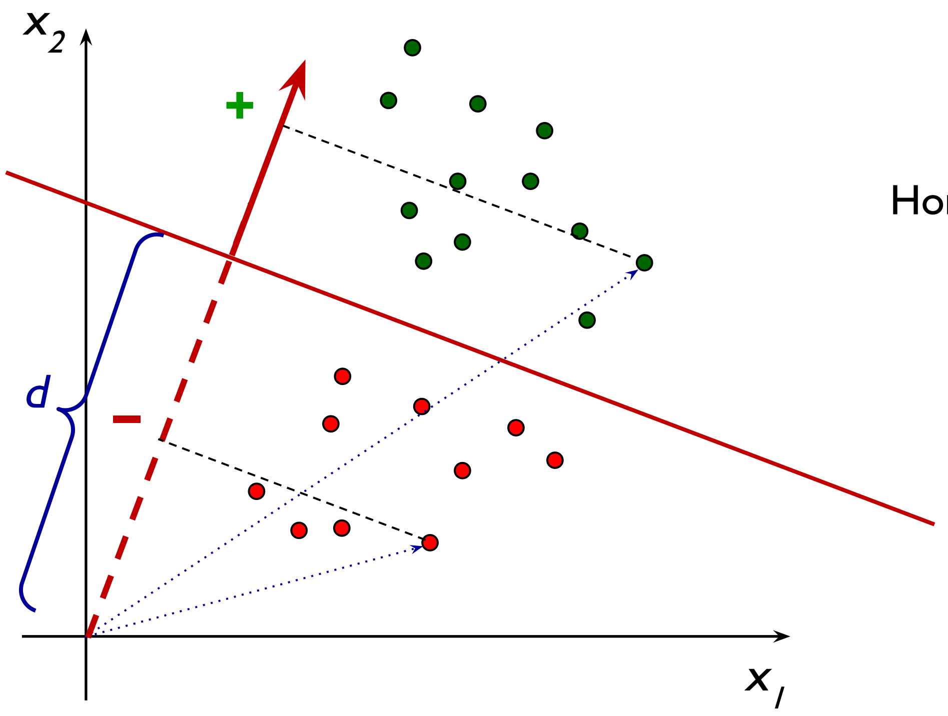
- If $x \cdot w > b$, then **+**
- Otherwise **–**

2 features means 2D classification and a 1D classification boundary

N features means N dimensional classification and an N-1 dimensional classification boundary

Dimensions	Linear boundary
1	Point
2	Line
3	Plane
>3	Hyperplane

Classifier geometry – $\boldsymbol{w}$ and b



Non-homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} - b = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - b > 0$$

Homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} = \begin{bmatrix} w_1 & w_2 & -b \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} > 0$$

Is $\boldsymbol{w}$ a unit vector?

Doesn't have to be

What's the relationship between $\boldsymbol{w}$ and b ?

$(w, b) \equiv (kw, kb)$

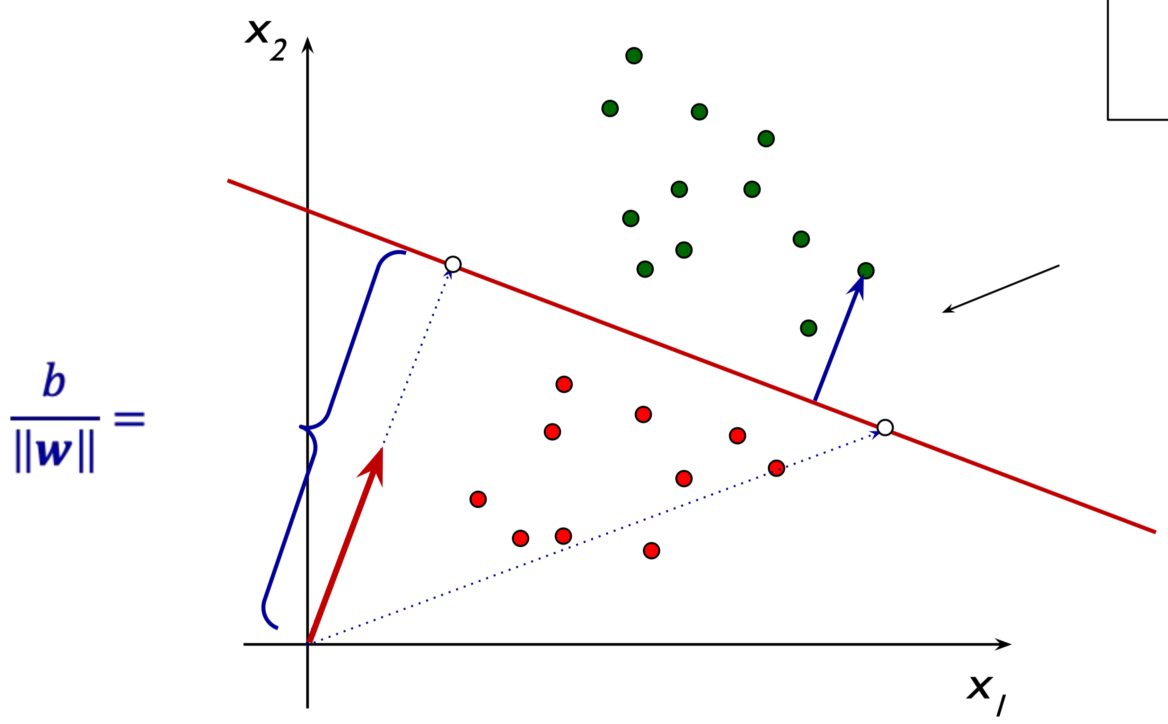
$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - b = 0 \qquad \begin{bmatrix} 2w_1 & 2w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 2b = 0$$

These describe the same line

Classifier geometry – $\boldsymbol{w}$ and b

Non-homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} - b = [w_1 \quad w_2]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - b > 0$$



Margin: $z_i = \frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b)}{\|\boldsymbol{w}\|}$

$\boldsymbol{w}^T\boldsymbol{x} = b$

Note:
Regular classifier: the margin is the distance from the decision boundary
Scoring classifier: the margin is the score

In both cases, value is positive for correctly classified, negative for incorrectly classified

Minimum Error Hyperplane

- Error of a linear model ($\mathbf{w}$, b) for an instance ($\mathbf{x_n}$, y_n)

$$\mathbf{1}[y_n(\mathbf{w}.\mathbf{x_n} + b) \leq 0]$$

Indicator function: 1 if stuff inside is true (incorrect prediction) and 0 otherwise (correct prediction)

- Objective function to minimize to find the minimum error hyperplane:

$$\min_{\mathbf{w},b} \sum_n \mathbf{1}[y_n(\mathbf{w}.\mathbf{x_n} + b) \leq 0]$$

Minimum Error Hyperplane

- Error of a linear model ($\mathbf{w}$, b) for an instance ($\mathbf{x_n}$, y_n)

$$\mathbf{1}[y_n(\mathbf{w}.\mathbf{x_n} + b) \leq 0]$$

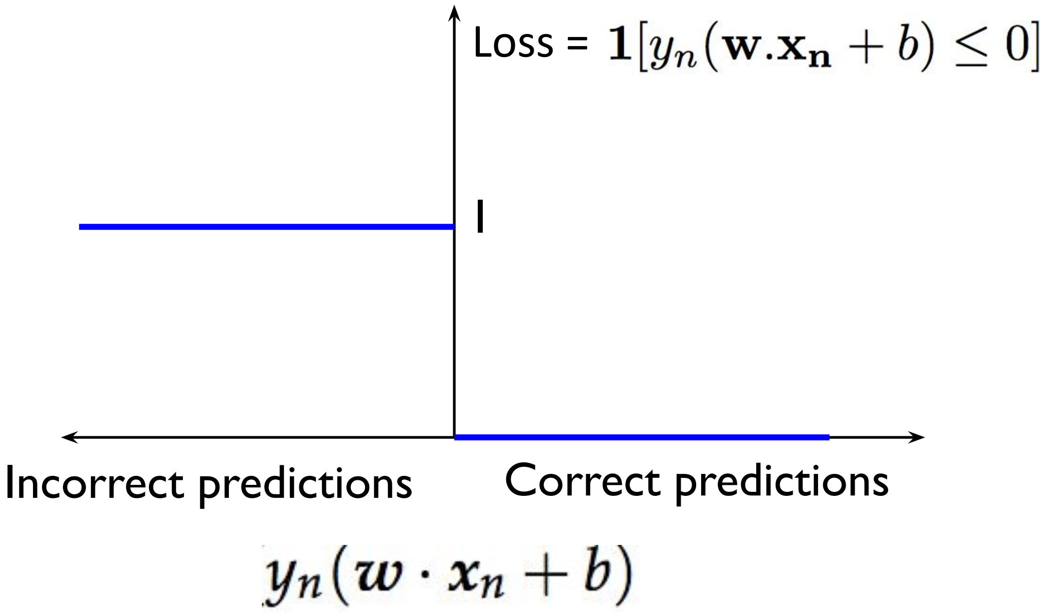
Indicator function: 1 if stuff inside is true (incorrect prediction) and 0 otherwise (correct prediction)

- Objective function to minimize to find the minimum error hyperplane:

$$\min_{\mathbf{w},b} \sum_n \mathbf{1}[y_n(\mathbf{w}.\mathbf{x_n} + b) \leq 0]$$

This optimization formulation is minimizing the 0/1 loss

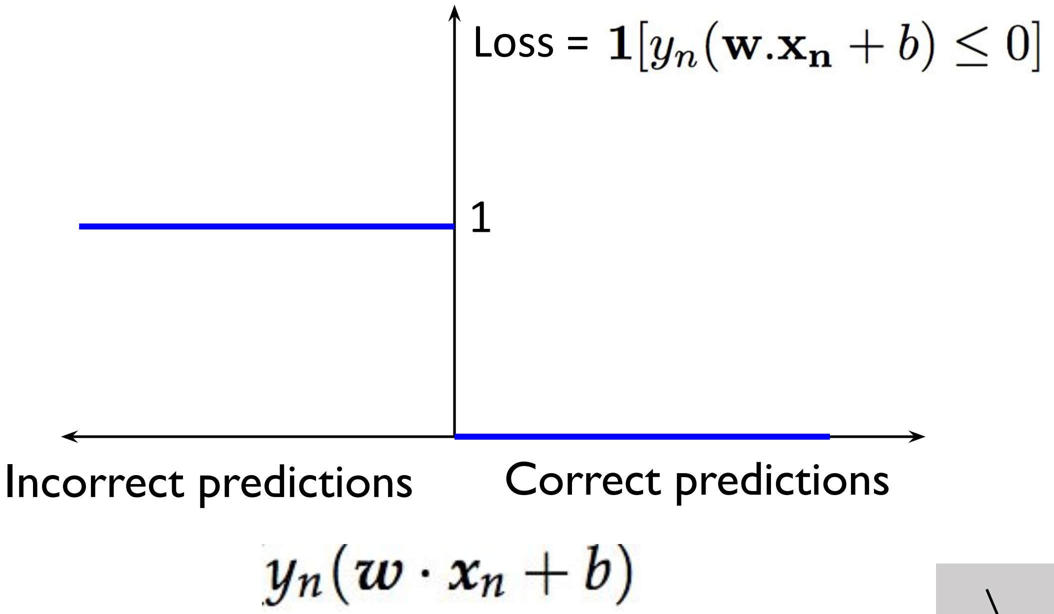
0/1 Loss

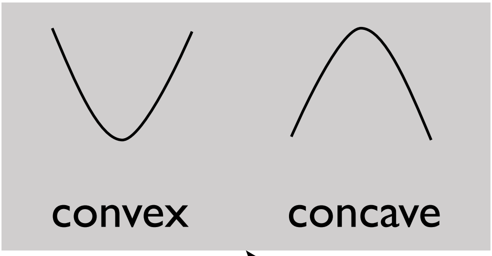


Minimizing 0-1 loss is difficult!

Why?

0/1 Loss





Minimizing 0-1 loss is difficult!

Why?

Not smooth, and not convex

Alternatives to 0/1 Loss

- Need to find an upper-bound to 0/1 loss that is convex

- Why do we require (1) convexity, and (2) upper-boundedness?

- (1) So that minimization is easy (we know how to minimize a convex function)
- (2) So that minimizing the upper bound also *pushes down* the real objective

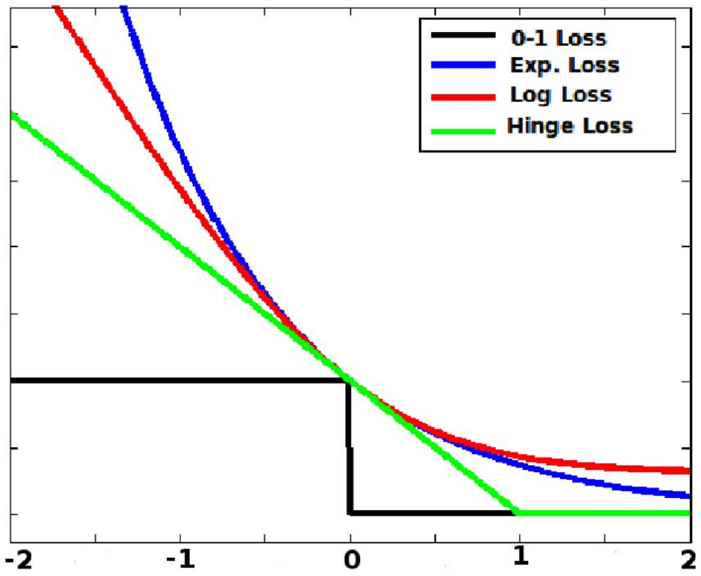
Convex upper-bounds of 0/1 Loss

0/1: $l^{(0/1)}(y,a) = 1[ya \leq 0]$

Hinge: $l^{(hin)}(y,a) = max\{0, 1-ya\}$

Logistic: $l^{(log)}(y,a) = \frac{1}{log2} log(1 + exp[-ya])$

Exponential: $l^{(exp)}(y,a) = exp[-ya]$



Different loss function can result in different classifiers

These are all convex and so can be minimized using algorithms like Gradient Descent

Why does it work?

- Law of large number

$$\frac{1}{|\text{data}|} \sum_{i \in \text{ data}} \ell(\theta; \mathbf{x}^{(i)}, y^{(i)}) \Rightarrow \mathbb{E}_{\text{distribution}}[\ell(\theta; \mathbf{x}, y)]$$

- Empirical risk minimization approximates the true optimal model

- Is the loss ℓ proper?

Classification calibration

- We say a loss function ℓ if there existsa non-decreasing function Φ such that for all f

$$\Phi\left(R_\ell(f) - \min_{f'} R_\ell(f')\right) \leq R(f) - \min_{f'} R(f')$$

where in above

$$R_\ell(f) := \mathbb{E}[\ell(f; X, Y)], \ R(f) := \mathbb{E}[1(f(X) \neq Y)]$$