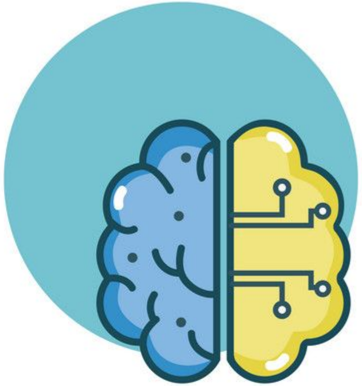
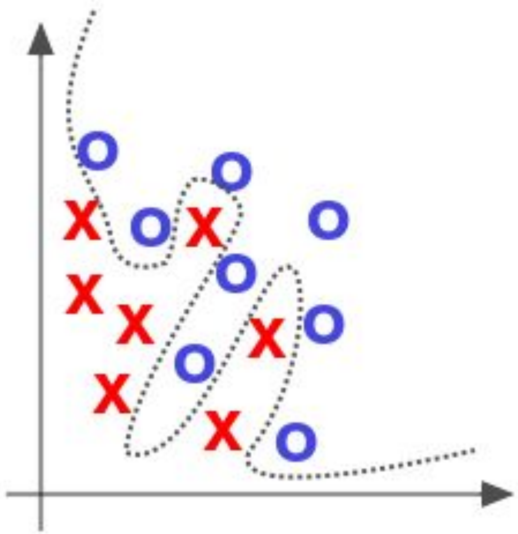
# INTRODUCTION TO MACHINE LEARNING

## REGULARIZATION

Elisa Ricci






# Avoid Overfitting

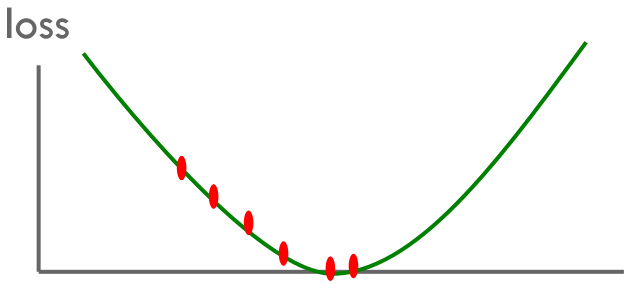
# One concern

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b))$$

We are calculating this on the **training set**

We still need to be careful about **overfitting**!

The $\min_{w,b} Loss$ on the training set is generally NOT the min for the test set




How did we deal with this?

# REGULARIZATION

- A regularizer is an additional criterion to the loss function to make sure that we do not overfit
- It is called a regularizer since it tries to keep the parameters more normal/regular
- It is a bias on the model that forces the learning to prefer certain types of weights over others

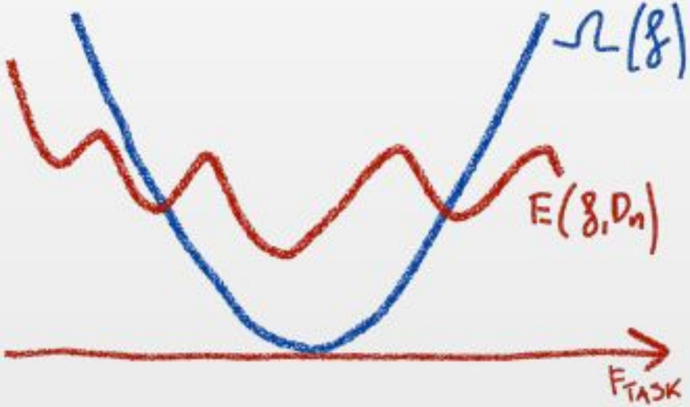
$$\text{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda \ regularizer(w, b)$$

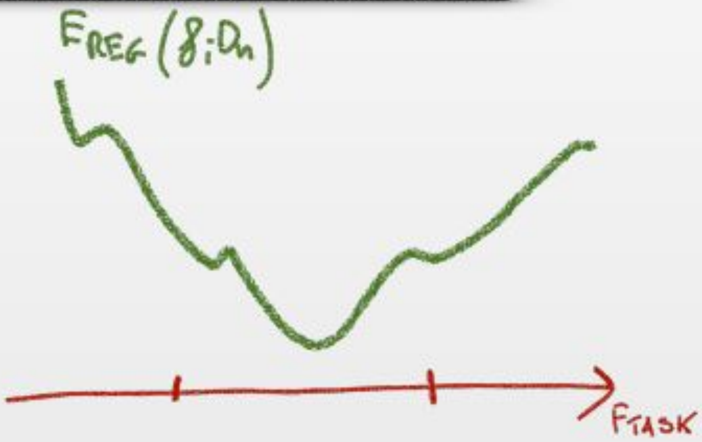
# So Far...

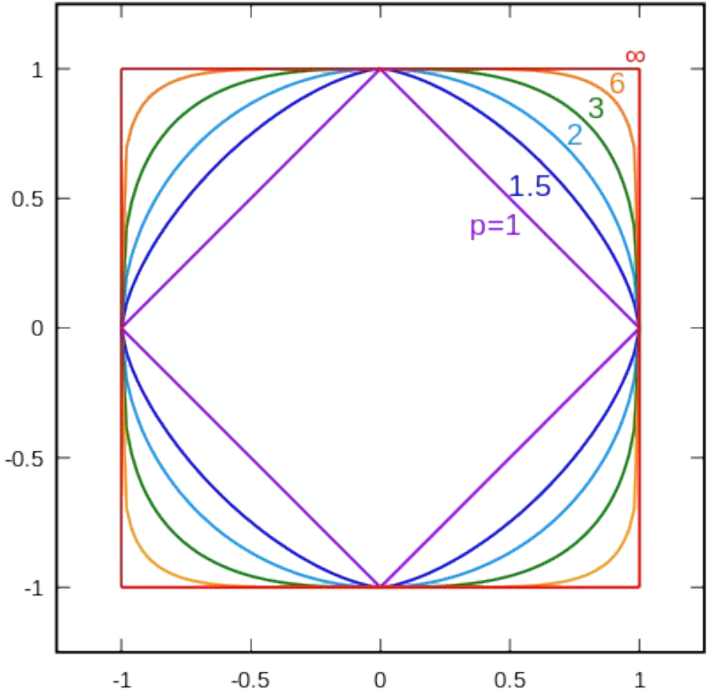
Modification of the training error function with a term $\Omega(f)$ that typically penalizes complex solutions

TRADE-OFF PARAMETER

$$\boxed{E_{\text{reg}}(f;\mathscr{D}_n) = E(f;\mathscr{D}_n) + \lambda_n \Omega(f)}$$






# Regularizers

# Regularizers

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

- Generally, we do not want huge weights: if weights are large, a small change in a feature can result in a large change in the prediction
- Might also prefer weights of 0 for features that are not useful

# Common regularizers

sum of the weights

$$r(w,b) = \sum \left| w_j \right|$$

sum of the squared weights

$$r(w,b) = \sqrt{\sum \left| w_j \right|^2}$$

Squared weights penalizes large values more

Sum of weights will penalize small values more

# P-NORM

sum of the weights (1-norm) $$r(w,b) = \sum |w_j|$$

sum of the squared weights (2-norm) $$r(w,b) = \sqrt{\sum |w_j|^2}$$

---

p-norm = family of norms

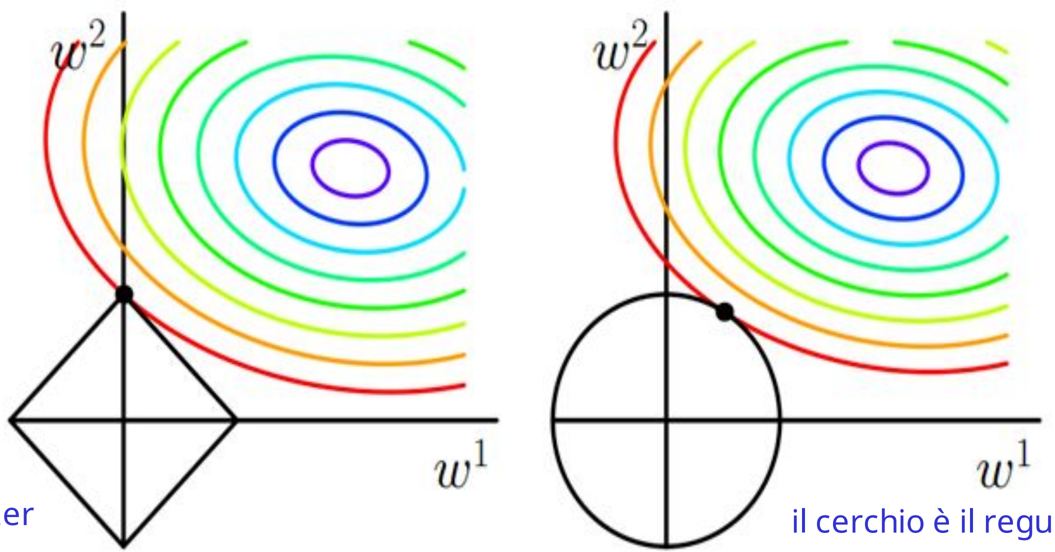
p-norm $$r(w,b) = \sqrt[p]{\sum |w_j|^p} = \|w\|^p$$

Smaller values of p (p < 2) encourage sparser vectors

Larger values of p discourage large weights more

# L1/L2-norms visualized

invece di trovare una soluzione al centro come prima ora la trovo nell'intersezione tra la loss function e il regularizer




il quadratro è il regularizer

il cerchio è il regularizer

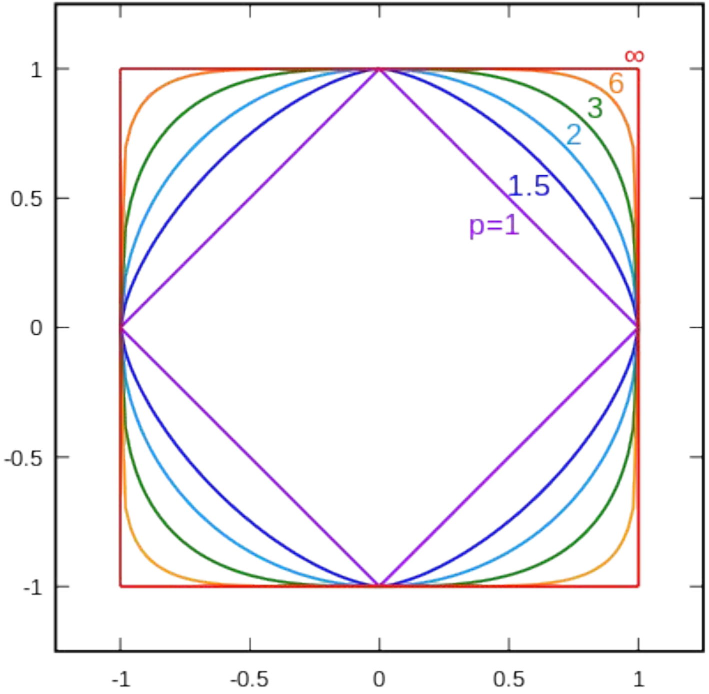
(a) $\ell_1$-ball meets quadratic function. $\ell_1$-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) $\ell_2$-ball meets quadratic function. $\ell_2$-ball has no corner. It is very unlikely that the meet-point is on any of axes."



https://zhuanlan.zhihu.com/p/28023308

# P-NORMS VISUALIZED




all p-norms penalize larger weights

$p < 2$ tends to create sparse (i.e. lots of 0 weights)

$p > 2$ tends to like similar weights

# Model-based machine learning

1. pick a model

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

2. pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} loss(yy') + \lambda regularizer(w)$$

non c'è una regola per scegliere λ
se non trial and error
è uno dei tanti hyperparameter

3. develop a learning algorithm

$$\text{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda regularizer(w)$$

Find w and b
that minimize

# Minimizing with a regularizer

We know how to solve convex minimization problems using gradient descent:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} loss(yy')$$

If we can ensure that the loss + regularizer is convex then we could still use gradient descent:

$$\text{argmin}_{w,b} \underbrace{\sum_{i=1}^{n} loss(yy') + \lambda regularizer(w)}_{\text{convex as long as both loss and regularizer are convex}}$$

# Model-based machine learning

1. Pick a model

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

2. Pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

3. Develop a learning algorithm

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

Find w and b that minimize

# Our optimization criterion

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

Loss function: penalizes examples where the prediction is different than the label

Regularizer: penalizes large weights

Key: this function is convex allowing us to use gradient descent

# Gradient descent

- pick a starting point (w)
- repeat until loss doesn't decrease in any dimension:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_j = w_j - \eta \frac{d}{dw_j}(loss(w) + regularizer(w,b))$$

---

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

# Some more maths

$$\frac{d}{dw_j} objective = \frac{d}{dw_j} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2} \|w\|^2$$

$\vdots$ (some math happens)

$$= -\sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) + \lambda w_j$$

# Gradient descent

- pick a starting point (w)
- repeat until loss doesn’t decrease in any dimension:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_j = w_j - \eta \frac{d}{dw_j}(loss(w) + regularizer(w,b))$$

---

$$w_j = w_j + \eta \sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta \lambda w_j$$

# The update

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta\lambda w_j$$

learning rate

direction to update

constant: how far from wrong

regularization

If $w_i$ is positive, reduces $w_i$

If $w_i$ is negative, increases $w_i$

moves $w_i$ towards 0

i -> j

# L1 regularization

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \|w\|$$

---

$$\frac{d}{dw_j} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \lambda \|w\|$$

$$= -\sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) + \lambda sign(w_j)$$

# The update

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \boxed{\eta \lambda sign(}w_j)$$

learning rate

direction to update

constant: how far from wrong

regularization

If $w_i$ is positive, reduces by a constant
If $w_i$ is negative, increases by a constant

moves $w_i$ towards 0 ***regardless of magnitude***

# Regularization with p-norms

**L1:**

$$w_j = w_j + \eta(loss\_correction - \lambda sign(w_j))$$

**L2:** $$w_j = w_j + \eta(loss\_correction - \lambda w_j)$$

**Lp:**

$$w_j = w_j + \eta(loss\_correction - \lambda c w_j^{p-1})$$

# Regularizers summarized

- L1 is popular because it tends to result in sparse solutions (i.e. lots of zero weights). However, it is not differentiable, so it only works for gradient descent solvers
- L2 is also popular because for some loss functions, it can be solved directly (no gradient descent required, though often iterative solvers still)
- Lp is less popular since they don't tend to shrink the weights enough

# The other loss functions

Without regularization, the generic update is:

$$w_j = w_j + \eta y_i x_{ij} c$$

where

$$c = \exp(-y_i(w \cdot x_i + b))$$ exponential

$$c = 1[yy' < 1]$$ hinge loss

---

$$w_j = w_j + \eta (y_i - (w \cdot x_i + b) x_{ij})$$ squared error

# QUESTIONS?

Some slides are taken from David Kauchak