MACHINE LEARNING & DATA MINING

Outline

We'll consider a few practical problems:

- imbalanced datasets
- differences in training and application domains
- one-class classification

Imbalanced datasets

Imbalanced datasets

Problem setting:

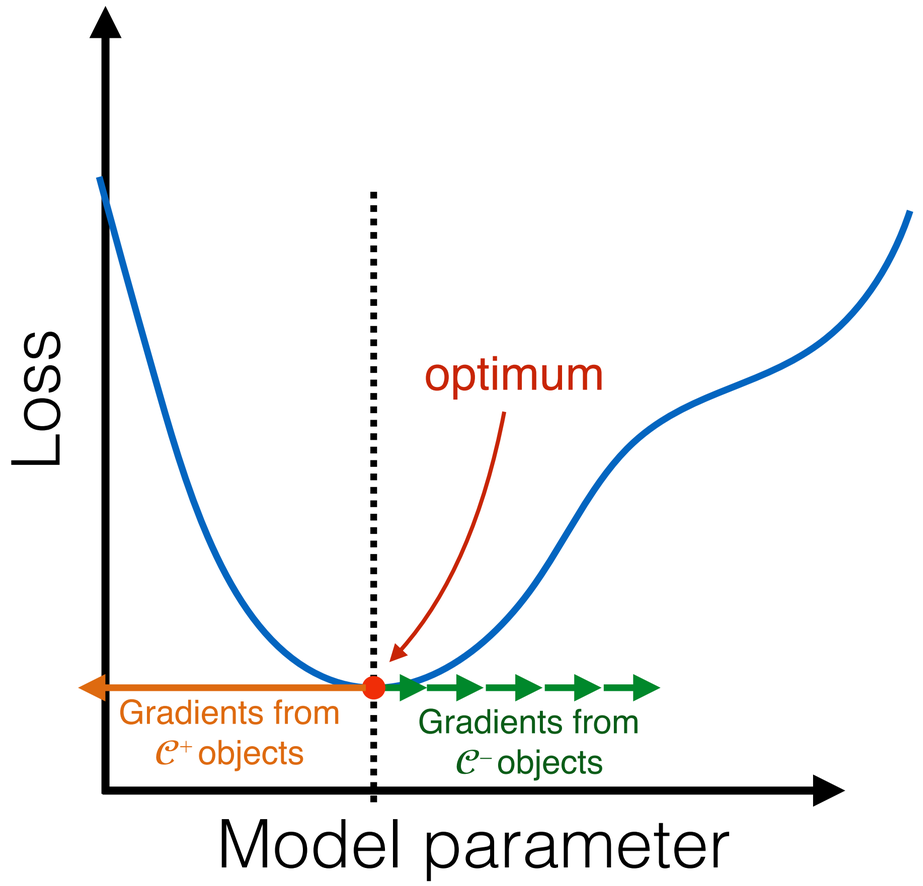
- Two class classification: $\mathcal{C}^+$ against $\mathcal{C}^-$
- Often in practice: $P(\mathcal{C}^+) \ll P(\mathcal{C}^-)$

This poses several problems:

- mini-batch learning procedures degrade
  - extremely slow learning
- imprecise results

Degradation of mini-batch learning

- Probability of picking an example from $\mathcal{C}^+$ into a mini-batch is low
- At the optimal point full gradient has to be 0
  - ⇒ Small gradients from $\mathcal{C}^-$ objects and large gradients from $\mathcal{C}^+$ objects
  - ⇒ Large gradients force us to reduce the learning rate
  - ⇒ slow learning



Change of priors

Decision function of the optimal Bayes classifier:

$$P(\mathcal{C}^+ \mid X) = \frac{P(X \mid \mathcal{C}^+)P(\mathcal{C}^+)}{P(X \mid \mathcal{C}^+)P(\mathcal{C}^+) + P(X \mid \mathcal{C}^-)P(\mathcal{C}^-)}$$

Thresholding the decision function defines a set of decision surfaces:

$$P(\mathcal{C}^+ \mid X) > \tau, \;\; \tau \in \mathbb{R}$$

The following change of the decision function defines the same set of decision surfaces (because there is 1 to 1 mapping between old and new thresholds):

$$P(\mathcal{C}^+ \mid X) \longrightarrow \frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)}$$

Change of priors

$$P(\mathcal{C}^+ \mid X) \longrightarrow \frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)}$$

$$\frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)} = \frac{P(\mathcal{C}^+ \mid X)}{P(\mathcal{C}^- \mid X)} = \frac{P(X \mid \mathcal{C}^+)P(\mathcal{C}^+)}{P(X \mid \mathcal{C}^-)P(\mathcal{C}^-)}$$

Change of priors

$$P(\mathcal{C}^+ \mid X) \longrightarrow \frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)}$$

$$\frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)} = \frac{P(\mathcal{C}^+ \mid X)}{P(\mathcal{C}^- \mid X)} = \frac{P(X \mid \mathcal{C}^+)P(\mathcal{C}^+)}{P(X \mid \mathcal{C}^-)P(\mathcal{C}^-)}$$

Multiplicative constant, that doesn't change the set of decision surfaces

Change of priors

$$P(\mathcal{C}^+ \mid X) \longrightarrow \frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)}$$

$$\frac{P(\mathcal{C}^+ \mid X)}{1 - P(\mathcal{C}^+ \mid X)} = \frac{P(\mathcal{C}^+ \mid X)}{P(\mathcal{C}^- \mid X)} = \frac{P(X \mid \mathcal{C}^+)P(\mathcal{C}^+)}{P(X \mid \mathcal{C}^-)P(\mathcal{C}^-)}$$

Multiplicative constant, that doesn't change the set of decision surfaces

⇒ We can artificially set the priors to be equal

- effectively done by sampling $\mathcal{C}^+$ and $\mathcal{C}^-$ objects at the same rates

Changing priors

An ideal classifier:

- Invariant under the change of priors w.r.t. the set of decision surfaces
- Change of priors may speed up and stabilize the learning
- **Only true for really good classifiers!**

Not-so-good classifier:

- Might change surfaces significantly
- This may render it useless

Importance sampling

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y \sim P_{x,y}} l(f(x), y) = \int P(x,y) l(f(x), y) dx dy =$$

$$\int P'(x,y) \frac{P(x,y)}{P'(x,y)} l(f(x), y) dx dy =$$

$$\mathop{\mathbb{E}}_{x,y \sim P'_{x,y}} w(x,y) \cdot l(f(x), y)$$

$$w(x,y) = \frac{P(x,y)}{P'(x,y)}$$

Importance sampling

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y \sim P_{x,y}} l(f(x), y) = \int P(x,y) l(f(x), y) dx dy =$$

$$\int P'(x,y) \frac{P(x,y)}{P'(x,y)} l(f(x), y) dx dy =$$

$$\mathop{\mathbb{E}}_{x,y \sim P'_{x,y}} w(x,y) \cdot l(f(x), y)$$

$$w(x,y) = \frac{P(x,y)}{P'(x,y)}$$

We can modify the sampling distribution and introduce corresponding weights without affecting the solution

Importance sampling

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y \sim P_{x,y}} l(f(x), y) = \int P(x,y) l(f(x), y) dx dy =$$

$$\int P'(x,y) \frac{P(x,y)}{P'(x,y)} l(f(x), y) dx dy =$$

$$\mathop{\mathbb{E}}_{x,y \sim P'_{x,y}} w(x,y) \cdot l(f(x), y)$$

$$w(x,y) = \frac{P(x,y)}{P'(x,y)}$$

We can modify the sampling distribution and introduce corresponding weights without affecting the solution

E.g. sample different classes at same rates

Importance sampling

$$\mathcal{L} = \mathop{\mathbb{E}}_{x,y \sim P_{x,y}} l(f(x), y) = \int P(x,y) l(f(x), y) dx dy =$$

$$\int P'(x,y) \frac{P(x,y)}{P'(x,y)} l(f(x), y) dx dy =$$

$$\mathop{\mathbb{E}}_{x,y \sim P'_{x,y}} w(x,y) \cdot l(f(x), y)$$

$$w(x,y) = \frac{P(x,y)}{P'(x,y)}$$

We can modify the sampling distribution and introduce corresponding weights without affecting the solution

E.g. sample different classes at same rates

Q: How is that different from the change of priors?

Importance sampling

Resampling trick allows to:

- stabilize the learning by increasing the frequency of rare but important samples
- increase the convergence rate

Can be predetermined, e.g.:

- uniform sampling across classes
- increased sampling probability of hard examples

or can be computed on the fly, e.g. adaptive sampling methods

Re-weighting

Re-weighting

Setting:

- training set X with distribution P
- target set X’ with distribution P’, with targets absent
- $P(x) \neq P'(x)$, but
- supp P = supp P’

Re-weighting

- train a classifier $r(x)$ on X' against X:

$$w(x) = \frac{r(x)}{1 - r(x)} \approx \frac{P'(x)}{P'(x) + P(x)} \cdot \frac{P'(x) + P(x)}{P(x)} = \frac{P'(x)}{P(x)}$$

- use output as weights (similarly to importance sampling):

$$\mathcal{L}_{\text{target}} = \mathop{\mathbb{E}}_{x,y \sim P'} l(f(x), y) = \mathop{\mathbb{E}}_{x,y \sim P} w(x) l(f(x), y)$$

Semi-supervised learning

Semi-supervised learning

- Semi-supervised learning targets cases with a large amount of unlabeled data:

$$\mathcal{D}_{\text{supervised}} = \{(x_i, y_i)\}_{i=1}^{N}$$

$$\mathcal{D}_{\text{unsupervised}} = \{x_i\}_{i=1}^{M}$$

$$M \gg N$$

- Distributions of X are equal in both datasets

Semi-supervised learning

Common techniques

- train dimensionality reduction method on $\mathcal{D}_{\text{unsupervised}}$
- apply dimensionality reduction to $\mathcal{D}_{\text{supervised}}$
- solve supervised problem in the reduced domain

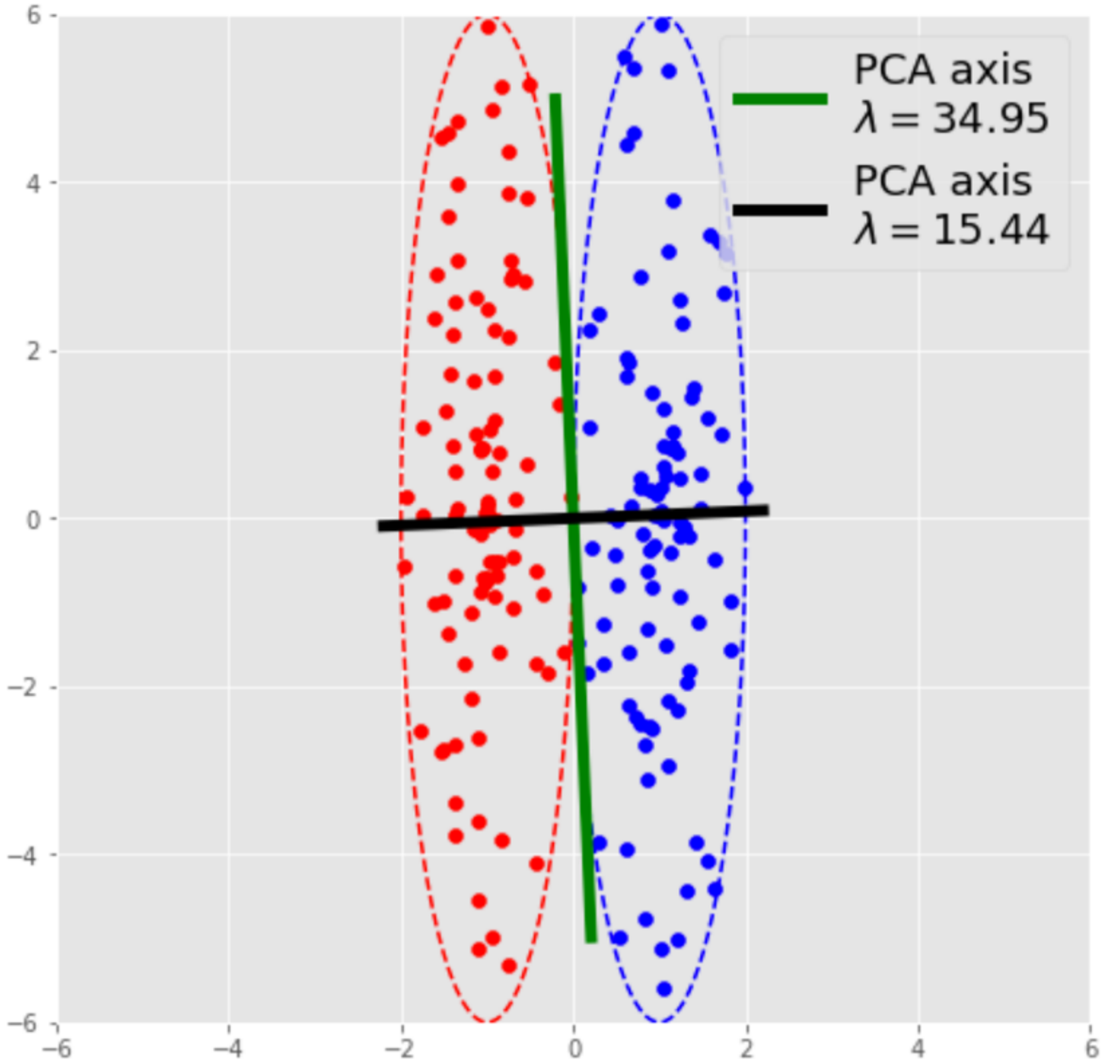
Examples:

- feature selection + classifier
- PCA + classifier

Possible problems

- Dimensionality reduction and supervised methods are trained independently
- The objectives of the two methods may be in conflict
- E.g. information lost in the compression might be important for the supervised task

Conflict of objectives



Semi-supervised deep learning

- In deep learning both objectives can be trained simultaneously
- E.g.:

$$\begin{aligned} \text{encoder} \quad & z = e(x); \\ \text{decoder} \quad & x' = d(z); \\ \text{classifier} \quad & f(z) \end{aligned}$$

$$\mathcal{L} = \mathop{\mathbb{E}}_{X,Y \sim \text{supervised}} l_1(f(e(x)), y) + \mathop{\mathbb{E}}_{X \sim \text{unsupervised}} l_2(x, d(e(x)))$$

- (see day-6 notebooks for an implementation example)

One-class classification

Setting

- training dataset consists of only one class $\mathcal{C}^+$
- target dataset might contain additional classes

- Examples:
  - anomaly detection
  - outlier detection
  - novelty detection

One-class classification

Density based:

- Estimate the probability density $P(X \mid \mathcal{C}^+)$
- Use it as the decision function: $P(X \mid \mathcal{C}^+) > \tau$
- Essentially a generative problem

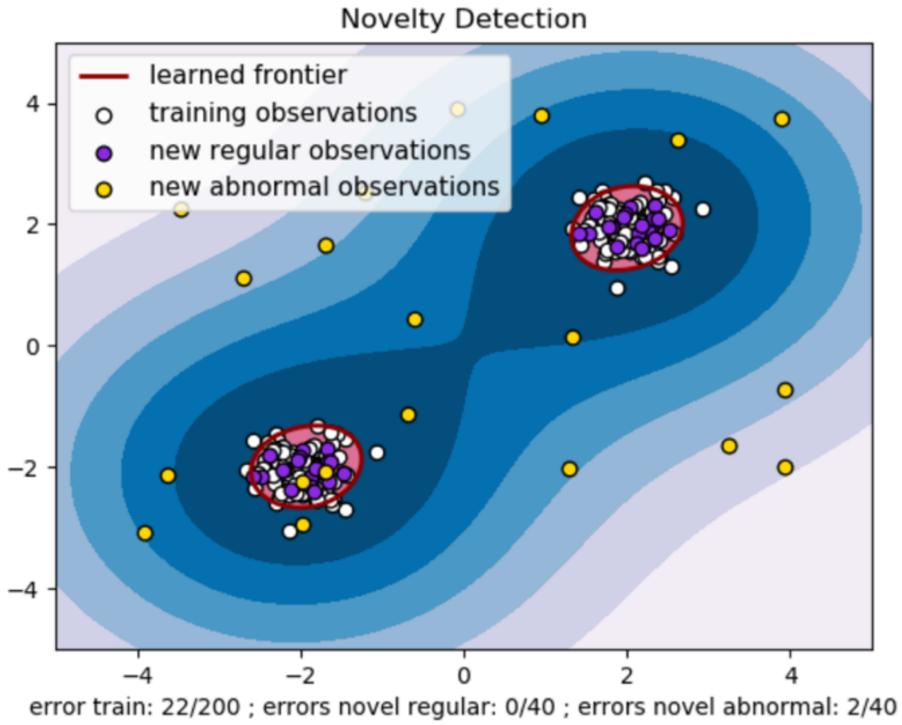
One-class classification

Density based:

- Estimate the probability density $P(X \mid \mathcal{C}^+)$
- Use it as the decision function: $P(X \mid \mathcal{C}^+) > \tau$
- Essentially a generative problem

Distance based:

- E.g. one-class SVM
  - minimizes the volume contained by the class
- (see https://scikit-learn.org/stable/modules/outlier_detection.html for more details



Summary

- Imbalanced datasets
  - changing priors should be done with care
  - resampling
- Re-weighting
  - different training and target distributions of X
  - special case of domain adaptation
- One-class classification
  - density and distance based approaches