ECEN 740: Machine Learning Engineering

Lecture 4: Support Vector Machine

Tie Liu

Texas A&M University

Linear predictors for binary classification

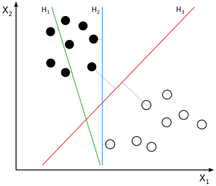
- In this lecture, we consider the problem of learning a binary classifier in which the observation $\mathsf{x} \in \mathbb{R}^d$ and the target $\mathsf{y} \in \{+1, -1\}$
- We focus the class of *linear predictors*:

$$\mathcal{H} = \{h_{(w,b)} : w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where

$$h_{(w,b)}(x) = sgn\left(\langle w, x\rangle + b\right)$$

- Note that for any $w \neq 0$, the *decision boundary* $\langle w, x\rangle + b = 0$ of $h_{(w,b)}$ is a *hyperplane* in $\mathbb{R}^d$



Learning via ERM

- Let $\{(x_i, y_i) : i \in [m]\}$ be a training data set. For each training data example (x_i, y_i), let

$$\tilde{\gamma}_i := y_i(\langle w, x_i \rangle + b)$$

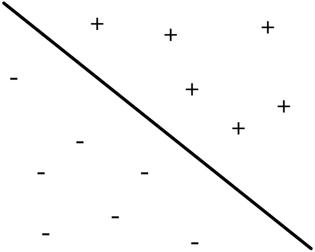
  be the *(functional) margin* of the classifier $h_{(w,b)}$ at (x_i, y_i)

- Note that at each training data example (x_i, y_i), the classifier $h_{(w,b)}$ makes an error *if and only if* the $\tilde{\gamma}_i \leq 0$

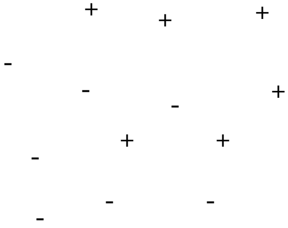
- Therefore, an ERM classifier (under the canonical 0-1 loss) can be learned by solving the following optimization problem:

$$\min_{(w,b)} \left[\frac{1}{m} \sum_{i=1}^{m} 1_{\{\tilde{\gamma}_i \leq 0\}} \right]$$

- We shall the following two cases separately:



Separable case



Non-separable case

Separable case

- For the separable case, an ERM solution is one that satisfies:

$$\tilde{\gamma}_i > 0, \quad \forall i \in [m]$$

- Note that for any $\alpha > 0$, $h_{(w,b)}$ and $h_{(w/\alpha,b/\alpha)}$ represent the *same* classifier

- We thus have the freedom to choose the scaling of (w, b) so that

$$\min_{i \in [m]} \tilde{\gamma}_i = 1$$

  In this case, we say that the ERM solution is *canonical*

- Therefore, for the separable case, we can find an ERM solution that satisfies:

$$\tilde{\gamma}_i \geq 1, \quad \forall i \in [m]$$

- Note that $\tilde{\gamma}_i$ is a *linear* function of (w, b), so an ERM solution that satisfies the above *linear* inequalities can be found via *linear programming (LP)*

Non-separable case

- For the non-separable case, the ERM problem:

$$\min_{(w,b)} \left[\frac{1}{m} \sum_{i=1}^{m} 1_{\{\tilde{\gamma}_i \leq 0\}} \right]$$

  is known to be *computationally hard*

- To make progress, note that in the separable case we enforced the *hard* constraints:

$$\tilde{\gamma}_i \geq 1, \quad \forall i \in [m]$$

- For the non-separable, a natural relaxation is to allow the constraints to be violated for some of the training examples. This can be modeled by introducing *non-negative slack variables* $(\psi_1, \ldots, \psi_m)$ and replacing each hard constraint

$$\tilde{\gamma}_i \geq 1$$

  by the *soft* constraint

$$\tilde{\gamma}_i \geq 1 - \psi_i$$

- It is then natural to consider minimizing the *average* of ψ_i, which corresponds to the *aggregate* violations of the hard constraints. This leads to the following optimization problem for learning a linear predictor in the non-separable case:

$$\begin{aligned} \min_{(w,b,\psi_1,\ldots,\psi_m)} \quad & \frac{1}{m}\sum_{i=1}^{m}\psi_i \\ \text{subject to} \quad & \tilde{\gamma}_i \geq 1-\psi_i, \quad \forall i \in [m] \\ & \psi_i \geq 0, \quad \forall i \in [m] \end{aligned}$$

- Fix (w, b) and consider minimizing over $(\psi_1, \ldots, \psi_m)$. Clearly, the best assignment for ψ_i is

$$\max(0, 1-\tilde{\gamma}_i)$$

- We thus have the following *equivalent* formulation for learning a linear predictor in the non-separable case:

$$\min_{(w,b)} \left[\frac{1}{m}\sum_{i=1}^{m}\max(0, 1-\tilde{\gamma}_i)\right]$$

Convex learning principle

- The objective in the above optimization problem suggests a *new* way of measuring the loss of a linear predictor $h_{(w,b)}$ on a data example (x_i, y_i):

$$\ell_{hinge}(h_{(w,b)}, (x_i, y_i)) = \max(0, 1 - \tilde{\gamma}_i)$$

- By contrast, the original ERM problem

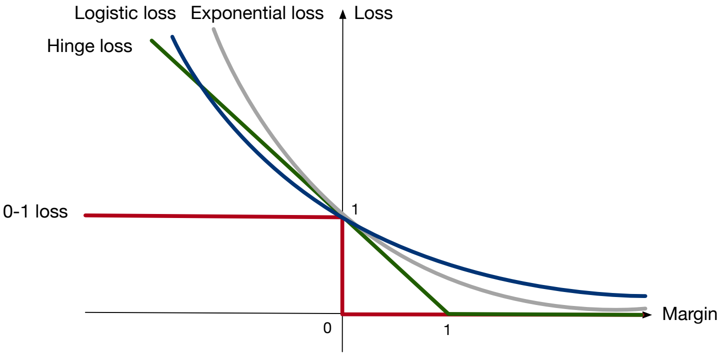
$$\min_{(w,b)} \left[\frac{1}{m} \sum_{i=1}^{m} 1_{\{\tilde{\gamma}_i \leq 0\}} \right]$$

uses the *0-1* loss below to learn a linear predictor:

$$\ell_{0-1}(h_w, (x, y)) = 1_{\{\tilde{\gamma}_i \leq 0\}}$$

- Note that both the *hinge loss* and the 0-1 loss depend on the linear predictor and the data example *only* through its margin

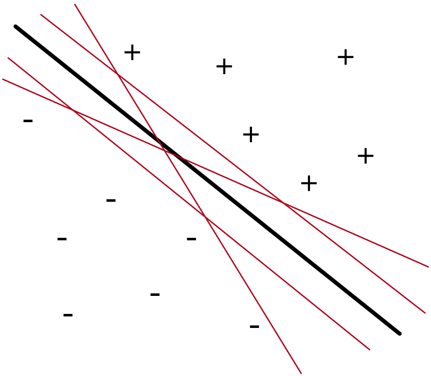
- More importantly, we observe that the hinge loss is a *convex, upper* bound on the 0-1 loss as a function of margin:



- As we shall see in the next lecture, the fact that hinge loss is a *convex* function of margin makes learning amenable to *efficient* algorithms

Large-margin separators

- Note that for any separable training data set, there are *infinite many* ERM hyperplanes:



- While all ERM classifiers render the same *empirical* error, we would like to pick one that can render a small *true* error as well

- To pick an ERM solution that also renders a small true error, let us define the *geometrical margin* γ_i as

$$\gamma_i := \tilde{\gamma}_i / \|w\|$$

- For a training data example (x_i, y_i) that is *correctly* classified, γ_i is simply the *distance* from x_i to the hyperplane (w, b) (for training examples that are *mis-classified*, the geometrical margin is the *negative distance* to the hyperplane)

- The *geometrical margin* γ for the *entire training data set* is defined as

$$\gamma := \min_{i \in [m]} \gamma_i$$

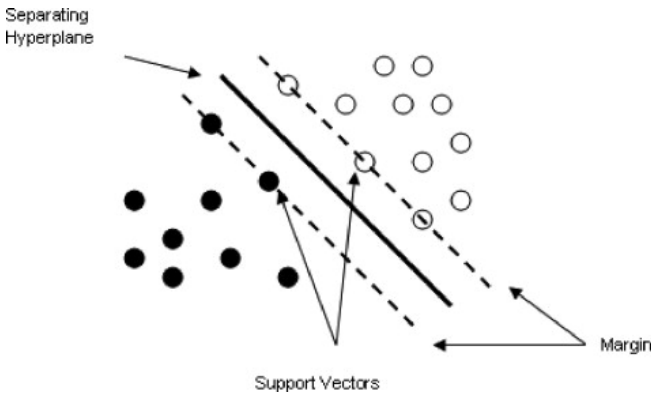
- Intuitively, one would prefer an ERM hyperplane that separates the training data set with a *large* geometrical margin (to the entire training data set): If a hyperplane has a large geometrical margin, then it will still separate the training examples even if we slightly perturb their observations, and this will help improve the *generalization* of the ERM solution

Hard support vector machine (SVM)

- *Hard support vector machine (SVM)* is the learning rule in which we return an ERM hyperplane that separates the training data set with the *largest* possible geometrical margin for the entire training data set
- For a *canonical* ERM solution $h_{(w,b)}$, the geometrical margin γ for the entire training data set is given by $1/\|w\|$
- By considering *canonical* ERM solutions, we are thus led to the following constrained optimization problem to determine the *hard-SVM* rule:

$$\begin{aligned} \min_{(w,b)} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & \tilde{\gamma}_i \geq 1, \quad \forall i \in [m] \end{aligned}$$

- For the hard-SVM solution, there will be at least one training data example *in each class* for which $\tilde{\gamma}_i = 1$. Let the hyperplanes that pass through these points be denoted H_+ and H_- respectively


Separating
Hyperplane
Margin
Support Vectors

- This constrained optimization problem can be set up using *Lagrange multipliers*, and solved using numerical methods for *quadratic programming (QP)* problems
- The form of the solution is

$$w = \sum_{i=1}^{m} \lambda_i y_i x_i$$

  where the λ_i's are non-negative Lagrange multipliers. Notice that the solution is a *linear* combination of the x_i's
- The key feature of the above solution is that λ_i is *zero* for every x_i except those which lie on the hyperplanes H_+ or H_-; these points are called the *support vectors*
- The fact that *not* all of the training data examples contribute to the final solution solution is referred to as the *sparsity* of hard-SVM

- To make predictions for a *new* observation x_*, we compute

$$sgn(\langle w, x_* \rangle + b) = sgn\left(\sum_{i=1}^{m} \lambda_i y_i \langle x_i, x_* \rangle + b\right)$$

  in which the training observations $\{x_i : i \in [m]\}$ and the test observation x_* enter the computation only in terms of their *inner products*

- Later on we will see that by using the *kernel trick*, we can replace the occurrences of the inner product by the kernel to obtain an equivalent result in *feature space*

Soft-SVM

- The hard-SVM formulation assumes that the training sequence is linearly separable, which is a rather strong assumption
- *Soft-SVM* can be viewed as a *relaxation* of the hard-SVM rule that can be applied even if the training sequence is not linearly separable
- Note that the quadratic optimization problem for hard-SVM enforces the hard constraints:

$$\tilde{\gamma}_i \geq 1, \quad \forall i \in [m]$$

- As before, a natural relaxation is to allow the constraints to be violated for some of the training examples. This can be modeled by introducing non-negative *slack* variables $(\psi_1, \ldots, \psi_m)$ and replacing each hard constraint

$$\tilde{\gamma}_i \geq 1$$

by the *soft* constraint

$$\tilde{\gamma}_i \geq 1 - \psi_i$$

- Soft-SVM jointly minimizes the norm of w (corresponding to the margin) and the average of ψ_i (corresponding to the aggregate violations of the hard constraints). The tradeoff between the two terms is controlled by a hyper-parameter λ
- This leads to the following *soft-SVM* optimization problem:

$$\begin{aligned} \min_{(w,b,\psi_1,\dots,\psi_m)} \quad & \frac{1}{m}\sum_{i=1}^{m}\psi_i + \frac{\lambda}{2}\|w\|^2 \\ \text{subject to} \quad & \tilde{\gamma}_i \geq 1-\psi_i, \quad \forall i \in [m] \\ & \psi_i \geq 0, \quad \forall i \in [m] \end{aligned}$$

- Fix (w, b) and consider minimizing over $(\psi_1, \dots, \psi_m)$. Clearly, the best assignment for ψ_i is

$$\max(0, 1-\tilde{\gamma}_i)$$

- We thus have the following *equivalent* formulation for the soft-SVM problem:

$$\min_{(w,b)} \left[\frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - \tilde{\gamma}_i) + \frac{\lambda}{2} \|w\|^2 \right]$$

  which can be viewed as *RLM* under the *hinge* loss and *Tikhonov* regularization

- By the *Representer Theorem* (which we will discuss in detail when we discuss the *kernel method* in a later lecture), the solution for w again takes the form of a *linear* combination of the training observations:

$$w = \sum_{i=1}^{m} \alpha_i x_i$$

- Similar to hard-SVM. soft-SVM can also be *kernelized*. However, unlike hard-SVM, the *support* vectors (those with $\alpha_i \neq 0$) in soft-SVM are not only those on H_+ or H_-, but also those that incur *penalties*

- In practice, hard-SVM is sensitive to *incorrect* labeling of the training data due to its *sparsity*. Thus, soft-SVM is often used in practice (even when the training data set is separable)