1 Exam revision

1.1 Simple Learning Framework

The simplest learning task is to *classify* features in two classes, in the case where there exists a unique and deterministic map between them.

- **Domain set** (*Instance space*) $\mathcal{X}$, where elements $\boldsymbol{x} \in \mathcal{X}$ are *vectors of features*.
- **Label set** $\mathcal{Y}$
- **Training data**: sequence (order matters, there can be repetitions) of elements in $\mathcal{X} \times \mathcal{Y}$: $S = ((x_1, y_1), \dots, (x_m, y_m))$. m denotes the number of training samples.
- The ML algorithm A, given S, outputs a **hypothesis function** (prediction rule, or *predictor*) $A(S) = h_S \colon \mathcal{X} \to \mathcal{Y}$ that represents the mapping between $\mathcal{X}$ and $\mathcal{Y}$ *learned* by the algorithm.
- We assume a **simple data generation model** for constructing S. $\mathcal{X}$ is generated by sampling a distribution (pdf) $\mathcal{D}$ which is *not known* by the learning model, and are then labelled by a function $f \colon \mathcal{X} \to \mathcal{Y}$ (labelling function, representing the *ground truth*). $\mathcal{D}$ allows to assign probabilities to *events*, i.e. subsets $A \subset \mathcal{X}$. We introduce the following notation:

  $$\mathcal{D}(A) = \mathbb{P}[x \in A] \qquad A \subset \mathcal{X}$$

  Equivalently, we can denote subsets A using their *characteristic function* $\pi_A(x) \colon \mathcal{X} \to \{0, 1\}$:

  $$\pi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \qquad A = \{x \in A \colon \pi(x) = 1\}$$

  And so $\mathcal{D}(A) = \mathbb{P}_{x \sim \mathcal{D}}[\pi_A(x)]$.

- We define the **generalization error** (or **risk**, **true error**) of h_S as the probability of randomly choosing a sample from $\mathcal{D}$ so that $h_S(x) \neq f(x)$:

  $$L_{\mathcal{D},f}(h_S) \equiv \underset{x\sim\mathcal{D}}{\mathbb{P}}[h_S(x) \neq f(x)] \equiv \mathcal{D}(\{x\colon h_S(x) \neq f(x)\})$$

  Note that $L_{\mathcal{D},f}$ depends on the *data generation* ($\mathcal{D}$, f), and so cannot be known by the learner. For simplicity, we denote $L_{\mathcal{D}} \equiv L_{\mathcal{D},f}$.

- The **empirical risk** (or **training error**) is defined as:

  $$L_S(h) \equiv \frac{|\{1 \leq i \leq m\colon h(x_i) \neq y_i\}|}{m}$$

  (Recall that $|\{\dots\}|$ denotes the cardinality, i.e. the number of elements, of the set $\{\dots\}$).

- **Empirical Risk Minimization** (ERM): the paradigm for choosing A so that it minimizes $L_S(h)$, hoping that this will translate to a minimum $L_{\mathcal{D}}(h)$ as well. Formally, this is written as:

  $$\mathrm{ERM}(S) \in \arg\min L_S(h)$$

  Note that there could be *many* possible choiches for h so that $L_S(h)$ is minimum, and the algorithm simply returns one of them.

- **Overfitting**: when $L_S(h) \sim 0$, but $L_{\mathcal{D}}(h) \gg 0$. This happens if the ERM search is not limited, meaning that the algorithm can simply *memorize* the training set S, leading to a performance on a different dataset which is no better than chance.

- **Hypothesis class**: set of predictors $h \in \mathcal{H}$ that are available for the ERM search (meaning that the final h_S must be $\in \mathcal{H}$). We denote a *constrained* ERM algorithm with $\mathrm{ERM}_{\mathcal{H}}$, so that:

  $$\mathrm{ERM}_{\mathcal{H}}(S) \in \arg\min_{h\in\mathcal{H}} L_S(h)$$

  The choice of $\mathcal{H}$ introduce a *inductive bias* in the learner, and should be made according to some prior knowledge about the problem.

- **PAC learnability**. A hypothesis class $\mathcal{H}$ is PAC learnable if, given a sufficient number of examples, an ERM algorithm will output an hypothesis which is *probably approximately correct*, meaning that with probability $1-\delta$ it is ϵ-accurate. We require the existence of a sufficient m for any choice of ϵ and δ, meaning that the model can be made more robust and accurate with more training samples.

  More formally, a hypothesis class $\mathcal{H}$ is PAC learnable if there exist a function (**sample complexity**) $m_{\mathcal{H}}\colon (0,1) \times (0,1) \to \mathbb{N}$, $(\epsilon, \delta) \mapsto m_{\mathcal{H}}(\epsilon, \delta)$, and a learning algorithm that $\forall \epsilon, \delta \in (0,1)$ and every distribution $\mathcal{D}$ over $\mathcal{X}$, and for every labelling function $f\colon \mathcal{X} \to \{0,1\}$, assuming realizability, when running over $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$ and labelled by f, the algorithm returns a hypothesis h such that, with probability $\geq 1-\delta$ (over the choice of the samples) satisfies $L_{\mathcal{D},f}(h) \leq \epsilon$. We also require $m_{\mathcal{H}}$ to be the minimum integer that guarantees the last property.

1.1.1 Theorems

PAC for finite hypothesis classes

The idea is to find a minimum number of samples m that are needed to construct an S so that the model h_S learned by the ERM algorithm will be $\epsilon-$*accurate*, i.e. satisfying:

$$L_{\mathcal{D},f}(h_S) \leq \epsilon$$

with probability $\geq 1-\delta$ (*confidence*) over the choice of an i.i.d. sample S of size m from the $\mathcal{D}$ distribution.

Hypotheses

1. **Finite hypothesis class**: $h_S \in \mathcal{H}$, with $|\mathcal{H}| < \infty$.
2. **Realizability**. There exists a "perfect" $h^* \in \mathcal{H}$, such that $L_{\mathcal{D},f}(h^*) = 0$. By the definition of *generalization error*, this means that $h^*(x) = f(x)$ with probability 1 for a random sample $x \in \mathcal{X}$, implying that $L_S(h^*) = 0$.
3. **i.i.d. assumption**. The examples in the training dataset S are **independently** and *identically distributed* according to $\mathcal{D}$, meaning that $S \sim \prod_{i=1}^m \mathcal{D} = \mathcal{D}^m$.

Proof. δ is the maximum probability of the learner failing, that is of not being ϵ-accurate:

$$\mathcal{D}^m(\{S\colon L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \delta$$

The learner fails only for certain "bad hypotheses", that are inside a set $\mathcal{H}_B$:

$$\mathcal{H}_B = \{h \in \mathcal{H}\colon L_{\mathcal{D},f}(h) > \epsilon\}$$

The ERM$_{\mathcal{H}}$ algorithm will choose an h_S that verifies $L_S(h_S) = 0$, due to realizability (generally $h_S \neq h^*$, because there could be many h_S with a 0 training error). So, it may choose a *bad hypothesis* only if it "appears good" on S, that is if S is an element of the set of *misleading training sets*:

$$M = \{S\colon \exists h \in \mathcal{H}_B, L_S(h) = 0\}$$

Note that any set S that results in the learner failing must be in M (because it contains all sets that *may be chosen* and result in failing), and so:

$$\{S\colon L_{\mathcal{D},f}(h_S) > \epsilon\} \subseteq M$$

The converse is not necessarily true: maybe there are certain sets in M that, even if misleading, result nonetheless in a good performing h_S, because they are compatible with more than one hypothesis.
We can now rewrite M as the union of misleading sets:

$$\{S\colon L_{\mathcal{D},f}(h_S) > \epsilon\} \subseteq M = \bigcup_{h\in\mathcal{H}_B} \{S\colon L_S(h) = 0\} \tag{1.1}$$

Applying the probability measure to (**??**) leads to:

$$\mathcal{D}^m(\{S\colon L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \mathcal{D}^m(M) = \mathcal{D}^m\left(\bigcup_{h\in\mathcal{H}_B} \{S\colon L_S(h) = 0\}\right) \tag{1.2}$$

Recall the **union bound**, that is the measure of the union of two sets is less or equal to the sum of the measure of each set (because there could be a non-empty intersection):

$$\mathcal{D}(A \cup B) \leq \mathcal{D}(A) + \mathcal{D}(B)$$

Applying it to the right hand side of (1.2) we have:

$$\mathcal{D}^m(\{S\colon L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \sum_{h\in\mathcal{H}_B} \mathcal{D}^m(\{S\colon L_S(h) = 0\})$$

Fix a certain *bad* hypothesis $h \in \mathcal{H}_B$. The event $L_S(h) = 0$ is equivalent to $\forall i$ $h(x_i) = f(x_i)$, and so:

$$\mathcal{D}^m(\{S\colon L_S(h) = 0\}) = \mathcal{D}^m(\{S\colon \forall i, h(x_i) = f(x_i)\})$$

which is just the product of probabilities that each $h(x_i) = f(x_i)$, as x_i are i.i.d:

$$= \prod_{i=1}^m \mathcal{D}(\{x_i\colon h(x_i) = f(x_i)\})$$

The probability of a good prediction is $1-$ the probability of error, which is $L_{\mathcal{D},f}$. As $h \in \mathcal{H}_B$, $L_{\mathcal{D},f} > \epsilon$, and so:

$$\mathcal{D}(\{x_i\colon h(x_i) = y_i\}) = 1 - L_{\mathcal{D},f}(h) \leq 1 - \epsilon$$

Substituting back leads to:

$$\mathcal{D}^m(\{S\colon L_S(h) = 0\}) \leq (1-\epsilon)^m \leq e^{-\epsilon m}$$

And finally:

$$\mathcal{D}^m(\{S\colon L_{\mathcal{D},f}(h_S) > \epsilon\}) \leq \sum_{h\in\mathcal{H}_B} \mathcal{D}^m(\{S\colon L_S(h) = 0\}) \leq |\mathcal{H}_B| e^{-\epsilon m} \leq |\mathcal{H}| e^{-\epsilon m}$$

We now want to choose m so that the red term is $\leq \delta$. So we impose:

$$|\mathcal{H}| e^{-\epsilon m} \leq \delta \Rightarrow m \geq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

This means that *every finite hypothesis class is PAC learnable with sample complexity*:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$$

1.2 Agnostic Simple Learning

We consider now a binary classification task where features are not sufficient to uniquely predict labels. In other words, a couple of exactly equal features can lead to opposite labels.

- **Data generation**. Consider a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, with $\mathcal{Y} = \{0, 1\}$. The probability of sampling a certain datapoint (x, y) is given by:

  $$\mathbb{P}(x, y) = \mathbb{P}(x)\mathbb{P}(y|x)$$

  For example $x = 4$ can be associated to $y = 0$ in 75% of the cases, and to $y = 1$ in the other 25%. In such a case realizability cannot hold, as the learner is deterministic, and must choose one output or the other, meaning that it is bound to make a certain error.

- **True error**. The risk of a predictor h is given by the probability of sampling $(x, y) \sim \mathcal{D}$ so that $h(x) \neq y$:

  $$L_{\mathcal{D}}(h) \equiv \mathop{\mathbb{P}}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \equiv \mathcal{D}(\{(x, y) \colon h(x) \neq y\})$$

- **Empirical risk** maintains the same formula as before:

  $$L_S(h) \equiv \frac{|\{1 \leq 1 \leq m \colon h(x_i) \neq y_i\}|}{m}$$

- **Bayes Optimal Predictor** Given any probability distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, the best label predicting function $\mathcal{X} \to \{0, 1\}$ is:

  $$f_{\mathcal{D}}(x) = \begin{cases} 1 & \mathbb{P}[y = 1|x] \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

  That is, predict 1 if the probability of x being $y = 1$ is greater than chance (1/2), and 0 otherwise.

- **Agnostic PAC learnability**. A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exists a function $m_{\mathcal{H}} \colon (0, 1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: for every $\epsilon, \delta \in (0, 1)$ and every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. samples generated by $\mathcal{D}$, the algorithm returns an hypothesis h such that, with probability $\geq 1 - \delta$ it satisfies:

  $$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$$

1.3 Agnostic PAC learnable

Finally, we generalize the previous case to a *general* learning task, i.e. classification with a larger number of classes, or regression.

- **Generalized Loss Functions**. A loss function $\ell\colon \mathcal{H} \times Z \to \mathbb{R}_+$, with Z being a certain domain, is a function that *evaluates* the performance of a model $h \in \mathcal{H}$ on the domain Z.
- **Risk function** is defined just as the expected value of the loss:

$$L_{\mathcal{D}}(h) \equiv \mathbb{E}_{z\sim\mathcal{D}}[\ell(h,z)]$$

- The **empirical risk** is the average loss over the training set:

$$L_S(h) \equiv \frac{1}{m}\sum_{i=1}^{m} \ell(h, z_i)$$

- **Agnostic PAC learnability with generalized loss function**. A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a domain Z and a loss function $\ell\colon \mathcal{H} \times Z \to \mathbb{R}_+$ if there exist a function $m_{\mathcal{H}}\colon (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: for every $\epsilon, \delta \in (0,1)$, and every distribution $\mathcal{D}$ over Z, running the algorithm over $m \geq m_{\mathcal{H}}$ i.i.d. samples S generated by $\mathcal{D}$ results in a hypothesis $h_S \in \mathcal{H}$ that, with probability $1-\delta$, satisfies:

$$L_{\mathcal{D}}(h_S) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$$

  where $L_{\mathcal{D}}(h_S) = \mathbb{E}_{z\sim\mathcal{D}}[\ell(h_S, z)]$.

1.4 Uniform Convergence

ERM algorithm minimizes the empirical risk, hoping that this will result in a low true risk as well. So, we want to prove which conditions are needed so that L_S is close to $L_{\mathcal{D}}$.

- **ϵ-representative**: A training set S is called ϵ-representative (with respect to a distribution $\mathcal{D}$ over a domain Z, a hypothesis class $\mathcal{H}$, and a loss ℓ) if:

$$\forall h \in \mathcal{H}, \quad |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$$

- **Uniform convergence**: A hypothesis class $\mathcal{H}$ has the *uniform convergence property* (with respect to a domain Z and a loss function ℓ) if there exists a function $m_{\mathcal{H}}^{\mathrm{UC}}\colon (0,1)^2 \to \mathbb{N}$ such that for every ϵ, $\delta \in (0,1)$ and for every probability distribution $\mathcal{D}$ over Z, if S is a sample of at least $m \geq m_{\mathcal{H}}^{\mathrm{UC}}(\epsilon, \delta)$ examples drawn i.i.d. from $\mathcal{D}$, then, with probability of at least $1-\delta$, S is ϵ-representative.

1.4.1 Theorems

- **Learned hypotheses on representative sample**. If a training set S is $\epsilon/2$ representative, then **any** output of the learner $\mathrm{ERM}_{\mathcal{H}}(S)$, i.e. any:

  $$h_S \in \arg\min_{h\in\mathcal{H}} L_S(h)$$

  satisfies:

  $$L_{\mathcal{D}}(h_S) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

  Proof. By the definition of $\epsilon/2$-representative we have:

  $$-\frac{\epsilon}{2} \leq L_S(h) - L_{\mathcal{D}}(h) \leq \frac{\epsilon}{2} \tag{1.3}$$

  Rearranging to isolate $L_{\mathcal{D}}(h)$:

  $$-\frac{\epsilon}{2} - L_S(h) \leq -L_{\mathcal{D}}(h) \leq \frac{\epsilon}{2} - L_S(h) \Rightarrow \frac{\epsilon}{2} + L_S(h) \geq L_{\mathcal{D}}(h) \geq -\frac{\epsilon}{2} + L_S(h)$$

  This inequality holds for any $h \in \mathcal{H}$, and in particular for the h_S outputted by the learner:

  $$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\epsilon}{2}$$

  h_S is a minimum point for L_S (by the definition of ERM algorithm) and so $L_S(h_S) \leq L_S(h)$, leading to:

  $$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\epsilon}{2} \leq L_S(h) + \frac{\epsilon}{2} \tag{1.4}$$

  We can now reuse (1.3) to get an inequality for $L_S(h)$:

  $$L_S(h) \leq L_{\mathcal{D}}(h) + \frac{\epsilon}{2}$$

  That can be applied in (1.4) completing the chain:

  $$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \frac{\epsilon}{2} \leq L_S(h) + \frac{\epsilon}{2} \leq L_{\mathcal{D}}(h) + \frac{\epsilon}{2} + \frac{\epsilon}{2} = L_{\mathcal{D}}(h) + \epsilon$$

  This holds for any $h \in \mathcal{H}$, and so we can pick the *minimum* $L_{\mathcal{D}}(h)$ to get the stronger condition:

  $$L_{\mathcal{D}}(h_S) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \epsilon$$

  In other words, if S is ϵ-representative, then the true risk and the empirical risk are close to each other.

- **Uniform convergence implies PAC learnability**. If a class $\mathcal{H}$ has the uniform convergence property with a sample complexity $m_{\mathcal{H}}^{\text{UC}}$ then the class is agnostically PAC learnable with the sample complexity $m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{\text{UC}}(\epsilon/2, \delta)$.

  Proof. Thanks to uniform convergence, we are certain that we can use the ERM algorithm to get *nice* results, meaning empirical risks that are close to true risks, and so, using the previous theorem, we reach the PAC learning condition.

- **Finite Classes are agnostic PAC learnable**. We want to prove that uniform convergence holds for finite classes, implying that they are agnostic PAC learnable.

  Fix some accuracy ϵ and confidence δ. We need to find a sufficient $m = |S|$ so that:

$$\mathcal{D}^m(\{S\colon \forall h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon\}) \geq 1 - \delta$$

  That is, if we pick m samples from $\mathcal{D}$ to generate S, then with probability $1 - \delta$, uniformly for *any* hypothesis $h \in \mathcal{H}$, the empirical risk and true risk are ϵ-close: $L_S(h) - L_{\mathcal{D}}(h)| \leq \epsilon$.

  If the probability of an event p is $\geq 1-\delta$, then the probability of the opposite event $\bar{p}$ is $< 1 - (1 - \delta) = \delta$:

$$\mathcal{D}^m(\{S\colon \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) < \delta$$

  (Note how $\forall \to \exists$).

  Consider now the argument of $\mathcal{D}^m$. This is the set of all samples sets S for which there is an hypothesis that satisfies some property. Equivalently, we could select all samples with that property on a *fixed* h, do the same for every other h, and join all the results:

$$\{S\colon \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\} = \bigcup_{h \in \mathcal{H}} \{S\colon |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}$$

  Applying $\mathcal{D}^m$ to both sides along with the union bound leads to:

$$\mathcal{D}^m(\{S\colon \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \leq \sum_{h \in \mathcal{H}} \mathcal{D}^m(\{S\colon |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\})$$

  We now consider a generic element of the sum, and search for a upper bound. The idea is that $L_S(h)$ is an *estimator* of $L_{\mathcal{D}}(h)$ (it is a sample mean vs an expected value), and so it should be *closer to it* if we increase the sample size m. In fact:

$$L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)] \qquad L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i)$$

meaning that $L_{\mathcal{D}}(h) = \mathbb{E}_{z\sim\mathcal{D}}[L_S(h)]$. The gap between averages (over i.i.d. samples) and their expected value can be quantified through **Hoeffding inequality**.

Hoeffding inequality. Let $\theta_1, \dots, \theta_m$ be a sequence of i.i.d. random variables and assume that for all i, $\mathbb{E}[\theta_i] = \mu$ and $\mathbb{P}[a \le \theta_i \le b] = 1$ (i.e. the support of the probability distribution lies in $[a, b]$). Then, for any $\epsilon > 0$:

$$\mathbb{P}\left[\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \epsilon\right] \le 2\exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right)$$

In our case, this means that:

$$\mathcal{D}^m(\{S\colon |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \le 2\exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right)$$

We assume $\ell \in [0, 1]$, meaning that $b - a = 1$. Then, computing the sum:

$$\mathcal{D}^m(\{S\colon \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \le \sum_{h\in\mathcal{H}} 2\exp\bigl(-2m\epsilon^2\bigr) = 2|\mathcal{H}|\exp\bigl(-2m\epsilon^2\bigr)$$

For our thesis, we want the red term to be $< \delta$, and so we impose:

$$2|\mathcal{H}|\exp\bigl(-2m\epsilon^2\bigr) < \delta \Rightarrow m \ge \frac{1}{2\epsilon^2}\log\left(\frac{2|\mathcal{H}|}{\delta}\right)$$

So, let $\mathcal{H}$ be a finite hypothesis class, let Z be a domain, and let $\ell\colon \mathcal{H}\times Z \to [0, 1]$ be a loss function. Then $\mathcal{H}$ enjoys the uniform convergence property with sample complexity:

$$m_{\mathcal{H}}^{\mathrm{UC}}(\epsilon, \delta) \le \left\lceil \frac{1}{2\epsilon^2}\log\biggl(\frac{2|\mathcal{H}|}{\delta}\biggr)\right\rceil$$

(Note that previously we found a *general* sufficient condition for the minimum size of m needed for uniform convergence. In a specific case, it is probable that a much lower m would allow the same property.)

1.5 Bias-complexity tradeoff

- **Error decomposition**. The true risk $L_{\mathcal{D}}(h_S)$ of an output h_S of the $\mathrm{ERM}_{\mathcal{H}}$ algorithm is a combination of the *bias* contained in the choice of $\mathcal{H}$ (which could be not optimal) - the **approximation error** - and the inefficiency of the ERM algorithm in minimizing the true risk (as it minimizes the empirical risk) - the **estimation error**:

$$L_{\mathcal{D}}(h_S) = \epsilon_{\text{approximation}} + \epsilon_{\text{estimation}}$$
$$\epsilon_{\text{app}} = \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h); \qquad \epsilon_{\text{est}} = L_{\mathcal{D}}(h_S) - \epsilon_{\text{app}}$$

  - ϵ_{app} usually decreases when enlarging $|\mathcal{H}|$, and does not depend on m. If realizability holds, it is 0, otherwise it includes always the error of the Bayes optimal predictor.
  - ϵ_{est} increases logarithmically with $|\mathcal{H}|$ (because it is easier to overfit) and decreases with m.

$|\mathcal{H}|$ too large makes ϵ_{est} dominate (overfitting), while $|\mathcal{H}|$ too small makes ϵ_{app} dominate (underfitting).

1.5.1 Theorems

- **No free lunch**. Let A be any learning algorithm for the task of binary classification with respect to the $0-1$ loss over a domain $\mathcal{X}$. Let m be any number smaller than $|\mathcal{X}|/2$, representing a training set size (so the algorithm knows *less* than half the elements). Then, there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:

  1. There exists a function $f\colon \mathcal{X} -> \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$
  2. With probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.

- **No prior knowledge implies not PAC learnable**. Let $\mathcal{X}$ be an *infinite* domain set and let $\mathcal{H}$ be the set of **all** functions $\mathcal{X} \to \{0,1\}$ (no bias, no prior knowledge). Then $\mathcal{H}$ is not PAC learnable.

  Proof. Just pick $\epsilon < 1/8$ and $\delta < 1/7$, suppose (by absurdity) that $\mathcal{H}$ is PAC learnable, and apply the no-free-lunch theorem to get a contradiction (no amount of m, as $|\mathcal{H}| = \infty$ is $> |\mathcal{H}|/2$, and so with probability $> \delta > 1/7$, $L_{\mathcal{D}}(A(S)) > 1/8 > \epsilon$).

1.6 VC-Dimension

- **Restriction of a hypothesis class to a set of samples**. Let $\mathcal{H}$ be a class of functions $\mathcal{X} \to \{0,1\}$ and let $C = \{c_1, \ldots, c_m\} \subset \mathcal{X}$. The restriction of $\mathcal{H}$ to C is the set of functions $C \to \{0,1\}$ that can be derived from $\mathcal{H}$.

  We can identify these functions by their outcomes on the elements of C:

  $$\mathcal{H}_C = \{(h(c_1), \ldots, h(c_m))\colon h \in \mathcal{H}\}$$

  Note that if two h result in the same outcomes on C, then they are effectively *the same function* on the restricted class. So $\mathcal{H}_C$ is the *intersection* between the set of all functions $C \to \{0,1\}$ and $\mathcal{H}^{|C|}$.

- **Shattering**. A hypothesis class $\mathcal{H}$ shatters a finite set $C \subset \mathcal{X}$ if the restriction of $\mathcal{H}$ to C is the set of all functions $C \to \{0,1\}$. That is, $|\mathcal{H}_C| = 2^{|C|}$.

Example. Let $\mathcal{H}$ be the hypothesis class of threshold functions $h_a \colon \mathbb{R} \to \{0,1\}$:

$$\mathcal{H} \ni h_a(x) = \begin{cases} 1 & x > a \\ 0 & x \leq a \end{cases}$$

Consider $C = \{c_1\}$. If we take $a = c_1 - 1$, then $h_a(c_1) = 1$, but for $b = c_1 + 1$ we have $h_b(c_1) = 0$. As the set of all functions $C \to \{0,1\}$ is just $\{0,1\}$, this means that $\mathcal{H}$ shatters C.

However, consider $D = \{c_1, c_2\}$ with $c_1 < c_2$. Here the set of all functions $D \to \{0,1\}$ is $\{(0,0),(0,1),(1,0),(1,1)\}$. By correctly choosing a, we can construct only $\{0,0\}$, $\{1,1\}$ or $\{0,1\}$, meaning that $\mathcal{H}$ does not shatter D.

1.6.1 Theorems

- **Shattering and No-Free-Lunch**. Let $\mathcal{H}$ be a hypothesis class of functions $\mathcal{X} \to \{0,1\}$. Let m be a training set size. Assume that there exists a set $C \subset \mathcal{X}$ of double the size ($|C| = 2m$) that is shattered by $\mathcal{H}$. Then, for *any* learning algorithm A, there exist a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ and a predictor $h \in \mathcal{H}$ such that $L_{\mathcal{D}}(h) = 0$ but with probability of at least $1/7$ over the choice of $S \sim \mathcal{D}^m$ we have that $L_{\mathcal{D}}(A(S)) \geq 1/8$.

1.7 Linear predictors

- **Affine Functions**

$$L_d = \{h_{\boldsymbol{w},b} \colon \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \qquad h_{\boldsymbol{w},b}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \left(\sum_{i=1}^{d} w_i x_i \right) + b_i$$

An affine function returns the dot product of the argument $\boldsymbol{x}$ by a *weight vector* $\boldsymbol{w}$, and sums a *bias* b to the result. These can be written in a more compact way by defining $\boldsymbol{w}' = (b, \boldsymbol{w}) \in \mathbb{R}^{d+1}$ and $\boldsymbol{x}' = (1, \boldsymbol{x})$, so that:

$$h_{\boldsymbol{w},b} = \langle \boldsymbol{w}', \boldsymbol{x}' \rangle = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$$

So an affine function of dimension d can be written as a linear function in dimension $d+1$ (homogeneous coordinates).

- **Linear functions**. Just a dot product with a weight $\boldsymbol{w}$ vector:

$$L'_d = \{h_{\boldsymbol{w}} \colon \boldsymbol{w} \in \mathbb{R}^d\} \qquad h_{\boldsymbol{w}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$$

- **Linear predictor**. The hypothesis class $\mathcal{H}$ of linear predictors contains all the functions $\Phi \circ L_d$, where $\Phi \colon \mathbb{R} \to \mathcal{Y}$ "connects" the output of the affine functions in L_d to the desired set of labels $\mathcal{Y}$.

- **Halfspaces**. If $\mathcal{Y} = \{-1, +1\}$ (binary classification), and $\mathcal{X} = \mathbb{R}^d$, we define the class of halfspaces as follows:

$$HS_d = \text{sign} \circ L_d = \{\boldsymbol{x} \mapsto \text{sign}(h_{\boldsymbol{w},b}(\boldsymbol{x}))\colon h_{\boldsymbol{w},b} \in L_d\}$$

  That is, we classify an element $\boldsymbol{x} \in \mathcal{X}$ as $+1$ if $h_{\boldsymbol{w},b}(\boldsymbol{x}) > 0$, and to -1 otherwise. Geometrically, $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ is the hyperplane in $\mathbb{R}^d$ that is $\perp$ $\boldsymbol{w}$, and is $b/\|\boldsymbol{w}\|$ away from the origin. Consider $\boldsymbol{w}$ as an affine vector "starting" from $\boldsymbol{w}b/\|\boldsymbol{w}\|^2$. Then any point that lies at an acute angle with respect to $\boldsymbol{w}$ is classified as $+1$, and all the others as -1.

  The distance between a point $\boldsymbol{x_0}$ and an hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ can be computed by considering the vector $\boldsymbol{x_0} - \boldsymbol{X}$, where $\boldsymbol{X}$ is a generic point on the hyperplane (i.e. such that $\boldsymbol{X} \cdot \boldsymbol{w} + b = 0$), and projecting it on the perpendicular unit vector $\boldsymbol{w}/\|\boldsymbol{w}\|$, leading to:

$$\text{dist} = \left|(\boldsymbol{x_0} - \boldsymbol{X}) \cdot \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right|$$

  Then using $\boldsymbol{X} \cdot \boldsymbol{w} = -b$ (as it is in the hyperplane), leads to:

$$\text{dist} = \frac{|\boldsymbol{x}_0 \cdot \boldsymbol{w} + b|}{\|\boldsymbol{w}\|} \tag{1.5}$$

  The **realizability** assumption holds if the points to be classified are *linearly separable*, meaning that there exist an halfspace that perfectly separates the ones labelled as $+1$ from the other ones.

- **Linear regression**. Take $\mathcal{H}_{\text{reg}} = L_d$, with L_2 loss $\ell(h, (\boldsymbol{x}, y)) = (h(\boldsymbol{x}) - y)^2$. This leads to the Mean Squared Error as empirical risk:

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}(h(\boldsymbol{x_i}) - y_i)^2$$

- **Polynomial regression**. Find the one dimensional polynomial of degree n that better predicts the data. $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$. The idea is to *produce nonlinear features* with the mapping $\psi\colon \mathbb{R} \to \mathbb{R}^{n+1}$, $x \mapsto (1, x, x^2, \ldots, x^n) = \boldsymbol{\psi}(x)$, and then note that $p(x) = \langle \boldsymbol{a}, \boldsymbol{\psi}(x)\rangle$, and find $\boldsymbol{a}$ with least squares.

- **Logistic regression**. Choose $\mathcal{H}\colon \Phi_{\text{sig}} \circ L_d$, where $\Phi_{\text{sig}}\colon \mathbb{R} \to [0, 1]$ is the sigmoid function:

$$\Phi_{\text{sig}}(z) = \frac{1}{1 + e^{-z}}$$

  Output is probability of being in class $+1$.
  Use loss: $\ell(h_{\boldsymbol{w}}, (\boldsymbol{x}, y)) = \log\left(1 + e^{-y\langle \boldsymbol{w}, \boldsymbol{x}\rangle}\right)$. Note that $y\langle \boldsymbol{w}, \boldsymbol{x}\rangle > 0$ for correctly classified samples, meaning that in this case the loss is low.

1.7.1 Theorems

- **ERM for halfspaces is a linear program**. A linear program is a way to maximize a linear function subject to linear inequalities, that is finding $\boldsymbol{w}$ such that:

  $$\max_{\boldsymbol{w}\in\mathbb{R}^d}\langle \boldsymbol{u}, \boldsymbol{w}\rangle \ \wedge \ A\boldsymbol{w} \geq \boldsymbol{v}$$

  with A is a $m \times d$ matrix and $\boldsymbol{v} \in \mathbb{R}^m$, $\boldsymbol{u} \in \mathbb{R}^d$ are vectors.

  Assume a **realizable**, **homogeneous** case, meaning that we take $b = 0$ (hypotheses are from L'_d, or from L_{d-1} with homogeneous coordinates). Let $S = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^m$ be the training dataset. From realizability, $L_S(h_S) = 0$.

  We search for a $\boldsymbol{w} \in \mathbb{R}^d$ so that:

  $$\operatorname{sign}(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle) = y_i \qquad \forall i = 1, \ldots, m$$

  This is equivalent to:

  $$y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle > 0 \qquad \forall i = 1, \ldots, m$$

  In fact, if $y_i = +1$ then a correct $\boldsymbol{w}$ will lead to $\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle > 0$, and so $y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle > 0$. On the other hand, if $y_i = -1$, then we will have $\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle < 0$, and so again $y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle > 0$.

  We now show that there exist a $\bar{\boldsymbol{w}}$ so that $y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle \geq 1$, and that this inequality can be mapped to the LP case.

  We construct $\bar{\boldsymbol{w}}$ starting from any $\boldsymbol{w}^*$ for which $y_i\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle$ (which exists due to realizability). Then consider the minimum value of that expression:

  $$\gamma \equiv \min_{1\leq i\leq m} y_i\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle$$

  Then:

  $$y_i\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle \geq \gamma \Rightarrow y_i\langle \frac{\boldsymbol{w}^*}{\gamma}, \boldsymbol{x_i}\rangle \geq 1$$

  and so $\bar{\boldsymbol{w}} = \boldsymbol{w}^*/\gamma$. We can now rewrite the previous inequality in vector form:

  $$y_i\langle \bar{\boldsymbol{w}}, \boldsymbol{x_i}\rangle \geq 1 \Rightarrow y_i\sum_{j=1}^d w_j(\boldsymbol{x_i})_j = \sum_{j=1}^d y_i(\boldsymbol{x}_i)_j w_j \geq 1 \Rightarrow X\boldsymbol{w} \geq \mathbf{1}$$

  where X is the $m \times d$ matrix with $X_{ij} = y_i(\boldsymbol{x_i})_j$ ($(\boldsymbol{x_i})_j$ is the j-th component of the i-th sample $\boldsymbol{x_i}$). So, if we take $\boldsymbol{v} = (1, \ldots, 1) \in \mathbb{R}^d$ we got the linear inequality needed for the linear program. As any $\boldsymbol{w}$ that satisfy that constraint is a valid solution, we do not need to maximize any function, and so we can set $\boldsymbol{u} = \mathbf{0} \in \mathbb{R}^d$.

- **Perceptron algorithm for halfspaces**. Let $S = \{(\boldsymbol{x_i}, y_i)\}_{i=1,\dots,m}$. Initialize $\boldsymbol{w}^{(1)} = \boldsymbol{0}$. At every step t, pick a misclassified sample, that is a i such that $y_i\langle \boldsymbol{w}^{(t)}, \boldsymbol{x_i}\rangle \leq 0$. Then update:

  $$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + y_i \boldsymbol{x_i}$$

  Note that:

  $$y_i\langle \boldsymbol{w}^{(t+1)}, \boldsymbol{x_i}\rangle = y_i\langle \boldsymbol{w}^{(t)} + y_i\boldsymbol{x_i}, \boldsymbol{x_i}\rangle = \underbrace{y_i\langle \boldsymbol{w}^{(t)}, \boldsymbol{x_i}\rangle}_{\leq 0} + \|\boldsymbol{x_i}\|^2$$

  So now it is *closer* to being > 0 (correct). The algorithm stops when there are no more misclassified samples (in the realizable case).

  Stopping condition. Let $R = \max_i \|\boldsymbol{x_i}\|$ (maximum norm of samples), and $B = \min\{\|\boldsymbol{w}\| \colon \forall i \in [m], y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle \geq 1\}$ (minimum norm of a correct weight). Then the algorithm stops after at most $(RB)^2$ iterations.

  Proof. We show that if T is the minimum number of iterations needed to reach the stopping condition, then $T \leq (RB)^2$. Let $\boldsymbol{w}^*$ be the solution with the smaller norm, meaning that:

  $$y_i\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle \geq 1 \qquad \|\boldsymbol{w}^*\| = B \tag{1.6}$$

  We expect $\boldsymbol{w}^{(t)}$ to be pointing closer and closer to the direction of $\boldsymbol{w}^*$ as t increases. Recall that the cosine of the angle α between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is:

  $$1 \geq \cos(\alpha) = \frac{\langle \boldsymbol{a}, \boldsymbol{b}\rangle}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$$

  In our case we examine:

  $$1 \geq \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)}\rangle}{\|\boldsymbol{w}^*\|\left\|\boldsymbol{w}^{(t)}\right\|}$$

  We expect this quantity to *increase* at every iteration, reaching a certain threshold after T iterations, after which the algorithm stops. We want to know what is this threshold.

  Let's examine how $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)}\rangle$ changes at every iteration. We start from:

  $$\langle \boldsymbol{w}^*, \boldsymbol{w}^{(1)}\rangle = 0$$

  At iteration t, if the update sample is $(\boldsymbol{x_i}, y_i)$, that scalar product increases by:

  $$\begin{aligned}\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t+1)}\rangle - \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)}\rangle &= \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\rangle = \\ &= \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} + y_i\boldsymbol{x_i} - \boldsymbol{w}^{(t)}\rangle = \\ &= \langle \boldsymbol{w}^*, y_i\boldsymbol{x_i}\rangle = y_i\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle \underset{(1.6)}{\geq} 1\end{aligned}$$

So $\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle$ *increases* by more than 1 at every iteration, meaning that if the algorithm stops at T iterations, we have:

$$\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)} \rangle = \sum_{t=1}^{T} \left(\langle \boldsymbol{w}^*, \boldsymbol{w}^{(t+1)} \rangle - \langle \boldsymbol{w}^*, \boldsymbol{w}^{(t)} \rangle \right) \geq T$$

Let's now examine how $\left\| \boldsymbol{w}^{(t)} \right\|$ changes. If at iteration t the sample $(\boldsymbol{x_i}, y_i)$ is selected, then:

$$\begin{aligned} \left\| \boldsymbol{w}^{(t+1)} \right\|^2 &= \left\| \boldsymbol{w}^{(t)} + y_i \boldsymbol{x_i} \right\|^2 = \\ &= \left\| \boldsymbol{w}^{(t)} \right\|^2 + 2 \underbrace{y_i \langle \boldsymbol{w}^{(t)}, \boldsymbol{x_i} \rangle}_{\leq 0} + \underbrace{y_i^2}_{1} \left\| \boldsymbol{x_i} \right\|^2 \\ &\leq \left\| \boldsymbol{w}^{(t)} \right\|^2 + (\max_i \left\| \boldsymbol{x_i} \right\|)^2 = \left\| \boldsymbol{w}^{(t)} \right\|^2 + R^2 \end{aligned}$$

As $\left\| \boldsymbol{w}^{(1)} \right\| = 0$, then $\left\| \boldsymbol{w}^{(T+1)} \right\|^2 \leq TR^2$ and so $\left\| \boldsymbol{w}^{(T+1)} \right\| \leq \sqrt{T}R$. This means that:

$$1 \geq \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^{(T+1)} \rangle}{\left\| \boldsymbol{w}^* \right\| \left\| \boldsymbol{w}^{(T+1)} \right\|} \geq \frac{T}{\sqrt{T}RB} = \frac{\sqrt{T}}{RB} \Rightarrow T \leq (RB)^2$$

- **Least squares**. An algorithm that solves the ERM problem for linear regression with respect to the L_2 loss. It outputs the $\boldsymbol{w}$ weight that satisfies:

  $$\arg\min_{\boldsymbol{w} \in \mathbb{R}^d} L_S(h_{\boldsymbol{w}}) = \arg\min_{\boldsymbol{w}} \frac{1}{m} \sum_{i=1}^{m} (\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle - y_i)^2$$

  To solve this, compute the gradient of the empirical risk and set it to 0:

  $$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{w}} L &= \frac{2}{m} \sum_{i=1}^{m} (\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle - y_i) \boldsymbol{x_i} \overset{!}{=} \boldsymbol{0} \\ &= \frac{2}{m} \sum_{i=1}^{m} \boldsymbol{x_i} (\boldsymbol{x_i}^T \boldsymbol{w} - y_i) = \frac{2}{m} \Big[\underbrace{\left(\sum_{i=1}^{m} \boldsymbol{x_i} \boldsymbol{x_i}^T \right)}_{A} \boldsymbol{w} - \underbrace{\sum_{i=1}^{m} y_i \boldsymbol{x_i}}_{\boldsymbol{b}} \Big] = \frac{2}{m} [A\boldsymbol{w} - \boldsymbol{b}] = 0 \\ &\Rightarrow \boldsymbol{w} = A^{-1} \boldsymbol{b} \end{aligned}$$

  (assuming A is invertible).

- **Logistic regression and MLE**

1.8 Validation

- **Cross-validation**.

1.8.1 Theorems

- **Bound for model selection**. Let $\mathcal{H} = \{h_1, \dots, h_r\}$ be an arbitrary set of predictors and assume that the loss function is in $[0, 1]$. Assume that a validation set V of size m_v is sampled independent of $\mathcal{H}$. Then, with probability $\geq 1 - \delta$ over the choice of V we have:

$$\forall h \in \mathcal{H}, |L_{\mathcal{D}}(h) - L_V(h)| \leq \sqrt{\frac{\log(2|\mathcal{H}|/\delta)}{2m_v}}$$

1.9 Regularization

- **Regularized Loss Minimization** (RLM): learning paradigm that generalizes ERM to make it more stable, by adding a **regularization function** $R\colon \mathbb{R}^d \to \mathbb{R}$ (where d is the number of parameters to be optimizes) to the empirical risk:

$$\text{Select } h_S \text{ as } \arg\min_{\boldsymbol{w}}(L_S(\boldsymbol{w}) + R(\boldsymbol{w}))$$

- **Tikhonov Regularization**. Use L_2 norm for R:

$$R(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|^2 = \lambda \sum_{i=1}^{d} w_i^2 \qquad R\colon \mathbb{R}^d \to \mathbb{R}_+$$

  $\|\boldsymbol{w}\|^2$ measures the *complexity* of the hypothesis defined by $\boldsymbol{w}$, and λ controls the trade-off between *empirical risk* and *complexity*. Generally, a higher $\|\boldsymbol{w}\|^2$ can reach a lower L_S, at the risk of overfitting (high $L_{\mathcal{D}}(h)$).

- **Ridge regression**. Linear regression with L_2 loss (least squares) and Tikhonov Regularization

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}} \left(\lambda \|\boldsymbol{w}\|^2 + \frac{1}{2m} \sum_{i=1}^{m} (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - y_i)^2 \right)$$

  The 2 is added for convenience (it will go away when computing the gradient).

- **Small perturbation**. Let $S = (z_1, \dots, z_m)$ be a training set of m samples. A *small perturbation* of S is denoted as $S^{(i)}$, and corresponds substituting the i-th element of S with another arbitrary sample $z' \notin S$:

$$S^{(i)} = (z_1, \dots, z_{i-1}, z', z_{i+1}, \dots, z_m)$$

- **Overfitting**. Let A be a learning algorithm and S a training set. We say that A is overfitting if the difference $L_{\mathcal{D}}(A(S)) - L_S(A(S))$ is very large.

- **On-Average-Replace-One-Stable** (OAROS). Intuitively, an algorithm is stable if by replacing an element in S it's prediction does not change much, that is if $\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)$ is small (note that $A(S^{(i)}))$ does not *see* z_i in training, as $z_i \notin S^{(i)}$, and so it will be likely higher). Moreover, we expect this difference to *decrease* with increasing m, because the perturbation becomes "diluted" in a large S (i.e. changing a single sample on a large S should have a negligible impact).

So, we say that an algorithm is OAROS if that difference is bounded by a decreasing function in m. More precisely, let $\epsilon\colon \mathbb{N} \to \mathbb{R}$ be a monotonically decreasing function. We say that a learning algorithm A is *on-average-replace-one-stable* with rate $\epsilon(m)$ if, for every distribution $\mathcal{D}$:

$$\mathop{\mathbb{E}}_{(S,z')\sim\mathcal{D}^{m+1}, i\sim U(m)} [\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)] \leq \epsilon(m)$$

Here we take the expected value over $S \sim \mathcal{D}^m$ and $z' \sim \mathcal{D}$, choosing uniformly the index i to *perturb*.

- **Convex functions**. Recall that a function $f\colon \mathbb{R}^d \to \mathbb{R}$ is *convex* if it stays *under* every segment drawn between two points of its graph:

  $$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha g(\boldsymbol{x}) + (1-\alpha) g(\boldsymbol{y}) \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d, \alpha \in [0,1] \quad (1.7)$$

  The right hand side is the *segment* joining $(\boldsymbol{x}, f(\boldsymbol{x}))$ and $(\boldsymbol{y}, f(\boldsymbol{y}))$. Equivalently, a convex function is greater than its tangents:

  $$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x})$$

- **Strongly convex function**. A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is λ-strongly convex if it remains convex even after subtracting a quadratic term:

  $$g(\boldsymbol{x}) = f(\boldsymbol{x}) - \frac{\lambda}{2}\|\boldsymbol{x}\|^2 \text{ is convex} \quad (1.8)$$

  That is:

  $$\begin{aligned} &g(\boldsymbol{y}) \geq g(\boldsymbol{x}) + \boldsymbol{\nabla} g(\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x}) \\ \Rightarrow\ & f(\boldsymbol{y}) - \frac{\lambda}{2}\|\boldsymbol{y}\|^2 \geq f(\boldsymbol{x}) - \frac{\lambda}{2}\|\boldsymbol{x}\|^2 + (\boldsymbol{\nabla} f(\boldsymbol{x}) - \lambda\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x}) \\ \Rightarrow\ & f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x}) + \frac{\lambda}{2}\|\boldsymbol{y}\|^2 - \frac{\lambda}{2}\|\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|^2 - \lambda\boldsymbol{x}\cdot\boldsymbol{y} \\ \Rightarrow\ & f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x}) \cdot (\boldsymbol{y} - \boldsymbol{x}) + \frac{\lambda}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 \qquad (1.9) \end{aligned}$$

  In the one-dimensional case this means that $f''(x) \geq \lambda$. In fact, by comparison with the second-order Taylor expansion with the Lagrange remainder we get:

  $$\begin{aligned} f(y) &= f(x) + f'(x)(y-x) + \frac{1}{2}f''(x)(y-x)^2 + \frac{1}{3!}f^{(3)}(z)(y-x)^3 \qquad z \in (x,y) \\ &\geq f(x) + f'(x)(y-x) + \frac{\lambda}{2}(y-x)^2 \\ &\Rightarrow f''(x) + \frac{1}{3!\cdot 2}f^{(3)}(z)(y-x) \geq \lambda \xrightarrow[y\to x]{} f''(x) \geq \lambda \end{aligned}$$

This generalizes to $\operatorname{Hess} f - \lambda\mathbb{I}$ is positive definite.

Applying (1.7) to (1.8) leads to an equivalent definition:

$$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \le \alpha f(\boldsymbol{x}) + (1-\alpha)f(\boldsymbol{y}) - \frac{\lambda}{2}\alpha(1-\alpha)\|\boldsymbol{x}-\boldsymbol{y}\|^2 \quad \forall \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d, \alpha \in [0,1] \tag{1.10}$$

- **Lipschitzness**. Let $C \subset \mathbb{R}^d$. A function $f\colon \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over C if:

$$\|f(\boldsymbol{w_1}) - f(\boldsymbol{w_2})\| \le \rho\|\boldsymbol{w_1} - \boldsymbol{w_2}\|$$

If f is differentiable, this means that its derivative is bounded by ρ.

- **Fitting-Stability tradeoff**. The expected risk of a learning algorithm can be written as:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S))] = \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_S(A(S))] + \underbrace{\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))]}_{\le\epsilon(m)}$$

where $\epsilon(m)$ is the bounding function for an OAROS algorithm, that, in the case of RLM is:

$$\epsilon(m) = \frac{2\rho^2}{\lambda m}$$

So increasing λ decreases the overfitting, but at the same time reduces the relative importance of $L_S(A(S))$ in the optimization, meaning that $L_S(A(S))$ will be larger. The following bound can be proven:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S))] \le L_{\mathcal{D}}(\boldsymbol{w}^*) + \lambda\|\boldsymbol{w}^*\|^2 + \frac{2\rho^2}{\lambda m}$$

where $\boldsymbol{w}^*$ is an hypothesis with low risk.

1.9.1 Theorems

- **Ridge regression solution**. The function being optimized is:

$$f(\boldsymbol{w}, S) = \lambda\|\boldsymbol{w}\|^2 + \frac{1}{2m}\sum_{i=1}^{m}(\langle\boldsymbol{w},\boldsymbol{x_i}\rangle - y_i)^2$$

Computing the gradient and setting it to 0:

$$\begin{aligned}
\nabla_{\boldsymbol{w}} f(\boldsymbol{w}, S) &= 2\lambda\boldsymbol{w} + \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x_i}(\langle\boldsymbol{w},\boldsymbol{x_i}\rangle - y_i) \overset{!}{=} \boldsymbol{0} \\
&= 2\lambda\boldsymbol{w} + \frac{1}{m}\underbrace{\left(\sum_{i=1}^{m}\boldsymbol{x_i}\boldsymbol{x_i}^T\right)}_{A}\boldsymbol{w} - \frac{1}{m}\underbrace{\left(\sum_{i=1}^{m}y_i\boldsymbol{x_i}\right)}_{\boldsymbol{b}} = 0 \\
&= 2\lambda\boldsymbol{w} + \frac{1}{m}(A\boldsymbol{w}) - \frac{1}{m}\boldsymbol{b} = \cancel{\frac{1}{m}}(2\lambda m\mathbb{I}_d + A)\boldsymbol{w} - \cancel{\frac{1}{m}}\boldsymbol{b} = 0 \\
&\Rightarrow \boldsymbol{w} = (2\lambda m\mathbb{I}_d + A)^{-1}\boldsymbol{b}
\end{aligned}$$

- **OAROS and overfitting**. Let $\mathcal{D}$ be a distribution, and $S = (z_1, \dots, m)$ an i.i.d. sequence of examples and let z' be another i.i.d. example. Let $U(m)$ be the uniform distribution over $[m]$. Then, for any learning algorithm:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] = \mathop{\mathbb{E}}_{(S,z')\sim\mathcal{D}^{m+1}, i\sim U(m)}[\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)] \tag{1.11}$$

  That is, the *response* from a perturbation tells us the *distance* between empirical and true risk.

  Proof. Recall that the true risk of $A(S)$ is the expected value of the loss over $\mathcal{D}$. So:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S))] = \mathop{\mathbb{E}}_{(S,z')\sim\mathcal{D}^{m+1}}[\ell(A(S), z')] = \mathop{\mathbb{E}}_{(S,z_i)\sim\mathcal{D}^{m+1}}[\ell(A(S^{(i)}), z_i)]$$

  Note that here we evaluate the loss over samples that are not in the training set ($z' \notin S$, $z_i \notin S^{(i)}$). If we were to use samples *from the training set* we would get the expectation of the empirical risk:

$$\mathbb{E}_S[L_S(A(S))] = \mathop{\mathbb{E}}_{S\sim\mathcal{D}^m, i\sim U(m)}[\ell(A(S), z_i)]$$

  Substituting in the left hand side of (1.11) and using the linearity of $\mathbb{E}$ we get the thesis.

- **Strongly convex functions**. The following conditions hold:

  1. The function $f(\boldsymbol{w}) = \lambda\|\boldsymbol{w}\|^2$ is 2λ-strongly convex.
  2. If f is λ-strongly convex and g is convex, then $f+g$ is λ-strongly convex.
  3. If f is λ-strongly convex and $\boldsymbol{u}$ minimizes f, then:

$$f(\boldsymbol{w}) - f(\boldsymbol{u}) \geq \frac{\lambda}{2}\|\boldsymbol{w} - \boldsymbol{u}\|^2$$

  Proof. For (1) and (2) just use the definition (1.10), and for (3) the definition (1.9) and the fact that $\boldsymbol{\nabla} f(\boldsymbol{u}) = \boldsymbol{0}$ by hypothesis.

- **RLM is stable**. The RLM rule is:

$$A(S) = \arg\min_{\boldsymbol{w}}(L_S(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|^2)$$

  Assuming that L_S is convex, then as $\lambda\|\boldsymbol{w}\|^2$ is 2λ-strongly convex:

$$f_S(\boldsymbol{w}) = L_S(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\| \text{ is } 2\lambda\text{-strongly convex}$$

  $A(S)$ by definition minimizes f_S, and so:

$$f_S(\boldsymbol{v}) - f_S(A(S)) \geq \lambda\|\boldsymbol{v} - A(S)\|^2 \qquad \forall \boldsymbol{v} \in \mathbb{R}^d$$

Then note that:

$$f_S(\boldsymbol{v}) - f_S(\boldsymbol{u}) = L_S(\boldsymbol{v}) + \lambda\|\boldsymbol{v}\|^2 - (L_S(\boldsymbol{u}) + \lambda\|\boldsymbol{u}\|^2)$$

Recall that $S^{(i)}$ is obtained from S by removing z_i and adding z' in its place, meaning that:

$$L_S(\boldsymbol{v}) = L_{S^{(i)}}(\boldsymbol{v}) + \frac{\ell(\boldsymbol{v}, z_i) - \ell(\boldsymbol{v}, z')}{m}$$

And so, if $\boldsymbol{u}$ minimizes f_S:

$$f_S(\boldsymbol{v}) - f_S(\boldsymbol{u}) = L_{S^{(i)}}(\boldsymbol{v}) + \lambda\|\boldsymbol{v}\|^2 - (L_{S^{(i)}}(\boldsymbol{u}) - \lambda\|\boldsymbol{u}\|^2) + \\ + \frac{\ell(\boldsymbol{v}, z_i) - \ell(\boldsymbol{u}, z_i)}{m} + \frac{\ell(\boldsymbol{u}, z') - \ell(\boldsymbol{v}, z')}{m} \geq \lambda\|\boldsymbol{v} - A(S)\|^2$$

Let $\boldsymbol{v} = A(S^{(i)})$ and $\boldsymbol{u} = A(S)$, leading to:

$$\underbrace{f_S(A(S^{(i)})) - f_S(A(S))}_{A} = \underbrace{L_{S^{(i)}}(A(S^{(i)})) + \lambda\left\|A(S^{(i)})\right\|^2}_{f_{S^{(i)}}(A(S^{(i)}))} - \underbrace{(L_{S^{(i)}}(A(S)) - \lambda\|A(S)\|^2)}_{f_{S^{(i)}}(A(S))} + \\ \underbrace{+ \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{m} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{m}}_{B}$$

Note that $A(S^{(i)})$ minimizes $f_{S^{(i)}}$ by definition, and so $f_{S^{(i)}}(A(S^{(i)})) \leq f_{S^{(i)}}(A(S))$, meaning that the red term is < 0, i.e. $= -|\epsilon|$. So we have $A = B - |\epsilon$, implying $A \leq B$:

$$f_S(A(S^{(i)})) - f_S(A(S)) \leq \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{m} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{m}$$

And from before we have:

$$\lambda\left\|A(S^{(i)}) - A(S)\right\|^2 \leq f_S(A(S^{(i)})) - f_S(A(S))$$

Meaning that:

$$\lambda\left\|A(S^{(i)}) - A(S)\right\|^2 \leq \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{m} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{m} \tag{1.12}$$

Suppose that the loss as a function of the model ($h \mapsto \ell(h, z_i)$) is ρ-Lipshitz. This means that the loss over a sample *outside* the training dataset can't grow too much compared to that of an element inside the sample:

$$\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i) \leq \rho\left\|A(S^{(i)}) - A(S)\right\|$$

Similarly, as $z' \notin S$, but $z' \in S^{(i)}$:

$$\ell(A(S), z') - \ell(A(S^{(i)}), z') \leq \rho \left\| A(S^{(i)}) - A(S) \right\|$$

Substituting in (1.12):

$$\lambda \left\| A(S^{(i)}) - A(S) \right\|^2 \leq \frac{2\rho \left\| A(S^{(i)}) - A(S) \right\|}{m}$$

Rearranging:

$$\left\| A(S^{(i)}) - A(S) \right\| \leq \frac{2\rho}{m\lambda}$$

And so:

$$\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i) \leq \frac{2\rho^2}{m\lambda}$$

Taking the expected value, this is equivalent to:

$$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(A(S)) - L_S(A(S))] \leq \frac{2\rho^2}{\lambda m}$$

1.10 Gradient Descent

- Find minimum of $f\colon \mathbb{R}^d \to \mathbb{R}$ by moving in the opposite direction of the gradient. Start with $\boldsymbol{w}^{(\mathbf{0})} = \mathbf{0} \in \mathbb{R}_d$, and then apply:

  $$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \nabla f(\boldsymbol{w}^{(t)})$$

  where $\epsilon \in \mathbb{R}$ is the learning rate.

- If f is a **convex** ρ**-Lipschitz** function, and the minimization domain is $\|\boldsymbol{w}\| \leq B$, then after T steps with $\eta = \sqrt{B^2/(\rho^2 T)}$ GD produces a $\bar{\boldsymbol{w}}$ so that:

  $$f(\bar{\boldsymbol{w}}) - f(\boldsymbol{w}^*) \leq \frac{B\rho}{T}$$

  where $\boldsymbol{w}^*$ is the minimum point.

- **Cons:** Needs all the training set at each iteration (high computational time)

- **Pros:** stable.

1.11 Stochastic Gradient Descent

- Instead of computing $\nabla f(\boldsymbol{w})$, pick a random $\boldsymbol{v_t}$ so that $\mathbb{E}[\boldsymbol{v_t}|\boldsymbol{w}^{(t)}] = \nabla f(\boldsymbol{w}^{(t)})$ (i.e. with expected value equal to the real gradient of f). Then apply the rule as before:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \boldsymbol{v_t}$$

- Can be used to minimize directly $L_{\mathcal{D}}$ (true risk). Recall that:

$$L_{\mathcal{D}}(\boldsymbol{w}) = \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\ell(\boldsymbol{w}, z)]$$

  And so:

$$\mathbb{E}[\boldsymbol{v_t}|\boldsymbol{w}^{(t)}] = \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\nabla\ell(\boldsymbol{w}^{(t)}, z)] = \nabla \mathop{\mathbb{E}}_{z\sim\mathcal{D}}[\ell(\boldsymbol{w}^{(t)}, z)] = \nabla L_{\mathcal{D}}(\boldsymbol{w}^{(t)})$$

- **Pros**: Faster, can jump out local minima
- **Cons**: Noisy (can be alleviated with adaptive step size)
- **In the case of regularization**. We want to minimize:

$$f(\boldsymbol{w}) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + L_S(\boldsymbol{w})$$

  At every step, choose $\boldsymbol{v_t} = \nabla\ell(\boldsymbol{w}^{(t)}, z)$ for some $z \sim \mathcal{D}$. Then the full gradient of f is $\lambda\boldsymbol{w}^{(t)} + \boldsymbol{v_t}$. Choosing $\eta = 1/(\lambda t)$ (useful for 2λ-strongly convex functions) leads to:

$$\begin{aligned}
\boldsymbol{w}^{(t+1)} &= \boldsymbol{w}^{(t)} - \frac{1}{\lambda t}\left(\lambda\boldsymbol{w}^{(t)} + \boldsymbol{v_t}\right) = \\
&= \left(1 - \frac{1}{t}\right)\boldsymbol{w}^{(t)} - \frac{1}{\lambda t}\boldsymbol{v_t} = \\
&= \frac{t-1}{t}\boldsymbol{w}^{(t)} - \frac{1}{\lambda t}\boldsymbol{v_t} = \\
&\underset{(a)}{=} \frac{t-1}{t}\left(\frac{t-2}{t-1}\boldsymbol{w}^{(t-1)} - \frac{1}{\lambda(t-1)}\boldsymbol{v_t}\right) - \frac{1}{\lambda t}\boldsymbol{v_t} = \\
&= \frac{t-1}{t}\frac{t-2}{t-1}\boldsymbol{w}^{(t-1)} - \frac{1}{\lambda t}(\boldsymbol{v_t} + \boldsymbol{v_{t-1}}) = \\
&= \frac{(t-1)!}{t!}\underbrace{\boldsymbol{w}^{(0)}}_{0} - \frac{1}{\lambda t}(\boldsymbol{v_t} + \cdots + \boldsymbol{v_0}) = -\frac{1}{\lambda t}\sum_{i=1}^{t}\boldsymbol{v_i}
\end{aligned}$$

  where in (a) we reiterated the last row with $t \to t-1$ to express $\boldsymbol{w}^{(t)}$.

2 SVM

- **Hard-SVM**. Consider a **separable** training set. The algorithm picks the ERM solution with the largest *margin*, i.e. the one where the halfspace boundary is furthest away from the samples. Let $\boldsymbol{w}$ be the unit vector $\perp$ to the separating hyperplane. The distance between a sample $\boldsymbol{x_i}$ and the hyperplane is $|\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b|$ (1.5). The margin is the *minimum* distance between a sample and the hyperplane:

$$\text{Margin} = \min_{i \in [m]} |\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b|$$

The hard-SVM problem chooses $(\boldsymbol{w}, b)$ so to maximize the margin (while still classificating correctly all the examples), i.e.:

$$\underset{(\boldsymbol{w},b)\colon \|\boldsymbol{w}\|=1}{\text{argmax}} \min_{i \in [m]} |\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b| \ \text{ such that } \forall i,\ y_i(\langle \boldsymbol{w}, \boldsymbol{x_i} \rangle + b) > 0 \tag{2.1}$$

In the separable case, if $\boldsymbol{w}^*$ is a solution, note that:

$$|\langle \boldsymbol{w}^*, \boldsymbol{x_i} \rangle + b| = y_i(\langle \boldsymbol{w}^*, \boldsymbol{x_i} + b \rangle) \tag{2.2}$$

And so the *constraint* can be integrated in the optimization:

$$\underset{(\boldsymbol{w},b)\colon \|\boldsymbol{w}\|=1}{\text{argmax}} \min_{i \in [m]} y_i(\langle \boldsymbol{w}, \boldsymbol{x_i} + b \rangle) \tag{2.3}$$

In fact, by separability we know that there exist a $\boldsymbol{w}^*$ that satisfies $\forall i, y_i(\langle \boldsymbol{w}^*, \boldsymbol{x_i} \rangle + b) > 0$, and in particular:

$$\min_{i \in [m]} y_i(\langle \boldsymbol{w}^*, \boldsymbol{x_i} \rangle + b) > 0$$

Then:

$$\max_{(\boldsymbol{w},b)\colon \|\boldsymbol{w}=1\|} \min_{i \in [m]} y_i(\langle \boldsymbol{w}, \boldsymbol{x_i} + b \rangle) \geq \min_{i \in [m]} y_i(\langle \boldsymbol{w}^*, \boldsymbol{x_i} \rangle + b) > 0$$

Meaning that (2.3) indeed generates a $\boldsymbol{w}$ that separates the data, justifying the use of (2.2) and thus the equivalence with (2.1).

The hard-SVM problem is also equivalent (see theorem below) to:

$$\boldsymbol{w_0} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\|^2 \text{ s.t. } \forall i, y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b_i) \geq 1$$

We can always use homogeneous coordinates, including the bias in $\boldsymbol{w}$, arriving to:

$$\boldsymbol{w_0} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{w}\|^2 \text{ s.t. } \forall i, y_i\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle \geq 1$$

There is some difference as now the bias appears in the $\|\boldsymbol{w}\|^2$, which proves to be not much significant in practice.

- **Soft SVM**. Allow some violation of the inequalities $y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) \geq 1$, by instead requiring:

  $$y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) \geq 1 - \xi_i$$

  The $\boldsymbol{\xi}$ are called *slack variables*, and measure how much each inequality is violated (i.e. how much misclassified samples are far from the separating hyperplane). We now minimize both $\|\boldsymbol{w}\|^2$ (as before) and the average of $\boldsymbol{\xi}$ (reduce the average violation), controlling the trade-off between the two with a λ hyper-parameter:

  $$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \left(\lambda\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^m \xi_i\right) \text{ s.t. } \forall i,\ y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

  This can be rewritten using the **hinge loss**:

  $$\ell^{\text{hinge}}((\boldsymbol{w}, b), (\boldsymbol{x}, y)) = \max(0, 1 - y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b))$$

  For a correctly classified sample, $y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b) \geq 1$, and so the loss is 0. If the sample is misclassified it is > 0, and increases with the distance from the hyperplane. Note that $\xi_i = \ell^{\text{hinge}}((\boldsymbol{w}, b), (\boldsymbol{x_i}, y_i))$ (to prove this, fix $\boldsymbol{w}, \boldsymbol{x}$ and i in the soft-svm rule and minimize for ξ_i) and so:

  $$\min_{\boldsymbol{w},b} \Big(\lambda\|\boldsymbol{w}\|^2 + \underbrace{\frac{1}{m}\sum_{i=1}^m \ell^{\text{hinge}}((\boldsymbol{w}, b), (\boldsymbol{x_i}, y_i))}_{L_S^{\text{hinge}}(\boldsymbol{w},b)}\Big)$$

2.1 Theorems

- **Hard-SVM is equivalent to a quadratic program**. A quadratic program solves an optimization problem in which the objective is a convex quadratic function and the constraints are linear inequalities.

  Start from the Hard-SVM formulation in the separable case:

  $$\underset{(\boldsymbol{w},b)\colon \|\boldsymbol{w}\|=1}{\text{argmax}} \min_{i\in[m]} y_i(\langle \boldsymbol{w}, \boldsymbol{x_i}\rangle + b)$$

Let $(\boldsymbol{w}^*, b^*)$ be a solution, and let γ^* be the margin of that solution, i.e.:

$$\gamma^* = \min_{i\in[m]} y_i(\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle + b^*) \Rightarrow y_i(\langle \boldsymbol{w}^*, \boldsymbol{x_i}\rangle + b) \geq \gamma^*$$

And dividing by γ^*:

$$y_i\left(\langle \frac{\boldsymbol{w}^*}{\gamma^*}, \boldsymbol{x_i}\rangle + \frac{b^*}{\gamma^*}\right) \geq 1$$

So, the pair $(\boldsymbol{w}^*, b^*)/\gamma^*$ is one of the solutions in the constraints of quadratic programming, which solves:

$$(\boldsymbol{w_0}, b_0) = \arg\min_{(\boldsymbol{w},b)} \|\boldsymbol{w}\|^2 \text{ s.t. } \forall i,\ y_i(\langle \boldsymbol{w}, \boldsymbol{x_i} + b\rangle) \geq 1$$

As $\|\boldsymbol{w}\|^2$ is minimized, the final solution will have $\|\boldsymbol{w_0}\| \leq \|\boldsymbol{w}^*\|/\gamma^* = 1/\gamma^*$.

To prove the equivalence, we need also to show that any normalized output $\hat{\boldsymbol{w}} = \boldsymbol{w_0}/\|\boldsymbol{w_0}\|$, $\hat{b} = b_0/\|\boldsymbol{w_0}\|$ of the quadratic problem is a solution. To do this, we evaluate the main condition:

$$y_i(\langle \hat{\boldsymbol{w}}, \boldsymbol{x_i}\rangle + \hat{b}) = \frac{1}{\|\boldsymbol{w_0}\|} \underbrace{y_i(\langle \boldsymbol{w_0}, \boldsymbol{x_i}\rangle + b_0)}_{\geq 1} \geq \frac{1}{\|\boldsymbol{w_0}\|} \geq \gamma^*$$

Which proves that it is indeed a solution.

Note that the quadratic paradigm is just a linear predictor with "regularization".

- **SGD for Soft SVM**. Recall the hinge loss:

$$\ell^{\text{hinge}}(\boldsymbol{w}, (\boldsymbol{x}, y)) = \max\{0, 1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle\}$$

(Here we assume the homogeneous case). The gradient becomes:

$$\boldsymbol{v} = \begin{cases} \boldsymbol{0} & (1 - y\langle \boldsymbol{w}, \boldsymbol{x}) \leq 1 \\ -y\boldsymbol{x} & \text{otherwise} \end{cases}$$

For the update rule we can use the one found for λ-strongly convex functions (i.e. with regularization):

$$\boldsymbol{w}^{(t+1)} = -\frac{1}{\lambda t}\sum_{j=1}^{t} \boldsymbol{v}^{(t)}$$

- **Duality**.

2.2 Clustering

- **Clustering**: Divide a set of objects (N-dimensional vectors) into groups (clusters), such that *similar objects end up in the same group* and *dissimilar objects are separated into different groups.*

  There are two main problems:

  - These two requirements may **contradict** each other, because similarity is not transitive (imagine a chain of similar elements, where x_i is similar to x_{i+1}, but the first element x_0 is very different from the last x_n). In such cases, one of the two objectives *dominates* over the other.
  - In general there is no unique solution (ground truth) to the problem: several alternatives are acceptable following different *implicit* notions of similarity. This means that it's difficult to **evaluate the performance**

- **Clustering Model**. Let's formalize the clustering problem and introduce a common notation.

  - **Input**: *set of elements to be grouped* $\mathcal{X}$, and a *distance function* $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ (symmetric $d(x,y) = d(y,x)$, definite positive $d(x,y) \geq 0$, $d(x,x) = 0$, satisfies the triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$).
  - **Output**: a *partition* of $\mathcal{X}$ into k disjoint clusters: $C = (C_1, C_2, \ldots, C_k)$ such that $\cup_{i=1}^{k} C_i = \mathcal{X}$, $\forall i \neq j\colon C_i \cap C_j = \varnothing$.

- **Linkage-based clustering**. All points are initially clusters. At every step, merge two clusters A and B that are the closest according to some condition, i.e. minimize $D(A,B)$. Some examples of D:

  - **Single linkage**: minimum distance between an element of A and one of B:

    $$D(A,B) = \min\{d(\boldsymbol{x}, \boldsymbol{x'})\colon \boldsymbol{x} \in A, \boldsymbol{x'} \in B\}$$

  - **Max linkage**: maximum distance between an element of A and one of B:

    $$D(A,B) = \max\{d(\boldsymbol{x}, \boldsymbol{x'})\colon \boldsymbol{x} \in A, \boldsymbol{x'} \in B\}$$

  - **Average linkage**: average distance between elements of A and elements of B:

    $$D(A,B) = \frac{1}{|A||B|} \sum_{\boldsymbol{x} \in A} \sum_{\boldsymbol{x'} \in B} d(\boldsymbol{x}, \boldsymbol{x'})$$

  Reiterate until a *terminating condition* (e.g. k clusters, all clusters are $> r$ apart, all points are a cluster (output the total tree - dendrogram))

- **Cost-based clustering**. Define an objective function $G\colon (\mathcal{X}, d), C \mapsto \mathbb{R}_+$, such that it evaluates a clustering C of $\mathcal{X}$ using the distance d.

- **K-Means**. The k-means objective function measures the squared distance between each point in $\mathcal{X}$ to the centroid of its cluster (assume that $\mathcal{X} \subseteq \mathcal{X}'$, with $(\mathcal{X}', d)$ a metric space, otherwise it would not make sense to consider centroids that are $\notin \mathcal{X}$). The centroid $\mu_i(C_i)$ of C_i is defined as:

$$\boldsymbol{\mu_i}(C_i) = \arg\min_{\boldsymbol{\mu} \in \mathcal{X}'} \sum_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \boldsymbol{\mu})^2 = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

That is, it's the point $\in \mathcal{X}'$ that lies at the minimum square distance from all the other points in C_i.

The *loss* of every cluster is the value at that minimum, and the objective is the sum of that loss over all the clusters:

$$G_{\text{k-means}}((\mathcal{X}, d), (C_1, \dots, C_k)) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \boldsymbol{\mu_i}(C_i))^2$$

Algorithm.

1. Select k random centroids, each representing a cluster.
2. Assigning each point to the closest centroid

$$\forall i\colon C_i = \{\boldsymbol{x} \in \mathcal{X} \colon i = \arg\min_{j \in [k]} \left\|\boldsymbol{x} - \boldsymbol{\mu_j}\right\|\}$$

3. Compute new centroids for the newly created clusters:

$$\forall i\colon \boldsymbol{\mu_i} = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

4. Reiterate 2-3 until convergence (e.g. if ΔG is lower than a certain threshold).

2.2.1 Theorems

- **Each iteration of k-means does not increase the objective function**. Fix $\mathcal{X}$ and $d = \|\cdot, \cdot\|^2$. The objective function of k-means becomes:

$$G(C_1, \dots, C_k) = \min_{\boldsymbol{\mu_1}, \dots, \boldsymbol{\mu_k} \in \mathbb{R}^n} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu_i}\|^2$$

Note that:

$$\min_{\boldsymbol{\mu_i} \in \mathbb{R}^n} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu_i}\|^2 = \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}(C_i)\|^2 \qquad \boldsymbol{\mu}(C_i) = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$$

Denote the partition at step t with $C_i^{(t)}$. So we have:

$$G(C_1^{(t)}, \dots, C_k^{(t)}) = \min_{\boldsymbol{\mu_1}, \dots, \boldsymbol{\mu_k} \in \mathbb{R}^n} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu_i}^{(t)}\right\|^2 \leq \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu_i}^{(t-1)}\right\|^2$$

By definition of minimum. The new partition $C_i^{(t)}$ is chosen by assigning each points to the closest $\boldsymbol{\mu}_{\boldsymbol{i}}^{(t-1)}$, meaning that:

$$\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i^{(t)}}\left\|\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{i}}^{(t-1)}\right\|^2=\min_{\{C_i\}}\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i}\left\|\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{i}}^{(t-1)}\right\|^2\leq\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i^{(t-1)}}\left\|\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{i}}^{(t-1)}\right\|^2$$

Putting it all together:

$$G(C_1^{(t)},\dots,C_k^{(t)})\leq\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i^{(t-1)}}\left\|\boldsymbol{x}-\boldsymbol{\mu}_{\boldsymbol{i}}^{(t-1)}\right\|^2=G(C_1^{(t-1)},\dots,C_k^{(t-1)})$$

2.3 PCA

- **Principal Component Analysis**. Let $\boldsymbol{x_1},\dots,\boldsymbol{x_m}$ be vectors in $\mathbb{R}^d$. The idea is to convert them to a *lower dimensionality representation* and then back again, trying to lose the least information possible.

  Consider a $n\times d$ matrix W with $n<d$ (compression matrix), representing a linear transformation $\boldsymbol{x}\mapsto W\boldsymbol{x}$, so that $W\boldsymbol{x}\in\mathbb{R}^n$ is the lower dimensionality representation of $\boldsymbol{x}$. Then consider a *recovering matrix* $d\times n$ U that *recovers* the vectors, i.e. $\mathbb{R}^n\ni\boldsymbol{y}\mapsto U\boldsymbol{y}\in\mathbb{R}^d$. We want to choose W and U so that $\boldsymbol{x_i}$ and the *compressed-recovered* vector $UW\boldsymbol{x_i}$ are close, i.e.:

$$\operatorname*{arg\,min}_{W\in\mathcal{M}_{n\times d}(\mathbb{R}),U\in\mathcal{M}_{d\times n}(\mathbb{R})}\sum_{i=1}^{m}\|\boldsymbol{x_i}-UW\boldsymbol{x_i}\|^2 \tag{2.4}$$

  There exists a solution (U^*,W^*) such that U^* is orthogonal ($(U^*)^TU^*=\mathbb{I}$) and $W^*=(U^*)^T$. To see this, let (U,V) be generic. Then note that $\boldsymbol{x}\mapsto W\boldsymbol{x}\in\mathbb{R}^n$, and so the range R of $U(W\boldsymbol{x})$ is "$U\mathbb{R}^n$", which is a n-dimensional linear subspace of $\mathbb{R}^d$. So $UW\boldsymbol{x}\in R\ \forall\boldsymbol{x}\in\mathbb{R}^d$.

  Now choose a ON basis of R, which consists of n vectors of d dimension, that can be arranged as columns in a matrix V $d\times n$, with $V^TV=\mathbb{I}$. We can write any vector $\boldsymbol{y}\in\mathbb{R}^n$ "in coordinates" with respect to that basis, i.e. $\forall\boldsymbol{r}\in R$, $\exists\boldsymbol{y}\in\mathbb{R}^n$ s.t. $\boldsymbol{r}=V\boldsymbol{y}$, where $\boldsymbol{y}$ are the *coordinates* of $\boldsymbol{r}$ in the V basis. Now pick any $\boldsymbol{x}\in\mathbb{R}^d$ and consider its distance with an arbitrary element $V\boldsymbol{y}$ of R:

$$\|\boldsymbol{x}-V\boldsymbol{y}\|^2=\|\boldsymbol{x}\|^2+(V\boldsymbol{y})\cdot V(\boldsymbol{y})-2(V\boldsymbol{y})\cdot\boldsymbol{x}$$

  And by writing dot products as matrix multiplications $\boldsymbol{a}\cdot\boldsymbol{b}=\boldsymbol{a}^T\boldsymbol{b}$ we arrive to:

$$\begin{aligned}\|\boldsymbol{x}-V\boldsymbol{y}\|^2&=\|\boldsymbol{x}\|^2+(V\boldsymbol{y})^TV\boldsymbol{y}-2(V\boldsymbol{y})^T\boldsymbol{x}=\\&=\|\boldsymbol{x}\|^2+\boldsymbol{y}^T\underbrace{V^TV}_{\mathbb{I}}\boldsymbol{y}-2\boldsymbol{y}^TV^T\boldsymbol{x}=\\&=\|\boldsymbol{x}\|^2+\|\boldsymbol{y}\|^2-\boldsymbol{2y}^T(V^T\boldsymbol{x})\end{aligned}$$

Call $V\boldsymbol{y} = \tilde{\boldsymbol{x}} \in R$. For a fixed $\boldsymbol{x} \in \mathbb{R}^d$, the closest $\tilde{\boldsymbol{x}} \in R$ is given by:

$$\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n} \|\boldsymbol{x} - V\boldsymbol{y}\|^2 = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n} \left(\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{y}^T(V^T\boldsymbol{x})\right)$$

We can minimize this expression by computing the gradient $\boldsymbol{\nabla}_{\boldsymbol{y}}$ and setting it to 0:

$$\boldsymbol{\nabla}_{\boldsymbol{y}} \left(\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2\boldsymbol{y}^T(V^T\boldsymbol{x})\right) = 2\boldsymbol{y} - 2V^T\boldsymbol{x} \overset{!}{=} 0 \Rightarrow \boldsymbol{y}_0 = V^T\boldsymbol{x}$$

And so the solution is $\tilde{\boldsymbol{x}} = V\boldsymbol{y}_0 = VV^T\boldsymbol{x}$. We can compute the $\tilde{\boldsymbol{x}}_i$ closest to $\boldsymbol{x}_i$ for $1 \leq i \leq m$, resulting in:

$$\sum_{i=1}^m \|\boldsymbol{x_i} - UW\boldsymbol{x_i}\|^2 \geq \sum_{i=1}^m \left\|\boldsymbol{x_i} - VV^T\boldsymbol{x_i}\right\|^2$$

for any generic U and W. So $(U, W) = (V, V^T)$ is a solution of (2.4), meaning that we can rewrite it as:

$$\arg\min_{U\in\mathcal{M}_{d\times n}, U^TU=\mathbb{I}} \sum_{i=1}^m \left\|\boldsymbol{x_i} - UU^T\boldsymbol{x_i}\right\|^2$$

Let's simplify the distance expression a bit more:

$$\begin{aligned}
\left\|\boldsymbol{x} - UU^T\boldsymbol{x}\right\|^2 &= \|\boldsymbol{x}\|^2 + (UU^T\boldsymbol{x})^TUU^T\boldsymbol{x} - 2(UU^T\boldsymbol{x})^T\boldsymbol{x} = \\
&= \|\boldsymbol{x}\|^2 + \boldsymbol{x}^TU\underbrace{U^TU}_{\mathbb{I}}U^T\boldsymbol{x} - 2\boldsymbol{x}^TUU^T\boldsymbol{x} = \\
&= \|\boldsymbol{x}\|^2 + \boldsymbol{x}^TUU^T\boldsymbol{x} - 2\boldsymbol{x}^TUU^T\boldsymbol{x} = \\
&= \|\boldsymbol{x}\|^2 - \boldsymbol{x}^TUU^T\boldsymbol{x} = \\
&= \|\boldsymbol{x}\|^2 - (U^T\boldsymbol{x})^TU^T\boldsymbol{x} = \|\boldsymbol{x}\|^2 - (U^T\boldsymbol{x}) \cdot (U^T\boldsymbol{x}) = \\
&= \|\boldsymbol{x}\|^2 - \sum_{i=1}^d \sum_{j,k=1}^d (U^T)_{ij}x_j(U^T)_{ik}x_k = \\
&= \|\boldsymbol{x}\|^2 - \sum_{i=1}^d \sum_{j,k=1}^d (U^T)_{ij}x_jx_kU_{ki} = \|\boldsymbol{x}\|^2 - \sum_{i=1}^d (U^T[\boldsymbol{x}\boldsymbol{x}^T]U)_{ii} = \\
&= \|\boldsymbol{x}\|^2 - \operatorname{trace}(U^T\boldsymbol{x}\boldsymbol{x}^TU)
\end{aligned}$$

Note that $\|\boldsymbol{x}\|^2$ is constant, and so can be removed from the optimization. The trace is linear and so:

$$\sum_{i=1}^m \operatorname{Trace}(U^T\boldsymbol{x_i}\boldsymbol{x_i}^TU) = \operatorname{Trace}\left(U^T\sum_{i=1}^m \boldsymbol{x_i}\boldsymbol{x_i}^TU\right)$$

And finally the $-$ sign transforms a min into a max, leading to the reformulation:

$$\arg\max_{U\in\mathcal{M}_{d\times n}(\mathbb{R}), U^TU=\mathbb{I}} \operatorname{Trace}\left(U^T\underbrace{\sum_{i=1}^m \boldsymbol{x_i}\boldsymbol{x_i}^T}_{A}U\right) \tag{2.5}$$

Now A is symmetric and so (spectral theorem) can be orthogonally diagonalized into $A = VDV^T$ with D diagonal and $V^TV = VV^T = \mathbb{I}$ with the columns of V being the eigenvectors of eigenvalue the elements of D.

The solution to the PCA is to set U to the matrix with columns equal to the n eigenvectors with highest eigenvalues, and $W = U^T$.

In fact, let $A = VDV^T$ be the spectral decomposition. Let's compute (2.5). Let U be a generic $d \times n$ matrix with $U^TU = \mathbb{I}$. We start by evaluating:

$$U^TAU = U^TVDV^TU = \underbrace{(V^TU)^T}_{B^T} D \underbrace{(V^TU)}_{B} = B^TDB$$

and so as B is $d \times n$ and D is $d \times d$:

$$\text{Trace}(U^TAU) = \text{Trace}(B^TDB) = \sum_{i=1}^{n} \sum_{k,j=1}^{d} B_{ki}D_{kj}\delta_{kj}B_{ji} = \sum_{i=1}^{n}\sum_{j=1}^{d} D_{jj}B_{ji}^2 =$$
$$= \sum_{j=1}^{d} D_{jj} \underbrace{\sum_{i=1}^{n} B_{ji}^2}_{\beta_j}$$

β_j is the norm of the j-th row of B. We now show that it's ≤ 1.

Note that the columns of B are orthonormal, in fact:

$$B^TB = (V^TU)^TV^TU = U^TVV^TU = \mathbb{I}$$

this implies that the columns of B have unit-norm:

$$\sum_{j=1}^{d} B_{ji}^2 = 1$$

Then, as B is $d \times n$ with $n < d$, the norm of its rows must be ≤ 1. In fact, we can extend the ON basis of $\mathbb{R}^n$ (the columns of B) to a ON basis of $\mathbb{R}^d$, constructing a $d \times d$ $\tilde{B}$ that is equal to B for the first n columns. Then:

$$\sum_{i=1}^{d} \tilde{B}_{ji}^2 = 1 \geq \sum_{i=1}^{n} B_{ji}^2$$

For the maximization, consider that $\sum_{j=1}^{d} B_j = n$, because they are rows of a $d \times n$ matrix with orthonormal columns. So:

$$\text{Trace}(U^TAU) = \sum_{j=1}^{d} D_{jj}\beta_j \leq \max_{\beta_j \in [0,1]; \|\beta\| \leq n} \sum_{j=1}^{d} D_{jj}\beta_j = \sum_{j=1}^{n} D_{jj}$$

For the last part, consider D_{jj} ordered so that D_{11} is maximum and D_{dd} is minimum. To maximize the sum, choose $B_j = 1$ for $j \leq n$, and $\beta_j = 0$ for $j > n$ (as $\sum_j B_j$ must be n).

Finally, note that the inequality is saturated by choosing U to be the matrix whose columns are the n leading eigenvectors of A, and this concludes the proof.