Other topics for future consideration & Concluding remarks
More on Linear Classifiers: Perceptron & SVMs
More into Linear Classifiers

Linearly separable case (i.e., realizability holds):

**Input:** training data \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)}) \in \mathbb{R}^m \times \{-1, +1\}\)

**Output:** linear classifier parameter vector \(\hat{w} = (w, b) \in \mathbb{R}^m \times \mathbb{R}\) such that

\[
y^{(i)} \langle \hat{w}, \tilde{x}^{(i)} \rangle > 0 \text{ for } i = 1, \ldots, n
\]

where \(\tilde{x}^{(i)} = (x^{(i)}, 1)\)

This is linear programming:

- Each data point is a linear constraint on \(w\)
- Want to find \(w\) that satisfies all these constraints

But we won’t use generic linear programming methods, such as simplex.
A simple alternative: **Perceptron algorithm** (Rosenblatt, 1958)

- $\tilde{w} = 0$
- while some $(x, y)$ is misclassified:
  \[
  \tilde{w} = \tilde{w} + y \tilde{x}
  \]
A better separator?

For a linearly separable data set, there are in general many possible separating hyperplanes, and Perceptron is guaranteed to find one of them.

But is there a better, more systematic choice of separator? The one with the most buffer around it, for instance?
Maximizing the margin

Want \( y^{(i)} \langle \tilde{w}, \tilde{x}^{(i)} \rangle > 0 \) for \( i = 1, ..., n \)
By scaling \( w, b \), we can equally ask for \( y^{(i)} \left( \langle w, x^{(i)} \rangle + b \right) \geq 1 \) \( \forall \ i \)

Maximize the margin \( \gamma \).
Maximizing the margin

Can easily show that maximizing $\gamma$ is equivalent to minimizing $\|w\|$ subject to

$$y^{(i)}(\langle w, x^{(i)} \rangle + b) \geq 1 \quad \forall \ i$$

This is a convex constrained optimization problem that has the following equivalent dual form:

$$\text{(DUAL)} \quad \max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)})$$

s.t.: $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$

$\alpha \geq 0$

At optimality, $w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ and moreover

$$\alpha_i > 0 \quad \Rightarrow \quad y^{(i)}(w \cdot x^{(i)} + b) = 1$$

Points $x^{(i)}$ with $\alpha_i > 0$ are called support vectors.
Support vectors

$\alpha_i$ is nonzero only for these support vectors

Linear classifier $w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ is a function of just the support vectors.
The non-separable case: Soft margin

Idea: allow each point $x^{(i)}$ to have some slack $\xi_i$
Model Selection and Validation
Model selection and validation

During the course, we usually assumed that we are given the model/hypothesis class.

In practice, choosing a certain family of models depends usually on experience (inductive bias). However, there could still be many parameters and/or degrees of freedom that need to be specified.

Model selection and validation is a formal and yet practical approach to help us hone in on a good model and properly tune its parameters.
Model selection and validation

$r$ models

$\mathcal{H}_1 \quad \mathcal{H}_2 \quad \ldots \quad \mathcal{H}_r$

$S_{\text{train}} \quad h_1 \quad h_2 \quad \ldots \quad h_r \quad S_{\text{Val}}$

$\hat{\mathcal{H}} = \{h_1, \ldots, h_r\}$

$h_{j^*}$ minimizes the empirical error on $S_{\text{Val}}$
The Bayesian approach to learning
The Bayesian learning model

Data is linked to hypotheses (i.e., parameters) via a probabilistic model that assumes:

- a certain conditional distribution \( p(y \mid x, w) \) called the **likelihood function**.

- a certain distribution over the parameters \( p(w) \) called the **prior**.

The approach mainly relies on computing the **posterior distribution**

\[
p(w \mid (x_1, y_1), \ldots, (x_n, y_n))
\]

given a training set \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \), and obtaining \( w \) that minimizes the expectation of a certain **loss function** w.r.t. the **posterior distribution**.
Concluding Remarks
We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.
- Sample complexity and its connection to VC dimension.
- For classification problems, ERM algorithms can learn with optimal sample complexity, or equivalently, optimal accuracy.
- Boosting: a paradigm to smoothly control the bias-complexity tradeoff by gradually increasing the model complexity.
- ERM is not always efficient when working the ‘0-1’ loss.
- When the direct ERM approach is computationally inefficient:
  - **boosting** can help circumventing the problem since it usually relies on multiple calls of a simple learning rule.
  - or, we may want to resort to a “surrogate” convex loss function instead of the ‘0-1’ loss, as in logistic regression. Hence, the problem becomes efficiently solvable (convex optimization).
We learned

• The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
• Many regression and classification problems can be expressed and dealt with using the convex learning framework.
• Can be used as a surrogate model for binary classification to circumvent efficiency problems. (E.g., linear classification under the ‘0-1’ loss for non-separable data is NP-hard, but classification via logistic regression or soft-margin SVM is efficient.)
• Under relatively weak assumptions on the model, e.g., Lipschitzness of the loss and boundedness of the hypothesis class, we can efficiently learn the model (in the agnostic PAC sense) via:
  ➢ Stochastic Gradient Descent
  ➢ Regularized Loss Minimization
• Regularization is tied to the algorithmic stability of the learner.
• Stability is directly related to the ability of the learner to generalize.