High level categories in ML

Based on objectives

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Parametric vs Non-Parametric

Learn a **fixed-size** set of parameters	Learnable parameters depends on size of training data
Can throw away data after training	Predictions uses training data

Despite the name, non-parametric **does not mean** “no parameters”

Simplest Example: k-Nearest Neighbors

- On test example x, predict the **most common label** among the **k neighbors of x**
- Nearby examples have the same label
- Need to define a **similarity function**

- **Low bias:** Makes no assumption about decision boundary
- **Potentially high variance:** Decisions are very sensitive to training data
  - “Curse of dimensionality”

Complex Example: Kernel Methods

- Like k-NN, decision depends on “nearby” training data
- **Kernel functions:** (loosely) **measures closeness** of two points

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x^{(i)}, x^{\text{test}})$$

- ɑ ‘s are parameters: one for each training data point
- How do we learn them?
  - Lec 8 covered connection between logistic regression and an algorithm for learning ɑ
  - SVMs

Kernel Trick

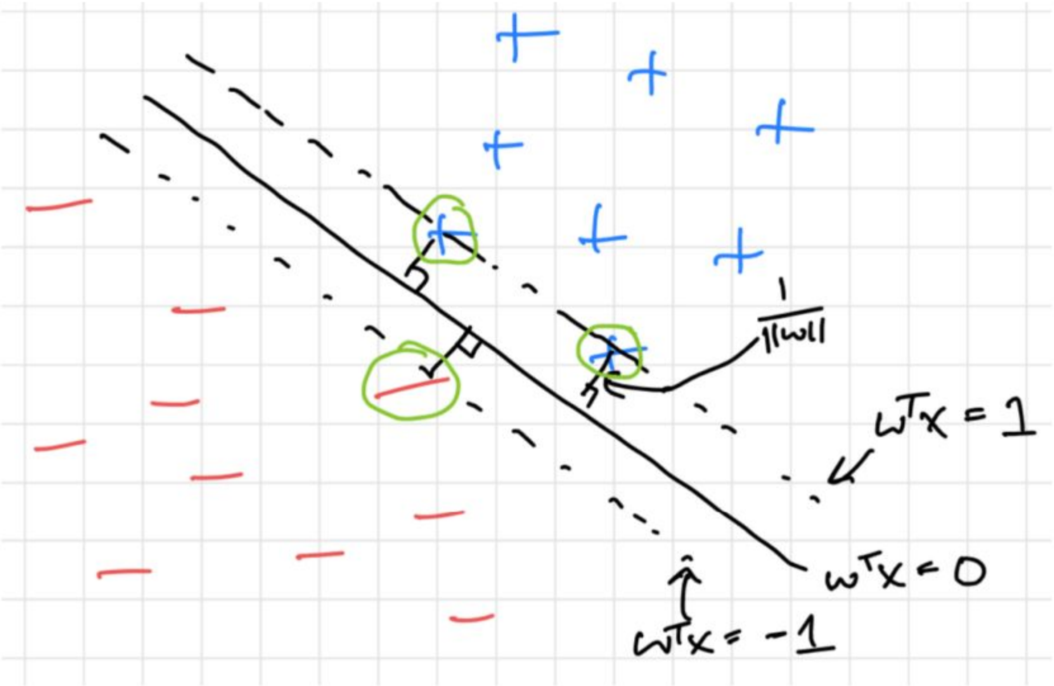
Use kernel functions between two data points
instead of
Computing dot product in large feature spaces

Why use kernel methods?

+ Get benefit of using large feature spaces (more expressive)
+ Avoid computation of explicitly writing out the data point in a large feature space
    - RBF kernels represent dot product in infinite dimensional space
    - You'll work through this in the homework
- Inference time has a dependence on number of training datapoints

Support Vector Machines

- Max-margin classifier
  - Objective function incentivizes a large margin between the two class



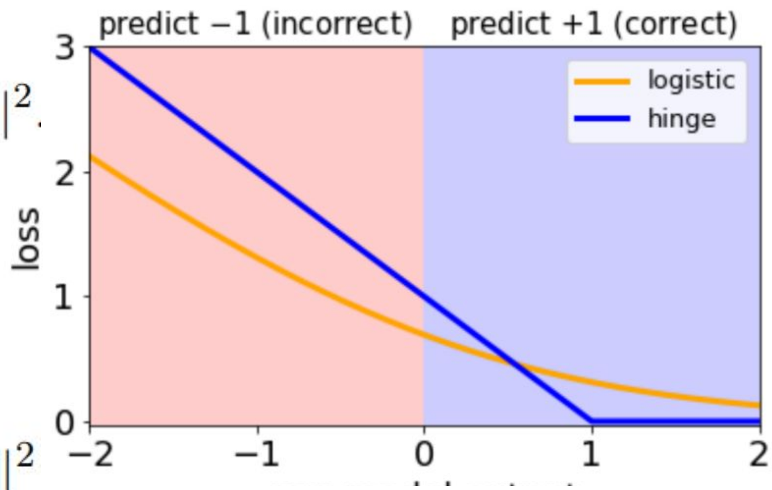
SVM

- Can be derived as a special case of replacing logistic loss with hinge loss

$$L(w) = \frac{1}{n}\sum_{i=1}^{n} -\log \sigma(y^{(i)} \cdot w^\top x^{(i)}) + \lambda \|w\|^2.$$

$$\Downarrow$$

$$L(w) = \left(\frac{1}{n}\sum_{i=1}^{n}[1 - y^{(i)} w^\top x^{(i)}]_+\right) + \lambda \|w\|^2$$



SVM

- Can then be kernelized to take advantage of kernel trick

$$L(\alpha) = \frac{1}{n}\sum_{i=1}^{n}\left[1 - y^{(i)}\sum_{j=1}^{n}\alpha_j k(x^{(j)}, x^{(i)})\right]_+ + \lambda\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(x^{(i)}, x^{(j)})\right)$$

SVM: What are support vectors?

- Like other kernel methods, the SVM decision function can be stated as:

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x^{(i)}, x^{\text{test}})$$

- However, optimizing the SVM objective leads to:
  - α being non-zero only for **support vectors:** examples that lie on the decision boundary or within the margin and misclassified examples
  - α being zero for examples that are correctly classified
- This means at test time, the decision depends only on a small number of support vectors and not the entire training set

Loss for the SVM with Soft Margin

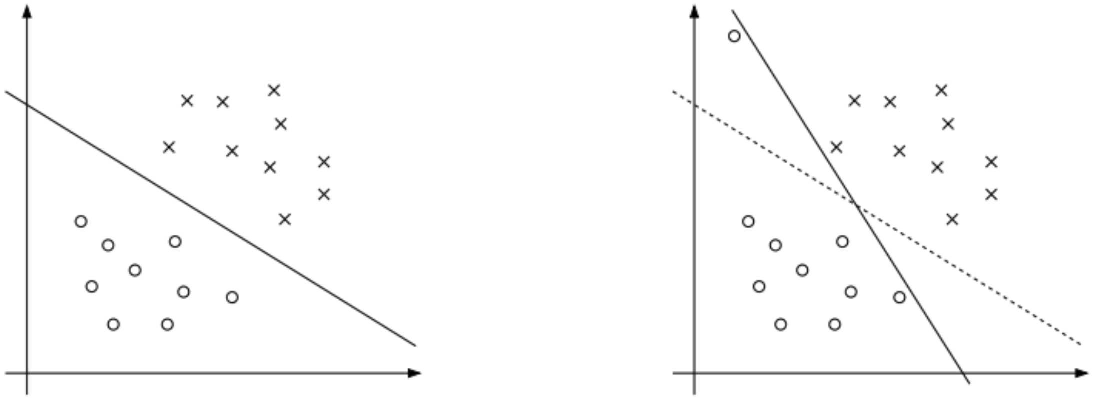
$$L(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \ - \sum_{i=1}^{n} \alpha_i \left[y^{(i)} \left(x^{(i)T} w \right) \ - 1 + \xi_i \right] - \sum_{i=1}^{n} r_i \xi_i$$

- $\boldsymbol{\frac{1}{2} w^T w}$: Regularization term that controls the margin size. Minimizing it ensures that the margin between the positive and negative classes is maximized

- $\mathbf{C} \boldsymbol{\sum_{i=1}^{n} \xi_i}$: $\boldsymbol{\xi_i}$ is called the slack variable that allow points to be on the wrong side of the margin or misclassified. This term penalizes the total amount of slack used. A larger C enforces stricter classification

- $\boldsymbol{\sum_{i=1}^{n} \alpha_i \left[y^{(i)} \left(x^{(i)T} w + b \right) \ - 1 + \xi_i \right]}$: This constraint forces each sample $x^{(i)}$ to be correctly classified with a margin of at least 1. If a point lies within the margin or on the wrong side of the decision boundary, the corresponding α_i becomes positive, reflecting the misclassification.
- $\boldsymbol{-\sum_{i=1}^{n} r_i \xi_i}$: Ensures non-negativity of the slack variables (i.e., $\boldsymbol{\xi_i} \geq 0$). This term ensures that all slack variables are zero or positive, corresponding to the fact that the misclassification penalties cannot be negative.

Regularization and Non-separable Case



The left figure below shows an optimal margin classifier, and when a single outlier is added in the upper-left region (right figure), it causes the decision boundary to make a dramatic swing, and the resulting classifier has a much smaller margin

- Mapping data to a high dimensional feature space via ϕ increase the likelihood that the data is separable but not guarantee it
- There are cases in which finding a separating hyperplane is not exactly what we'd want to do
- Therefore, we need some trick make the SVM algorithm work for non-linearly separable datasets as well as be less sensitive to outliers

SVM: Optimization

- There exist alternative techniques to minimize SVM loss
  - Linear optimization libraries
- These approaches only require dot-products between the data points
  - Thus, they can use kernel functions

- Combine this with efficient prediction using kernels

This is the big reason why SVMs and kernels are a good match!

Quick Note:
All non-parametric approaches discussed, have corresponding variants for regression

References:

Slides adopted from previous year discussion: https://usc-csci467.github.io/
Additional Information for soft can be found here:
https://cs229.stanford.edu/notes2020fall/notes2020fall/cs229-notes3.pdf