CS221 Problem Workout Solutions

Week 2

Key Takeaways from this Week

The goal of ML is to learn a function f parameterized by w s.t. $f_w(x)$ is very close to y. Each algorithm is a triplet of three design decisions:

1. **Hypothesis class** – How will I write down my prediction for y as a function of x? Which parameters w do I need to learn?
2. **Loss function** – How do I measure how far my prediction is from the real y?
3. **Optimization algorithm** – What algorithm will I use to minimize my loss function?

		Hypothesis class	**Loss function**	**Optimization algorithm**
$y \in \mathbb{R}$	Linear regression	$f_w(x) := w \cdot \phi(x)$	Squared loss: $(f_w(x) - y)^2$	GD or SGD
$y \in \{-1, 1\}$	(Binary) linear classification	$f_w(x) := \text{sign}(w \cdot \phi(x))$	0-1 loss: $1[f_w(x) \neq y]$	Cannot use GD, SGD
			Hinge loss: $\max\{1 - (w \cdot \phi(x))y, 0\}$	GD or SGD
			Logistic loss: $\log\left(1 + e^{-(w \cdot \phi(x))y}\right)$	GD or SGD

Dimension check. Above, $w, \phi(x) \in \mathbb{R}^d$, while y is a scalar.

1) Problem 1: Non-linear features

Consider the following two training datasets of (x, y) pairs:

- $\mathcal{D}_1 = \{(-1, +1), (0, -1), (1, +1)\}$.
- $\mathcal{D}_2 = \{(-1, -1), (0, +1), (1, -1)\}$.

Observe that neither dataset is linearly separable if we use $\phi(x) = x$, so let's fix that.

Define a two-dimensional feature function $\phi(x)$ such that:

- There exists a weight vector $\mathbf{w}_1$ that classifies $\mathcal{D}_1$ perfectly (meaning that $\mathbf{w}_1 \cdot \phi(x) > 0$ if x is labeled $+1$ and $\mathbf{w}_1 \cdot \phi(x) < 0$ if x is labeled -1); and
- There exists a weight vector $\mathbf{w}_2$ that classifies $\mathcal{D}_2$ perfectly.

Note that the weight vectors can be different for the two datasets, but the features $\phi(x)$ must be the same.

Solution One option is $\phi(x) = [1, x^2]$, and using $\mathbf{w}_1 = [-1, 2]$ and $\mathbf{w}_2 = [1, -2]$.

Then in $\mathcal{D}_1$:

- For $x = -1$, $\mathbf{w}_1 \cdot \phi(x) = [-1, 2] \cdot [1, 1] = 1 > 0$
- For $x = 0$, $\mathbf{w}_1 \cdot \phi(x) = [-1, 2] \cdot [1, 0] = -1 < 0$
- For $x = 1$, $\mathbf{w}_1 \cdot \phi(x) = [-1, 2] \cdot [1, 1] = 1 > 0$

In $\mathcal{D}_2$:

- For $x = -1$, $\mathbf{w}_2 \cdot \phi(x) = [1, -2] \cdot [1, 1] = -1 < 0$
- For $x = 0$, $\mathbf{w}_2 \cdot \phi(x) = [1, -2] \cdot [1, 0] = 1 > 0$
- For $x = 1$, $\mathbf{w}_2 \cdot \phi(x) = [1, -2] \cdot [1, 1] = -1 < 0$

Note that there are many options that work, so long as -1 and 1 are separated from 0.

Some additional food for thought: Is every dataset linearly separable in some feature space? In other words, given pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, can we find a feature extractor ϕ such that we can perfectly classify $(\phi(\mathbf{x}_1), y_1), \ldots, (\phi(x_n), y_n)$ for some linear model $\mathbf{w}$? If so, is this a good feature extractor to use?

Solution In theory, yes we can. If we assume that our inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are distinct, then we can construct a feature map $\phi : x_i \mapsto y_i$ for $i = 1, \ldots, n$. By setting $\mathbf{w}^\star = [1]$, it's clear that

$$y_i \mathbf{w}^\star \cdot \phi(x_i) = y_i * y_i = 1 > 0, \quad i = 1, \ldots, n, \tag{1}$$

so $\mathbf{w}^\star$ correctly classifies all the points in the dataset.

Hopefully, it's clear that this is a poor choice of feature map. For one, this feature extractor is undefined for any points outside of the training set! But even more broadly, this process is not at all *generalizeable*. We are essentially just memorizing our dataset instead of learning patterns and structures within the data that will allow us to accurately predict new points in the future. While minimizing training loss is an important part of the machine learning process (the aforementioned procedure gives you zero training loss!), it does not guarantee you good performance in the future.

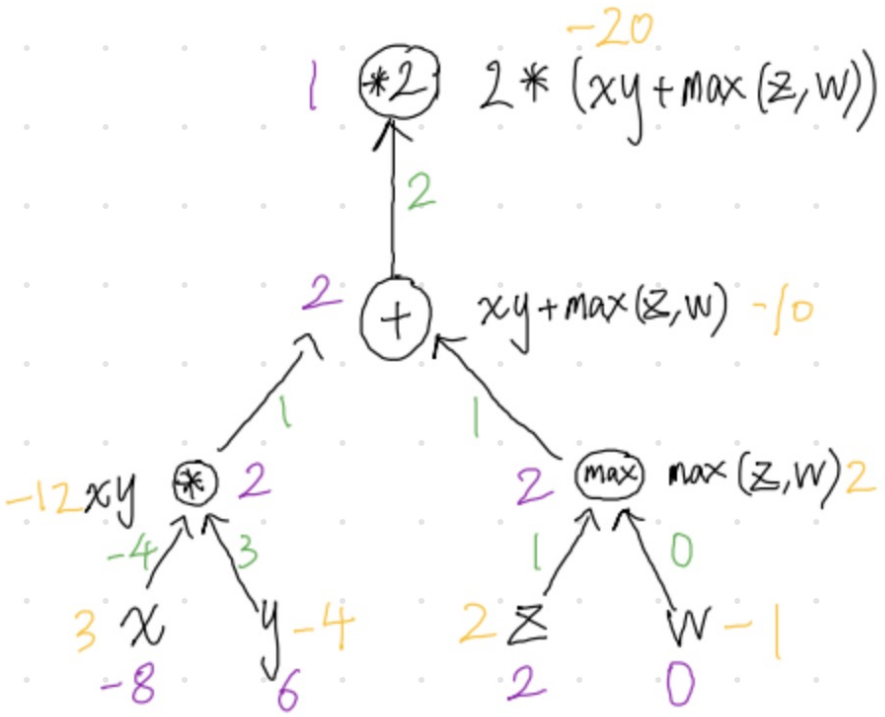
2) Problem 2: Backpropagation

Consider the following function

$$\text{Loss}(x, y, z, w) = 2(xy + \max\{w, z\})$$

Run the backpropagation algorithm to compute the four gradients (each with respect to one of the individual variables) at $x = 3$, $y = -4$, $z = 2$ and $w = -1$. Use the following nodes: addition, multiplication, max, multiplication by a constant.

Solution When calculating the gradients, we run backpropagation from the root node to the leaves nodes. As shown on the computation graph below, the purple values are the gradients of Loss with respect to each node. The yellow values are the computed values of each term for the forward pass. The green values are the partial derivative of the loss with respect to that node.



3) Problem 3: K-means

Consider doing ordinary K-means clustering with $K = 2$ clusters on the following set of 3 one-dimensional points:

$$\{-2, 0, 10\}. \tag{2}$$

Recall that K-means can get stuck in local optima. Describe the precise conditions on the initialization $\mu_1 \in \mathbb{R}$ and $\mu_2 \in \mathbb{R}$ such that running K-means will yield the global optimum of the objective function. Notes:

- Assume that $\mu_1 < \mu_2$.
- Assume that if in step 1 of K-means, no points are assigned to some cluster j, then in step 2, that centroid μ_j is set to ∞.
- Hint: try running K-means from various initializations μ_1, μ_2 to get some intuition; for example, if we initialize $\mu_1 = 1$ and $\mu_2 = 9$, then we converge to $\mu_1 = -1$ and $\mu_2 = 10$.

Solution The objective is minimized for $\mu_1 = -1$ and $\mu_2 = 10$. First, note that if all three points end up in one cluster, K-means definitely fails to recover the global optimum. Therefore, -2 must be assigned to the first cluster, and 10 must be assigned to the second cluster. 0 can be assigned to either: If 0 is assigned to cluster 1, then we're done. If it is assigned to cluster 2, then we have $\mu_1 = -2, \mu_2 = 5$; in the next iteration, 0 will be assigned to cluster 1 since its closer. Therefore, the condition on the initialization written formally is $|-2-\mu_1| < |-2-\mu_2|$ and $|10-\mu_1| > |10-\mu_2|$.

4) [optional] Problem 4: Non-linear decision boundaries

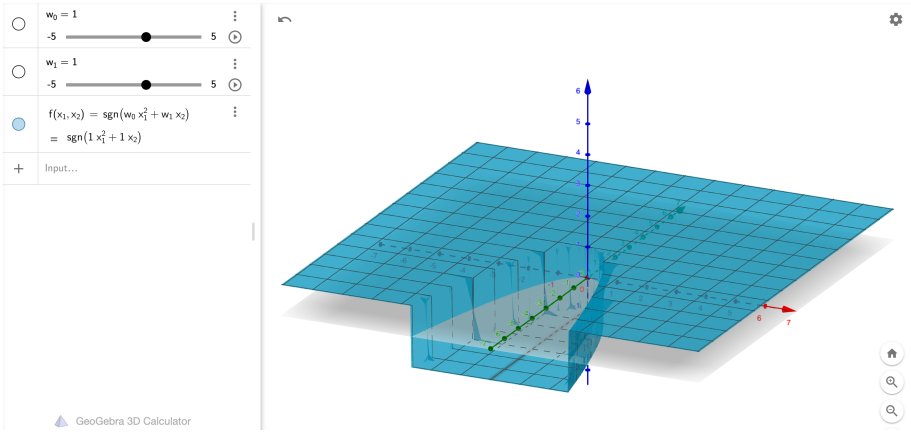
Suppose we are performing classification where the input points are of the form $(x_1, x_2) \in \mathbb{R}^2$. We can choose any subset of the following set of features:

$$\mathcal{F} = \left\{ x_1^2, x_2^2, x_1 x_2, x_1, x_2, \frac{1}{x_1}, \frac{1}{x_2}, 1, \mathbf{1}[x_1 \geq 0], \mathbf{1}[x_2 \geq 0] \right\} \tag{3}$$

For each subset of features $F \subseteq \mathcal{F}$, let $D(F)$ be the set of all decision boundaries corresponding to linear classifiers that use features F.

For each of the following sets of decision boundaries E, provide the minimal F such that $D(F) \supseteq E$. If no such F exists, write 'none'.

For example the set of features $F = \{x_1^2, x_2\}$ allows the decision boundary of parabolas opening in the x_2 axis, centered at the origin:



- E is all lines [CA hint]:

  _____ (4)

- E is all circles centered at the origin:

  _____ (5)

- E is all circles:

  _____ (6)

- E is all axis-aligned rectangles:

  _____ (7)

- E is all axis-aligned rectangles whose lower-right corner is at $(0, 0)$:

  _____ (8)

Solution

- Lines: $x_1, x_2, 1$ $(ax_1 + bx_2 + c = 0)$
- Circles centered at the origin: $x_1^2, x_2^2, 1$ $(x_1^2 + x_2^2 = r^2)$
- Circles centered anywhere in the plane: $x_1^2, x_2^2, x_1, x_2, 1$ $((x_1 - a)^2 + (x_2 - b)^2 = r^2)$
- Axis aligned rectangles: not possible (need features of the form $\mathbf{1}[x_1 \leq a]$)
- Axis aligned rectangles with lower right corner at (0, 0): not possible