CS 181 Spring 2018 Section 4
Solution

1 Generative Models

A **generative model** is one that models the joint distribution $p(x, y)$. This factors into $p(x|y)p(y)$.

- $p(y)$ is called the **class prior** and is always a categorical distribution (generalization of a Bernoulli to multiple classes). Do not confuse "prior" with a Bayesian prior over model parameters. $p(y)$ gives an a priori probability of an observation being a certain class, without even considering the observation's features.
- $p(x|y)$ is called the **class-conditional distribution** and its form is model-specific. $p(x|y)$ specifies how likely an observation (set of features) is given a class.

During classification, we are interested in picking the class k that maximizes $p(y = k|\mathbf{x})$. This conditional is related to the joint $p(\mathbf{x}, y)$ through Bayes' Rule:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \propto p(\mathbf{x}|y)p(y)$$

$p(\mathbf{x})$ is constant because it does not depend on the choice of class, so we can disregard it.

Compare the above definition of generative models to logistic regression (introduced in section 3 notes and on T2). In logistic regression, you optimize a set of vectors $\mathbf{w}_k$ and classify an observation $\mathbf{x}$ with the class k that maximizes the score $\mathbf{w}_k^\top \mathbf{x}$. Logistic regression is called a **discriminative model** because you directly optimize for model parameters and do not estimate the distribution of $p(x, y)$. **Brainstorm:** is this a shortcoming of logistic regression?

2 Naive Bayes

Naive Bayes is one type of generative model for classification. It's "naive" because we assume that each feature in $\mathbf{x}$ is independently distributed, conditioned on the class. This means that we can write $p(\mathbf{x}|y = k) = \prod_{j=1}^{D} p(x_j|y = k)$ where D is the number of features and k is the class. Choosing to use Naive Bayes does not fully specify the model. One must still choose the particular distribution $p(x_j|y)$. Let's choose a Multinomial class-conditional distribution over discrete feature counts. That is,

$$p(\mathbf{x}_i|y = k; \boldsymbol{\pi}_k) = \mathrm{Mu}(\mathbf{x}_i|N_i = \sum_j x_{ij}, \boldsymbol{\pi}_k) = \frac{N_i!}{\prod_{j=1}^{D} x_{ij}!} \prod_{j=1}^{D} \pi_{kj}^{x_j} \propto \prod_{j=1}^{D} \pi_{kj}^{x_j}$$

In this model, each class k has a vector of probabilities over each of D features.

$$\boldsymbol{\pi}_k = [\pi_{k1}, \pi_{k2}, \ldots, \pi_{kD}]$$

This vector sums to one for each class k: $\sum_{j=1}^{D} \pi_{kj} = 1$. Two notes. First, notice that we drop the normalizing coefficient in the PMF. It does not affect our predictions and is specific to each $\mathbf{x}_i$, not y. Second, notice that this model is different than if each feature was Bernoulli distributed.

Intuition: You can think of each observation $\mathbf{x}_i$ as the result of rolling one of K D-sided dice several times and recording how many times each side came up. These are our feature counts. Each die corresponds to a class and might have its own biases toward particular sides (so seeing a high count for a given feature may be more likely given one class than other)

Let's consider just two classes. Let $p(y = 1) = \theta$ and $p(y = 0) = (1 - \theta)$. One die might be more common overall, regardless of the features we observe in a particular $\mathbf{x}_i$.

We have specified the full model and the role of its parameters θ, $\boldsymbol{\pi}_0$, and $\boldsymbol{\pi}_1$. Classification corresponds to identifying which of two dice generated $\mathbf{x}$. During prediction, we pick the k that maximizes $p(y = k|x)$ for an observation $\mathbf{x}$. To be able to do this, we must estimate the parameters of our model from data. For MLE, we want to find $\theta, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1$ that maximize the log-likelihood:

$$\begin{aligned}
\theta^*, \boldsymbol{\pi}_0^*, \boldsymbol{\pi}_1^* &= \arg\max_{\theta, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1} \sum_{i=1}^{N} \ln p(\mathbf{x}_i, y_i) \\
&= \arg\max_{\theta, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1} \sum_{i=1}^{N} \ln p(\mathbf{x}_i | y_i; \boldsymbol{\pi}_0, \boldsymbol{\pi}_1) + \sum_{i=1}^{N} \ln p(y_i; \theta) \\
&= \arg\max_{\theta, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1} \sum_{i=1}^{N} \sum_{k \in \{0,1\}} \mathbb{I}\{y_i = k\} \left[\Big(\sum_{j=1}^{D} x_{ij} \ln \pi_{kj} \Big) + k \ln \theta + (1 - k) \ln(1 - \theta) \right]
\end{aligned}$$

The indicator $\mathbb{I}\{y_i = k\}$, the k that multiplies $\ln \theta$ and the $1 - k$ that multiplies $\ln(1 - \theta)$ in the last line are all just used for notation tricks to make sure that our likelihood only evaluates $p(\mathbf{x}, y)$ for each $\mathbf{x}$'s true class rather than both classes. This is used a lot, so make sure you see what is going on.

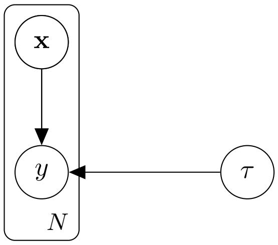
Is NB a linear classifier? In the case of our Multinomial example, you can write the classifier in the form of $\mathbf{w}^\top \mathbf{x} + w_0$:

$$\begin{aligned}
h(\mathbf{x}) &= [\ln p(\mathbf{x}|y = 1) + \ln p(y = 1)] - [\ln p(\mathbf{x}|y = 0) + \ln p(y = 0)] \\
&= \big[\ln \prod_{j=1}^{m} \pi_{1j}^{x_j} - \ln \prod_{j=1}^{m} \pi_{0j}^{x_j} \big] + \big[\ln \theta - \ln(1 - \theta) \big] \\
&= \sum_{j=1}^{m} x_j \ln \frac{\pi_{1j}}{\pi_{0j}} + \ln \left(\frac{\theta}{1 - \theta} \right) \\
&= \mathbf{x}^\top \mathbf{ln}(\frac{\boldsymbol{\pi}_1}{\boldsymbol{\pi}_0}) + \ln \left(\frac{\theta}{1 - \theta} \right)
\end{aligned}$$

where $\mathbf{ln}(\mathbf{q})$ for vector $\mathbf{q}$ applies ln element-wise, and we write $\boldsymbol{\pi}_1 / \boldsymbol{\pi}_0$ to mean element-wise division. The resulting equation is in the same form as linear regression, and in particular the decision boundary $h(\mathbf{x}) = 0$ is linear in $\mathbf{x}$. NB is not a linear classifier for all choice of the class-conditional distribution.

1. **Naive Bayes Graphical Model**

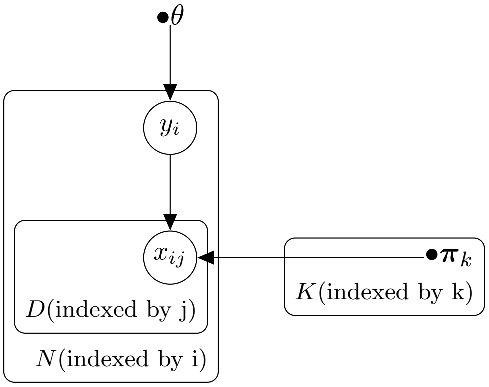
In the notes above, we algebraically described Naive Bayes as a generative model. It's very helpful to also approach the problem graphically. The language of **probabilistic graphical models** has become a rich tool for doing so. In such graphical models, each random variable gets a vertex, and conditional distributions are represented with directed edges between vertices. Repeated variables (e.g. N datapoints) are placed on rectangular plates. Consider the following (not Naive Bayes) example. We are using the `tikzbayesnet` library:



Here, there are N copies of $\mathbf{x}$ and y. y is conditionally distributed given $\mathbf{x}$ and τ.

Draw a graphical model for Naive Bayes with x_{ij} ($N \times D$ of these), y_i (N of these), $\boldsymbol{\pi}_k$ (K of these) and θ (you can think about there being 1 or K of these). Treat $\mathbf{x}$ and y as random and treat the rest as fixed parameters. To specify a non-random parameter, draw it next to a dot without a vertex around it. Remember, in Naive Bayes, the class and the class-conditional parameters generate the observation.

Solution



End Solution

2. **Redundant Features in Naive Bayes**

Suppose, as in Bishop 4.2.3, that we use a Naive Bayes classifier to classify binary feature vectors $\mathbf{x} \in \mathbb{R}^D$ into two classes. The class conditional distributions will then be of the form

$$p(\mathbf{x} \mid y = C_k) = \prod_{j=1}^{D} \pi_{kj}^{x_j} \, (1 - \pi_{kj})^{(1-x_j)}, \qquad \text{(Bishop 4.81)}$$

where $x_j \in \{0, 1\}$, and $\pi_{kj} = p(x_j = 1 \mid y = C_k)$. This is a Bernoulli Naive Bayes, different from Multinomial model in the notes in that all the features are binary instead of representing count data. Assume also that the class priors are $p(y = C_1) = p(y = C_2) = \frac{1}{2}$.

a. How is the quantity $\ln(p(y = C_1 \mid \mathbf{x})/p(y = C_2 \mid \mathbf{x}))$ used for classification of a new example $\mathbf{x}$?

b. If $D = 1$ (i.e., there is only one feature), use the equations above to write out $\ln \frac{p(y=C_1 \mid x)}{p(y=C_2 \mid x)}$ for a single binary feature x.

c. Now suppose we change our feature representation so that instead of using just a single feature, we use two redundant features. (i.e., two features that always have the same value). With this feature representation, instead of x we will use $\mathbf{x} = [x, x]^\top$. What is $\ln \frac{p(y=C_1 \mid \mathbf{x})}{p(y=C_2 \mid \mathbf{x})}$ in terms of the value for $\ln \frac{p(y=C_1 \mid x)}{p(y=C_2 \mid x)}$ you calculated in part (a.)?

d. Is this a bug or a feature?

Solution

a. We will predict class C_1 if $p(y = C_1 \,|\, \mathbf{x}) \geq p(y = C_2 \,|\, \mathbf{x})$, and class C_2 otherwise. Equivalently, we will predict C_1 if $\ln(p(y = C_1 \,|\, \mathbf{x})/p(y = C_2 \,|\, \mathbf{x})) \geq 0$, and C_2 otherwise.

b. Because the class priors are the same and the denominators cancel, we have $p(y = C_1 \,|\, x)/p(y = C_2 \,|\, x) = p(y = C_1)p(x \,|\, y = C_1)/p(y = C_2)p(x \,|\, y = C_2) = p(x \,|\, y = C_1)/p(x \,|\, y = C_2)$, and we have:

$$\begin{aligned}\ln \frac{p(y = C_1 \,|\, x)}{p(y = C_2 \,|\, x)} &= \ln \frac{\pi_{11}^{x}\,(1-\pi_{11})^{(1-x)}}{\pi_{21}^{x}\,(1-\pi_{21})^{(1-x)}} \\ &= x\ln\pi_{11} + (1-x)\ln(1-\pi_{11}) - x\ln\pi_{21} - (1-x)\ln(1-\pi_{21})\end{aligned}$$

c. Because the two features are identical, we will have

$$\begin{aligned}\ln \frac{p(y = C_1 \,|\, \mathbf{x})}{p(y = C_2 \,|\, \mathbf{x})} &= \ln \frac{\left(\pi_{11}^{x}\,(1-\pi_{11})^{(1-x)}\right)^2}{\left(\pi_{21}^{x}\,(1-\pi_{21})^{(1-x)}\right)^2} \\ &= \ln \left[\left(\frac{\pi_{11}^{x}\,(1-\pi_{11})^{(1-x)}}{\pi_{21}^{x}\,(1-\pi_{21})^{(1-x)}}\right)^2\right] \\ &= 2\ln \frac{p(y = C_1 \,|\, x)}{p(y = C_2 \,|\, x)}\end{aligned}$$

(above, we use x to mean either redundant feature in $\mathbf{x}$ and dropped the subscripts)

d. This is a feature! We see that the classifier with the two identical features has exactly the same behavior as the classifier with just a single feature. In particular,

$$\ln \frac{p(y = C_1 \,|\, \mathbf{x})}{p(y = C_2 \,|\, \mathbf{x})} \geq 0 \quad \Leftrightarrow \quad \ln \frac{p(y = C_1 \,|\, x)}{p(y = C_2 \,|\, x)} \geq 0$$

Note: it does not matter that there is a new constant 2 in front of the expression. Only the sign is important for classification.

End Solution

3. **Shapes of Decision Boundaries - Part I**

Consider now a generative model with $K > 2$ classes, and output label $\mathbf{y}$ encoded as a "one hot" vector of length K. We adopt class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$ for all $k \in \{1, \ldots, K\}$ (where π_k is a parameter of the prior). Let $p(\mathbf{x} \mid \mathbf{y} = C_k)$ denote the class-conditional density of features $\mathbf{x}$ (in this case for class C_k). Let the class-conditional probabilities be Gaussian distributions

$$p(\mathbf{x} \mid \mathbf{y} = C_k) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \text{ for } k \in \{1, \ldots, K\}$$

We will predict the class of a new example $\mathbf{x}$ as the class with the highest conditional probability, $p(\mathbf{y} = C_k \mid \mathbf{x})$. Luckily, a little bird came to the window of your dorm, and claimed that you can classify an example $\mathbf{x}$ by finding the class that maximizes the following function:

$$f_k(\mathbf{x}) = \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k).$$

Derive this formula by comparing two different classes' conditional probabilities. What can we claim about the shape of the decision boundary given this formula?

Solution

Let's start with the conditional probability:

$$p(\mathbf{y} = C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{y} = C_k)p(\mathbf{y} = C_k)}{\sum_\ell^K p(\mathbf{x} \mid \mathbf{y} = C_\ell)p(\mathbf{y} = C_\ell)}$$

{We can take out the denominator since it will be the same for each class}

$$\begin{aligned} &\propto p(\mathbf{x} \mid \mathbf{y} = C_k)p(\mathbf{y} = C_k) \\ &= \frac{1}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)\pi_k \end{aligned}$$

{We can factor out constants that will be shared across classes}

$$\propto \frac{\pi_k}{|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)$$

{We take the logarithm, which is a monotonically increasing function and won't change the maximum across classes}

$$\propto \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

Now, what can we say about the shape of our decision boundary? $(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$ gives us intuition about the shape of our decision boundary. We see it is quadratic.

For further intuition, we can look at the other two terms to see what affects the probability of an example being classified to a given class. As the prior π_k increases, we see that the probability increases, which fits our intuition. As the determinant of the covariance matrix $\boldsymbol{\Sigma}_k$ increases, we see that the probability decreases, which also fits our intuition.

End Solution

4. **Shapes of Decision Boundaries - Part II**

Let's say the little bird comes back and now tells you that every class has the same covariance matrix, and so $\boldsymbol{\Sigma}_\ell = \boldsymbol{\Sigma}_\ell'$ for all classes C_ℓ and $C_{\ell'}$. Simplify this formula down further. What can we claim about the shape of the decision boundaries now?

Solution

Here was our original formula.

$$\ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \tag{1}$$

Given that all of the classes have the same covariance matrix, which we will write as $\boldsymbol{\Sigma}$, we have:

$$\begin{aligned}
f_k(\mathbf{x}) &= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) && (2)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T(\boldsymbol{\Sigma}^{-1}\mathbf{x}-\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) && (3)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}-\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k-\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}+\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) && (4)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}\boldsymbol{\Sigma}^{-1}\mathbf{x}^T-2\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k+\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) && (5)\\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}\mathbf{x}\boldsymbol{\Sigma}^{-1}\mathbf{x}^T+\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k-\frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k && (6)
\end{aligned}$$

{We can drop $-\frac{1}{2}\ln(|\boldsymbol{\Sigma}|)$ and $-\frac{1}{2}\mathbf{x}\boldsymbol{\Sigma}^{-1}\mathbf{x}^T$ since they are class independent}

$$\propto \ln(\pi_k) + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \tag{7}$$

Thus, we conclude that we can adopt function

$$\tilde{f}_k(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln(\pi_k) \tag{8}$$

Looking at this formula, we can see that our decision boundaries are no longer quadratic but linear. We've proven that if the classes share the same covariance matrix, our decision boundaries will be linear! You will see this yourselves when coding up the Gaussian model in T2 Q3.

End Solution

3 Neural Networks

Recall that in the case of binary classification, we can think about neural network as being equivalent to logistic regression with parameterized, adaptive basis functions. Adaptive means that you don't need to specify a feature basis. We can train a matrix that linearly transforms the data, run the resulting vector through an element-wise non-linearity, potentially repeat this process, and then finally run logistic regression on the resulting vector, where the logistic regression itself has weights that need to be trained.

Let's think about a neural network binary classifier with $\mathbf{x} \in \mathbb{R}^2$ $(D = 2)$ and with a single two-dimensional hidden layer.

For the non-linear activation function, we use the *ReLU* function defined by the following:

$$\text{ReLU}(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Let's consider a function h defined by the following:

$$\begin{align} h(\mathbf{x}) &= \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}) + w_0 \tag{10} \\ &= \mathbf{w}^{\mathsf{T}}\mathbf{ReLU}\left(\mathbf{W}^{hid}\mathbf{x} + \mathbf{w}_0^{hid}\right) + w_0, \tag{11} \end{align}$$

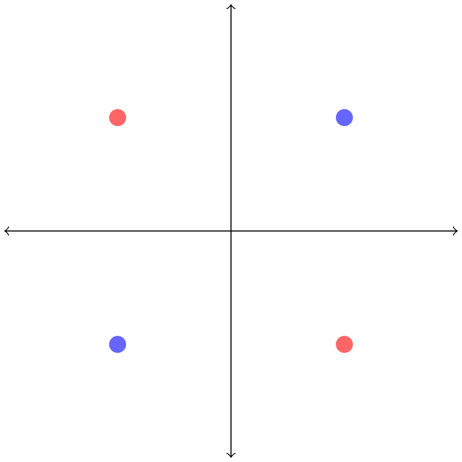
where example $\mathbf{x} \in \mathbb{R}^{2\times1}$, weights in the hidden layer $\mathbf{W}^{hid} \in \mathbb{R}^{2\times2}$, bias in the hidden layer $\mathbf{w}_0^{hid} \in \mathbb{R}^{2\times1}$, output weight vector $\mathbf{w} \in \mathbb{R}^{2\times1}$, and output bias $w_0 \in \mathbb{R}$. Everything within the **ReLU()** is the "adaptive basis" and the parameters outside are that of the logistic regression model.

Suppose we want to fit the following data:

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad y_1 = 1, \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad y_2 = -1$$

$$\mathbf{x}_3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad y_3 = -1, \quad \mathbf{x}_4 = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad y_4 = 1$$

This looks as follows (blue = 1, red = -1):



Why can't we solve this problem with a linear classifier? What values of parameters $\mathbf{W}^{hid}, \mathbf{w}_0^{hid}, \mathbf{w}$, and w_0 will allow the neural network to solve the problem? Show that your choice of parameters allows the network to correctly classify the data.

Note: You do not need to learn to find these weights systematically by hand. This is not very possible to do manually in general, but you might see something in this low-dimensional case.

Hint: Think carefully about why we need a nonlinearity in finding a basis function for the examples. What can the ReLU do for us? What does it do to various kinds of vectors? Think geometrically.

Solution

We can solve the problem with

$$\mathbf{W}^{hid} = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}, \quad \mathbf{w}_0^{hid} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad w_0 = -1$$

How does this solve the problem? Note that

$$h(\mathbf{x}_1) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} 2 \\ -2 \end{pmatrix} - 1 = 1$$

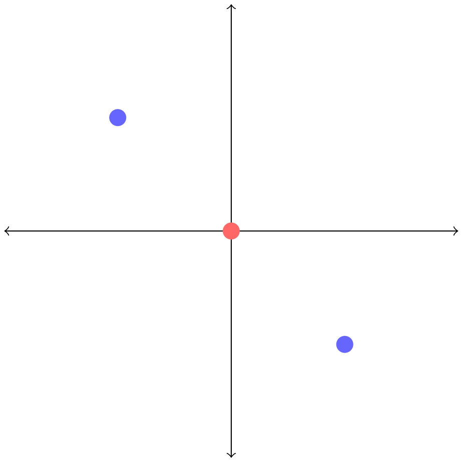
$$h(\mathbf{x}_2) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 1 = -1$$

$$h(\mathbf{x}_3) = \begin{pmatrix} -1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 1 = -1$$

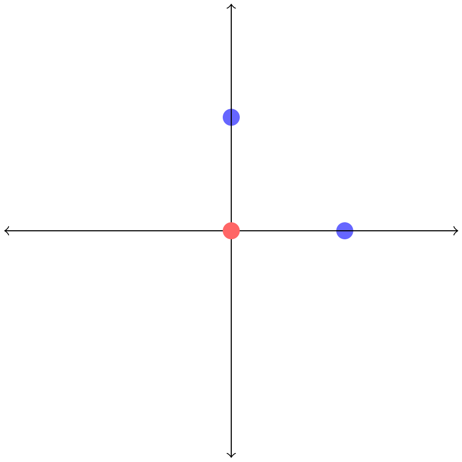
$$h(\mathbf{x}_4) = \begin{pmatrix} 1 & 1 \end{pmatrix} \mathbf{ReLU} \begin{pmatrix} -2 \\ 2 \end{pmatrix} - 1 = 1$$

This works because we have a ReLU! Having a nonlinearity in the activation function allows us to find a linear classifier for these examples in the new basis.

What did we do? With $\mathbf{W}^{hid}$, we map our data linearly so it looks as follows:



This is great, but still not linearly separable. Applying the ReLU, however, maps this picture to the following one:



Now our data is linearly separable and we can classify the points correctly. Note that the parameters we chose here are not unique (far from it). The important thing is that we transformed our input space well enough to apply a linear classifier.

End Solution