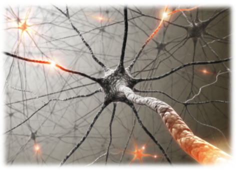
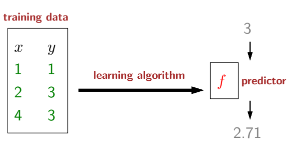
Machine learning: non-linear features



- In this module, we'll show that even using the machinery of **linear** models, we can obtain much more powerful **non-linear** predictors.

Linear regression



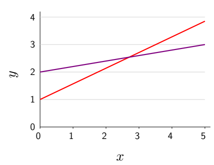
Which predictors are possible?
Hypothesis class

$$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^d\}$$

$\phi(x) = [1, x]$

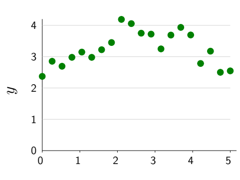
$f(x) = [1, 0.57] \cdot \phi(x)$

$f(x) = [2, 0.2] \cdot \phi(x)$



- We will look at regression and later turn to classification.
- Recall that in linear regression, given training data, a learning algorithm produces a predictor that maps new inputs to new outputs. The first design decision: what are the possible predictors that the learning algorithm can consider (what is the hypothesis class)?
- For linear predictors, remember the hypothesis class is the set of predictors that map some input x to the dot product between some weight vector $\mathbf{w}$ and the feature vector $\phi(x)$.
- As a simple example, if we define the feature extractor to be $\phi(x) = [1, x]$, then we can define various linear predictors with different intercepts and slopes.

More complex data



How do we fit a non-linear predictor?

- But sometimes data might be more complex and not be easily fit by a linear predictor. In this case, what can we do?
- One immediate reaction might be to go to something fancier like neural networks or decision trees.
- But let's see how far we can get with the machinery of linear predictors first.

Quadratic predictors

$$\phi(x) = [1, x, x^2]$$

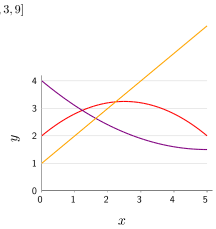
Example: $\phi(3) = [1, 3, 9]$

$f(x) = [2, 1, -0.2] \cdot \phi(x)$

$f(x) = [4, -1, 0.1] \cdot \phi(x)$

$f(x) = [1, 1, 0] \cdot \phi(x)$

$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^3\}$



Non-linear predictors just by changing ϕ

- The key observation is that the feature extractor ϕ can be arbitrary.
- So let us define it to include an x^2 term.
- Now, by setting the weights appropriately, we can define a non-linear (specifically, a **quadratic**) predictor.
- The first two examples of quadratic predictors vary in intercept, slope and curvature.
- Note that by setting the weight for feature x^2 to zero, we recover linear predictors.
- Again, the hypothesis class is the set of all predictors $f_{\mathbf{w}}$ obtained by varying $\mathbf{w}$.
- Note that the hypothesis class of quadratic predictors is a **superset** of the hypothesis class of linear predictors.
- In summary, we've seen our first example of obtaining non-linear predictors just by changing the feature extractor ϕ!
- Advanced: here $x \in \mathbb{R}$ is one-dimensional, so x^2 is just one additional feature. If $x \in \mathbb{R}^d$ were d-dimensional, then there would be $O(d^2)$ quadratic features of the form $x_i x_j$ for $i, j \in \{1, \ldots, d\}$. When d is large, then d^2 can be prohibitively large, which is one reason that using the machinery of linear predictors to increase expressivity can be problematic.

Piecewise constant predictors

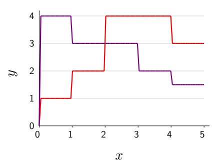
$$\phi(x) = [\mathbf{1}[0 < x \le 1], \mathbf{1}[1 < x \le 2], \mathbf{1}[2 < x \le 3], \mathbf{1}[3 < x \le 4], \mathbf{1}[4 < x \le 5]]$$

Example: $\phi(2.3) = [0, 0, 1, 0, 0]$

$f(x) = [1, 2, 4, 4, 3] \cdot \phi(x)$

$f(x) = [4, 3, 3, 2, 1.5] \cdot \phi(x)$

$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^5\}$



Expressive non-linear predictors by partitioning the input space

- Quadratic predictors are still a bit restricted: they can only go up and then down smoothly (or vice-vera).
- We introduce another type of feature extractor which divides the input space into regions and allows the predicted value of each region to vary independently, yielding piecewise constant predictors (see figure).
- Specifically, each component of the feature vector corresponds to one region (e.g., $[0, 1)$) and is 1 if x lies in that region and 0 otherwise.
- Assuming the regions are disjoint, the weight associated with a component/region is exactly the predicted value.
- As you make the regions smaller, then you have more features, and the expressivity of your hypothesis class increases. In the limit, you can essentially capture any predictor you want.
- Advanced: what happens if x were not a scalar, but a d-dimensional vector? Then if each component gets broken up into B bins, then there will be B^d features! For each feature, we need to fit its weight, and there will in generally be too few examples to fit all the features.

Predictors with periodicity structure

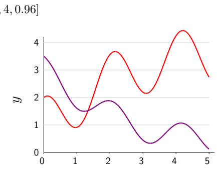
$$\phi(x) = [1, x, x^2, \cos(3x)]$$

Example: $\phi(2) = [1, 2, 4, 0.96]$

$f(x) = [1, 1, -0.1, 1] \cdot \phi(x)$

$f(x) = [3, -1, 0.1, 0.5] \cdot \phi(x)$

$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) : \mathbf{w} \in \mathbb{R}^4\}$



Just throw in any features you want

- Quadratic and piecewise constant predictors are just two examples of an unboundedly large design space of possible feature extractors.
- Generally, the choice of features is informed by the prediction task that we wish to solve (either prior knowledge or preliminary data exploration).
- For example, if x represents time and we believe the true output y varies according to some periodic structure (e.g., traffic patterns repeat daily, sales patterns repeat annually), then we might use periodic features such as cosine to capture these trends.
- Each feature might represent some type of structure in the data. If we have multiple types of structures, these can just be "thrown in" into the feature vector.
- Features represent what properties **might** be useful for prediction. If a feature is not useful, then the learning algorithm can assign a weight close to zero to that feature. Of course, the more features one has, the harder learning becomes.

Linear in what?

Prediction:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

Linear in $\mathbf{w}$? Yes

Linear in $\phi(x)$? Yes

Linear in x? No!

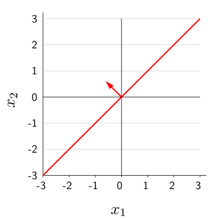
Key idea: non-linearity

- Expressivity: score $\mathbf{w} \cdot \phi(x)$ can be a **non-linear** function of x
- Efficiency: score $\mathbf{w} \cdot \phi(x)$ always a **linear** function of $\mathbf{w}$

- Wait a minute...how are we able to obtain non-linear predictors if we're still using the machinery of linear predictors? It's a linguistic sleight of hand, as "linear" is ambiguous.
- The score is $\mathbf{w} \cdot \phi(x)$ linear in $\mathbf{w}$ and $\phi(x)$. However, the score is not linear in x (it might not even make sense because x need not be a vector at all — it could be a string or a PDF file).
- The significance is as follows: From the feature extractor's viewpoint, we can define arbitrary features that yield very **non-linear** functions in x.
- From the learning algorithm's viewpoint (which only looks at $\phi(x)$, not x), **linearity** us to optimize the weights efficiently.
- Advanced: if the score is linear in $\mathbf{w}$ and the loss function Loss is convex (which holds for the squared, hinge, logistic losses but not the zero-one loss), then minimizing the training loss TrainLoss is a convex optimization problem, and gradient descent with a proper step size is guaranteed to converge to the global minimum.

Linear classification

$$\phi(x) = [x_1, x_2]$$
$$f(x) = \text{sign}([-0.6, 0.6] \cdot \phi(x))$$



Decision boundary is a line

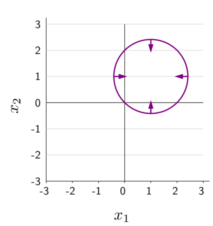
- Now let's turn from regression to classification.
- The story is pretty much the same: you can define arbitrary features to yield non-linear classifiers.
- Recall that in binary classification, the classifier (predictor) returns the sign of the score.
- The classifier can be therefore be represented by its decision boundary, which divides the input space into two regions: points with positive score and points with negative score.
- Note that the classifier $f_{\mathbf{w}}(x)$ is a non-linear function of x (and $\phi(x)$) no matter what (due to the sign function), so it is not helpful to talk about whether $f_{\mathbf{w}}$ is linear or non-linear. Instead we will ask whether the **decision boundary** corresponding to $f_{\mathbf{w}}$ is linear or not.

Quadratic classifiers

$$\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$$
$$f(x) = \text{sign}([2, 2, -1] \cdot \phi(x))$$

Equivalently:

$$f(x) = \begin{cases} 1 & \text{if } \{(x\_1 - 1)^2 + (x\_2 - 1)^2 \le 2\} \\ -1 & \text{otherwise} \end{cases}$$



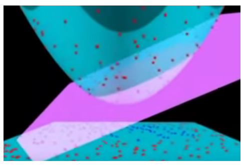
Decision boundary is a circle

- Let us see how we can define a classifier with a non-linear decision boundary.
- Let's try to construct a feature extractor that induces a decision boundary that is a circle: the inside is classified $+1$ and the outside is classified -1.
- We will add a new feature $x_1^2 + x_2^2$ into the feature vector, and define the weights to be as follows.
- Then rewrite the classifier to make it clear that it is the equation for the interior of a circle with radius $\sqrt{2}$.
- As a sanity check, we you can see that $x = [0, 0]$ results in a score of 0, which means that it is on the decision boundary. And as either of x_1 or x_2 grow in magnitude (either $|x_1| \to \infty$ or $|x_2| \to \infty$), the contribution of the third feature dominates and the sign of the score will be negative.

Visualization in feature space

Input space: $x = [x_1, x_2]$, decision boundary is a circle

Feature space: $\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$, decision boundary is a hyperplane



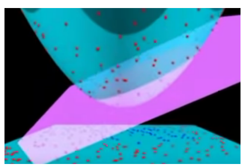
- Let's try to understand the relationship between the non-linearity in x and linearity in $\phi(x)$.
- Click on the image to see the linked video (which is about polynomial kernels and SVMs, but the same principle applies here).
- In the input space x, the decision boundary which separates the red and blue points is a circle.
- We can also visualize the points in **feature space**, where each point is given an additional dimension $x_1^2 + x_2^2$.
- In this three-dimensional feature space, a linear predictor (which is now defined by a hyperplane instead of a line) can in fact separate the red and blue points.
- This corresponds to the non-linear predictor in the original two-dimensional space.

Summary

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

linear in $\mathbf{w}, \phi(x)$

non-linear in x



- Regression: non-linear predictor, classification: non-linear decision boundary
- Types of non-linear features: quadratic, piecewise constant, etc.

Non-linear predictors with linear machinery

- To summarize, we have shown that the term "linear" is ambiguous: a predictor in regression is non-linear in the input x but is linear in the feature vector $\phi(x)$.
- The score is also linear with respect to the weights $\mathbf{w}$, which is important for efficient learning.
- Classification is similar, except we talk about (non-)linearity of the decision boundary.
- We also saw many types of non-linear predictors that you could create by concocting various features (quadratic predictors, piecewise constant predictors).
- So next time someone on the street asks you about linear predictors, you should first ask them "linear in what?"