University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 2: Linear Regression

Lecturer: Alexander Schwing Scribe: Timothy Klem

By the end of this lecture, you should have an understanding of the following:

- How model parameters in linear regression are used to map inputs to outputs
- How these model parameters are selected by minimizing squared error
- How linear regression extends to higher-dimensional inputs
- Why regularization is useful in linear regression
- How linear regression models can be trained on feature vectors
- How linear regression can be formulated by the probability distribution of a dataset
- Why linear regression tends to perform poorly for classification tasks

1 A Simple One-Dimensional Model

For this lecture, we'll take a bottom-up approach to linear regression. We'll first consider simple datasets that we can model, and then we'll augment our model with new features to explain different inputs.

1.1 Framing the Problem

Suppose we have a dataset of N pairs $(x^{(i)}, y^{(i)})$, where $x^{(i)} \in \mathbb{R}$ and $y^{(i)} \in \mathbb{R}$ for all i. We pose the standard *supervised learning* task: we want to develop a model that can relate each **covariate** $x^{(i)}$ to its corresponding **outcome** $y^{(i)}$ in this dataset or similar datasets.

1.1.1 Creating an Error Expression

Let's assume that a linear model is appropriate to link the covariates to the outcomes. Suppose that the model can be expressed in the form

$$y = w_1 \cdot x + w_2$$

for some **model parameters** w_1, w_2. To choose the best parameters, we want to find some w_1, w_2 such that the squared **error**

$$\frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - w_1 \cdot x^{(i)} - w_2 \right)^2$$

is minimized. We sometimes refer to the squared error as a **loss function** because we want to minimize it over some space of constraints. In other words, we want to evaluate the expression

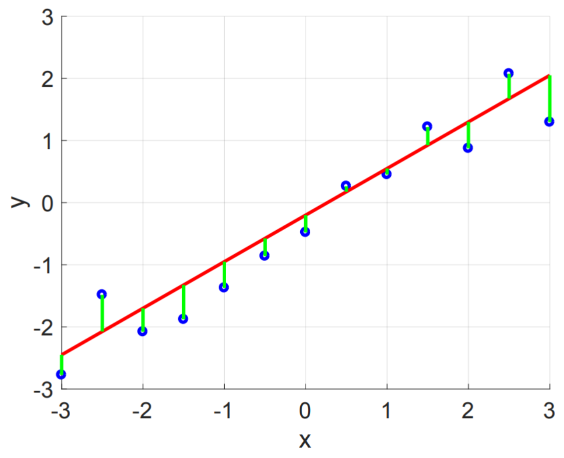
$$\arg\min_{w_1, w_2} \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - w_1 \cdot x^{(i)} - w_2 \right)^2$$

where the argmin operator indicates that we desire the values in the subscript that make the expression a minimum.

Why the $\frac{1}{2}$ in front? We'll eventually want to take the gradient of the expression, so the $\frac{1}{2}$ would help unclutter our the gradient expression by eliminating $2's$ from the expression.

1.1.2 Visualizing Error

We'll use the following plot.



The blue dots represent points in our dataset, and the red line represents the linear model. In linear regression, we wish to minimize the square of the lengths of each green bar. Note that the green lines are vertical because the length of a green bar is the difference between the observed outcome $y^{(i)}$ and the outcome $\hat{y}^{(i)}$ predicted by the linear model. (i.e. The length is the *residual* corresponding to $(x^{(i)}, y^{(i)})$.) In this example, note that the green lines are *not* orthogonal to the red line – we are not trying to minimize the squared distance of all of the data points to our linear model.

1.1.3 Expressing Error Using Vector Notation

It may be helpful to express our loss function

$$\frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - w_1 \cdot x^{(i)} - w_2\right)^2$$

as a series of matrix operations shown by

$$\arg\min_{w_1,w_2} \frac{1}{2}\left\| \underbrace{\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{\mathbf{Y}\in\mathbb{R}^N} - \underbrace{\begin{bmatrix} x^{(1)} & 1 \\ \vdots & \vdots \\ x^{(N)} & 1 \end{bmatrix}}_{\mathbf{X}^T\in\mathbb{R}^{N\times 2}} \cdot \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{\mathbf{w}\in\mathbb{R}^2} \right\|_2^2.$$

In this course, we usually represent vector-valued inputs as column vectors. So, when we coalesce all of the input into a single matrix $\mathbf{X}$, we take the transpose of the matrix so that each *row* of the matrix represents a unique input into the model.

The extra column of 1's in $\mathbf{X}^T$ are needed to represent the subtraction of w_2 in the summation form of the loss function.

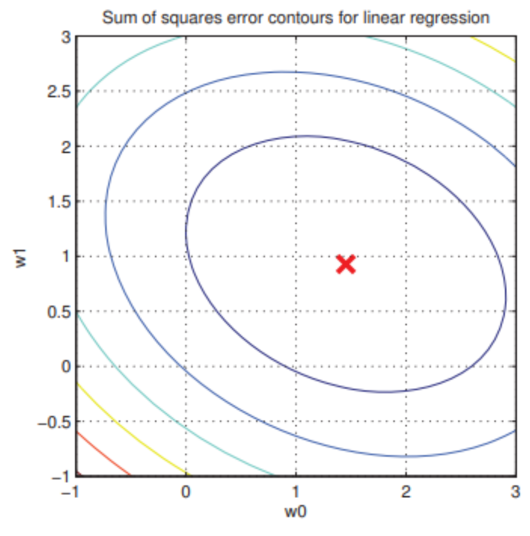
The subscript 2 at the end of the expression indicates that we want to take the 2-norm of the result $\mathbf{Y} - \mathbf{X}^T\mathbf{w}$. The 2-norm is simply the square root of the dot product of a vector with itself.

Now, we can construct the expression to obtain our model parameters (now grouped together in a vector $\mathbf{w}$) as

$$\arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{Y} - \mathbf{X}^T\mathbf{w}\|_2^2$$

1.2 Solving for the Loss Function Minimum

Most loss functions of the form $\frac{1}{2}\|\mathbf{Y} - \mathbf{X}^T\mathbf{w}\|_2^2$ with matrix sizes shown in section 1.1.3 have contour plots whose shape mirrors the one below. The plot was taken from section 7.3 of [1].



(b)

Therefore, we can solve for the solution $\mathbf{w}^*$ that minimizes our loss function by simply setting the derivative of the loss with respect to $\mathbf{w}$ to zero.

Theorem 1 *The optimal* $\boldsymbol{w}$ *that minimizes the loss function for linear regression is* $\boldsymbol{w}^* = \left(\boldsymbol{X}\boldsymbol{X}^T\right)^{-1}\boldsymbol{X}\boldsymbol{Y}$*, assuming that the inverse of* $\boldsymbol{X}\boldsymbol{X}^T$ *exists.*

Proof: The loss function can be rewritten as follows.

$$\begin{aligned}
\frac{1}{2}\|\mathbf{Y} - \mathbf{X}^T\mathbf{w}\|_2^2 &= \frac{1}{2}(\mathbf{Y} - \mathbf{X}^T\mathbf{w})^T(\mathbf{Y} - \mathbf{X}^T\mathbf{w}) \\
&= \frac{1}{2}(\mathbf{Y}^T - \mathbf{w}^T\mathbf{X})(\mathbf{Y} - \mathbf{X}^T\mathbf{w}) \\
&= \frac{1}{2}(\mathbf{Y}\mathbf{Y}^T - \mathbf{Y}^T\mathbf{X}^T\mathbf{w} - \mathbf{w}^T\mathbf{X}\mathbf{Y} + \mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w})
\end{aligned}$$

Each of the matrix products in the last line are 1-by-1 matrices, so $\mathbf{Y}^T\mathbf{X}^T\mathbf{w} = (\mathbf{Y}^T\mathbf{X}^T\mathbf{w})^T =$

$\mathbf{w}^T\mathbf{X}\mathbf{Y}$. Then, the equation for the loss function can be simplified to

$$\begin{aligned}\frac{1}{2}\|\mathbf{Y} - \mathbf{X}^T\mathbf{w}\|_2^2 &= \frac{1}{2}(-2\mathbf{w}^T\mathbf{X}\mathbf{Y} + \mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}) \\ &= -\mathbf{w}^T\mathbf{X}\mathbf{Y} + \frac{1}{2}\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}\end{aligned}$$

Differentiating with respect to $\mathbf{w}$ and setting the expression equal to zero yields

$$\begin{aligned}-\mathbf{X}\mathbf{Y} + \mathbf{X}\mathbf{X}^T\mathbf{w}^* &= 0 \\ \mathbf{X}\mathbf{X}^T\mathbf{w}^* &= \mathbf{X}\mathbf{Y} \\ \mathbf{w}^* &= \left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{X}\mathbf{Y}\end{aligned}$$

It's nice that we are able to derive a closed form solution for $\mathbf{w}^*$, but in practice gradient descent algorithms are commonly used to compute $\mathbf{w}^*$.

2 Extending Our Model

2.1 Higher Dimensional Inputs

2.1.1 Analogous Structures

We now wish to construct a linear model for a dataset of N pairs $(\mathbf{x}^{(i)}, y^{(i)})$ where $\mathbf{x}^{(i)}$ is a d-dimensional real-valued vector and $y^{(i)} \in \mathbb{R}$ for all i. Our linear model will now consider each $y^{(i)}$ to be some linear combination of the elements x_k in the vector $\mathbf{x}^{(i)}$, plus a constant. The model we seek can be written as

$$y = w_0 + \sum_{k=1}^{d} \mathbf{x}_k w_k$$

Notice that now the constant term is the lowest-indexed w_i in the model. This affects how we apply subscripts in matrix form. Similar to before, we can construct a loss function by summing the squares of the errors. We obtain the model parameters $w_0, w_1, \ldots, w_d$ by evaluating the corresponding argmin expression.

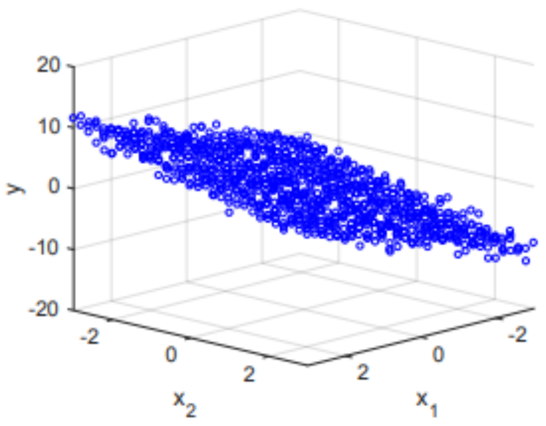
$$\arg\min_{w_0, w_1, \ldots, w_d} \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - w_0 - \sum_{k=1}^{d} w_k \cdot x_k^{(i)}\right)^2$$

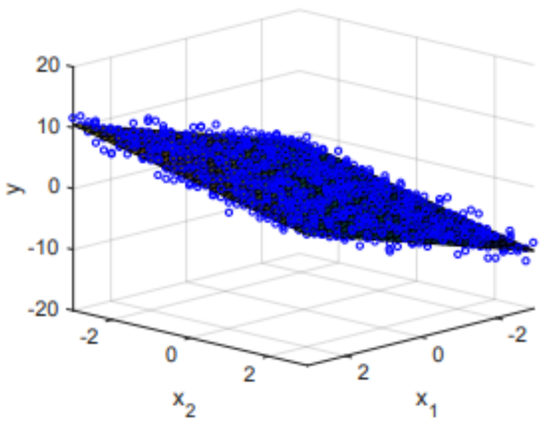
We rewrite the expression in matrix form.

$$\arg\min_{w_0, w_1, \ldots, w_d} \frac{1}{2}\left\| \underbrace{\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{\mathbf{Y}\in\mathbb{R}^N} - \underbrace{\begin{bmatrix} x_d^{(1)} & \cdots & x_1^{(1)} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_d^{(N)} & \cdots & x_1^{(N)} & 1 \end{bmatrix}}_{\mathbf{X}^T\in\mathbb{R}^{N\times(d+1)}} \cdot \underbrace{\begin{bmatrix} w_d \\ \vdots \\ w_1 \\ w_0 \end{bmatrix}}_{\mathbf{w}\in\mathbb{R}^{d+1}} \right\|_2^2$$

Note that there are $d+1$ elements in $\mathbf{w}$ – one element to describe a weight for each feature in the input vector, and one element to describe the *bias term* w_0. Because of this, $\mathbf{X}^T$ has dimensions $N \times (d+1)$ and has its last column to filled with 1's to be multiplied against the bias term.

In higher dimensions, the graphical representation of the linear model changes from a line to a hyperplane.





The plots show a dataset consisting of two dimensional covariates $(x_1^{(i)}, x_2^{(i)})$ and outcomes $y^{(i)}$. The plot on left only shows the dataset, while the plot on right also shows the hyperplane representing the linear model. Like before, the hyperplane parameters are tuned so that the sum of the squares of the *vertical distances* between a data point and the hyperplane for each point is minimized.

2.1.2 Inversion Problem

Since we can describe the loss function in the form $\frac{1}{2}\|\mathbf{Y} - \mathbf{X}^T\mathbf{w}\|_2^2$, we can use the theorem from section 1.2 to show that the optimal solution that minimizes loss is $\mathbf{w}^* = \left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{X}\mathbf{Y}$.

There is one problem: $\mathbf{X}\mathbf{X}^T$ might not be invertible! This occurs when $N < d + 1$, causing $\mathbf{X}^T$ to not have full rank. As a result, there would be no closed solution to the optimization problem.

There were a couple of remedies we proposed in class to solve this issue:

- Use more data, increasing the size of N.
- Reduce dimensionality of the input, decreasing the size of d.
- Instead of an inverse matrix, use the pseudo-inverse to compute $\mathbf{w}^*$.
- Regularize the loss function.

Of these strategies, we decided to dive deeply into the last one.

2.2 Regularization

2.2.1 New Cost Function

In linear regression, we are concerned with creating a model that overfits our dataset and does not generalize well to other datasets. Many times, overfitting occurs when our linear model chooses parameters that are very extreme. Therefore, one heuristic that we can use to limit overfitting is to introduce a penalty in our loss function for parameter vectors whose norms are very large.

Thus, we add a regularization term to our cost function from section 2.1 to obtain

$$\frac{1}{2}\|\mathbf{Y}-\mathbf{X}^T\mathbf{w}\|_2^2+\frac{C}{2}\|\mathbf{w}\|_2^2,$$

where C is a positive regularization constant that we can tune to control $\mathbf{w}$. Many texts and references such as [1] refer to this new regression model as **ridge regression**. When C is small, the chosen model parameter $\mathbf{w}^*$ does not change much from the one chosen in linear regression. However, when C is large, the chosen $\mathbf{w}^*$ begins to approach zero.

2.2.2 Fixing Invertibility Issues

Surprisingly, ridge regression also solves the problem in section 2.1.2 by providing an analytic solution whose matrix inverses can always be computed. With the addition of the $\frac{C}{2}\|\mathbf{w}\|_2^2$ term in the loss function, the derivative of the loss with respect to $\mathbf{w}$ is

$$-\mathbf{X}\mathbf{Y}+\mathbf{X}\mathbf{X}^T\mathbf{w}+C\mathbf{w}.$$

Solving for the optimal $\mathbf{w}^*$ when the derivate is set to zero yields

$$\mathbf{w}^*=\left(\mathbf{X}\mathbf{X}^T+C\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{Y}.$$

$\mathbf{X}\mathbf{X}^T$ can be shown to be positive semidefinite and have all nonnegative eigenvalues. Therefore, the matrix

$$\mathbf{X}\mathbf{X}^T+C\mathbf{I}$$

must have all eigenvalues greater than or equal to C, so the matrix is invertible and $\mathbf{w}^*$ can always be computed.

2.3 Using Feature Vectors

2.3.1 Higher Dimensional Polynomials

We'll return to our basic one-dimensional covariates $x^{(i)}\in\mathbb{R}$ and consider constructing an $(M-1)$th degree polynomial model to link the covariates to their outcomes. The polynomial model would take the form

$$y=w_{M-1}x^{M-1}+\ldots+w_1x+w_0.$$

Like before, we can construct the loss function by summing the squares of the errors. We obtain the model parameters by evaluating the corresponding argmin expression.

$$\arg\min_{w_0,w_1,\ldots,w_{M-1}}\frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)}-w_0-\sum_{k=1}^{M-1}w_k\cdot(x^{(i)})^k\right)^2$$

Like before, we can construct this expression in matrix form.

$$\arg\min_{w_0,w_1,\ldots,w_{M-1}}\frac{1}{2}\left\|\underbrace{\begin{bmatrix}y^{(1)}\\ \vdots\\ y^{(N)}\end{bmatrix}}_{\mathbf{Y}\in\mathbb{R}^N}-\underbrace{\begin{bmatrix}(x^{(1)})^{M-1} & \cdots & x^{(1)} & 1\\ \vdots & \ddots & \vdots & \vdots\\ (x^{(N)})^{M-1} & \cdots & x^{(N)} & 1\end{bmatrix}}_{\Phi^T\in\mathbb{R}^{N\times M}}\cdot\underbrace{\begin{bmatrix}w_{M-1}\\ \vdots\\ w_1\\ w_0\end{bmatrix}}_{\mathbf{w}\in\mathbb{R}^M}\right\|_2^2$$

Instead of using the symbol $\mathbf{X}$ to represent our data in the least squares expression, we use Φ because we applied some transformation to our covariates before composing the matrix to be multiplied with $\mathbf{w}$.

Using an argument very similar to before, we can determine analytically that the optimal solution $\mathbf{w}^*$ that solves our least squares problem here has the form

$$\mathbf{w}^* = \left(\Phi\Phi^T\right)^{-1} \Phi\mathbf{Y}.$$

2.3.2 Feature Vectors

Our polynomial model is a specific case of transforming our input vectors $x^{(i)}$ into feature vectors. We can define the transformation for input $x^{(i)}$ as $\phi(x^{(i)}) \in \mathbb{R}^M$. The $(M-1)$th degree polynomial model earlier performs the transformation

$$\phi(x^{(i)}) = [(x^{(i)})^{M-1}, \dots, x^{(i)}, 1]^T.$$

For a generalized feature vector, the linear model whose parameters $\mathbf{w}$ we want to determine will have the form

$$y = \phi(x)^T \mathbf{w}.$$

$\phi(x)$ could be used to create a bias term in the linear model by producing a 1 in the same index of the feature vector for all inputs. Like before, we can construct the argmin expression to find the model parameters.

$$\arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} \left(y^{(i)} - \phi(x^{(i)})^T \mathbf{w}\right)^2$$

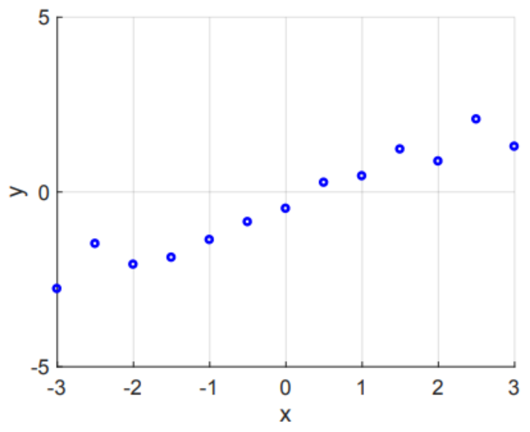
We can reuse the loss function in matrix form from the polynomial model to express our generalized model that uses feature vectors. Now, we define Φ as

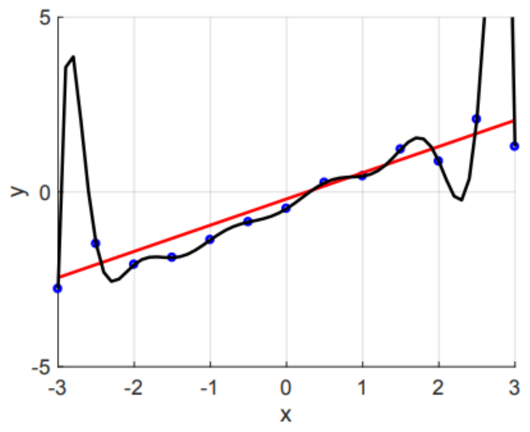
$$\Phi = [\phi(x^{(1)}), \dots, \phi(x^{(N)})]$$

where each $\phi(x^{(i)})$ is a column in Φ. Like before, the analytic solution to the general model is $\mathbf{w}^* = \left(\Phi\Phi^T\right)^{-1} \Phi\mathbf{Y}$.

2.3.3 Overfitting Risk

Consider the plots below where we fit a dataset with a linear model in red and a higher-dimensional polynomial model in black.





We can observe that polynomial model overfits the dataset and would not generalize to a different set of samples. The polynomial model is one of many cases where constructing more features from our input data would likely cause our model to overfit the dataset.

3 Probabilistic Model of Linear Regression

So far, we've developed our linear regression model by first modelling our outcomes as a linear combination of elements in feature vectors and then minimizing our hand-crafted loss function. Now, we will describe our outcomes as samples from a probability distribution that uses a similar linear combination. Then, we will choose model parameters that maximize the likelihood of observing the outcomes in our dataset.

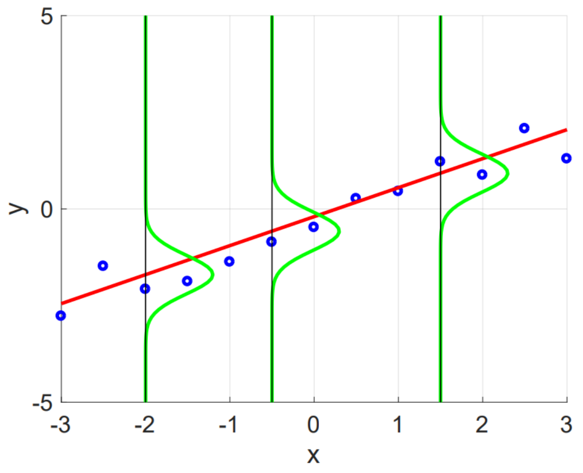
3.1 Gaussian Model

We will model the output y given an input vector x to follow a Gaussian distribution whose mean is $\mathbf{w}^T\phi(x)$ and standard deviation is an arbitrary σ. We assume no prior knowledge of the distribution of $\mathbf{w}$ or x.

The probability density function (pdf) of $p(y|x)$ can be written as

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{1}{2\sigma^2}(y - \mathbf{w}^T\phi(x))).$$

We can visualize the pdf of $p(y|x)$ using the following plot.



Here, we use the same one-dimensional dataset used in the beginning of these notes. The green curves represent the Gaussian distribution for observed inputs $x = -2$, $x = -0.5$, and $x = 1.5$. Each curve's peak aligns with the intersection of a vertical line and the red model because each curve's mean is equal to $\mathbf{w}^T\phi(x)$, the output value predicted by the linear model. Each curve also has the same shape because all of them share the same standard deviation.

3.2 Maximizing Likelihood

We will assume that each sample in the dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}$ of N samples is drawn independently from the identical conditional Gaussian distribution with pdf $p(y|x)$. Then, we can construct the probability $p(\mathcal{D})$ of observing the dataset as follows.

$$\begin{aligned}
p(\mathcal{D}) &= \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}) \\
&= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y^{(i)} - \mathbf{w}^T\phi(x^{(i)}))\right) \\
&= \frac{1}{(2\pi\sigma^2)^{N/2}} \prod_{i=1}^{N} \exp\left(-\frac{1}{2\sigma^2}(y^{(i)} - \mathbf{w}^T\phi(x^{(i)}))\right) \\
&= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^T\phi(x^{(i)}))\right) \\
&= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{Y} - \Phi^T\mathbf{w}\|_2^2\right)
\end{aligned}$$

We wish to find the parameter $\mathbf{w}$ that maximizes the likelihood $p(\mathcal{D})$. The coefficients $\frac{1}{(2\pi\sigma^2)^{N/2}}$ and $-\frac{1}{2\sigma^2}$ are constant with respect to $\mathbf{w}$, so $p(\mathcal{D})$ is maximized when the expression $\|\mathbf{Y} - \Phi^T\mathbf{w}\|_2^2$ is minimized. More succinctly,

$$\arg\min_{\mathbf{w}} p(\mathcal{D}) = \arg\max_{\mathbf{w}} \|\mathbf{Y} - \Phi^T\mathbf{w}\|_2^2.$$

Therefore, maximizing the dataset's likelihood in the probabilistic view of linear regression produces the same model parameters as those found when minimizing the loss function in the loss view.

4 Linear Regression for Classification Tasks

4.1 New Model

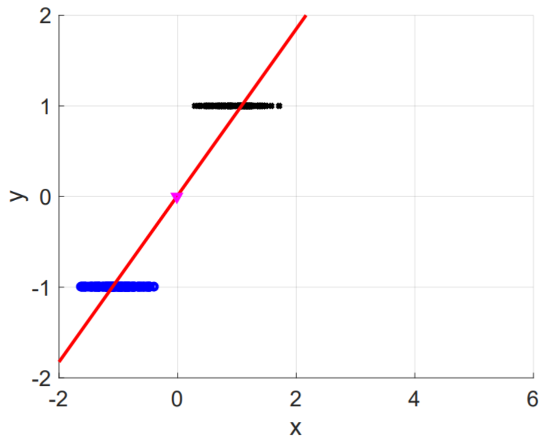
For now, we wish to explore whether linear regression is suitable for binary classification tasks. In this scenario, our dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}$ is similar to before where each $x^{(i)}, y^{(i)} \in \mathbb{R}$, except that every outcome $y^{(i)}$ is a member of the set $\{-1, 1\}$. To map our real-valued predictions generated from linear regression, we will threshold the output of our linear model at $y = 0$. Our model for classification now becomes

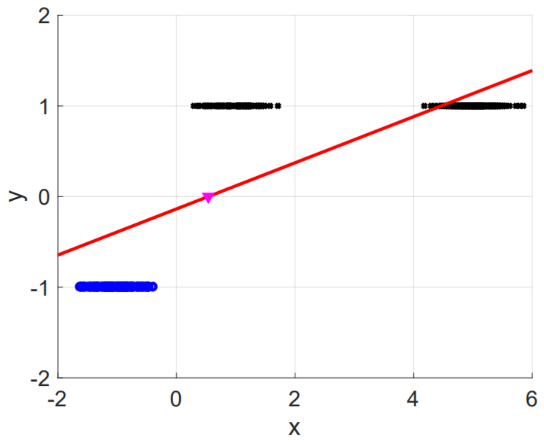
$$y = \operatorname{sgn}(w_1 \cdot x + w_0)$$

where w_1, w_0 are model parameters that minimize the squared loss that we've seen before.

4.2 Performance

We generate two datasets where an ideal classifier will label all $x^{(i)} > 0$ with 1 and all $x^{(i)} < 0$ with -1. A sample $(x^{(i)}, y^{(i)})$ is colored black if $y^{(i)} = 1$ and colored blue if $y^{(i)} = -1$. The small pink triangle identifies where the linear model intersects the x-axis. In both plots below, all data points to the left of the triangle are predicted to have an outcome of -1, and all points to the right are predicted to have outcome 1.





In the picture on the left, we see that the linear model with a threshold correctly labels all of the data because the pink triangle is located on the line $x = 0$. However, the addition of more samples labelled 1 in the plot of the right causes the decision boundary to shift to the right. The pink triangle is now located on $x = 0.4$, so some of the black data points near $x = 0$ are incorrectly classified as -1 by our model.

4.3 Why the Model Performs Poorly

For classification tasks, let's define a function $F(x^{(i)}, \mathbf{w})$ that generates a prediction and its strength for $y^{(i)}$ using covariate $x^{(i)}$ and a model with parameter vector $\mathbf{w}$. The prediction may need a threshold to map to a possible label for output $y^{(i)}$. For our linear models, $F(x^{(i)}, \mathbf{w}) = \phi(x^{(i)})^T \mathbf{w}$. Let's further define a *score function* $F(x^{(i)}, \mathbf{w}, y^{(i)})$ that indicates how well a model with parameter $\mathbf{w}$ classified a given covariate $x^{(i)}$ whose label should be identical to $y^{(i)}$.

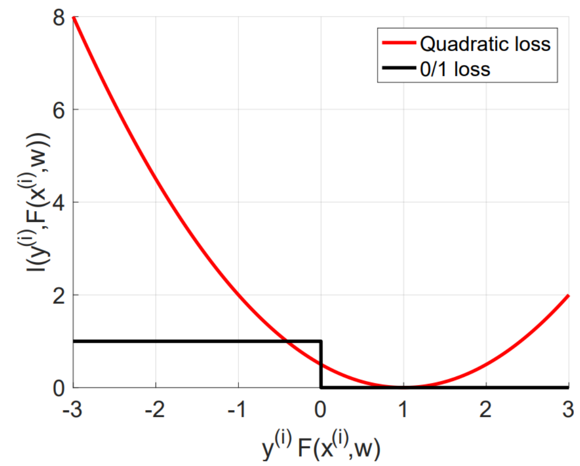
For binary classification, it is suitable to further define $F(x^{(i)}, \mathbf{w}, y^{(i)}) = y^{(i)}F(x^{(i)}, \mathbf{w})$. $F(x^{(i)}, \mathbf{w})$ would assign positive values to predict that the outcome should be 1 and negative values for -1. Thus, the product $y^{(i)}F(x^{(i)}, \mathbf{w})$ would be positive for correct classification and negative for incorrect classification.

Let $\ell(y^{(i)}, F(x^{(i)}, \mathbf{w}))$ represent the contribution to the dataset's loss function by one sample $(x^{(i)}, y^{(i)})$. When we are modelling data using linear regression, we term $\ell(y^{(i)}, F(x^{(i)}, \mathbf{w}))$ as the quadratic loss for a sample. Ideally, we want each sample to contribute a 0/1 loss to the overall dataset's loss function. A 0/1 loss assigns zero loss to a correct classification and a loss of 1 to incorrect classification. This way we would punish all misclassifications evenly and do not deter our model from classifying data correctly.

Now, we want to examine the relationship between the score function $y^{(i)}F(x^{(i)}, \mathbf{w})$, which indicates whether a correct classification for a sample has been made and how difficult it was to make the judgment, and the loss $\ell(y^{(i)}, F(x^{(i)}, \mathbf{w}))$, which indicates the relative need for model parameters to change due to a sample. For quadratic loss, note that $(y^{(i)})^2 = 1$ for all samples in our binary classification task, so we can establish the following relationship between the score and loss.

$$\begin{aligned}
\ell(y^{(i)}, F(x^{(i)}, \mathbf{w})) &= \frac{1}{2}(y^{(i)} - \phi(x^{(i)})^T\mathbf{w})^2 \\
&= \frac{1}{2}((y^{(i)})^2 - 2y^{(i)}\phi(x^{(i)})^T\mathbf{w} + (\phi(x^{(i)})^T\mathbf{w})^2) \\
&= \frac{1}{2}(1 - 2y^{(i)}\phi(x^{(i)})^T\mathbf{w} + ((y^{(i)})^2)(\phi(x^{(i)})^T\mathbf{w})^2) \\
&= \frac{1}{2}(1 - y^{(i)}\phi(x^{(i)})^T\mathbf{w})^2
\end{aligned}$$

For 0/1 loss, we want $\ell(y^{(i)}, F(x^{(i)}, \mathbf{w})) = 1$ when $y^{(i)}F(x^{(i)}, \mathbf{w}) < 0$ and misclassification occurs, and $\ell(y^{(i)}, F(x^{(i)}, \mathbf{w})) = 0$ otherwise. We now plot the relationship between score and loss for 0/1 and quadratic cases.



From the plot, we can pose that it is difficult to develop a learning model for 0/1 loss because the plot is discontinuous when the score is equal to zero and flat everywhere else. For quadratic loss, which linear regression uses, notice that loss is increasing when the score is greater than 1 and correct classification occurs. Therefore, linear regression accumulates losses for samples that are correctly and easily classified, so linear regression actually discourages model parameters from strongly classifying a sample correctly. As a result, linear regression is a poor choice for classification tasks.

References

[1] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.