Linear regression
- Training data
- Learning task
  - Error functions
    - Residuals
    - Ordinary least squares (OLS)
  - Bias term
  - Design matrix
    - Solving the OLS problem (with the design matrix)
- Regularization
  - L_1 and L_2 regularized least squares problems
  - Elastic net regularization
    - Comparison with standalone L_1 and L_2 regularization
- Basis functions
  - The identity basis function
  - Polynomial regression
  - Multivariate basis functions

Resources

Linear regression

Linear regression addresses the supervised learning problem of approximating the relationship between the **input variables** and **output variables** of some data.

Training data

Training data for linear regression problems comes in the form:

$$\mathcal{D}_{\text{train}} = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$$

> **Where**:
>
> - $\mathbf{x}^{(i)} \in \mathbb{R}^D$, a D-dimensional vector of real numbers, $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_D^{(i)} \right)^\mathsf{T}$.
> - $y^{(i)} \in \mathbb{R}$

This is typically represented as an $N \times D$ matrix of **feature vectors** $\mathbf{X} = \left(\mathbf{x}^{(1)^\mathsf{T}}, \ldots, \mathbf{x}^{(N)^\mathsf{T}} \right)^\mathsf{T}$ and a corresponding **output vector** $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(N)} \right)^\mathsf{T}$.

Each of the columns $\mathbf{x}_j = \left(x_j^{(1)}, \ldots, x_j^{(N)} \right)^\mathsf{T}$ of $\mathbf{X}$ represents a **feature**, which is simply a **random variable** (sometimes called an **explanatory variable**). In practicality, these features are often specific attributes of the system or object being modeled, e.g. *height*, *rent*, *temperature*.

$\mathbf{X}$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_D$	$\mathbf{y}$
$\mathbf{x}^{(1)^\mathsf{T}}$	$x_1^{(1)}$	$x_2^{(1)}$	$\cdots$	$x_D^{(1)}$	$y^{(1)}$
$\mathbf{x}^{(2)^\mathsf{T}}$	$x_1^{(2)}$	$x_2^{(2)}$	$\cdots$	$x_D^{(2)}$	$y^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbf{x}^{(N)^\mathsf{T}}$	$x_1^{(N)}$	$x_2^{(N)}$	$\cdots$	$x_D^{(N)}$	$y^{(N)}$

The accompanying output variable $\mathbf{y}$ contains the output for each feature vector where the output $y^{(i)}$ is assumed to be a **linear combination** of its feature values $\mathbf{x}^{(i)}$—that is:

$$\begin{aligned} y^{(i)} &= \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_D x_D^{(i)} \\ &= \sum_{j=1}^{D} \theta_j x_j^{(i)} \end{aligned}$$

> **Where**: $\theta_j \in \mathbb{R}$ is an arbitrary coefficient, referred to as a **weight**.

If we let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_D)^\mathsf{T}$ then we can see that the expression for $y^{(i)}$ as a sum of products, can be represented as a dot product:

$$\begin{aligned} y^{(i)} &= \sum_{j=1}^{D} \theta_j x_j^{(i)} \\ &= \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_D \end{pmatrix} \cdot \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_D^{(i)} \end{pmatrix} \\ &= \boldsymbol{\theta} \cdot \mathbf{x}^{(i)} \\ &= \boldsymbol{\theta}^\mathsf{T} \mathbf{x}^{(i)} \end{aligned}$$

> **Where**: $\boldsymbol{\theta}$ is referred to as the **weight vector**.

Which further allows us to represent $\mathbf{y}$ more concisely, as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta}$$

Note: Remember that $\boldsymbol{\theta}$ is the same for each feature vector, since it's trying to model a linear relation between the features—**not** the feature vectors!

Learning task

Suppose we have training data $\mathcal{D}_{\text{train}} = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$.

Linear regression aims to learn from this data in order to create a **regression line**, $\hat{\mathbf{y}}$. This line can then be used to estimate the value of the output variable $y^{(k)}$ for some new unlabeled feature vector $\mathbf{x}^{(k)}$.

Note: The regression line is simply a D-dimensional hyperplane in D-dimensional feature space. This is often called a **line-of-best-fit**.

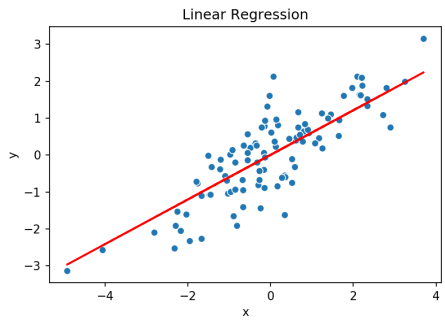


Figure 1: Search for an optimal regression line in one-dimentional feature space. (source)

Error functions

In order to determine which of the infinitely many regression lines have the least *error*, we need to introduce **error functions**.

An **error function** (also called **cost function**) for linear regression should act as a form of aggregate measure of how far the output values of the training samples are, from the values that would be predicted by the regression line—that is, for some regression line with fixed $\boldsymbol{\theta}$, how far are the predicted outputs (lying on the hyperplane) $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$ from the actual output values, $\mathbf{y}$.

The optimum regression line should have the **minimal** error—which in turn makes this a minimization problem.

Residuals

A **residual** is the error in a single result—how far an individual predicted $\hat{y}^{(i)}$ is from the actual output $y^{(i)}$.

The most commonly used residuals are **vertical-offset**, only considering the distance in the plane of the output variable.

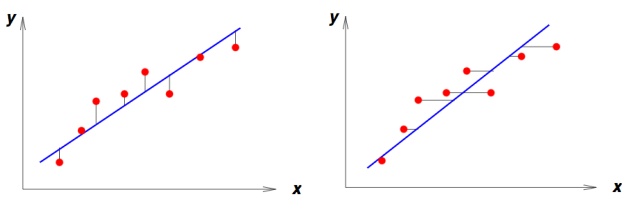


Figure 2: Vertical and horizontal offset residuals. (source)

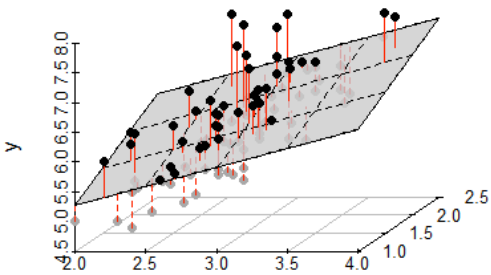


Figure 3: Vertical-offset residuals in a two-dimensional feature space. (source)

Ordinary least squares (OLS)

Ordinary least squares (OLS) is a form of regression analysis that uses the **sum-squared error** (or **residual sum of squares (RSS)**) function as a cost function.

Sum-squared error is simply the sum of the squared (vertical-offset) residual lengths:

$$\begin{aligned} C(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \\ &= \sum_{i=1}^{N} \left(y^{(i)} - \boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}^{(i)} \right)^2 \end{aligned}$$

> **Note**: We square the lengths because the subtraction may result in a negative number.

Which can be represented as $(\mathbf{y} - \hat{\mathbf{y}})^{\mathsf{T}} (\mathbf{y} - \hat{\mathbf{y}})$ in matrix notation, giving us:

$$C(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^{\mathsf{T}} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

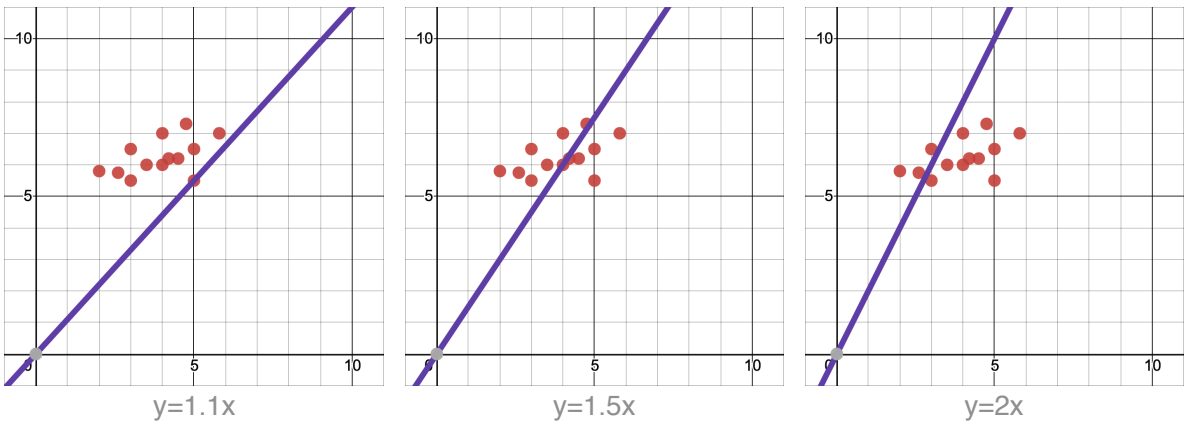
Then the optimum weight vector $\hat{\boldsymbol{\theta}}$ is given by minimising this cost function:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) \\ &= \arg\min_{\boldsymbol{\theta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^{\mathsf{T}} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \end{aligned}$$

Bias term

Despite there being infinitely many possible values for $\boldsymbol{\theta}$, the hyperplane is restricted to passing through the origin $\mathbf{0}$—as there is no intercept term in $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$.

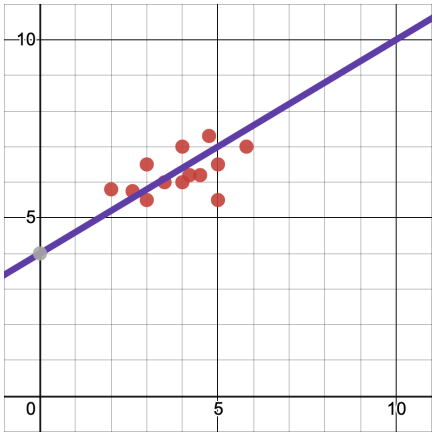
This can often be a very limiting restriction, as it essentially means that the hyperplane cannot be translated on the $\mathbf{y}$ plane. This may make it difficult to create a *good* regression line from the training data.

> **Example**: Due to the origin restriction, it is not possible to create an optimal regression line for the following collection of one-dimensional feature vectors.
>
> 
>
>
> **Figure 4**: Sub-optimal origin-restricted regression lines for one-dimensional feature vectors.

If an intercept term θ_0 (called a **bias**) is incorporated into the regression line $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} + \theta_0$, it becomes possible for the regression line to move around in space more freely.

In practicality, we will usually want a bias for our linear model anyway.

> **Example**: If we are modelling the prediction the price of a house, $y^{(i)}$, with explanatory variables valued at $x_1^{(i)}, \ldots, x_D^{(i)}$, we don't want the house price to be 0 when all of the explanatory variables are valued at 0.
>
> For example, when the explanatory variables are all zero, we want the default house price will be $\$1000$. The bias term allows us to capture this information by defining the value of the regression line when the explanatory variables are all zero—in other words, the point of intercept with the $\mathbf{y}$ axis.



y=0.6x+4

Figure 5: A better-fitting regression line as a result of the bias term.

Design matrix

In order for the bias term introduced above to be consistent with the previous linear algebra, a few changes need to be made to the representations of matrices and vectors, to ensure that dot products to remain well-defined.

With the introduction of the bias term θ_0, an individual training example can be represented linearly as:

$$\begin{aligned} y^{(i)} &= \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_D x_D^{(i)} \\ &= \theta_0 + \sum_{j=1}^{D} \theta_j x_j^{(i)} \end{aligned}$$

Prior to the introduction of the bias term, we could express this as a dot product. However, the θ_0 term in the expression above is **no longer** attached to the value of any explanatory variable $x_j^{(i)}$.

We can address this issue by extending the feature vector $\mathbf{x}^{(i)}$ to have 1 as it's first element, $x_0^{(i)}$. Let this new vector be $\phi\left(\mathbf{x}^{(i)}\right)$:

$$\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right) = \underbrace{\left(1, x_1^{(i)}, \ldots, x_D^{(i)}\right)^\mathsf{T}}_{D+1}$$

If we express $y^{(i)}$ as a linear combination of the explanatory variables as defined in $\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)$, we can now say:

$$\begin{aligned}
y^{(i)} &= \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_D x_D^{(i)} \\
&= \theta_0 \cdot 1 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_D x_D^{(i)} \\
&= \theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_D x_D^{(i)} \\
&= \sum_{j=0}^{D} \theta_j x_j^{(i)}
\end{aligned}$$

Further if we treat the bias term θ_0 as an extension of the weight vector $\boldsymbol{\theta}$, so that $\boldsymbol{\theta} = \underbrace{(\theta_0, \ldots, \theta_D)^\mathsf{T}}_{D+1}$ we can express this as a dot product:

$$\begin{aligned}
y^{(i)} &= \sum_{j=0}^{D} \theta_j x_j^{(i)} \\
&= \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_D \end{pmatrix} \cdot \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_D^{(i)} \end{pmatrix} \\
&= \boldsymbol{\theta}^\mathsf{T} \boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)
\end{aligned}$$

The modified feature vectors $\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right)$ can be represented by a new matrix—known as the **design matrix**, $\boldsymbol{\Phi}$:

$\boldsymbol{\Phi}$	$\mathbf{x}_0$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\cdots$	$\mathbf{x}_D$	$\mathbf{y}$
$\boldsymbol{\phi}\left(\mathbf{x}^{(1)}\right)^\mathsf{T}$	1	$x_1^{(1)}$	$x_2^{(1)}$	$\cdots$	$x_D^{(1)}$	$y^{(1)}$
$\boldsymbol{\phi}\left(\mathbf{x}^{(2)}\right)^\mathsf{T}$	1	$x_1^{(2)}$	$x_2^{(2)}$	$\cdots$	$x_D^{(2)}$	$y^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\boldsymbol{\phi}\left(\mathbf{x}^{(N)}\right)^\mathsf{T}$	1	$x_1^{(N)}$	$x_2^{(N)}$	$\cdots$	$x_D^{(N)}$	$y^{(N)}$

Using the previous result along with matrix algebra, we can represent the output vector $\mathbf{y}$ as:

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta}$$

Solving the OLS problem (with the design matrix)

We previously saw that the optimal $\hat{\boldsymbol{\theta}}$ with no bias term is given by solving the following minimization problem for OLS:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^{\mathsf{T}} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})
\end{aligned}
$$

By extending $\boldsymbol{\theta}$ with the bias term θ_0 and using the design matrix $\boldsymbol{\Phi}$ instead of $\mathbf{X}$, this becomes:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^{\mathsf{T}} (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta})
\end{aligned}
$$

Minimization for OLS has a **closed-form (analytical) solution** which can be derived by taking partial derivatives with respect to $\boldsymbol{\theta}$ and setting them to 0. This leads to the following analytical solution for the optimal weight vector $\hat{\boldsymbol{\theta}}$:

$$
\hat{\boldsymbol{\theta}} = \underbrace{\left(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathsf{T}}}_{\text{pseudo-inverse}}\mathbf{y}
$$

> **Where**: $\left(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathsf{T}}$ is referred to as the **pseudo-inverse** of $\boldsymbol{\Phi}$.
>
> This is not the actual inverse matrix since $\boldsymbol{\Phi}$ is **not invertible** as it is not square, since it has shape $N \times (D+1)$.

Note: Although OLS has an analytical solution, it is also possible to use other iterative optimization methods such as **gradient descent**, **stochastic gradient descent**, **BFGS**, etc. to minimize the cost function. However, these methods are **not** guaranteed to converge or find a global minimum.

Regularization

Regularization refers to various methods used to penalize specific terms in a cost function in order to prevent overfitting to the training data. This is done by adding a **regularization term** (also called a **regularizer**).

Regularization discourages or decreases the complexity of a linear model.

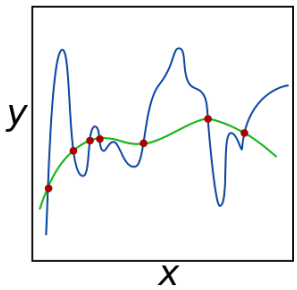


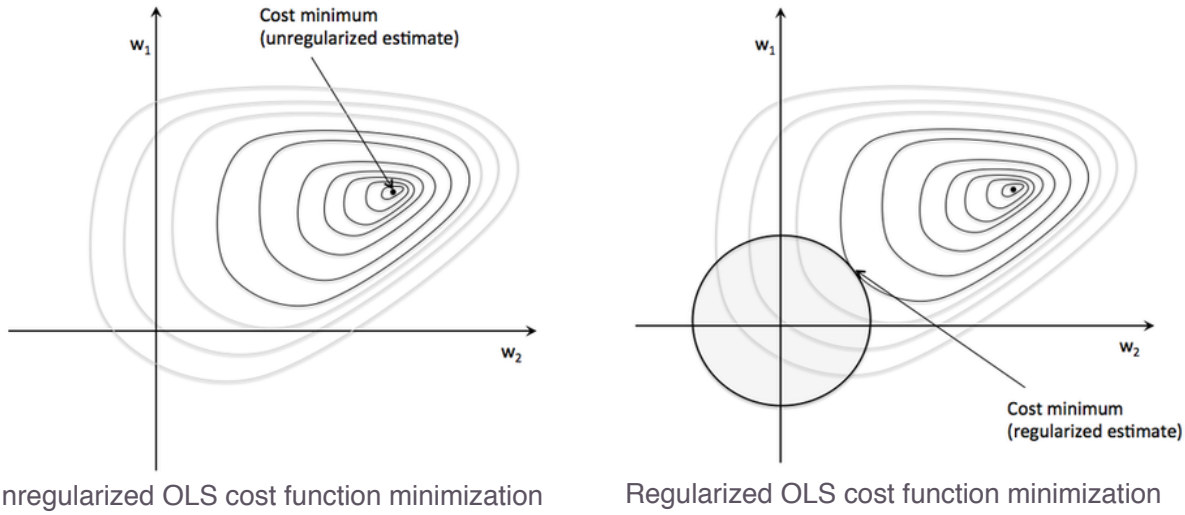
Figure 6: Simplification of a polynomial regression model's *complexity* as a result of regularization. *Blue represents the unregularized and overfitted model, and green represents a regularized model which generalizes better.* (source)

For least squares problems, the regularized cost function looks like:

$$C(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(y^{(i)} - \boldsymbol{\theta}^{\mathsf{T}} \mathbf{x}^{(i)} \right)^2 + \lambda \underbrace{R(\boldsymbol{\theta})}_{\text{reg. term}}$$

> **Where**:
>
> - λ is a **tuning parameter** that controls the importance of the regularization term—higher λ leads to more penalization. This parameter is selected through cross-validation.
> - $R(\boldsymbol{\theta})$ is a regularization term chosen to penalize coefficients by a specific quantity—shrinking them towards zero.

Regularization essentially forces the minimization of a cost function within the constraints of the provided regularization term.



Unregularized OLS cost function minimization

Regularized OLS cost function minimization

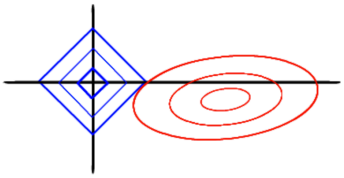
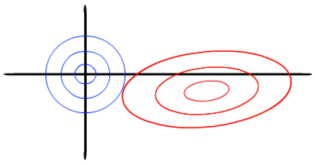
Figure 7: Graphical depiction of the constraint placed on the minimization of the RSS cost function as a result of L2 regularization in a two-dimensional feature space.
This constraint is in place due to the fact that we now have to minimize a combined sum of the RSS and regularization term. To solve this minimization problem we must get as close to the minimum of the RSS contour, whilst still remaining in the constrained region imposed by the regularization term (the circle in this case, but n-sphere in general). (source)

L_1 and L_2 regularized least squares problems

Regularization terms in the form of the L_1 (Manhattan) and L_2 (Euclidean) **norms** are commonly used for linear regression—these norms form the basis for **L_1 and L_2 regularization**.

The table below provides information about both of these regularization methods when used to solve least squares problems:

	L_1	L_2
Name	Lasso	Ridge
Regularization term — $R(\boldsymbol{\theta})$	$\sum_{i=1}^{N} \lvert\theta_i\rvert$ (or $\lvert\lvert\boldsymbol{\theta}\rvert\rvert_1$ — the L_1 norm)	$\sum_{i=1}^{N} \theta_i^2$ (or $\lvert\lvert\boldsymbol{\theta}\rvert\rvert_2^2$ — the squared L_2 norm)

***Note**: Observe that we don't penalize the bias term, θ_0.*		
Regularized least squares $- C(\boldsymbol{\theta})$	$\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^{(i)}\right)^2 + \lambda\sum_{i=1}^{N}\lvert\theta_i\rvert$	$\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^{(i)}\right)^2 + \lambda\sum_{i=1}^{N}\theta_i^2$
Analytic solution	None—use iterative optimization methods.	$\hat{\boldsymbol{\theta}} = \left(\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \lambda\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\mathbf{y}$
Affected weights	All weights—uniformly.	All weights—but low valued weights will be penalized less since we are squaring. Conversely, large weights will face more penalty.
When to use	When there are many features which are irrelevant to the output variable—since it **can** shrink them to zero, completely disregarding them. Also works well when $n \gg d$ (number of instances is far greater than the number of features).	When all (or most) features are relevant to the output variable—since it can **not** shrink them to zero, meaning that all features will have **some** impact. Also works better when there is high collinearity between features.
Constraint region visualization **Note**: Regularization constraint region is depicted in red, RSS contour is depicted in blue.		

Elastic net regularization

Elastic net is a compromise regularization method that involves the usage of a regularization term which linearly combines the L_1 and L_2 norms of the weights, using two tuning parameters, λ_1 and λ_2.

The elastic net regularized cost function for least squares problems is given as:

$$C(\boldsymbol{\theta}) = \underbrace{\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^{(i)}\right)^2}_{\text{RSS}} + \lambda_1\underbrace{\sum_{i=1}^{N}\lvert\theta_i\rvert}_{L_1} + \lambda_2\underbrace{\sum_{i=1}^{N}\theta_i^2}_{L_2}$$

Comparison with standalone L_1 and L_2 regularization

This form of regularization is often used to counteract the limitations of the L_1 and L_2 penalties.

- With highly-correlated features:
  - L_1 regularization generally picks one and effectively discards the others by setting their weights to zero. However, it is often difficult to determine which feature was chosen.
  - L_2 regularization shrinks the weights of highly-correlated features towards one another.

  Elastic net is a compromise between the two that attempts to shrink and do a **sparse selection** simultaneously.
- In regards to penalty:
  - L_1 regularization penalizes weights more uniformly.
  - L_2 regularization penalizes higher-valued weights more than the smaller ones.

  Once again, elastic net acts as a compromise between this property of the two regularization methods.

Basis functions

The main requirement for a linear regression model is that the weights must be linear. However, it is not necessary that the explanatory variables are linear—they can be defined by any non-linear function of the explanatory variables too.

This allows us to more generally define a linear model as:

$$\begin{aligned} y^{(i)} &= \theta_0 \phi_0\left(\mathbf{x}^{(i)}\right) + \theta_1 \phi_1\left(\mathbf{x}^{(i)}\right) + \theta_2 \phi_2\left(\mathbf{x}^{(i)}\right) + \cdots + \theta_D \phi_D\left(\mathbf{x}^{(i)}\right) \\ &= \sum_{j=0}^{D} \theta_j \phi_j\left(\mathbf{x}^{(i)}\right) \\ &= \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right) \end{aligned}$$

> **Where**:
>
> - Each ϕ_j is called a **basis function**, which is a function of the current input $\mathbf{x}^{(i)}$.
> - $\boldsymbol{\phi}$ is a vector-valued function such that $\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right) = \left(\phi_0\left(\mathbf{x}^{(i)}\right), \ldots, \phi_D\left(\mathbf{x}^{(i)}\right)\right)^{\mathsf{T}}$.
> - $\phi_0\left(\mathbf{x}^{(i)}\right) = 1$ by convention—so that the bias term is not affected by the basis function.

The identity basis function

The most simple case of a linear regression model that we have seen before is where the output variable may be modeled as:

$$y^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_D x_D^{(i)}$$

This regression model can be defined by the identity basis function:

$$\begin{aligned} \boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right) &= \mathbf{x}^{(i)} \\ &= \left(1, x_1^{(i)}, \ldots, x_D^{(i)}\right)^{\mathsf{T}} \end{aligned}$$

> **Where**: The individual basis functions would be $\phi_j\left(\mathbf{x}^{(i)}\right) = x_j^{(i)}$—a function only of the feature with the same index as the basis function.

This is more clearly seen by looking at the design matrix $\mathbf{\Phi}$ of this basis function when applied to the training set $\mathcal{D}_{\text{train}}$:

$$
\begin{array}{c|cccccc|c}
\mathbf{\Phi} & \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_D & & \mathbf{y} \\
\hline
\boldsymbol{\phi}\left(\mathbf{x}^{(1)}\right)^{\mathsf{T}} & \phi_0\left(\mathbf{x}^{(1)}\right) & \phi_1\left(\mathbf{x}^{(1)}\right) & \phi_2\left(\mathbf{x}^{(1)}\right) & \cdots & \phi_D\left(\mathbf{x}^{(1)}\right) & & y^{(1)} \\
\boldsymbol{\phi}\left(\mathbf{x}^{(2)}\right)^{\mathsf{T}} & \phi_0\left(\mathbf{x}^{(2)}\right) & \phi_1\left(\mathbf{x}^{(2)}\right) & \phi_2\left(\mathbf{x}^{(2)}\right) & \cdots & \phi_D\left(\mathbf{x}^{(2)}\right) & = & y^{(2)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & & \vdots \\
\boldsymbol{\phi}\left(\mathbf{x}^{(N)}\right)^{\mathsf{T}} & \phi_0\left(\mathbf{x}^{(N)}\right) & \phi_1\left(\mathbf{x}^{(N)}\right) & \phi_2\left(\mathbf{x}^{(N)}\right) & \cdots & \phi_D\left(\mathbf{x}^{(N)}\right) & & y^{(N)}
\end{array}
\quad
\begin{array}{c|cccccc|c}
\mathbf{\Phi} & \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_D & \mathbf{y} \\
\hline
\boldsymbol{\phi}\left(\mathbf{x}^{(1)}\right)^{\mathsf{T}} & 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} & y^{(1)} \\
\boldsymbol{\phi}\left(\mathbf{x}^{(2)}\right)^{\mathsf{T}} & 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} & y^{(2)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\boldsymbol{\phi}\left(\mathbf{x}^{(N)}\right)^{\mathsf{T}} & 1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_D^{(N)} & y^{(N)}
\end{array}
$$

Polynomial regression

Standard linear regression with the identity basis function is powerful for modelling an output variable $y^{(i)}$ which is assumed to be linearly dependent upon the explanatory variables $x_j^{(i)}$.

However, it is not always the case that the explanatory variables have a linear relationship with the output variable.

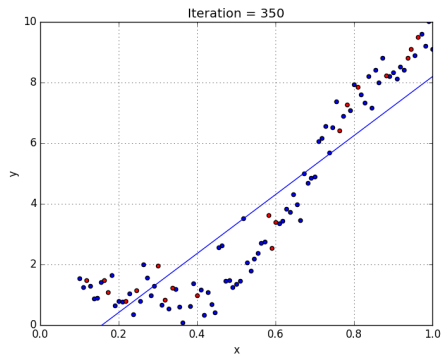


Figure 8: Example of a relationship that cannot accurately be modeled with a hyperplane.
This data would be more accurately represented with a polynomial regression model. (source)

In this case, it may be more appropriate to assume a different relationship, such as a polynomial one. The output variable can be modeled as a D-degree polynomial—a linear combination of the **monomials** of each feature:

$$y^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 \left(x_2^{(i)}\right)^2 + \cdots + \theta_D \left(x_D^{(i)}\right)^D$$

Which can be defined with the following basis function:

$$\boldsymbol{\phi}\left(\mathbf{x}^{(i)}\right) = \left(1, x_1^{(i)}, \left(x_2^{(i)}\right)^2, \ldots, \left(x_D^{(i)}\right)^D\right)^{\mathsf{T}}$$

Multivariate basis functions

Despite the identity and polynomial basis functions $\phi_j\left(\mathbf{x}^{(i)}\right)$ only operating on the jth feature, $x_j^{(i)}$, this isn't a strict requirement of basis functions—remember that each basis function ϕ_j is a function of the entire feature vector $\mathbf{x}^{(i)}$, and can therefore be dependent upon the values of other features. This brings rise to what are known as **multivariate basis functions**.

> **Example**: A basis function with multivariate inputs.
>
> $$\phi\left(\mathbf{x}^{(i)}\right)^{\mathsf{T}} = \left(1,\ x_1^{(i)},\ x_2^{(i)},\ x_3^{(i)},\ x_1^{(i)}x_2^{(i)},\ x_1^{(i)}x_3^{(i)},\ x_2^{(i)}x_3^{(i)},\ \left(x_1^{(i)}\right)^2,\ \cdots\right)$$

Resources

- *Iain Murray (School of Informatics, University of Edinburgh)*
  Machine Learning and Pattern Recognition: Linear Regression
- *Nigel Goddard (School of Informatics, University of Edinburgh)*
  Introductory Applied Machine Learning: Linear Regression - Solving for Model Parameters
- *Hiroshi Shimodaira, Iain Murray, Steve Renals (School of Informatics, University of Edinburgh)*
  Algorithms, Data Structures and Learning: Introduction to statistical pattern recognition and optimization
- *Gordon Ross (School of Mathematics, University of Edinburgh)*
  Statistical Learning: Nonlinearity and Dimensionality Reduction
- *Nguyen (StackOverflow)*
  Role of the bias term in linear regression
- *Wikipedia*
  Residual sum of squares
  Regularization (mathematics)
  Norm (mathematics)
  Taxicab geometry
  Elastic net regularization
- *Renu Khandelwal (Medium)*
  L_1 and L_2 Regularization
- *Jae Duk Seo (Towards Data Science)*
  Only Numpy: Implementing different combinations of L_1/L_2 norm/regularization
- *Stephanie (Statistics How To)*
  Tuning Parameter / Penalty Parameter
- *Sebastian Raschka*
  Does regularization in logistic regression always results in better fit and better generalization?
- *Sebastian Raschka (Mlxtend)*
  Regularization of Generalized Linear Models
- *balaks (StackOverflow)*
  Ridge, lasso and elastic net
- *Martin Krasser*
  Bayesian regression with linear basis function models
- *Ayush Pant (Towards Data Science)*
  Introduction to Linear Regression and Polynomial Regression
- *Ignacio P. Pozuelo (Stack Overflow)*
  Lasso or Ridge for correlated variables