Lecture 4.2
Linear Regression.

Linear Regression with Gradient Descent. **Regularization**

1. https://www.youtube.com/watch?v=vMh0zPT0tLI
2. https://www.youtube.com/watch?v=Q81RR3yKn30
3. https://www.youtube.com/watch?v=NGf0voTMlcs
4. https://www.youtube.com/watch?v=1dKRdX9bfIo

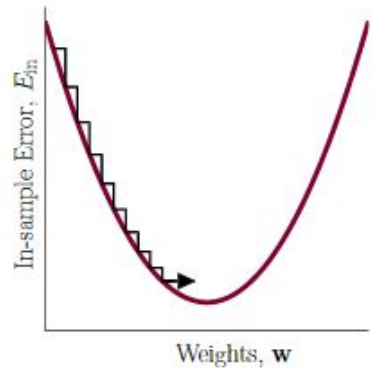
Gradient descent

is a method of numerical optimization that can be used in many algorithms where it is required to find the extremum of a function

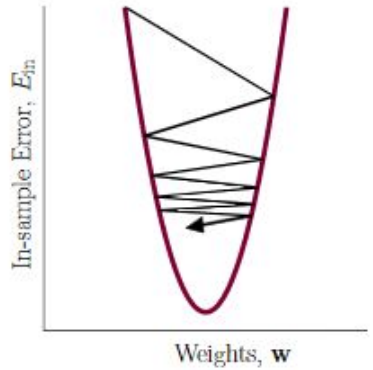
- **Gradient Descent** is the most common optimization algorithm in *machine learning* and *deep learning*. It is a first-order optimization algorithm. This means it only takes into account the first derivative when performing the updates on the parameters. On each iteration, we update the parameters in the opposite direction of the gradient of the objective function *J(w)* w.r.t the parameters where the gradient gives the direction of the steepest ascent. The size of the step we take on each iteration to reach the local minimum is determined by the learning rate α. Therefore, we follow the direction of the slope downhill until we reach a local minimum.

The 'Goldilocks' Step Size

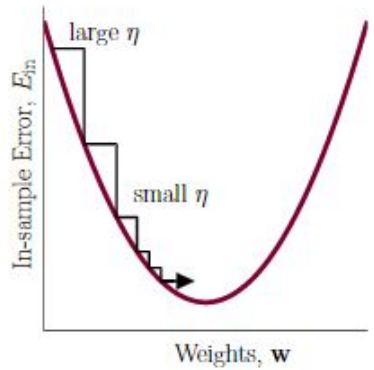
η too small

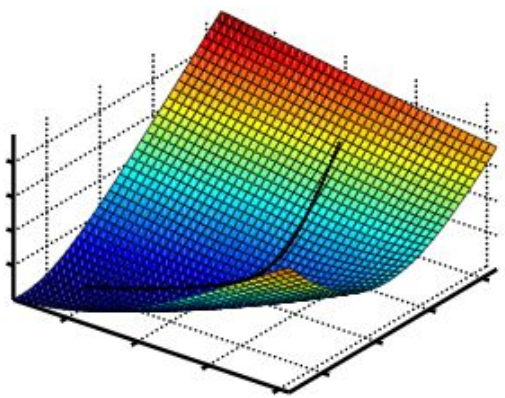


η too large

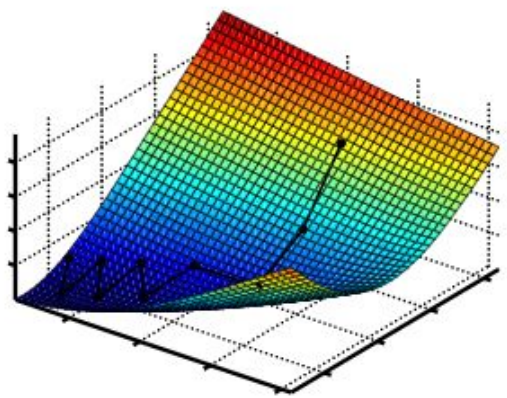


variable η_t – just right

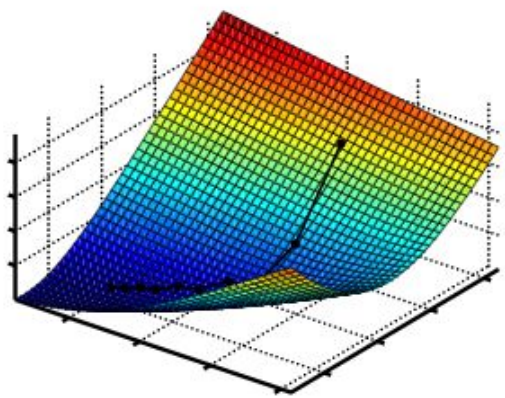




$\eta = 0.1$; 75 steps



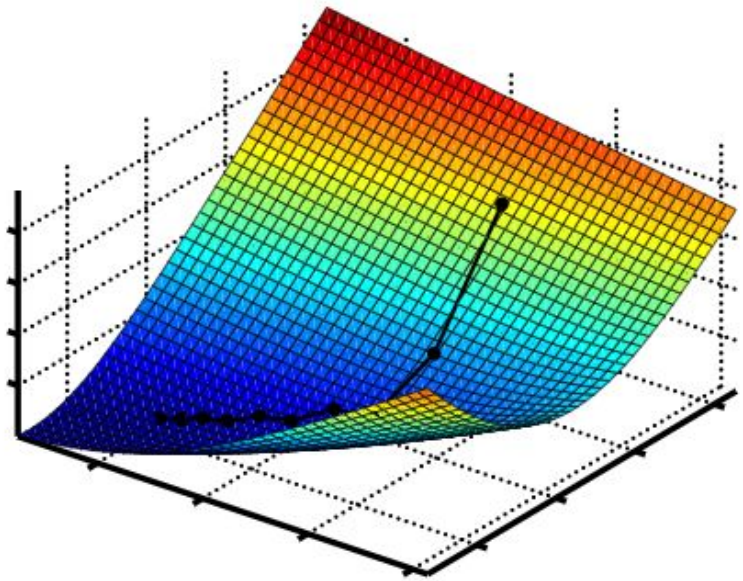
$\eta = 2$; 10 steps



variable η_t; 10 steps

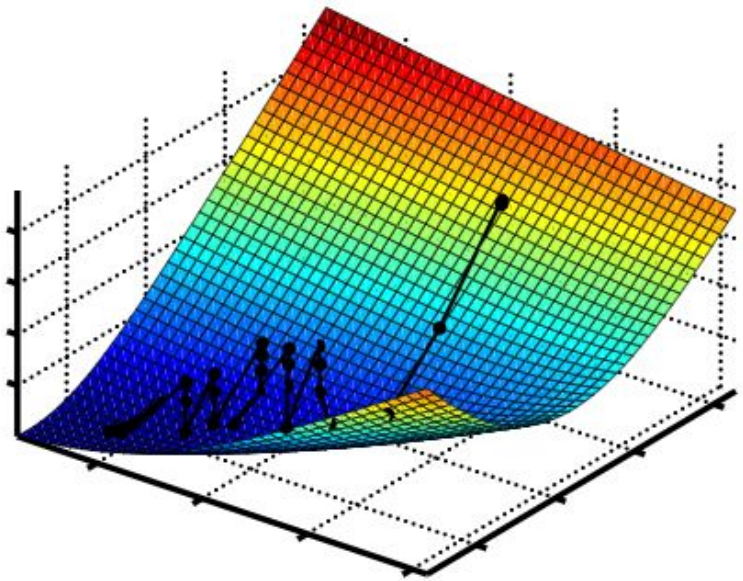
Stochastic Gradient Descent

GD



$\eta = 6$
10 steps
$N = 10$

SGD



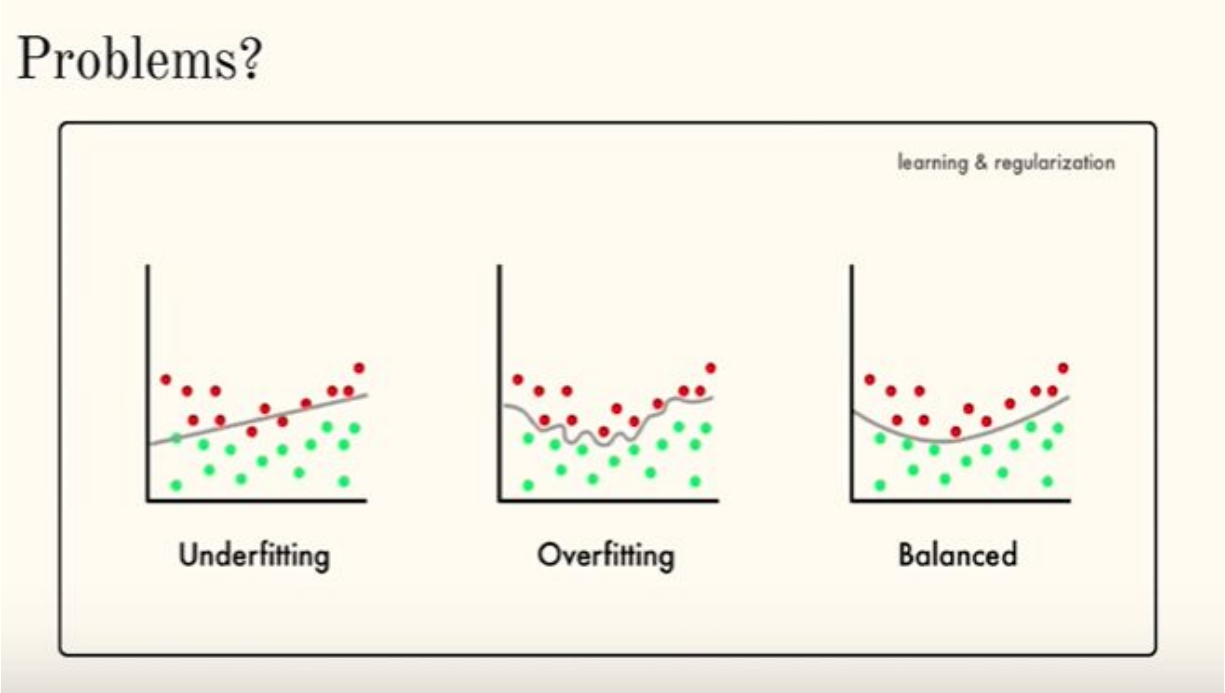
$\eta = 2$
30 steps

Linear Regression in Python using gradient descent

```
import sklearn
from sklearn.linear_model import SGDRegressor
# Create a linear regression object

regr = linear_model.SGDRegressor(max_iter=10000, tol =0.001)
```

- For many machine learning problems with a large number of features or a low number of observations, a linear model tends to **overfit** and variable selection is tricky.



Regularization: Ridge, Lasso and Elastic Net

- Models that use shrinkage such as Lasso and Ridge can improve the prediction accuracy as they reduce the estimation variance while providing an interpretable final model.
- In this tutorial, we will examine Ridge and Lasso regressions, compare it to the classical linear regression and apply it to a dataset in Python. Ridge and Lasso build on the linear model, but their fundamental peculiarity is regularization. The goal of these methods is to improve the loss function so that it depends not only on the sum of the squared differences but also on the regression coefficients.
- One of the main problems in the construction of such models is the correct selection of the regularization parameter. Comparing to linear regression, Ridge and Lasso models are more resistant to outliers and the spread of data. Overall, their main purpose is to prevent overfitting.
- The main difference between Ridge regression and Lasso is how they assign a penalty term to the coefficients.

Lasso Regression Basics

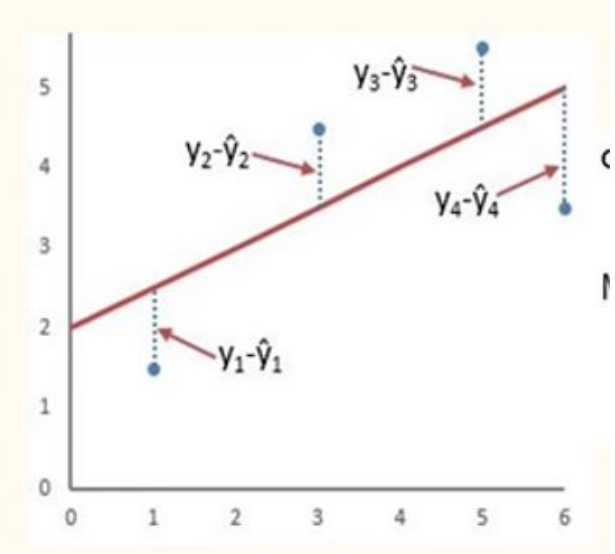
- Lasso performs a so called L1 regularization (a process of introducing additional information in order to prevent overfitting), i.e. adds penalty equivalent to absolute value of the magnitude of coefficients.

- In particular, the minimization objective does not only include the residual sum of squares (RSS) - like in the OLS regression setting - but also the sum of the absolute value of coefficients.

Ordinary least squares (OLS)

The residual sum of squares (RSS) is calculated as follows:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



This formula can be stated as:

$$RSS = \sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \right) \right)^2$$

- n represents the number of distinct data points, or observations, in our sample.
- p denotes the number of variables that are available in the dataset.
- x_{ij} represents the value of the jth variable for the ith observation, where i = 1, 2, . . ., n and

$$\widehat{y}_\star = \beta_0 + \beta_1 x_{\star 1} + \beta_2 x_{\star 2} + \cdots + \beta_p x_{\star p}.$$

In the lasso regression, the minimization objective becomes:

$$\sum_{i=1}^{n}\left(y_i - \left(\beta_0 + \sum_{j=1}^{k}\beta_j x_{ij}\right)\right)^2 + \alpha\sum_{j=1}^{k}|\beta_j|$$

which equals:

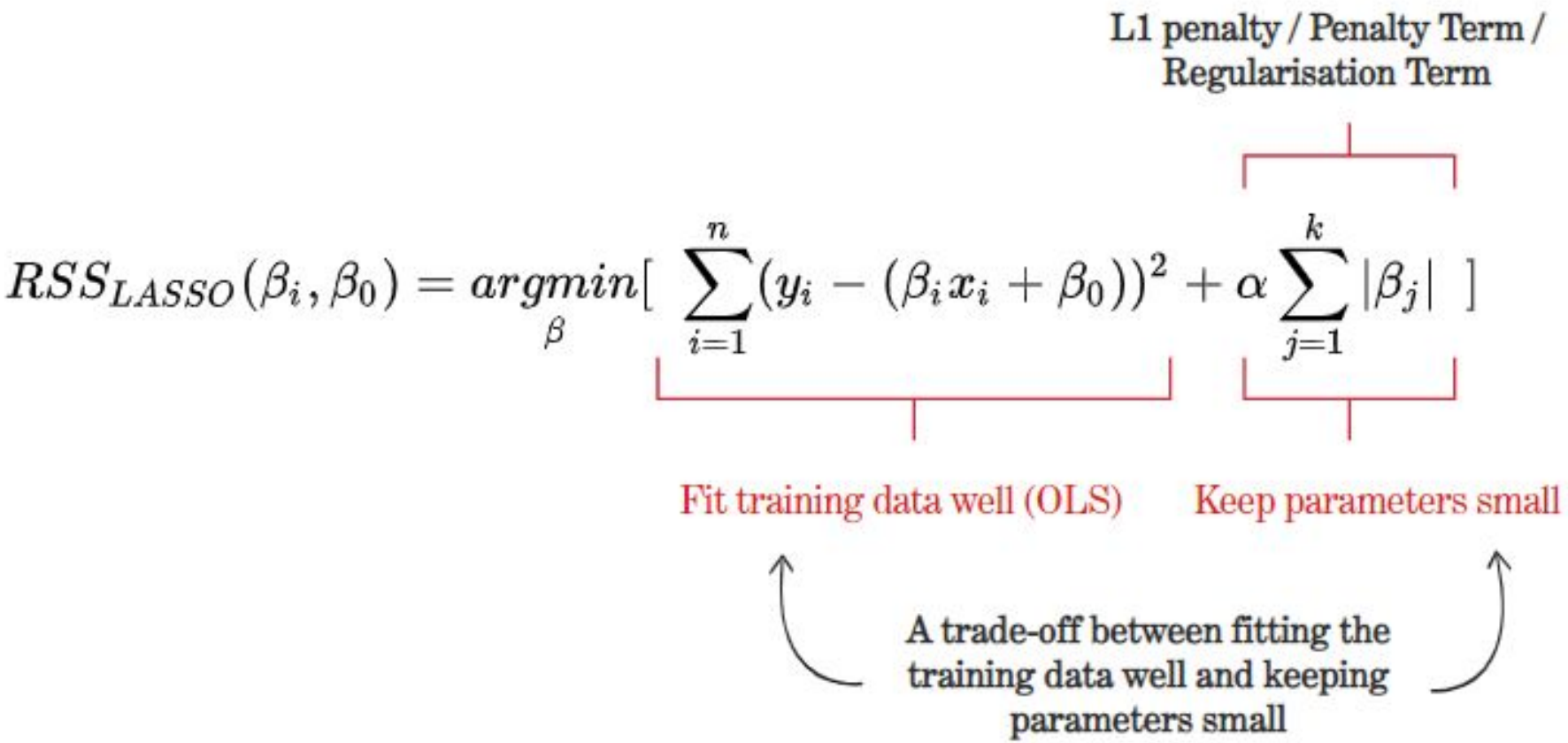
$$RSS + \alpha\sum_{j=1}^{k}|\beta_j|$$

λ (lambda) provides a trade-off between balancing RSS and magnitude of coefficients.

λ can take various values:

- $\lambda = 0$: Same coefficients as simple linear regression
- $\lambda = \infty$: All coefficients zero (same logic as before)
- $0 < \lambda < \infty$: coefficients between 0 and that of simple linear regression

The LASSO minimizes the sum of squared errors, with an upper bound on the sum of the absolute values of the model parameters. The lasso estimate is defined by the solution to the L1 optimization problem:



Parameter α

- In practice, the tuning parameter that controls the strength of the penalty assumes great importance. Indeed, when α is sufficiently large, coefficients are forced to be exactly equal to zero. This way, dimensionality can be reduced. T
- The larger the parameter , the more the number of coefficients are shrunk to zero. On the other hand, if α = 0, we have just an OLS (Ordinary Least Squares) regression.
- Alpha simply defines regularization strength and is usually chosen by cross-validation.

- This additional term penalizes the model for having coefficients that do not explain a sufficient amount of variance in the data. It also has a tendency to set the coefficients of the bad predictors mentioned above 0.
- This makes Lasso useful in feature selection.
- Lasso however struggles with some types of data. If the number of predictors (p) is greater than the number of observations (n), Lasso will pick at most n predictors as non-zero, even if all predictors are relevant. Lasso will also struggle with colinear features (they're related/correlated strongly), in which it will select only one predictor to represent the full suite of correlated predictors. This selection will also be done in a random way, which is bad for reproducibility and interpretation.

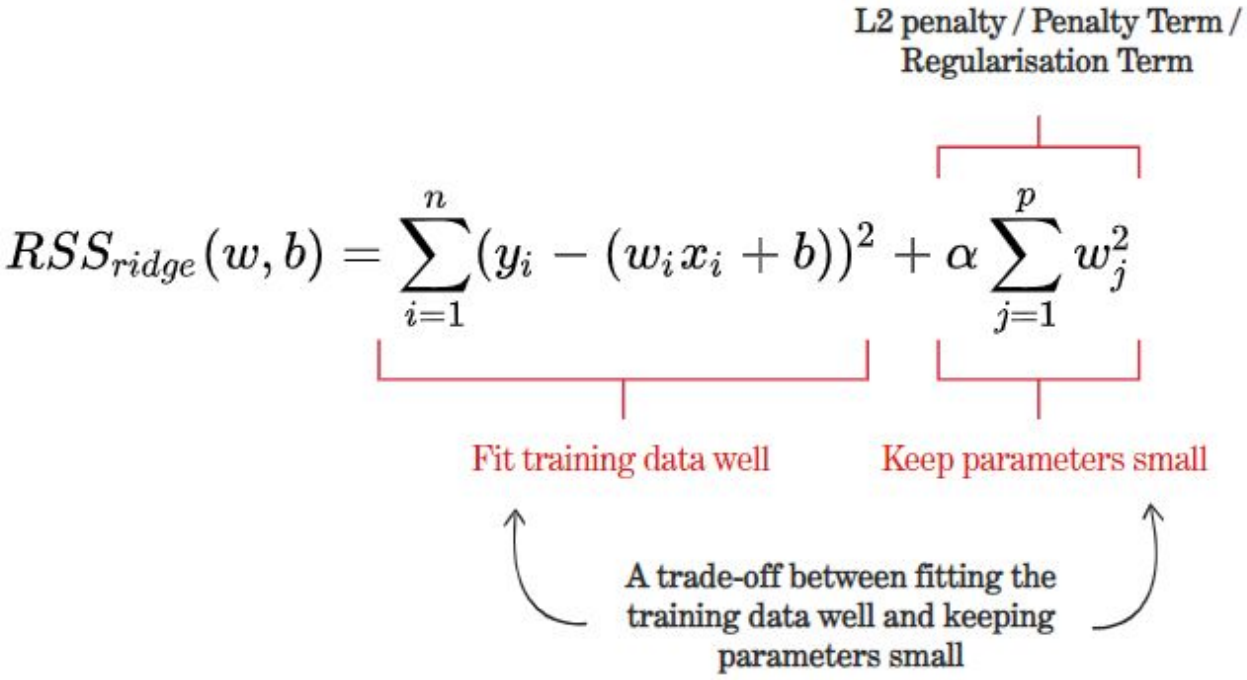
Lasso Regression with Python

```
from sklearn.linear_model import Lasso

reg = Lasso(alpha=0.5)
reg.fit(X_train, y_train)
```

Ridge regression

- Ridge regression also adds an additional term to the cost function, but instead sums the squares of coefficient values (the L-2 norm) and multiplies it by some constant lambda.



- Compared to Lasso, this regularization term will decrease the values of coefficients, but is unable to force a coefficient to exactly 0. This makes ridge regression's use limited with regards to feature selection. However, when p > n, it is capable of selecting more than n relevant predictors if necessary unlike Lasso. It will also select groups of colinear features, which its inventors dubbed the 'grouping effect.'
- Much like with Lasso, we can vary lambda to get models with different levels of regularization with lambda=0 corresponding to OLS and lambda approaching infinity corresponding to a constant function.

- rr = Ridge(alpha=0.01)
- rr.fit(X_train, y_train)

Elastic Net

- Elastic Net includes both L-1 and L-2 norm regularization terms.
- This gives us the benefits of both Lasso and Ridge regression.
- It has been found to have predictive power better than Lasso, while still performing feature selection.
- We therefore get the best of both worlds, performing feature selection of Lasso with the feature-group selection of Ridge.

- the elastic net adds a quadratic part to the L1 penalty, which when used alone is a ridge regression (L2). The estimates from the elastic net method are defined by
-

$$\hat{\beta}(\alpha) = \underset{\beta}{argmin}[\ \frac{\sum_{i=1}^{n}(y_i - (\beta_i x_i + \beta_0))^2}{n} + \alpha_1 \sum_{j=1}^{k} |\beta_j| + \alpha_2 \sum_{j=1}^{k} (\beta_j)^2\ ]$$

$$where\ \alpha_1 \geq 0\ and\ \alpha_2 \geq 0\ are\ two\ regularization\ parameters.$$

- #Elastic Net
- model_enet = ElasticNet(alpha = 0.01)
- model_enet.fit(X_train, y_train)

Lecture for Home work

- http://subtitlelist.com/en/Lecture-25-%E2%80%94-Linear-Regression-With-One-Variable-Gradient-Descent-%E2%80%94-Andrew-Ng-10360