ЛЕКЦИЯ 7. ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ. АППРОКСИММАЦИЯ ПОЛИНОМАМИ ВЫСОКОЙ СТЕПЕНИ

Что, если наши данные в действительности сложнее обычной прямой линии? Удивительно, но мы на самом деле можем применять линейную модель для подгонки к нелинейным данным. Простой способ предполагает добавление степеней каждого признака в виде новых признаков и последующее обучение линейной модели на таком расширенном наборе признаков. Этот прием называется полиномиальной регрессией. Давайте рассмотрим пример. Для начала сгенерируем нелинейные данные, основываясь на простой квадратичной функции и плюс некоторый шум. Сгенерируем данные.

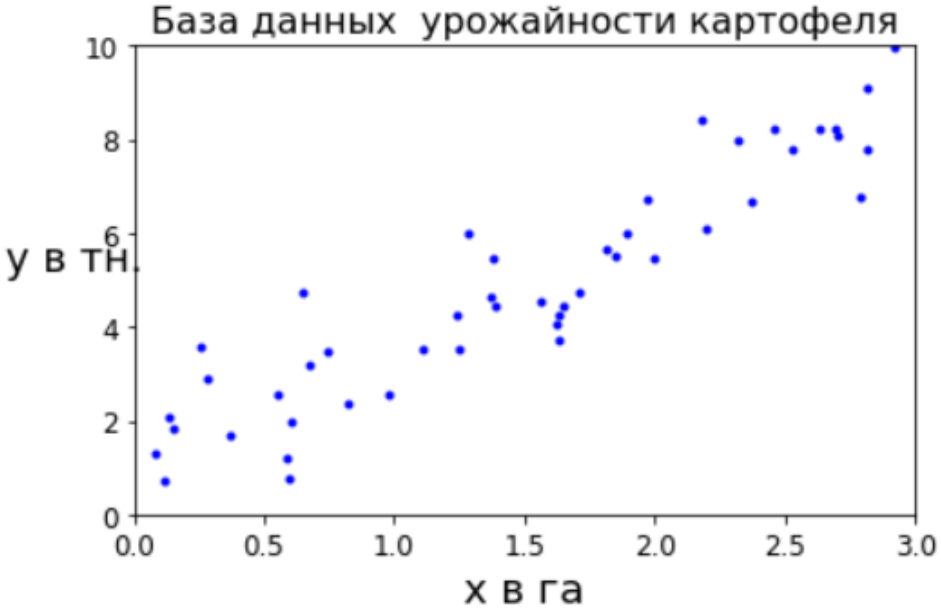

Рис.4.1. Распределение базы данных

Безусловно, прямая линия никогда не будет подогнана под такие данные должным образом. Потому воспользуемся классом PolynomialFeatures из Scikit-Learn. Построим кривую прогноза

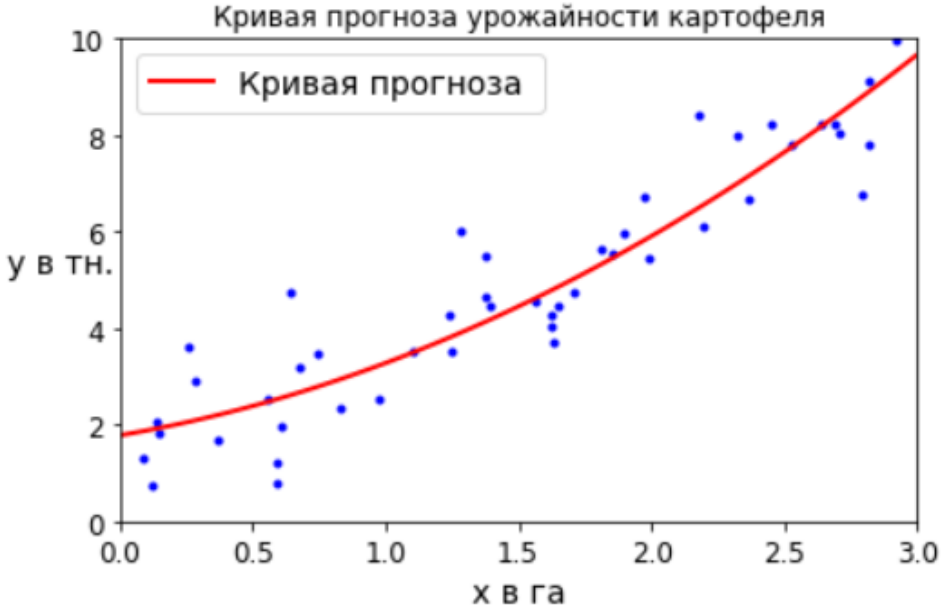

Рис.4.2. Распределение базы данных

У нас получилось модель, которая оценивает функцию как y = 0.56 x*x+ 0.93 x + 1 .78, когда на самом деле исходной функцией была y = 0.5 x*x+ x+ 2.0 + гауссов шум.

Давайте рассмотрим более сложный случай моделирования с полиномами высоких и даже очень высоких степеней. Наиболее реальный модель данных, например для урожайности картофеля в зависимости от посевной площади имеет следующее распределение данных f(x)=8x+sin(x)+N(0,0.01), где $N(\mu,\sigma^2)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ нормальное распределения Гаусса. В данном случае видим, как и выше при аппроксимации линейной функцией, математическое ожидание данного распределения равно 0, а дисперсия равно 0,01. Проанализируем нелинейные данные полученные по данной формуле, которое наиболее вероятнее описывает модель урожайности картофеля. Вот результат моделирования с полиномами высоких степеней

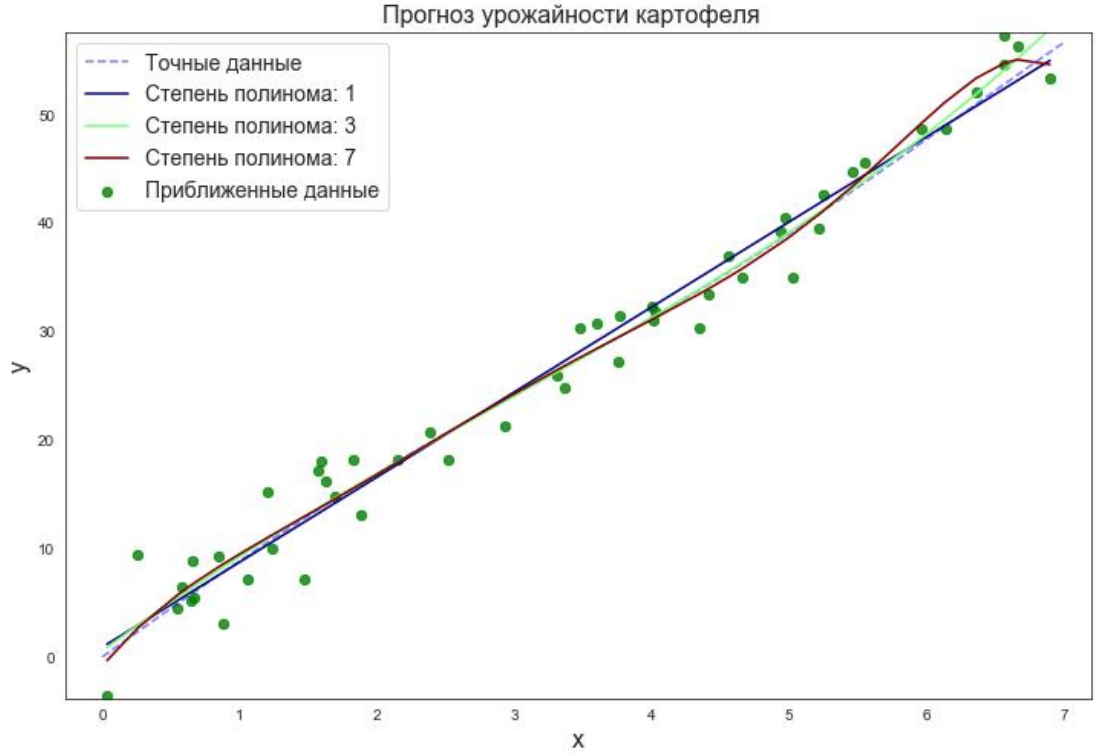

Рис.4.3. Аппроксимация полиномом 7 степени для нелинейных данных урожайности картофеля

При аппроксимации полиномами 9 степени получаем, следующий прогноз урожайности картофеля

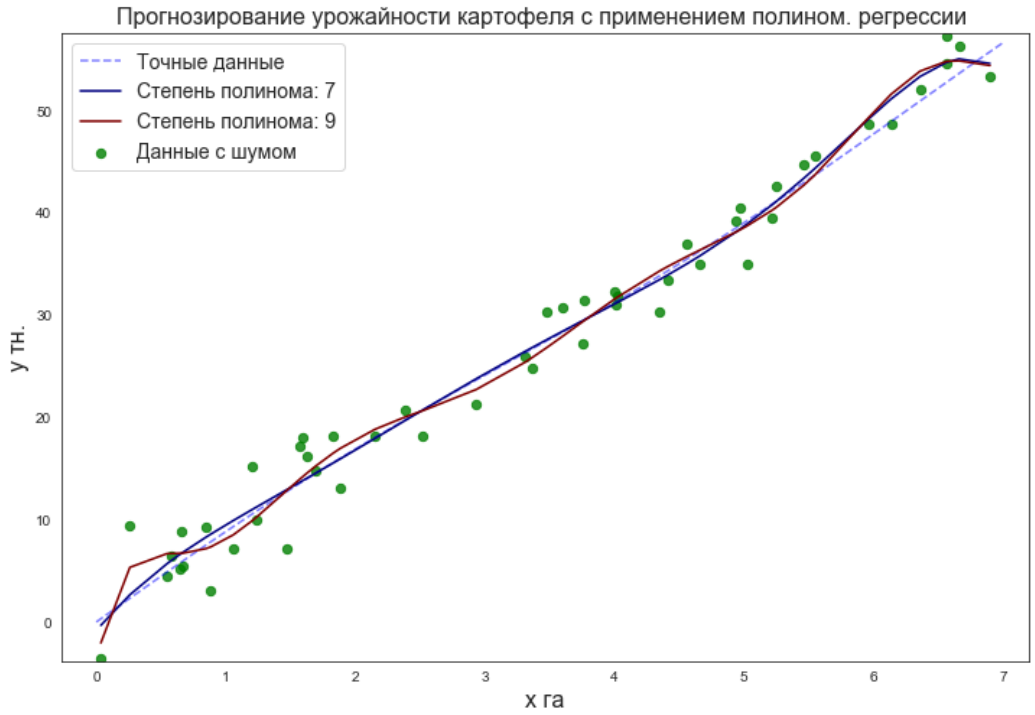

Рис.4.4. Аппроксимация полиномами 9 степени для нелинейных данных урожайности картофеля

Вычислим коэффициенты корреляции матрицы X

```
np.corrcoef(X.T)
```

```
array([[       nan,        nan,        nan,        nan,        nan,
               nan,        nan,        nan,        nan,        nan],
       [       nan, 1.        , 0.96906472, 0.91338345, 0.85690607,
        0.80683434, 0.76432569, 0.72858959, 0.69842006, 0.67268472],
       [       nan, 0.96906472, 1.        , 0.9842355 , 0.95269134,
        0.9181019 , 0.88521499, 0.85542955, 0.82886648, 0.80520787],
       [       nan, 0.91338345, 0.9842355 , 1.        , 0.99109359,
        0.97233476, 0.95050675, 0.92850086, 0.90745252, 0.88772222],
       [       nan, 0.85690607, 0.95269134, 0.99109359, 1.        ,
        0.99465191, 0.98283298, 0.96833034, 0.95295531, 0.93755387],
       [       nan, 0.80683434, 0.9181019 , 0.97233476, 0.99465191,
        1.        , 0.99657063, 0.98864817, 0.9784666 , 0.96719847],
       [       nan, 0.76432569, 0.88521499, 0.95050675, 0.98283298,
        0.99657063, 1.        , 0.99765746, 0.9920301 , 0.98450323],
       [       nan, 0.72858959, 0.85542955, 0.92850086, 0.96833034,
        0.98864817, 0.99765746, 1.        , 0.9983065 , 0.99409714],
       [       nan, 0.69842006, 0.82886648, 0.90745252, 0.95295531,
        0.9784666 , 0.9920301 , 0.9983065 , 1.        , 0.99871304],
       [       nan, 0.67268472, 0.80520787, 0.88772222, 0.93755387,
        0.96719847, 0.98450323, 0.99409714, 0.99871304, 1.        ]])
```

Матрица корреляции выглядит следующим образом

```
# Матрица корреляции
plt.figure(figsize=(20,14))
plt.matshow(np.corrcoef(X.T), cmap=plt.cm.bwr)
plt.title('Корреляционная матрица  X', fontsize=20)
plt.show()
```

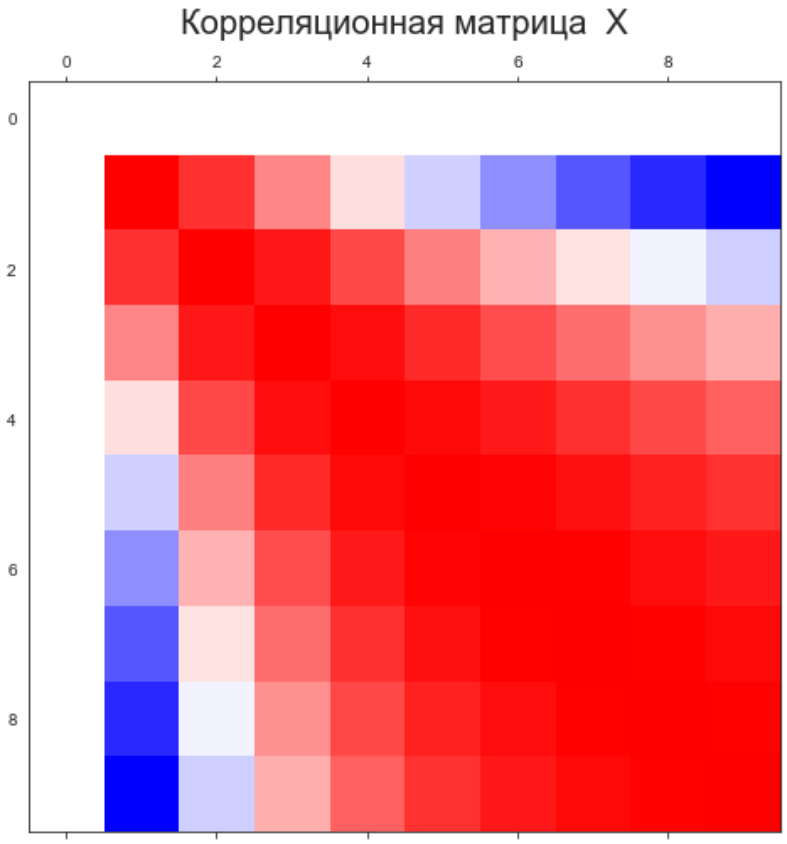

Рис.4.5. Корреляционная матрица данных урожайности картофеля

```
# Собственные значения для полинома -12 объектов, мы можем видеть мультиколлинеарность
np.linalg.eigvals(np.cov(X[:, 1:-1].T))
```

```
array([1.40443523e+12, 1.14224590e+08, 8.19425270e+04, 1.84496412e+02,
       1.27766937e+00, 3.34108323e-02, 1.55854381e-03, 2.73843638e-05])
```

```
# Собственные значения для поли-13 признаков, мы можем видеть мультиколлинеарность и сложные собственн
np.linalg.eigvals(np.cov(X[:, 1:].T))
```

```
array([6.25236219e+13, 3.83162844e+09, 2.15685112e+06, 3.46853302e+03,
       1.24305779e+01, 2.58662579e-01, 8.87407501e-03, 4.33189273e-04,
       3.55787981e-06])
```

Для тестирование на линейную зависимость или мультиколлинеарность можно использовать число обусловленности матрицы X^TX. Число обусловленности равно отношению большего собственного числа к меньшему. Большое число определенности или наличие близких к нулю собственных чисел является признаком мультиколлинеарности.

К сожалению инвертирование X^TX при нечеткой мультиколлинеарности численно нестабильно, но существует решение. Вспомним, что любую полноранговую матрицу X размера $n \times m$ можно представить в виде: $X=QR$, где R треугольная матрица размера $m \times m$

$$Q^TQ = E$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \quad &\Leftrightarrow \quad \frac{1}{2n}\left(-2X^T\vec{y} + 2X^T X\vec{w}\right) = 0 \\
&\Leftrightarrow \quad -X^T\vec{y} + X^T X\vec{w} = 0 \\
&\Leftrightarrow \quad X^T X\vec{w} = X^T\vec{y} \\
&\Leftrightarrow \quad (QR)^T (QR)\,\vec{w} = (QR)^T\vec{y} \\
&\Leftrightarrow \quad R^T\left(Q^T Q\right) R\vec{w} = R^T Q^T\vec{y} \\
&\Leftrightarrow \quad R^T R\vec{w} = R^T Q^T\vec{y} \\
&\Leftrightarrow \quad \left(R^T\right)^{-1} R^T R\vec{w} = \left(R^T\right)^{-1} R^T Q^T\vec{y} \\
&\Leftrightarrow \quad R\vec{w} = Q^T\vec{y} \\
&\Leftrightarrow \quad \vec{w} = (R)^{-1} Q^T\vec{y}
\end{aligned}$$

Треугольная матрица легко инвертируется, поэтому решение заметно стабильнее. Если бы нам был интересен только прогноз, то можно было бы и не выводить значения параметров модели:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \vec{w}} = 0 \quad &\Leftrightarrow \quad \frac{1}{2n}\left(-2X^T\vec{y} + 2X^T X\vec{w}\right) = 0 \\
&\Leftrightarrow \quad R\vec{w} = Q^T\vec{y} \\
&\Leftrightarrow \quad QR\vec{w} = QQ^T\vec{y} \\
&\Leftrightarrow \quad X\vec{w} = QQ^T\vec{y}
\end{aligned}$$

Реализуем данную технологию, с применением полиномов 5,7 и 9 степеней для урожайности фермерских хозяйств , получим следующий результат прогноза

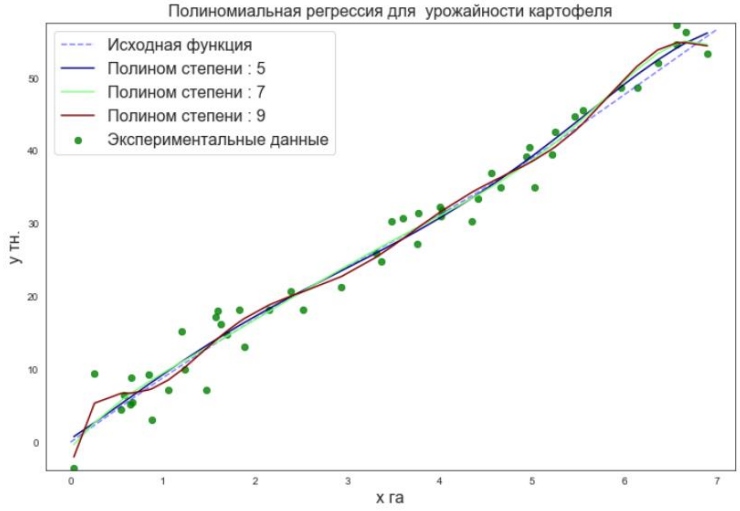

Рис.4.6. Прогноз с полиномами высоких степеней урожайности картофеля с применением разложения матрицы на треугольные.

Посмотрим теперь на значение параметров, которые получились в результате применения данного алгоритма обучения, при различных степеней полиномов

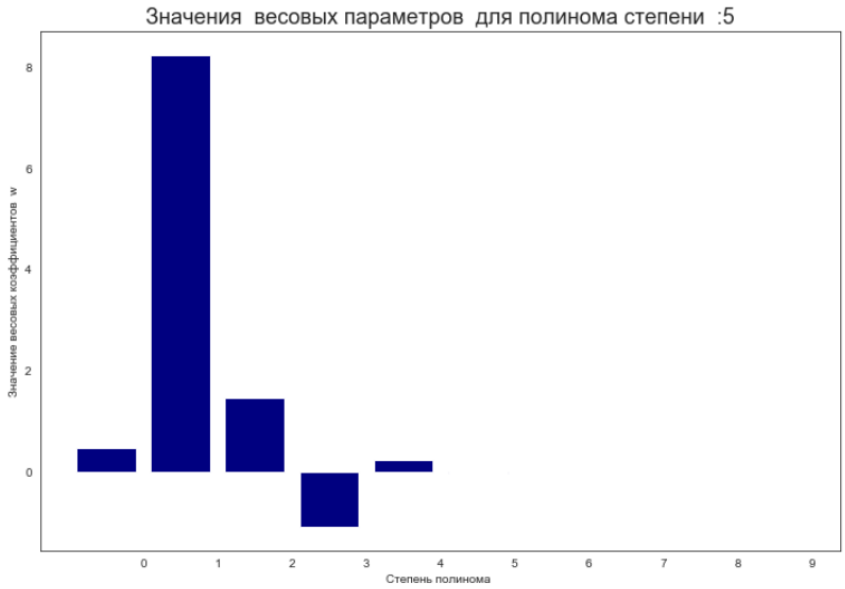

Рис.4.7. Распределение весовых коэффициентов при аппроксимации полиномом 5 степени урожайности картофеля

Как видно из гистограммы некоторые коэффициенты имею большие значения, и при увеличении степени полинома просто равны 0.В большинство случаях не удается использовать точные методы определения параметров модели. В таких случаях необходимо применять численные методы нахождения параметров модели.

Оптимизация кривых обучения с полиномами очень высоких степеней

В случае выполнения полиномиальной регрессии высокой степени, вероятно, мы гораздо лучше подгоним обучающие данные, чем с помощью линейной регрессии использующие полиномы низших степеней. Например, на рис. 4.8 демонстрируется применение полиномиальной модели 300-й степени к предшествующим обучающим данным,а результат сравнивается с чистой линейной моделью и квадратичной моделью (полиномиальной 2-й степени).

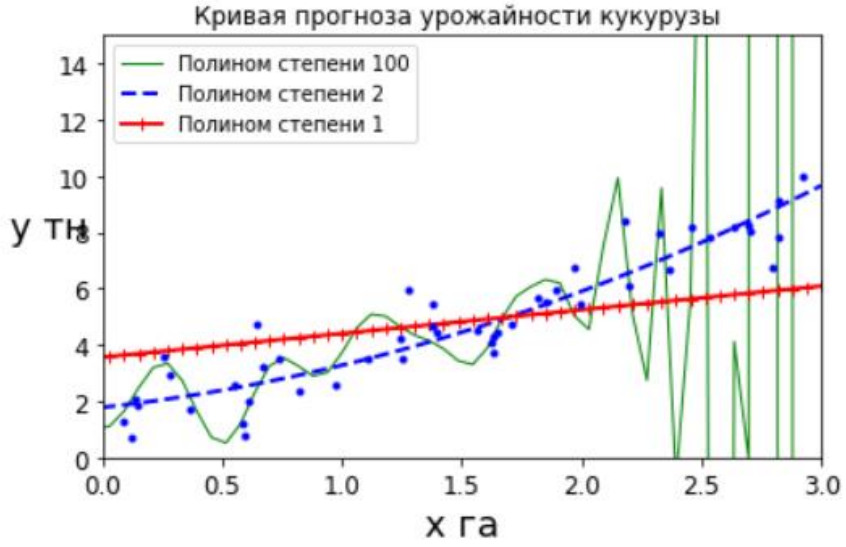

Рис.4.8. Распределение весовых коэффициентов при аппроксимации полиномом 100 степени урожайности картофеля

Для сравнения приведем модель при комбинация полиномов 2 и первой степеней

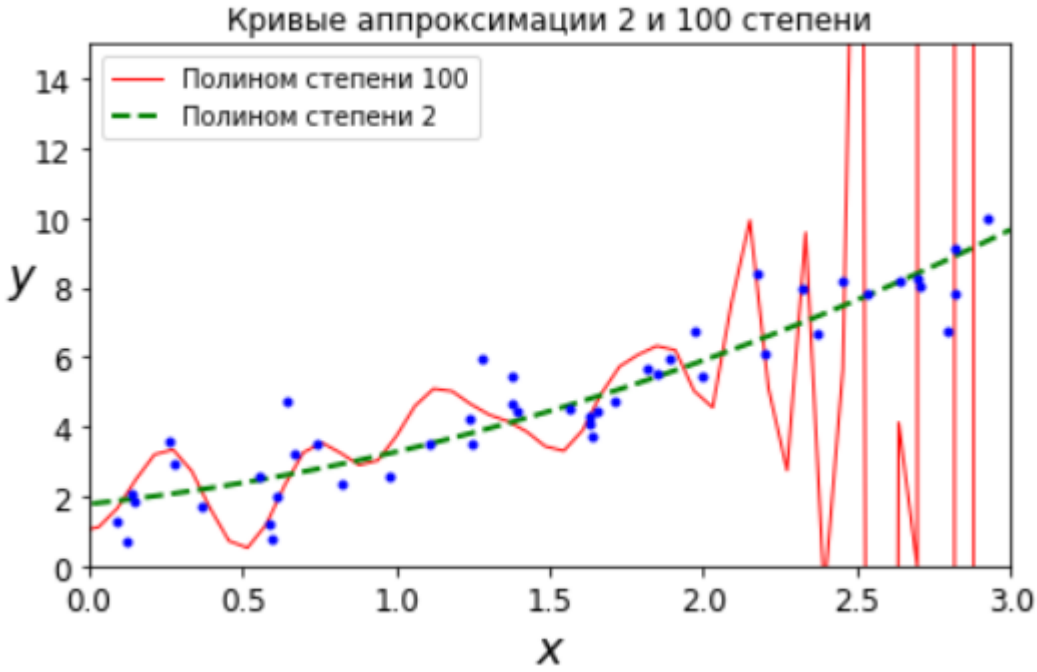

Рис.4.9. Распределение весовых коэффициентов при аппроксимации полиномом 1 и 2 степени урожайности картофеля

Обратите внимание на то, как полиномиальная модель 100-й степени колеблется, чтобы как можно больше приблизиться к обучающим образцам. Разумеется, такая полиномиальная регрессионная модель высокой степени вызывает сильное переобучение обучающими данными, тогда как линейная модель - недообучение на них. В рассматриваемом случае будет хорошо обобщаться квадратичная модель. Это имеет смысл, поскольку данные генерировались с использованием квадратного уравнения, но с учетом того, что обычно вы не будете знать функцию, применяемую для генерации данных, каким же образом принимать решение о том, насколько сложной должна быть модель? Как выяснить, что модель переобучается или недообучается на данных? Если согласно метрикам перекрестной проверки модель хорошо выполняется на обучающих данных, но плохо обобщается, то модель переобучена. Если модель плохо выполняется в обоих случаях, тогда она недообучена. Так выглядит один из способов сказать, что модель слишком проста или чрезмерно сложна. Другой способ предусматривает просмотр кривых обучения (learningcurve): они представляют собой графики производительности модели на обучающем наборе и проверочном наборе как функции от размера обучающего набора (или итерации обучения). Чтобы получить такие графики, нужно просто обучить модель несколько раз на подмножествах разных размеров, взятых из обучающего набора. Давайте взглянем на кривые обучения обыкновенной линейной регрессионной модели прямая линия на рис. 4. 10:

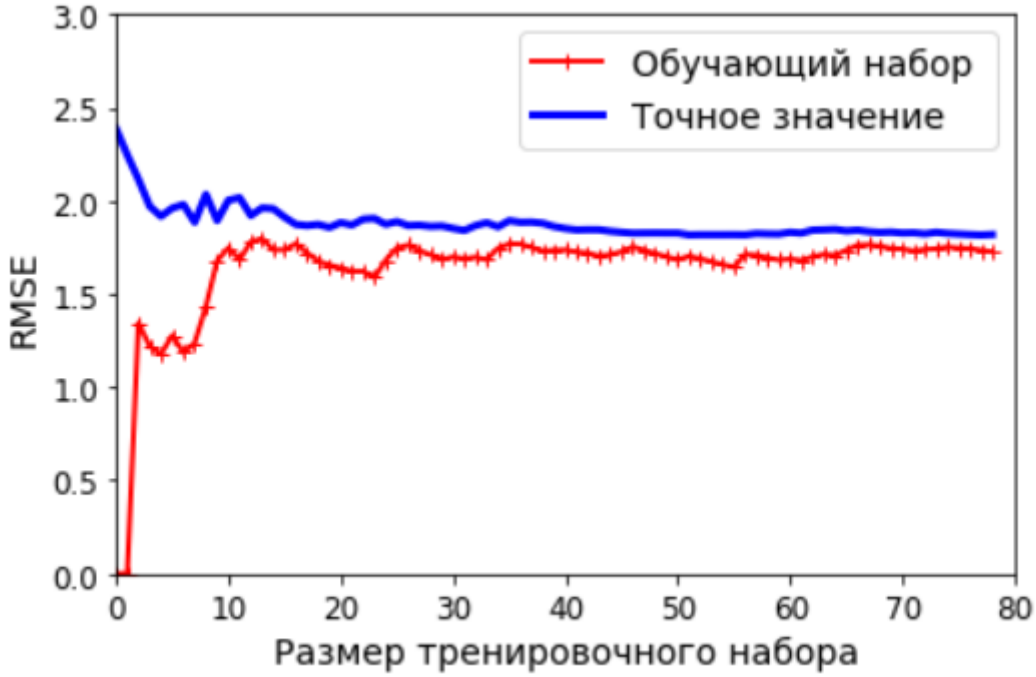

Рис.4.10. Поведение кривых обучения при различных количествах признаков модели

Здесь необходимы некоторые пояснения. Первым делом обратите внимание на производительность модели в случае использования обучающих данных: когда в обучающем наборе есть только один или два образца, модель может быть в полной мере подогнана к ним, что и объясняет начало кривой с нулевой ошибки. Но по мере добавления образцов в обучающий набор идеальная подгонка модели к обучающим данным становится невозможной, как из-за того, что данные зашумлены, так и потому, что они совершенно отличаются от линейных. Следовательно, ошибка на обучающих данных двигается вверх, пока не стабилизируется, когда добавление новых образцов в обучающий набор не делает среднюю ошибку намного лучше или хуже. Теперь перейдем к производительности модели на проверочных данных. Когда модель обучалась на незначительном количестве обучающих образцов, она неспособна обобщаться надлежащим образом, а потому ошибка проверки изначально довольно велика. Затем по мере того, как модель видит все больше обучающих образцов, она обучается, а ошибка проверки соответственно медленно снижается. Однако прямая линия снова не в состоянии хорошо смоделировать данные, так что ошибка стабилизируется поблизости к другой кривой. Такие кривые обучения типичны для недообученной модели. Обе кривые стабилизируются; они расположены близко друг к другу и довольно высоко. Если ваша модель недообучена на обучающих данных, тогда добавление дополнительных обучающих образцов не поможет. Вам нужно выбрать более сложную модель или отыскать лучшие признаки. Теперь рассмотрим кривые обучения полиномиальной модели 100-й степени на тех же самых данных см. рис. ниже:

```
from sklearn.pipeline import Pipeline
polynomial_regression = Pipeline([
        ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
        ("lin_reg", LinearRegression()),
    ])
plot_learning_curves(polynomial_regression, X, y)
plt.axis([0, 80, 0, 3])
save_fig("learning_curves_plot")
plt.show()
```

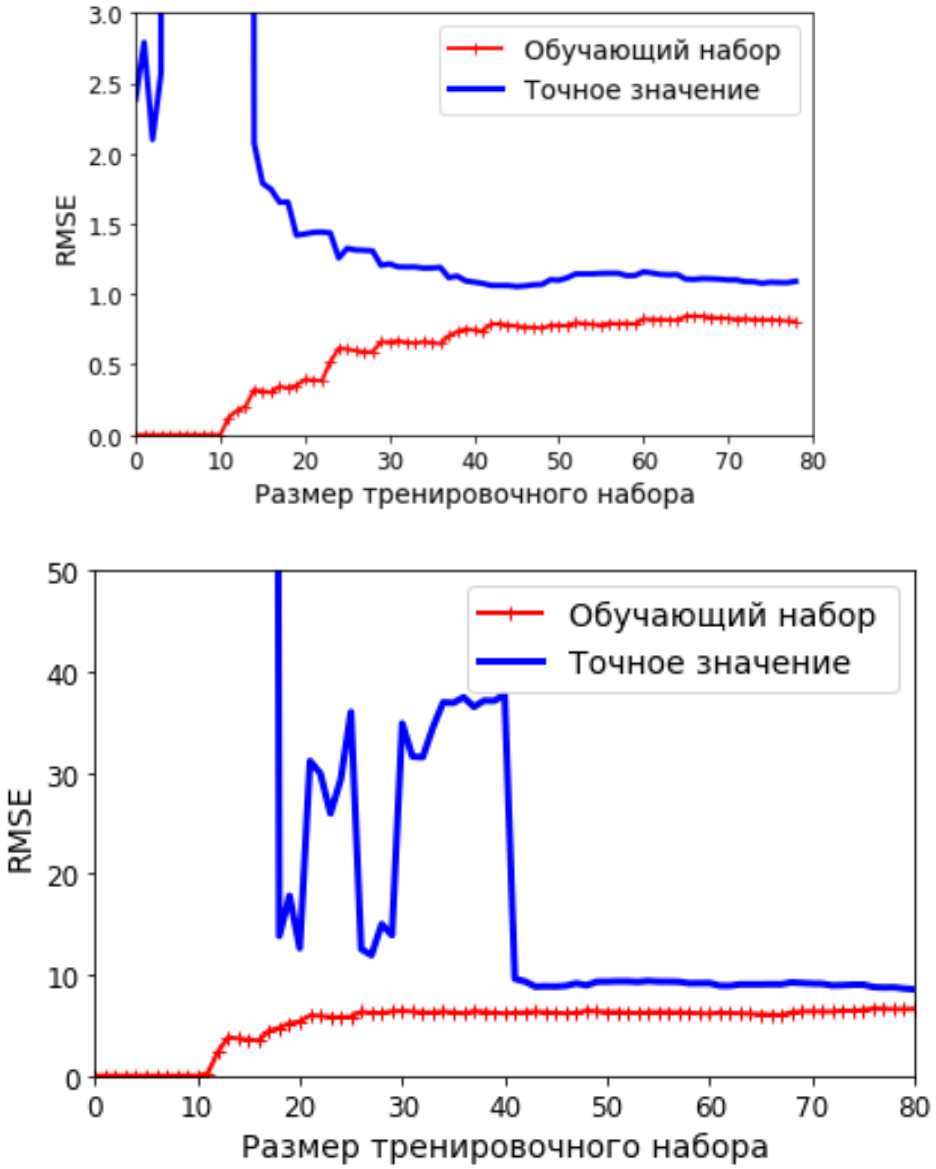

Рис.4.11. Поведение кривых обучения при различных количествах признаков модели

Кривые обучения выглядят чуть лучше предыдущих, но есть два очень важных отличия. • Ошибка на обучающих данных гораздо ниже, чем в случае линейной регрессионной модели. • Между кривыми имеется промежуток. Это значит, что модель выполняется существенно лучше на обучающих данных, чем на проверочных данных, демонстрируя признак переобучения. Тем не менее, если вы примените намного более крупный обучающий набор, то две кривые продолжат сближение. Один из способов улучшения переобученной модели заключается в предоставлении ей добавочных обучающих данных до тех пор, пока ошибка проверки не достигнет ошибки обучения.

Использование технологий регуляризации для построения линейных моделей

Выше мы обучали модели полиномами высшего порядка. При такой полиномиальной регрессии легко заметить, симптомы переобучения. Выученная функция интерполирует данные, вместе экстрополяции при тестировании модели. Так же мы видим, что абсолютные значения весов растут вместе с увеличением степени полинома. В данном случае мы должны наложит какой-нибудь штраф на амплитуду весов. В общем случае штраф выглядит следующим образом:

Рассмотрим следующий функционал и попробуем добавить ограничение на L_2 норму вектора параметров модели:

$L_{reg}(X,\vec{y},\vec{w})=L(X,\vec{y},\vec{w})+\lambda R(\vec{w})$, где (4.1)

$R(\vec{w})=\frac{1}{2}||\overrightarrow{w}||_2^2||=\frac{1}{2}\sum_{j=1}^{n}\omega_j^2=\frac{1}{2}\vec{w}^T\vec{w}$

тогда целевая функция в векторной форме примет вид:

$L(X,\vec{y},\vec{w})=\frac{1}{2}(\vec{y}-X\vec{\omega})^T(\vec{y}-X\vec{\omega})+\frac{\lambda}{2}\vec{\omega}^T\vec{\omega},$ (4.2)

Разложив выражения производной функционала и приравняя к ее к нулю найдем решение

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\vec{w}} &= \frac{\partial}{\partial\vec{w}}\left(\frac{1}{2}\left(\vec{y}-X\vec{w}\right)^T\left(\vec{y}-X\vec{w}\right)+\frac{\lambda}{2}\vec{w}^T\vec{w}\right)\\ &= \frac{\partial}{\partial\vec{w}}\left(\frac{1}{2}\left(\vec{y}^T\vec{y}-2\vec{y}^TX\vec{w}+\vec{w}^TX^TX\vec{w}\right)+\frac{\lambda}{2}\vec{w}^T\vec{w}\right)\\ &= -X^T\vec{y}+X^TX\vec{w}+\lambda\vec{w}\end{aligned} \quad (4.3)$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\vec{w}}=0 &\Leftrightarrow -X^T\vec{y}+X^TX\vec{w}+\lambda\vec{w}=0\\ &\Leftrightarrow X^TX\vec{w}+\lambda\vec{w}=X^T\vec{y}\\ &\Leftrightarrow \left(X^TX+\lambda E\right)\vec{w}=X^T\vec{y}\\ &\Leftrightarrow \vec{w}=\left(X^TX+\lambda E\right)^{-1}X^T\vec{y}\end{aligned} \quad (4.4)$$

Таким образом из условия мимнимума функционала мы нашли точное решение исходной задачи. Такая регрессия называется гребневой регрессией –Тихоновской . При увеличении параметра регуляризации, матрица $X^TX+\lambda E$ становится менее сингулярной-регулярной, а задача становится более определенной. Для такой матрицы число обусловленности будет равно

$$\frac{e_{\max}+\lambda}{e_{\min}+\lambda},$$

где e_x — это собственные числа матрицы. Таким образом, увеличивая параметр регуляризации мы уменьшаем число обусловленности матрицы полученной системы алгебраического уравнения. Как было показано в предыдущих параграфах , хороший способ сократить переобучение заключается в том, чтобы регуляризировать модель (т.е. ограничить ее): чем меньше степеней свободы она имеет, тем труднее ее будет переобучить данными. Например, простой метод регуляризации полиномиальной модели предполагает понижение количества полиномиальных степеней. Для линейной модели регуляризация обычно достигается путем ограничения весов модели. Мы рассмотрим гребневую регрессию (ridge regression), лассо-регрессию (lasso regression) и эластичную сеть (elastic net), которые реализуют три разных способа ограничения весов. Гребневая регрессия Гребневая регрессия (также называемая регуляризацией Тихонова) является регуляризированной версией линейной регрессии: к функции издержек добавляется член регуляризации (regularization tert).Это заставляет алгоритм обучения не только приспосабливаться к данным, но также удерживать веса модели насколько возможно небольшими. Обратите внимание, что член регуляризации должен добавляться к функции издержек только во время обучения. После того как модель обучена, вы захотите оценить производительность модели с использованием нерегуляризированной меры производительности. Отличие функции издержек, применяемой во время обучения, от меры производительности, используемой для проверки довольно распространенное явление. Еще одна причина, по которой они могут быть разными, не считая регуляризации, заключается в том, что хорошая функция издержек при обучении должна иметь удобные для оптимизации производные, тогда как мера производительности, применяемая для проверки, обязана быть насколько возможно близкой к финальной цели. Подходящим

примером может служить классификатор, который обучается с использованием такой функции издержек, как логарифмическая потеря (log loss; обсуждается далее в главе), но оценивается с применением точности/полноты. Гиперпараметр а управляет тем, насколько необходимо регуляризировать модель. Когда $\alpha = 0$, гребневая регрессия оказывается просто линейной регрессией. При очень большом значении а все веса становятся крайне близкими к нулю, и результатом будет ровная линия, проходящая через середину данных. В уравнении (2.12) представлена функция издержек гребневой регрессии

$$J(\theta) = MSE(\theta) + \frac{\alpha}{2}\sum_{i=1}^{n}\theta_i^2, \qquad (4.5)$$

Обратите внимание, что член смещения θ_0 не регуляризирован (сумма начинается с i = 1, не с 0). Если мы объявим w как весовой вектор признаков (от θ_0 до θ_n), тогда член регуляризации просто равен $\frac{1}{2}||w||_2^2$, где $||.||_2^2$ представляет норму l_2 весового вектора.

Для градиентного спуска нужно просто добавить $\alpha\mathbf{w}$ к вектору-градиенту MSE .Перед выполнением гребневой регрессии важно масштабировать данные, т.к. она чувствительна к масштабу входных признаков. Это справедливо для большинства регуляризированных моделей. На рис. 4. 12 показаны несколько гребневых моделей, обученных на некоторых линейных данных с использованием разных значений.

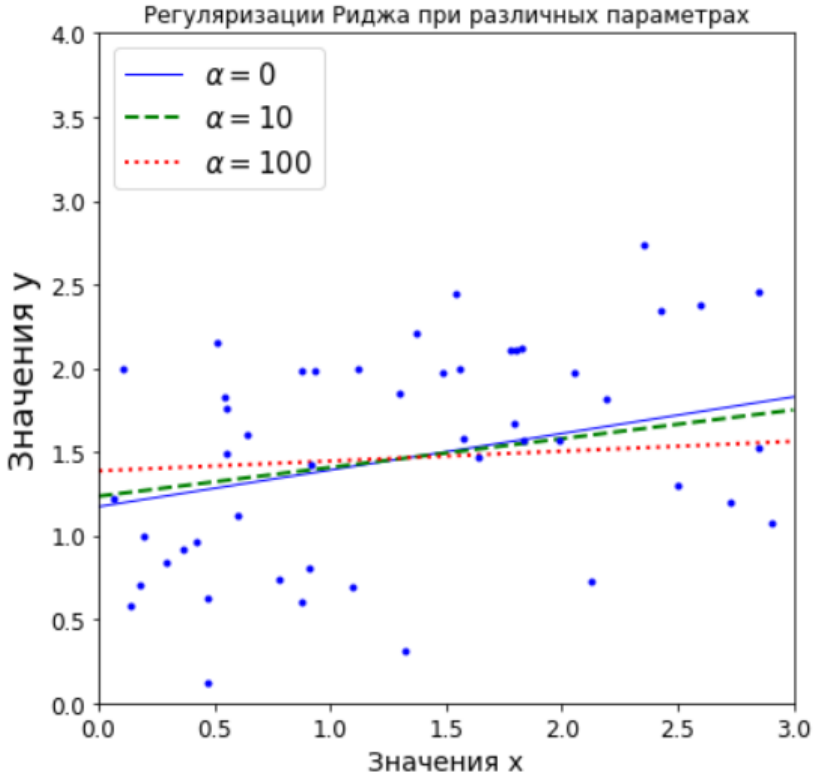

Рис.4.12. Гребневая регрессия при различных α=0,10,100

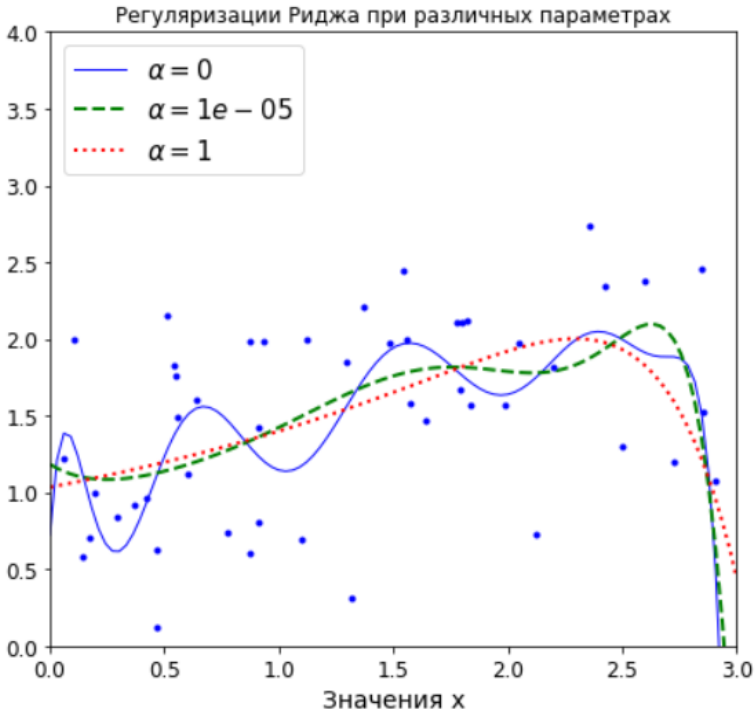

Рис.4.13. Гребневая регрессия при различных α=0,1e-05,1

Вверху применялись обыкновенные гребневые модели, приводя к линейным прогнозам. На втором графике данные были сначала расширены с применением PolynomialFeatures (degree= 10) , затем масштабированы с использованием StandardScaler и в заключение к результирующим признакам были применены гребневые модели: это полиномиальная регрессия с гребневой регуляризацией. Обратите внимание на то, как увеличение a ведет к более ровным (т.е. менее предельным, более рациональным) прогнозам; в итоге повышается дисперсия модели, но снижается ее смещение. Как и в случае линейной регрессии, мы можем производить гребневую регрессию, либо вычисляя уравнение в аналитическом виде, либо выполняя градиентный спуск. Доводы за и против одинаковы. В уравнении 4.6 решение в аналитическом виде (где A - это единичная матрица, размерности n x n за исключением нуля в левой верхней ячейке, соответствующего члену смещения). Решение в аналитическом виде для гребневой регрессии

$$\hat{\theta}=(\boldsymbol{X}^T\boldsymbol{X}+\alpha\boldsymbol{A})^{-1}\boldsymbol{X}^T\mathbf{y}, \quad (4.6)$$

Вот как выполнять гребневую регрессию с помощью Scikit-Learn, используярешение в аналитическом виде (вариант уравнения 4.6 применяет метод разложения матрицы Андре-Луи Холецкого):

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1, solver="cholesky", random_state=42)
ridge_reg.fit(X, y)
ridge_reg.predict([[1.5]])
array([[1.55071465]])
```

И с применением стохастического градиентного спуска

```
sgd_reg = SGDRegressor(max_iter=50, tol=-np.infty, penalty="l2", random_state=42)
sgd_reg.fit(X, y.ravel())
sgd_reg.predict([[1.5]])
array([1.49905184])
ridge_reg = Ridge(alpha=1, solver="sag", random_state=42)
ridge_reg.fit(X, y)
ridge_reg.predict([[1.5]])
array([[1.5507201]])
```

Гиперпараметр penal ty устанавливает используемый тип члена регуляризации. Указание " 1 2 " обозначает то, что вы хотите, чтобы алгоритм SGD добавлял к функции издержек член регуляризации, равный одной второй квадрата нормы i2 весового вектора: это просто гребневая регрессия.

Лассо-регрессия

Регрессия методом наименьшего абсолютного сокращения и выбора, называемая просто лассо-регрессией, представляет собой еще одну регуляризированную версию линейной регрессии: в точности как гребневая регрессия она добавляет к функции издержек член регуляризации, но вместо одной второй квадрата нормы L_2 весового вектора использует норму L_1 весового вектора (уравнение 4. 10).Функционал Лассо

$J(\theta) = MSE(\theta) + \frac{\alpha}{2}\sum_{i=1}^{n} |\theta_i|$, (4.9)

На рис. 2. 29 демонстрируется то же самое, что и на рис. 4. 14, но гребневые модели заменены лассо-моделями и применяются меньшие значения α.

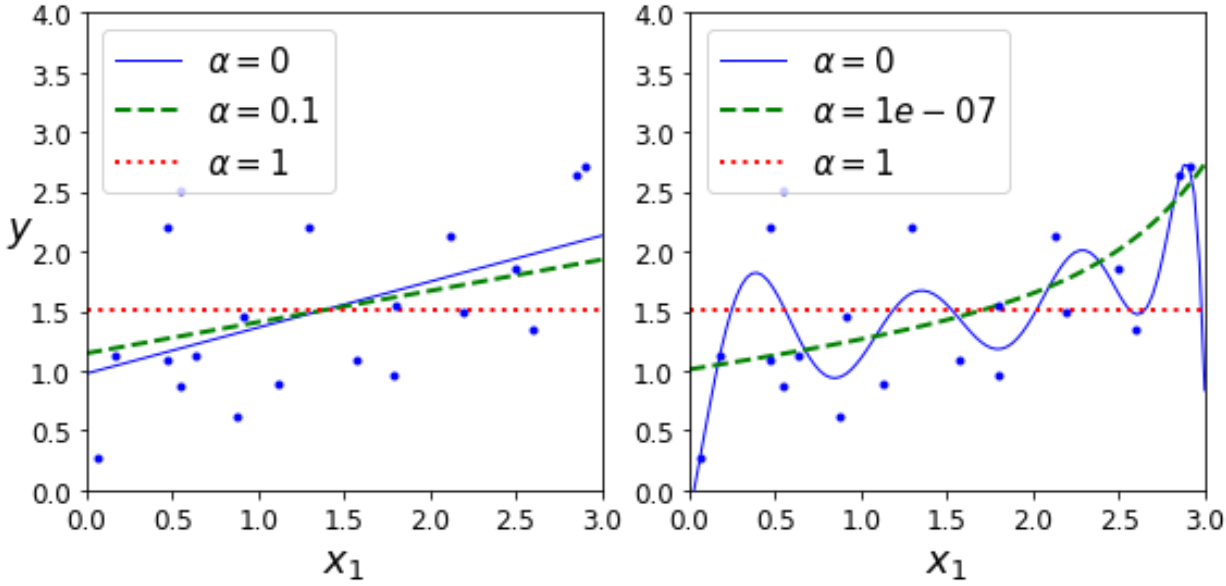

Рис.4.14. Гребневая регрессия при различных α

Важной характеристикой лассо-регрессии является то, что она стремится полностью исключить веса наименее важных признаков (т.е. устанавливает их в ноль). Например, пунктирная линия на графике справа на рис. 4. 14 (с α = 10-7) выглядит квадратичной, почти линейной: все веса для полиномиальных признаков высокой степени равны нулю. Другими словами, лассо-регрессия автоматически выполняет выбор признаков и выпускает разреженную модель (т.е. с незначительным числом ненулевых весов признаков). Осознать, почему это так, можно с помощью рис. 2.30: на левом верхнем графике фоновые контуры (эллипсы) представляют нерегуляризированную функцию издержек MSE ($\alpha = 0$), а белые кружочки показывают путь пакетного градиентного спуска (BGD) с этой функцией издержек. Контуры переднего плана (ромбы) представляют штраф l_1, а треугольники показывают путь BGD только для данного штрафа $\alpha \to \infty$. Обратите внимание на то, как путь сначала достигает $\theta_1 = 0$, после чего катится вниз по желобу до тех пор, пока не добирается до $\theta_2 = 0$. На правом верхнем графике контуры представляют ту же самую функцию издержек плюс штраф l_1 с α= 0.5. Глобальный минимум находится на оси $\theta_2 = 0$. Путь BGD сначала достигает $\theta_2 = 0$. и затем катится по желобу, пока не доходит до глобального минимума.

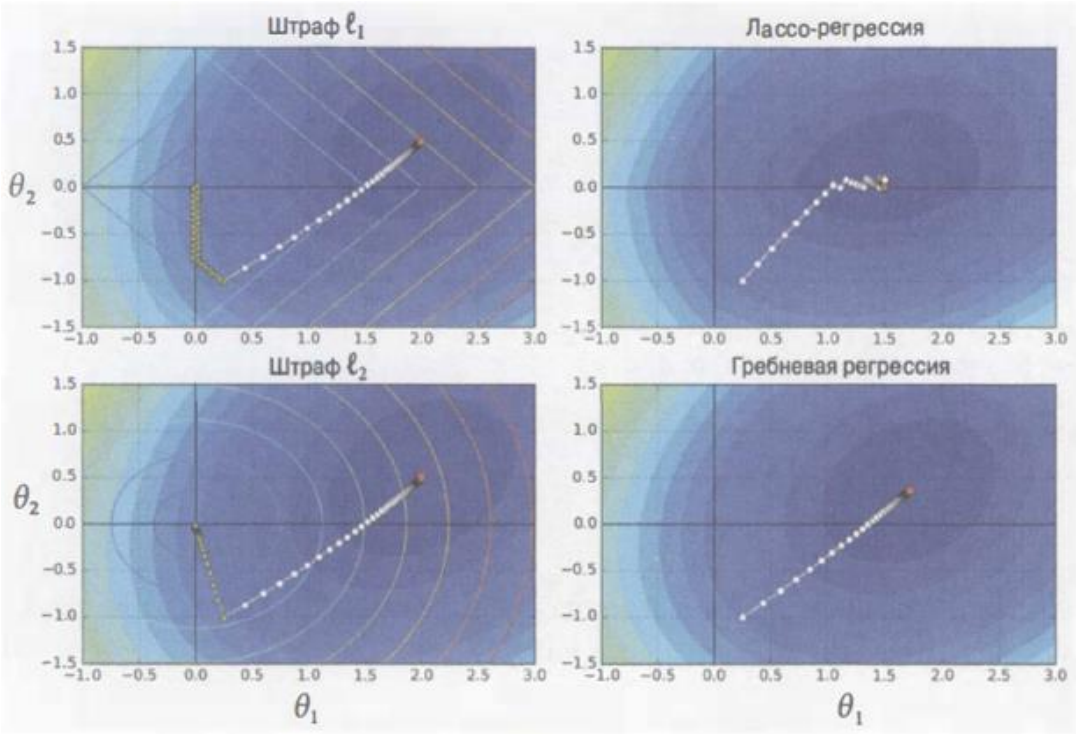

Рис.4.15. Сравнение Лассо-регрессии и Гребневая регрессия при различных α.

Два нижних графика демонстрируют те же вещи, но взамен используют штраф l_2.Регуляризированный минимум ближе к $\theta = 0$, чем нерегуляризированный минимум, но веса не полностью устранены. С функцией издержек лассо-регрессии путь BGD имеет тенденцию колебаться поперек желоба ближе к концу. Причина в том, что в точке $\theta_2 = 0$ наклон внезапно изменяется. Чтобы действительно сойтись в глобальном минимуме, вам понадобится постепенно снижать скорость обучения. Функция издержек лассо-регрессии не является дифференцируемой при θ_i= 0 (для j = 1, 2, . . . , n), но градиентный спуск по-прежнему хорошо работает, если вы взамен применяете вектор-субградиент (subgradient vector), когда θ_i= 0. В (4.16) показано уравнение вектора-субградиента, которое можно использовать для градиентного спуска с функцией издержек лассо-регрессии.

g (θ, J)=∇_θMSE(θ)+α(sign(θ_1), sign(θ_2),… sign(θ_n)), где (4.16)

sign(θ_i)=-1,если θ_i<0, sign(θ_i)=0,если θ_i=0 sign(θ_i)=+1,если θ_i>0

Ниже приведен небольшой пример Scikit-Learn, в котором применяется класс Lasso. Обратите внимание, что взамен вы могли бы использовать SGDRegressor(penalty="11").

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)
lasso_reg.predict([[1.5]])
array([1.53788174])
```