# TP 2 – AOS1
# Regularization
# Corrigé

In the following, we will study the differences between the classical linear regression model and their most popular regularized versions: the ridge regression model and the Lasso model.

We will use the sklearn Python library and some tools from the scipy library.

```
from sklearn.linear_model import Ridge, LinearRegression
from scipy.linalg import svd
```

A custom class is provided in the file utils.py that models a linear random phenomena. You have to first create the random phenomena

```
from utils import Event
evt = Event(n_features=n_features, effective_rank=3, noise_level=1)
```

Then you can repeatedly sample from it

```
batch1_X, batch1_y = evt.sample(n_samples=100)
batch2_X, batch2_y = evt.sample(n_samples=200)
```

More formally the following model is implemented inside Event

$$y = X\beta + \varepsilon.$$

The coefficients are randomly generated unless given as argument to `Event`. They are accessible with `evt.coefficients`.

The matrix $X$ is created on demand when getting samples with the `sample` method. It is controlled by the argument `n_features` (10 features by default) and the `effective_rank` integer that is the "denoised rank" of $X$.

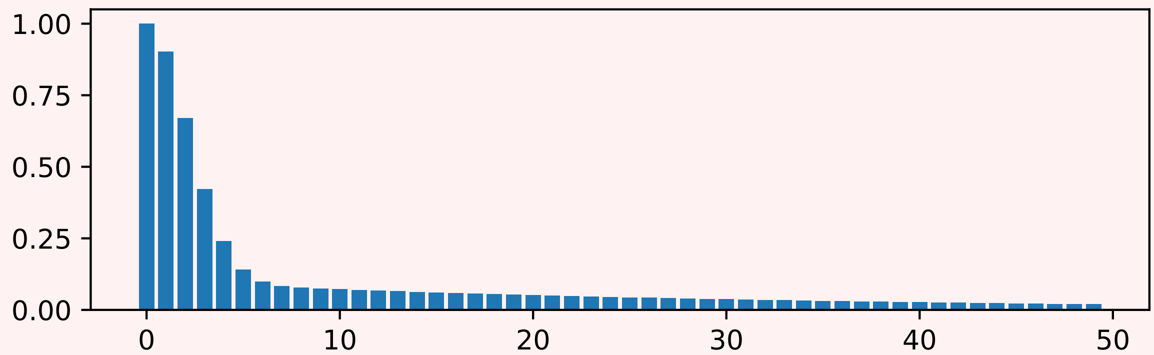
The standard deviation $\sigma$ of the noise $\varepsilon$ is controlled by the noise_level parameter in Event. More precisely, noise_level indicates a ratio in percentage between $\sigma$ and $\|\boldsymbol{\beta}\|$.

(1) Using a SVD, compute the singular values $d_i$, $1 \leqslant i \leqslant k$ of $X$. What is the effect of the parameter `effective_rank`? What is the effective rank supposed to model ?

In [1]:
```python
import scipy.linalg as linalg
sys.path.append('.')
from src.utils import Event
n_features = 50
evt = Event(n_features=n_features, effective_rank=3)
X, y = evt.sample()
U, s, VT = linalg.svd(X, full_matrices=False)
plt.bar(range(len(s)), s)
```

Out [1]:
```
<BarContainer object of 50 artists>
```

In [2]:
```python
plt.show()
```

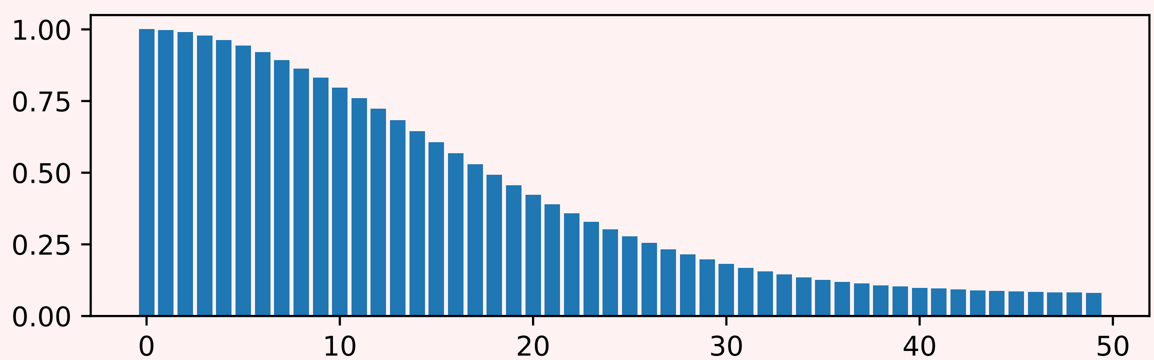



The rate of decay of singular values is high. There seem to be 3 or 4 significant singular values. After that they are exponentially decreasing.

In [3]:
```python
evt = Event(n_features=n_features, effective_rank=20)
X, y = evt.sample()
U, s, VT = linalg.svd(X, full_matrices=False)
plt.bar(range(len(s)), s)
```

Out [3]:
```
<BarContainer object of 50 artists>
```

In [4]:
```python
plt.show()
```




Second plot shows a decaying rate of singular values that is much smaller and it shows a change in curvature around 20, the chosen number of effective rank.

The effective rank is then the underlying number of factors that explains the event we sample from.

2 By repeatedly fitting each model on the same Event object, give an estimate of the bias, variance and risk of the estimator at a chosen point.

```
In [5]: import numpy as np
        from sklearn.linear_model import Ridge, LinearRegression, Lasso, LassoCV, RidgeCV
        n_features = 50
        n_samples = 200
        evt = Event(n_features=n_features, effective_rank=3, noise_level=1)
        X0 = np.random.randn(1, n_features)
        ntimes = 1000
```

Define a function `risk` that is returning an estimation of the variance, bias and risk of prediction at point `X0`.

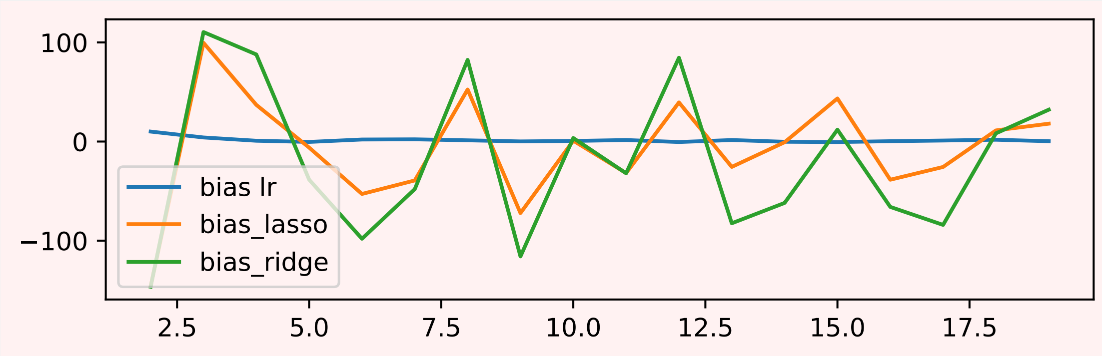
```
In [6]: def compute_risk(model, evt, ntimes, n_samples, X0):
            pred_model = []
            y0 = np.dot(X0, evt.coefficients)
            for i in range(ntimes):
                X, y = evt.sample(n_samples=n_samples)
                model.fit(X, y)
                pred = model.predict(X0)
                pred_model.append(pred)
            bias = np.mean(pred_model) - y0
            var = np.var(pred_model)
            risk = np.mean((pred_model - y0)**2)
            return (bias, var, risk)
        risk_lin = compute_risk(LinearRegression(), evt, ntimes, n_samples, X0)
        risk_ridge = compute_risk(Ridge(alpha=0.1), evt, ntimes, n_samples, X0)
        risk_lasso = compute_risk(Lasso(alpha=0.1), evt, ntimes, n_samples, X0)
        output = "Bias: %f\tVariance: %f\tRisk: %f"
        print("Linear regression: ", output % risk_lin)

Out [6]: Linear regression:  Bias: -4.256442        Variance: 11454.814940        Risk:
         ↪ 11472.932242

In [7]: print("Ridge regression: ", output % risk_ridge)

Out [7]: Ridge regression:  Bias: 69.582132        Variance: 4.290764        Risk: 4845.963790

In [8]: print("Lasso regression: ", output % risk_lasso)

Out [8]: Lasso regression:  Bias: 72.651781        Variance: 0.002695        Risk: 5278.283991
```

(3) Show the influence of the dataset's effective rank on the 3 algorithms and interpret.

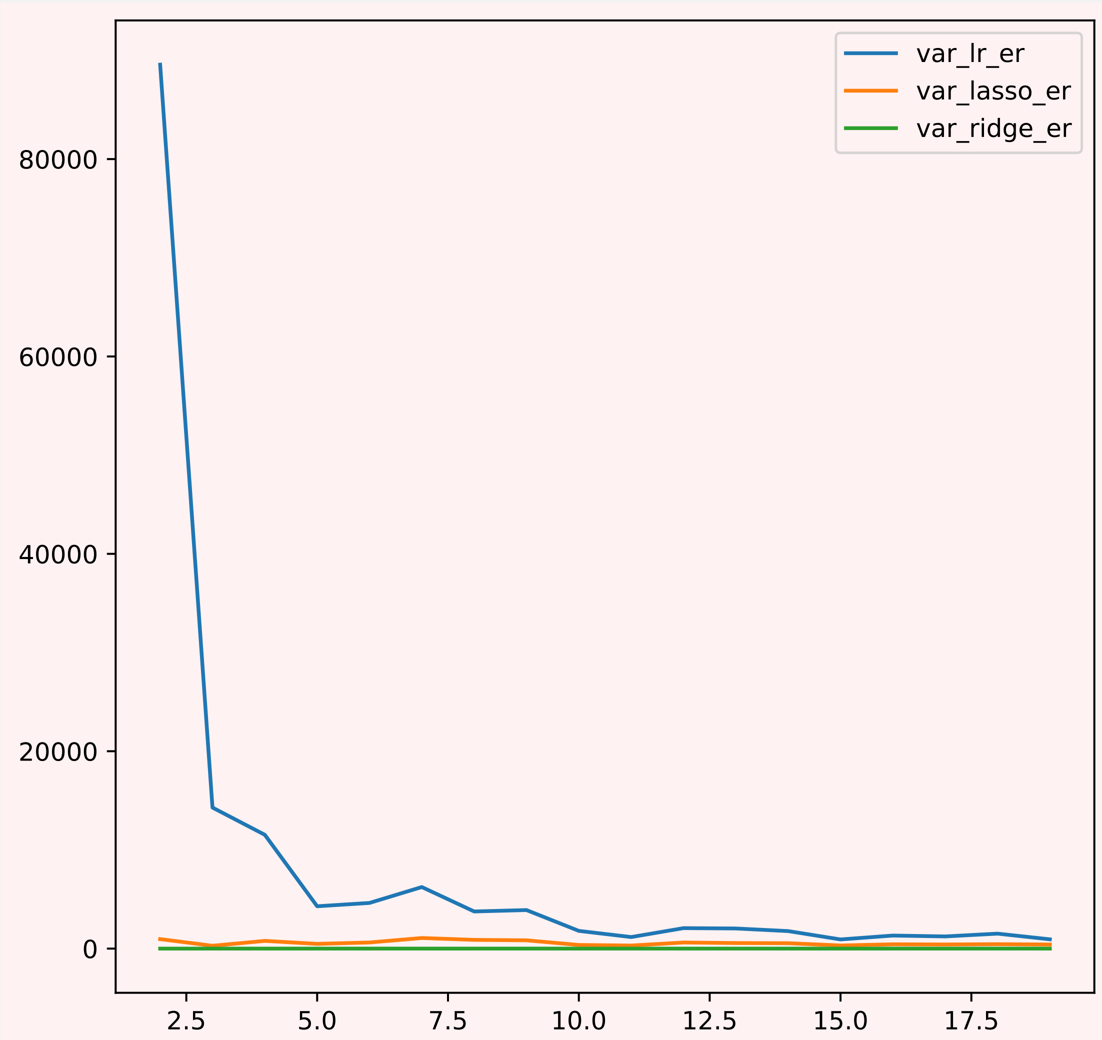
In [9]:
```python
X, y = evt.sample(n_samples=100)
lasso_alpha = LassoCV(cv=5, random_state=0).fit(X, y).alpha_
```

Out [9]:
```
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.013585333177825731, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.014613349221235694, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.015519848484395027, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.015671298279517742, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.01612506082045151, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.015436443511065434, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.014798256902690099, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.01416004803865789, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.013568568911857426, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.013907861210356032, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.013343431582359955, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.011726202251111317, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.011198312620315676, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
↪  number of iterations. Duality gap: 0.010639265986736035, tolerance:
↪  0.006317653499993123
  model = cd_fast.enet_coordinate_descent_gram(
/home/sylvain/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:470:
↪  ConvergenceWarning: Objective did not converge. You might want to increase the
```

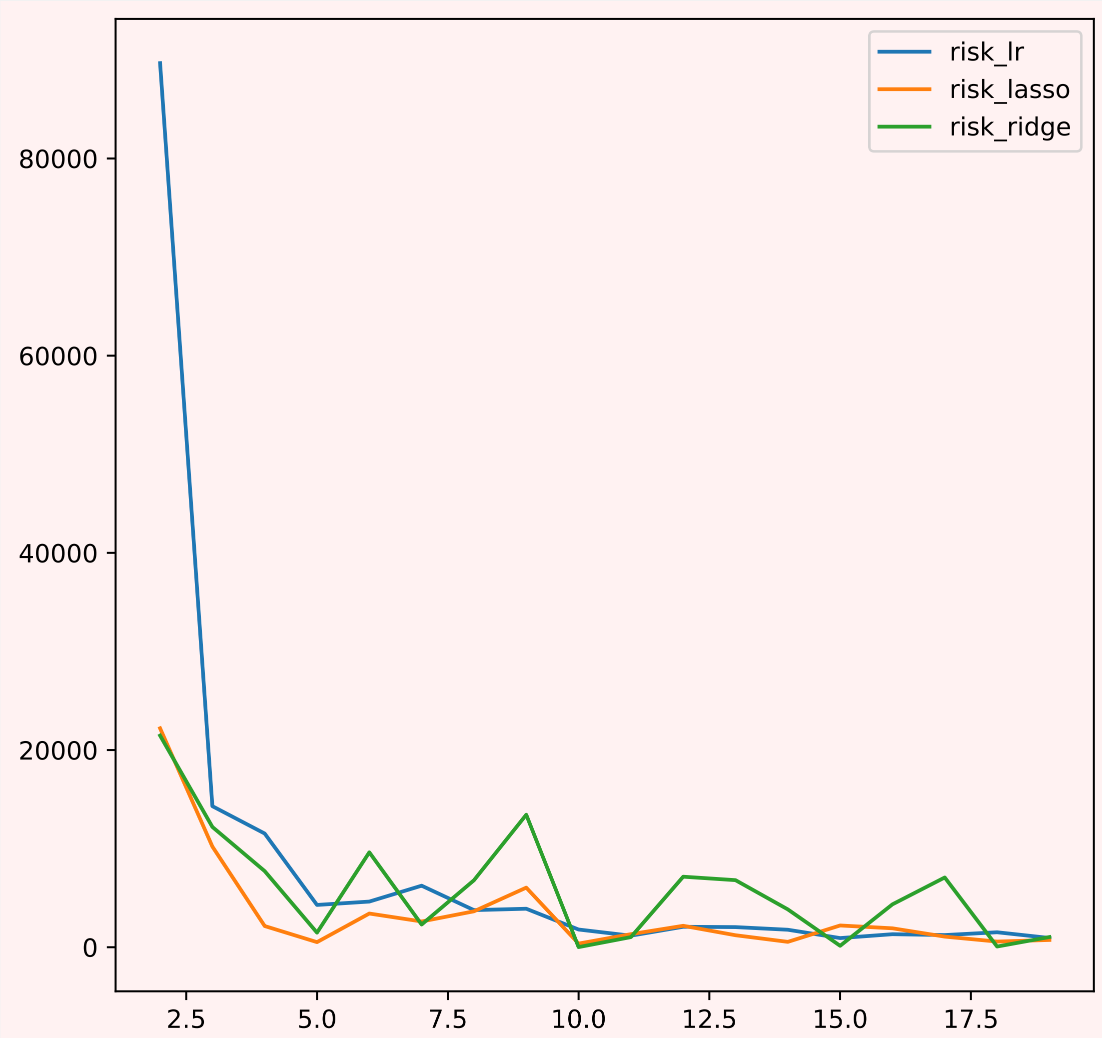
In [13]:
```
plt.plot(effective_ranks, bias_lr_er, label='bias lr')
plt.plot(effective_ranks, bias_lasso_er, label='bias_lasso')
plt.plot(effective_ranks, bias_ridge_er, label='bias_ridge')
plt.legend()
plt.show()
```



In [14]:
```
plt.figure()
plt.plot(effective_ranks, var_lr_er, label='var_lr_er')
plt.plot(effective_ranks, var_lasso_er, label='var_lasso_er')
plt.plot(effective_ranks, var_ridge_er, label='var_ridge_er')
plt.legend()
plt.show()
```



In [15]:
```
plt.figure()
plt.plot(effective_ranks, risk_lr_er, label='risk_lr')
plt.plot(effective_ranks, risk_lasso_er, label='risk_lasso')
plt.plot(effective_ranks, risk_ridge_er, label='risk_ridge')
plt.legend()
plt.show()
```




As expected, the linear regression gives an unbiased estimator that has a high variance when the effective rank is small. On the contrary, the ridge regression and lasso always have a nonzero bias but a smaller variance when the effective rank is small. The risk combining the bias and the variance is smaller for ridge regression or lasso.

## Problem: make elastic net outshine the lasso

The elastic net regression was invented to compensate for the lack of robustness of the lasso regression. The elastic net especially outshines the lasso when some variables are highly correlated and on the same scale. You have been shown in the last course's slides some intuitive arguments demonstrating why it should particularly be more robust in this case. But could you demonstrate this experimentally?

Design a regression dataset in which we want to select variables and where the elastic net gives better results in terms of stability of the set of selected variables.