In [1]:
```
# set tf 1.x for colab
%tensorflow_version 1.x
```

```
UsageError: Line magic function `%tensorflow_version` not found.
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore', category=DeprecationWarning)
warnings.filterwarnings('ignore', category=FutureWarning)
```

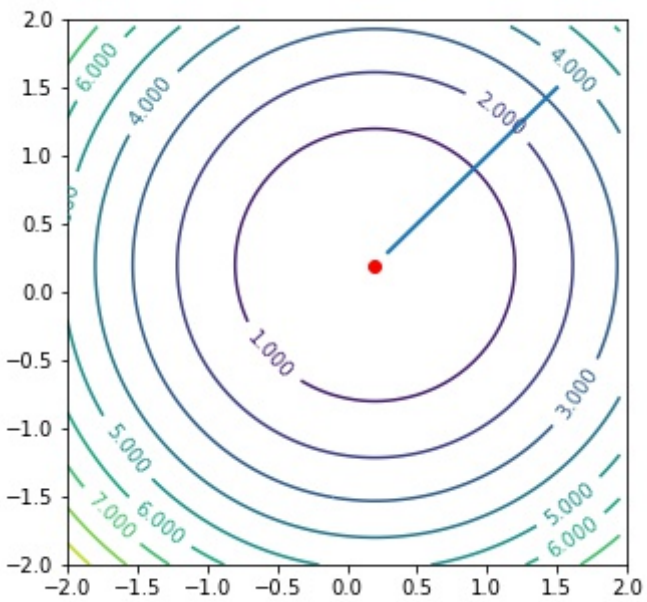
Read about ill-conditioning: http://cnl.salk.edu/~schraudo/teach/NNcourse/precond.html (http://cnl.salk.edu/~schraudo/teach/NNcourse/precond.html)

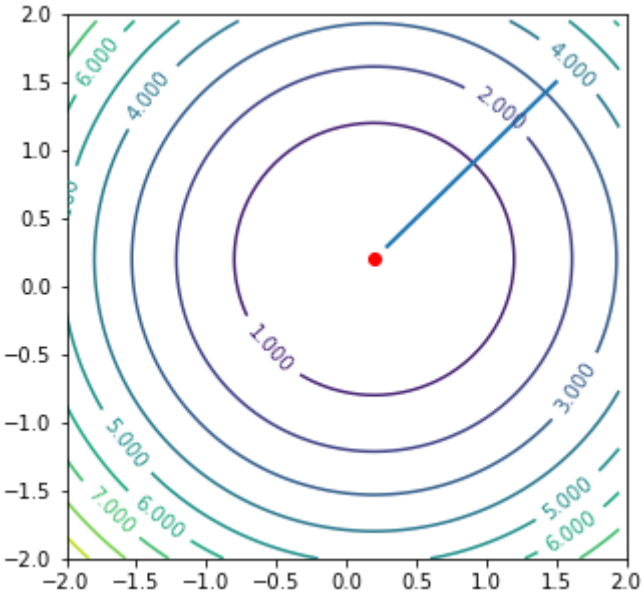
In [3]:
```
import tensorflow as tf
import sys
sys.path.append("../..")
from keras_utils import reset_tf_session
s = reset_tf_session()
print("We're using TF", tf.__version__)
from matplotlib import animation, rc
import matplotlib.pyplot as plt
import matplotlib_utils
from IPython.display import HTML, display_html
import numpy as np
```

```
WARNING:tensorflow:From ../..\keras_utils.py:68: The name tf.get_default_session is deprecated. Please use tf.compat.v
1.get_default_session instead.

WARNING:tensorflow:From ../..\keras_utils.py:75: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigP
roto instead.

WARNING:tensorflow:From ../..\keras_utils.py:77: The name tf.InteractiveSession is deprecated. Please use tf.compat.v1.
InteractiveSession instead.


Using TensorFlow backend.

We're using TF 1.14.0
```

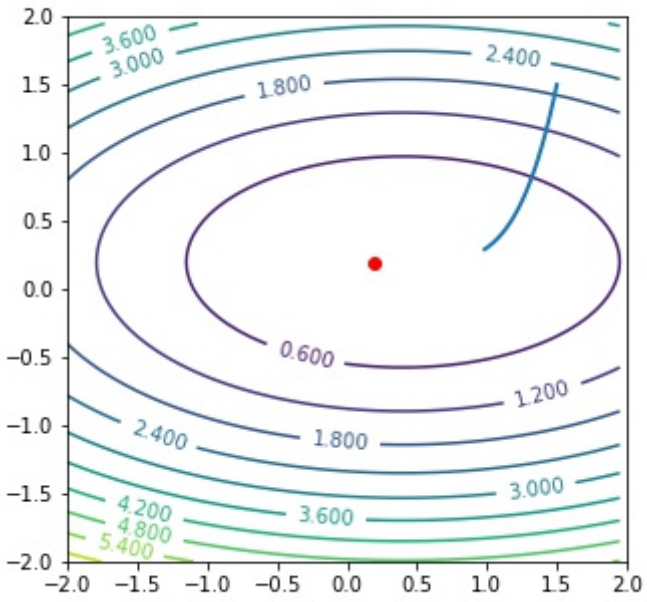
In [4]:
```
def plot_gd(x_scale, lr, steps):
    y_guess = tf.Variable([1.5, 1.5], dtype='float32')
    y_true = tf.constant([0.2, 0.2], dtype='float32')

    # x is x_scale times more important in loss (creates valleys)
    loss = tf.reduce_mean((tf.multiply(y_guess, tf.constant([x_scale, 1.])) - y_true)**2)

    step = tf.train.GradientDescentOptimizer(lr).minimize(loss, var_list=y_guess)

    # nice figure settings
    fig, ax = plt.subplots(figsize=(5, 5))
    y_true_value = s.run(y_true)
    level_x = np.arange(-2, 2, 0.05)
    level_y = np.arange(-2, 2, 0.05)
    X, Y = np.meshgrid(level_x, level_y)
    Z = (X * x_scale - y_true_value[0])**2 + (Y - y_true_value[1])**2
    ax.set_xlim(-2, 2)
    ax.set_ylim(-2, 2)
    s.run(tf.global_variables_initializer())
    ax.scatter(*s.run(y_true), c='red')
    contour = ax.contour(X, Y, Z, 10)
    ax.clabel(contour, inline=1, fontsize=10)
    line, = ax.plot([], [], lw=2)

    # start animation with empty trajectory
    def init():
        line.set_data([], [])
        return (line,)

    trajectory = [s.run(y_guess)]

    # one animation step (make one GD step)
    def animate(i):
        s.run(step)
        trajectory.append(s.run(y_guess))
        line.set_data(*zip(*trajectory))
        return (line,)

    anim = animation.FuncAnimation(fig, animate, init_func=init,
                                   frames=steps, interval=20, blit=True)

    anim.save(None, writer=matplotlib_utils.SimpleMovieWriter(0.0001))
```

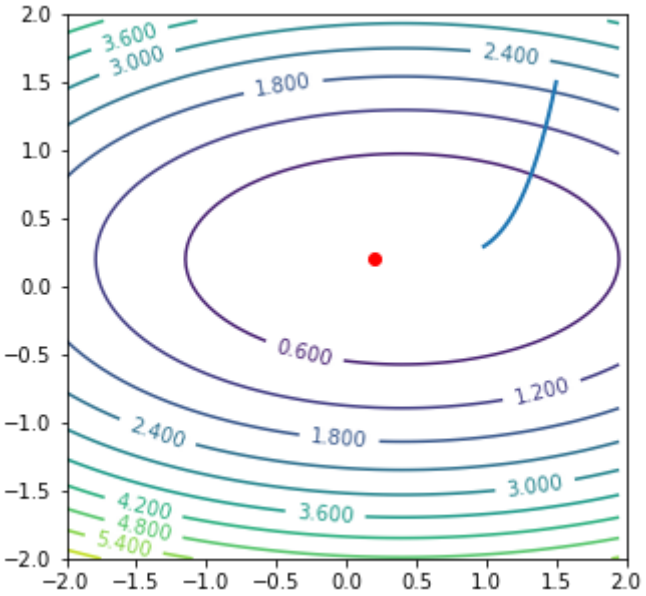
In [5]:
```
1 plot_gd(x_scale=1.0, lr=0.1, steps=25)
```



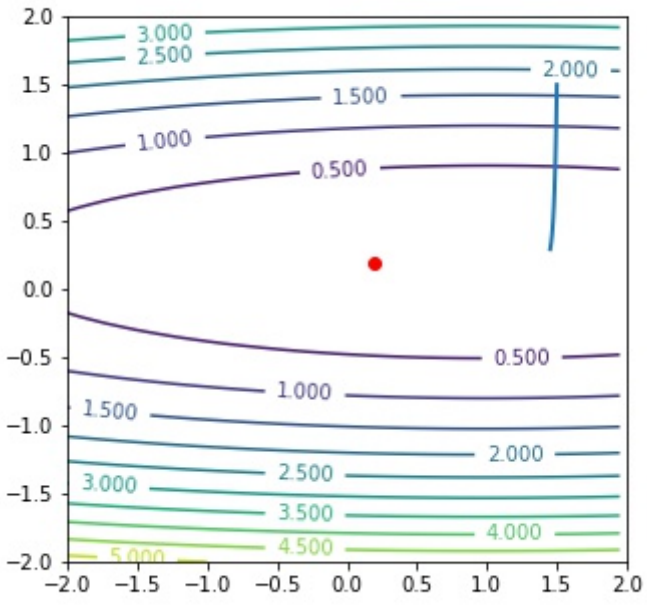


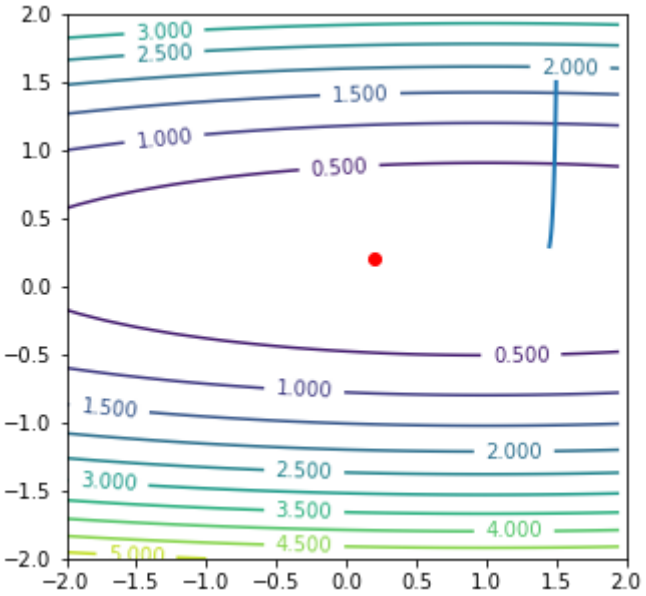
In [6]:
```
# narrow valleys
plot_gd(x_scale=0.5, lr=0.1, steps=25)
```



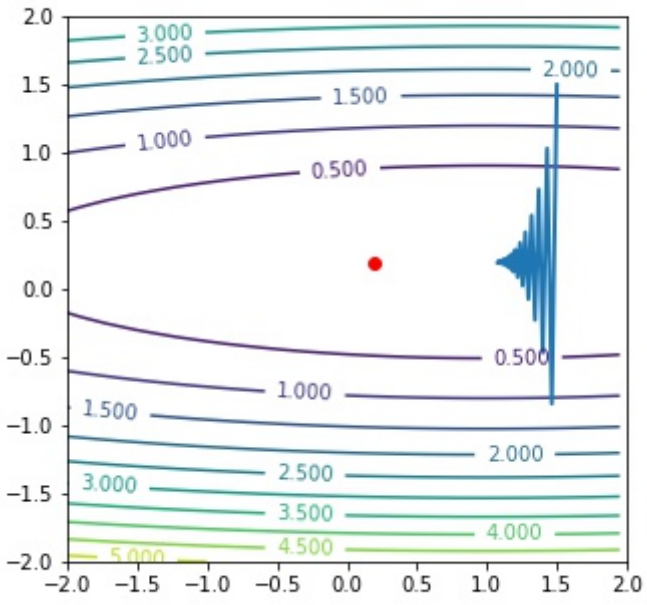


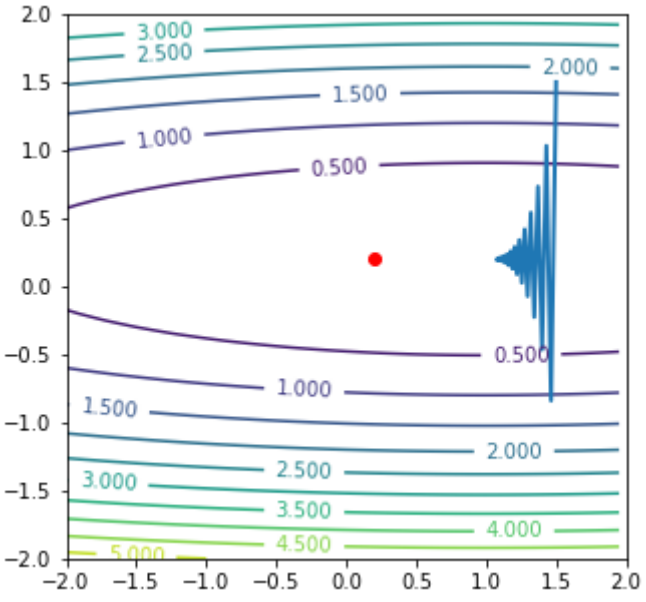
In [7]:
```
# narrower valleys
plot_gd(x_scale=0.2, lr=0.1, steps=25)
```





In [8]:
```
# bigger learning rate then?
# x is changed faster, but y changes are too big, leads to oscillation
plot_gd(x_scale=0.2, lr=1.8, steps=25)
```





In [ ]:
```
1
```