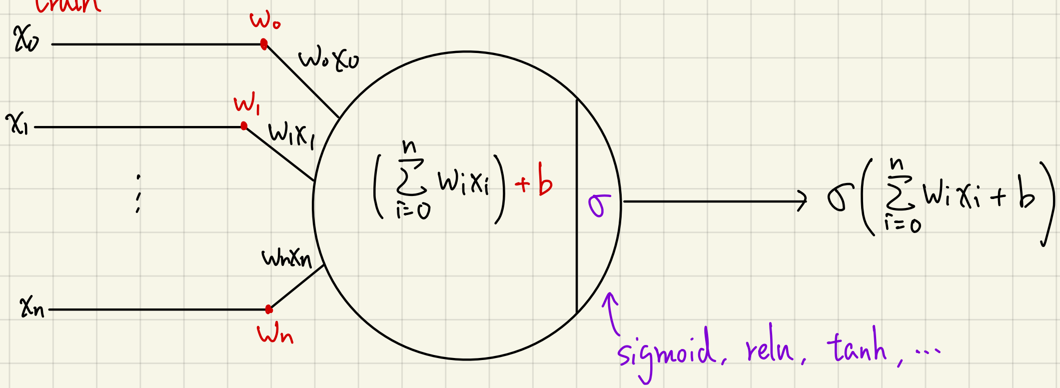
Training

The most enigmatic procedure in machine learning is training of neural networks, or, in general, parametric families of functions. This is of course an inverse problem, on which we have developed two perspectives above: the optimization perspective (with regularization) and the Bayesian perspective.

Essentially training is described as minimization of loss, i.e. for a given loss function L on a space of functions f parametrized by a set of parameters $\theta \in \Theta$

$$\operatorname{arginf}_{\theta\in\Theta} L(f^\theta)$$

w, b : need to train



f : our neural network

Θ : weights, biases

L : loss function

Assume that Θ is some open subset of points in $\mathbb{R}^d$ and $U : \Theta \to \mathbb{R}$, $\theta \mapsto L(f^\theta)$ a sufficiently regular function with a unique minimum $\theta^* \in \Theta$, then one can describe essentially one local and one global method to find the infimum:

1. If U is strictly convex and C^2 in a neighborhood of the unique minimizer θ^*, then

$$d\theta_t = \underline{-\nabla_\Theta U(\theta_t)}dt \quad \text{towards minimum}$$

converges to θ^* as $t \to \infty$. For any $t \geq 0$ it holds that

$$dU(\theta_t) = -||\nabla U(\theta_t)||^2 dt\,,$$

i.e. the value of U is strictly increasing along the path $t \mapsto \theta_t$. Together with the fact that U is strictly convex we obtain a convergence of $||\theta_t - \theta^*|| \leq C\exp(-\lambda_{\min} t)$ as $t \to \infty$, where $\lambda_{\min}$ is the minimum of the smallest eigenvalue of the Hessian of U on Θ. This holds remarkably for any starting point $\theta_0 \in \Theta$ and is the basis of all sorts of gradient descent algorithms.

$d\theta_t \rightsquigarrow \theta_{t+1} - \theta_t$ * θ_t is a vector, not function

$\Rightarrow \theta_{t+1} - \theta_t = -\delta \nabla U(\theta_t)\, dt$ (here, set $\delta = 1$) $= -\delta_t \nabla U(\theta_t)$ (check the proof of G.D.)

Since $U \in C^2$ and (strictly) convex, the gradient descent algorithm converges to θ^*.

Rmk.

- The convexity of U is necessary in the proof the convergence of G-D.
- gradient $\longrightarrow$ (global / local) maximum
- $-$ gradient $\longrightarrow$ (global / local) minimum
- δ_t determines the length $-\nabla U(\theta_t)$, also called the "learning rate"

2. A far reaching generalization is given by the following consideration: consider U on Θ having a unique minimizer $\theta^* \in \Theta$, then the probability measure given by the density with respect to Lebesgue measure on $\mathbb{R}^d$

$$p_\epsilon := \frac{1}{Z_\epsilon} \exp\big(-\frac{U}{\epsilon}\big)$$

tends in law to δ_{θ^*} as $\epsilon \to 0$. The denominator Z_ϵ is just the integral $\int_\Theta \exp(-U(\theta)/\epsilon) d\lambda(\theta)$ and the above statement nothing else than the fact that the described density function concentrates at θ^*. If one manages to sample from the measure $p_\epsilon d\lambda$, then one can approximate empirically θ^*.

The measure $p_\epsilon d\lambda$ is the invariant measure of the stochastic differential equation

$$d\theta_t = -\frac{1}{2} \nabla U(\theta_t) dt + \sqrt{\epsilon} dW_t \,,$$

which is just checked by the following equality

$$\int_\Theta \big(-\frac{1}{2} \nabla U(\theta) \nabla f(\theta) + \frac{\epsilon}{2} \Delta f(\theta)\big) p_\epsilon(\theta) d\lambda(\theta) = 0$$

for all test functions f.

One method, which generalizes this thought in a time-dependent way, is to sample from a measure concentrating at θ^* is to simulate from a stochastic differential equation (for $\Theta = \mathbb{R}^N$) of the type

$$d\theta_t = -\nabla U(\theta_t) dt + \alpha(t) dW_t$$

where W is an N-dimensional Brownian motion and the non-negative quantity, called cooling schedule, $\alpha(t) = O(\frac{1}{\log(t)})$ as $t \to \infty$. For appropriate constants we obtain that θ_t converges in law to δ_{θ^*} as $t \to \infty$. This procedure is called simulated annealing and is the fundament for global mimimization algorithms of the type 'steepest descent plus noise'.

U: unique minimizer $\theta^* \in \Theta$

Consider a measure $p_\varepsilon = \frac{1}{Z_\varepsilon} \exp(-\frac{U}{\varepsilon})$, $Z_\varepsilon = \int_\Theta \exp(-\frac{U(\theta)}{\varepsilon})\, d\lambda(\theta)$

$\rightsquigarrow$ ① concentrate at θ^* as $\varepsilon \to 0$.

② if one manages to sample from the measure $p_\varepsilon d\lambda$, then one can approximate θ^*.

before: $d\theta_t = -\nabla U(\theta_t)\, dt$

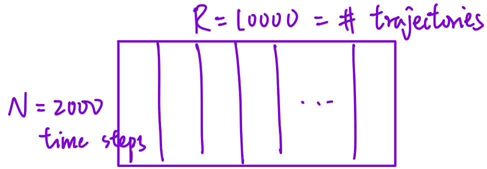
now: $d\theta_t = -\nabla U(\theta_t)\, dt + \alpha(t)\, dW_t$, W_t: non-negative Brownian motion

$\alpha(t)$: violatility

* variance of Brownian motion: $s-t$ if W_t to W_s
* violatility affects the variance

In [4]:

```python
import numpy as np

# Trajectories of d \theta = - \nabla U (\theta) dt + \alpha d W

N=2000 # time disrectization
theta0=1 # initial value of theta
T=100 # maturity
alpha = 0.1 # volatility in Black Scholes

R=10**5 # number of Trajectories

def U(theta):
  return theta**2*(theta-2)**2
  #return theta**2*(theta-2)**2*np.sin(theta)**2

def gradU(theta):
  return 2*theta*(theta-2)**2 + theta**2*2*(theta-2)
  #return (2*theta*(theta-2)**2 + theta**2*2*(theta-2))*np.sin(theta)**2+2*np.sin(theta)*np.cos(theta)*th

theta = np.zeros((N,R))
theta[0,]=theta0*np.ones((1,R))


for j in range(N-1):
    increment = np.random.normal(0,np.sqrt(alpha)*np.sqrt(T)/np.sqrt(N),(1,R))
    theta[j+1,:] =theta[j,:]+ increment - 0.5*gradU(theta[j,:])*(T/N)
```



→ initial value of each trajectory is 1

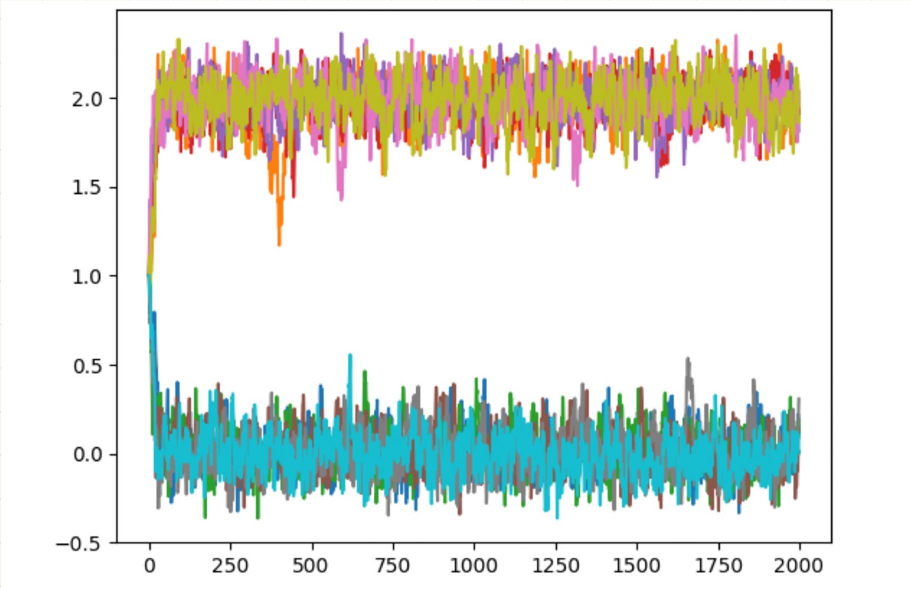
std. deviation

output shape (row vector)

$[N(\mu,\sigma^2), N(\mu,\sigma^2), \cdots, N(\mu,\sigma^2)]$

result →

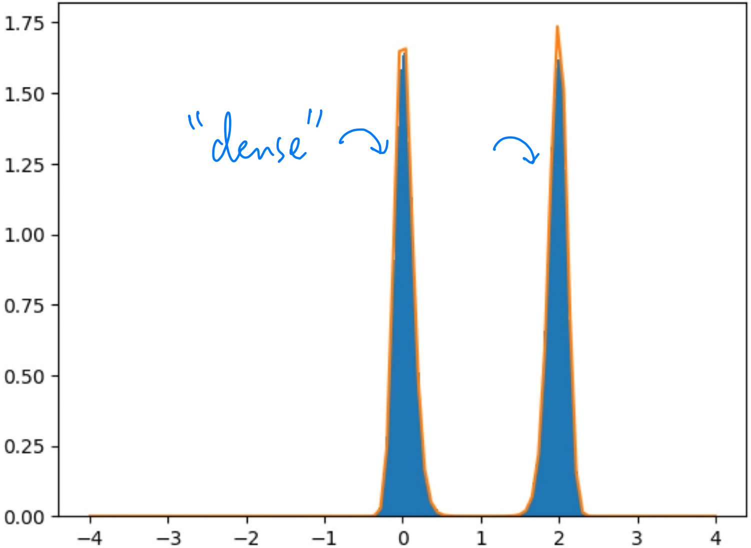
(前10條 trajectories)



In [6]:

```
a=4
x=np.linspace(-a,a,100)
plt.hist(theta[N-1,:],density=True,bins=100)
plt.plot(x,np.exp(-U(x)/alpha)/(2*a/100*sum(np.exp(-U(x)/alpha))))
plt.show()
```

$$p_\varepsilon = \frac{\exp\left(-\frac{U(\theta)}{\alpha}\right)}{\sum_{\theta} \exp\left(-\frac{U(\theta)}{\alpha}\right)}$$



The stochastic gradient descent algorithm essentially says for a function of expectation type $U(\theta) = E[V(\theta)]$

$$\theta_{n+1} = \theta_n - \gamma_n \nabla V(\theta_n, \omega_n)$$

for independently chosen samples ω_n converges in law to θ^*. Notice first that all our examples, in particular the ones from mathematical finance, are of expectation type, where the samples ω are usually seen as elements from the training data set.

GD

```
for i in range(# epochs):
    θ = θ - η grad(L, data, θt)
```

e.g. $L = \sum_i (\hat{y}^i - (b + \sum_j w_j x_j))^2$ (MSE)

summation over all training examples

$$U(\theta) = E[V(\theta)]$$

rewrite

$$U(\theta) = \mathbb{E}_{n \sim Unif}[U_n(\theta)] = \mathbb{E}_{n \sim Unif}[U(\theta, w_n)]$$

Let N = data size = # training examples.

SGD

```
for i in range(# epochs):
    np.random.shuffle(data)
    for example in data:
        θ = θ - η grad(L, example, θt)
```

or

```
    example = np.random.uniform(1, len(data)+1)
    θ = θ - η grad(L, Wexample, θ)
```

for n in range N:

$n \sim Unif[1, N]$, w_n: sample

$$\theta = \theta - \eta \nabla U_n(\theta)$$
$$= \theta - \eta \nabla U(\theta, w_n)$$

Lemma $\mathbb{E}[\nabla U(w_n, \theta)] = \nabla U(\theta)$

$\theta \longrightarrow \theta^*$

By the central limit theorem and appropriate sub-sampling one can understand

$$\nabla V(\theta, \omega) = \nabla U(\theta) + \text{ 'Gaussian noise with a certain covariance structure' } \Sigma(\theta)$$

where $\Sigma(\theta)$ is essentially given by

$$\operatorname{cov}(\nabla V(\theta)) \,.$$

Please check the paper: On the Noisy Gradient Descent that Generalizes as SGD