Module 5.8 : Line Search

Just one last thing before we move on to some other algorithms ...

- In practice, often a line search is done to find a relatively better value of η

- In practice, often a line search is done to find a relatively better value of η

```python
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

- In practice, often a line search is done to find a relatively better value of η
- Update w using different values of η

```python
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

- In practice, often a line search is done to find a relatively better value of η
- Update w using different values of η
- Now retain that updated value of w which gives the lowest loss

```python
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

- In practice, often a line search is done to find a relatively better value of η
- Update w using different values of η
- Now retain that updated value of w which gives the lowest loss
- Esentially at each step we are trying to use the best η value from the available choices

```python
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

- In practice, often a line search is done to find a relatively better value of η
- Update w using different values of η
- Now retain that updated value of w which gives the lowest loss
- Esentially at each step we are trying to use the best η value from the available choices
- What's the flipside?

```python
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

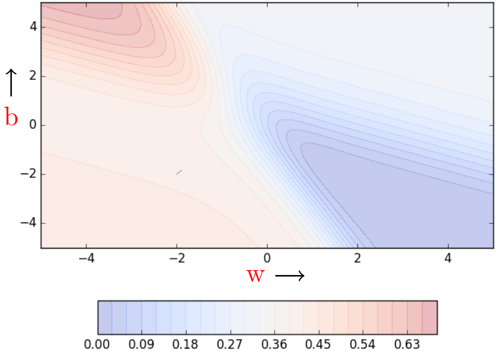
- In practice, often a line search is done to find a relatively better value of η
- Update w using different values of η
- Now retain that updated value of w which gives the lowest loss
- Esentially at each step we are trying to use the best η value from the available choices
- What's the flipside? We are doing many more computations in each step

```
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

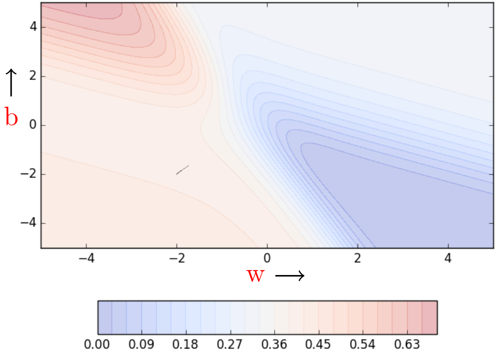
- In practice, often a line search is done to find a relatively better value of η
- Update w using different values of η
- Now retain that updated value of w which gives the lowest loss
- Esentially at each step we are trying to use the best η value from the available choices
- What's the flipside? We are doing many more computations in each step
- We will come back to this when we talk about second order optimization methods

```python
def do_line_search_gradient_descent():
    w, b, etas = init_w, init_b, [0.1, 0.5, 1.0, 5.0, 10.0]
    for i in range(max_epochs):
        dw, db = 0, 0
        for x,y in zip(X, Y):
            dw += grad_w(w, b, x, y)
            db += grad_b(w, b, x, y)
        min_error = 10000 #some large value
        best_w, best_b = w, b
        for eta in etas:
            tmp_w = w - eta * dw
            tmp_b = b - eta * db
            if error(tmp_w, tmp_b) < min_error:
                best_w = tmp_w
                best_b = tmp_b
                min_error = error(tmp_w, tmp_b)
        w, b = best_w, best_b
```

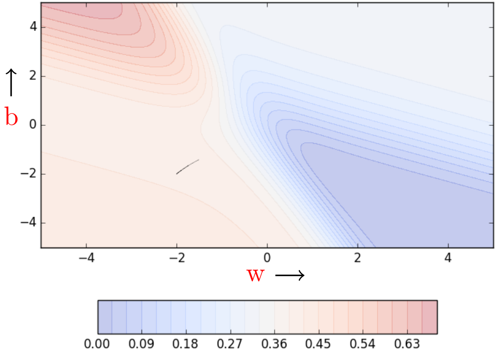
- Let us see line search in action



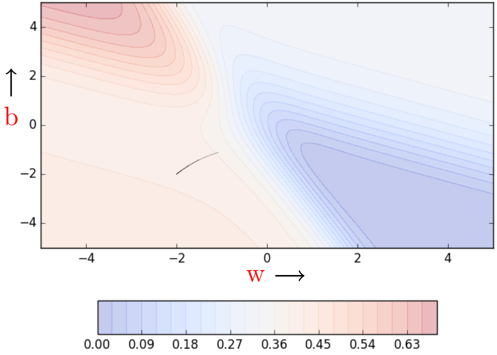
- Let us see line search in action



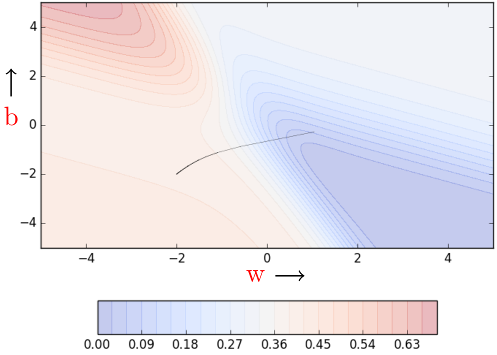
- Let us see line search in action



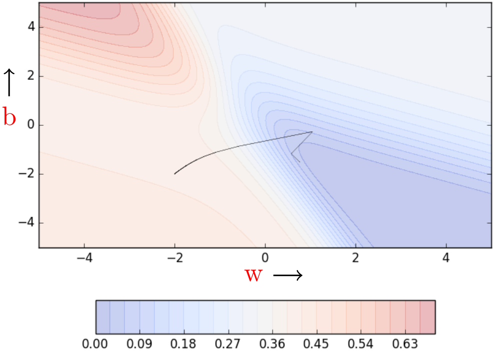
- Let us see line search in action



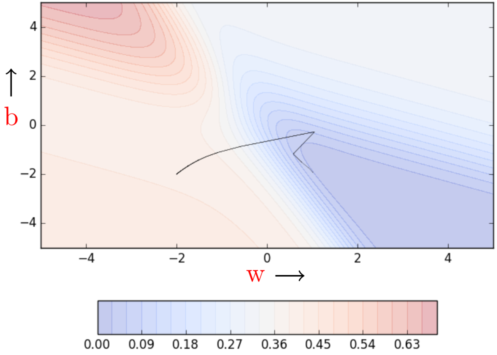
- Let us see line search in action



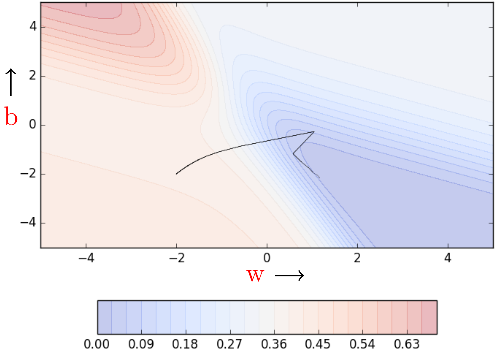
- Let us see line search in action



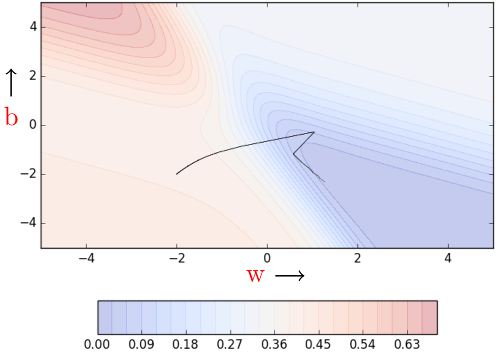
- Let us see line search in action



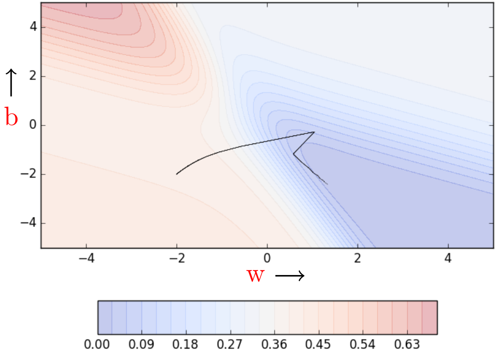
- Let us see line search in action



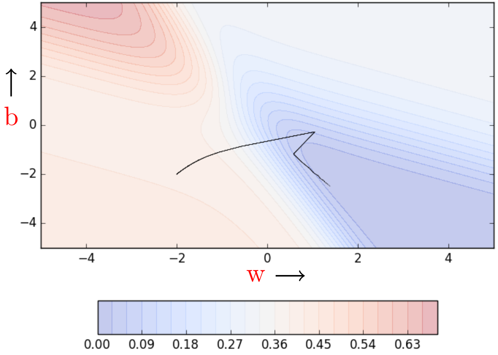
- Let us see line search in action



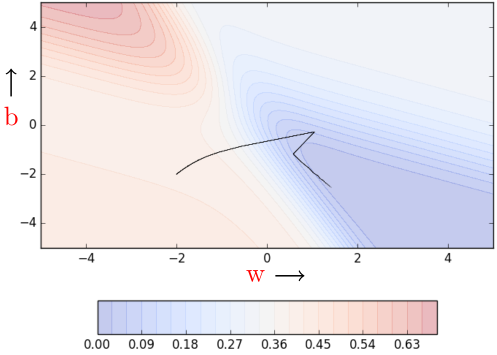
- Let us see line search in action



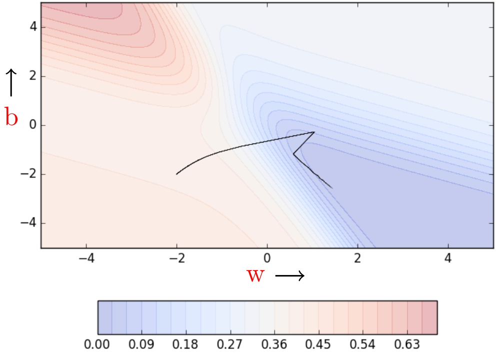
- Let us see line search in action



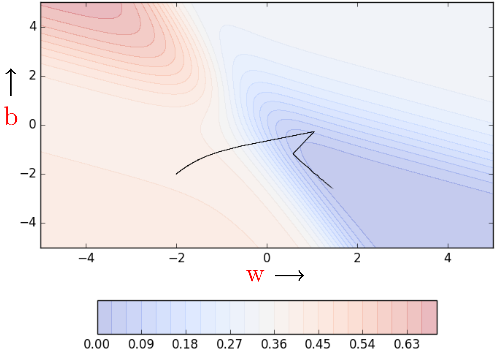
- Let us see line search in action



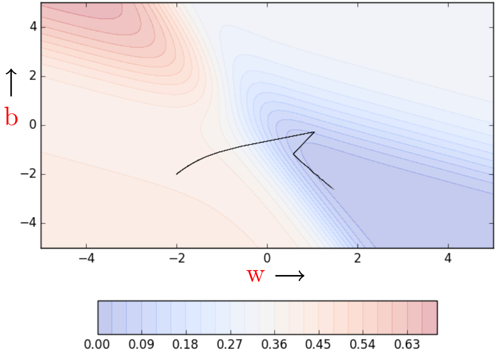
- Let us see line search in action



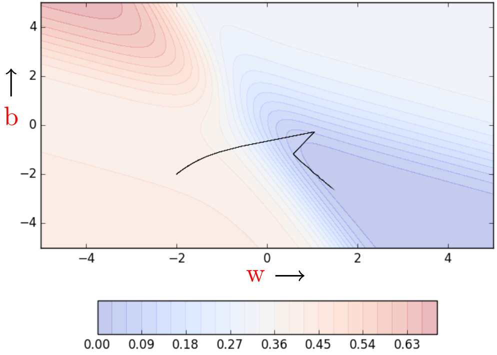
- Let us see line search in action



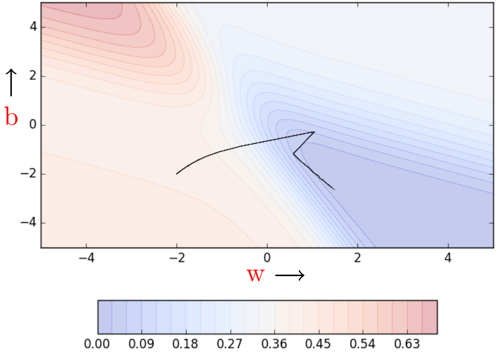
- Let us see line search in action



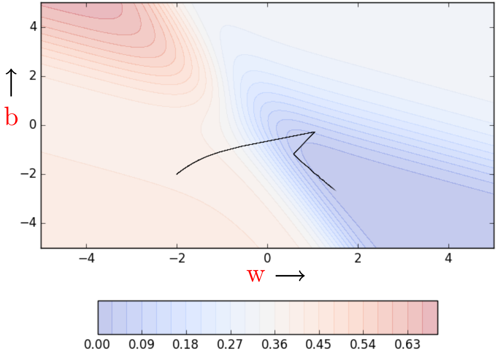
- Let us see line search in action



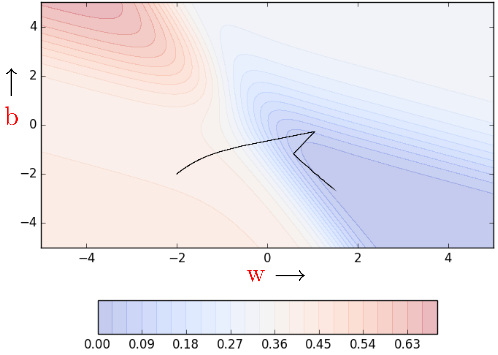
- Let us see line search in action
- Convergence is faster than vanilla gradient descent



- Let us see line search in action
- Convergence is faster than vanilla gradient descent
- We see some oscillations,



- Let us see line search in action
- Convergence is faster than vanilla gradient descent
- We see some oscillations, but note that these oscillations are different from what we see in momentum and NAG



- Let us see line search in action
- Convergence is faster than vanilla gradient descent
- We see some oscillations, but note that these oscillations are different from what we see in momentum and NAG