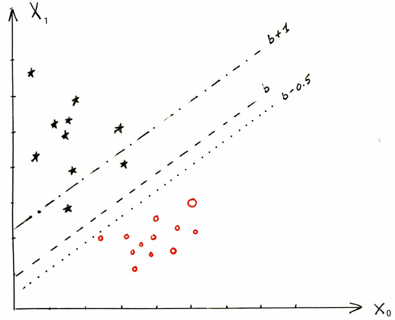
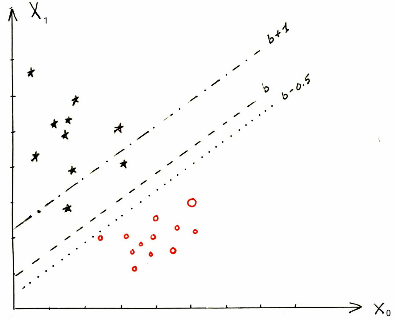
Best bias value

Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



Best bias value

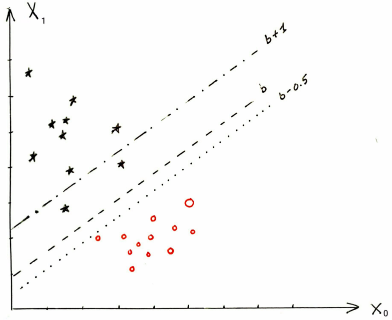
Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the hyperplane separating the instances.

Best bias value

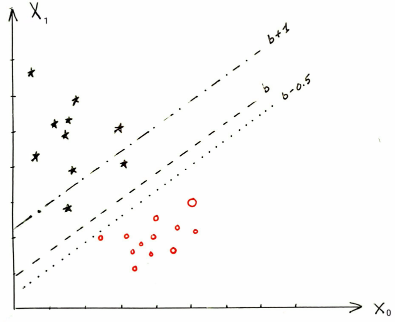
Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the hyperplane separating the instances.
- This decision boundary is actually our learned classifier.

Best bias value

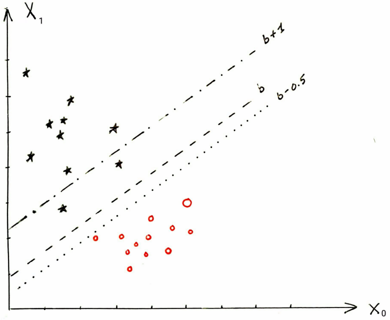
Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the hyperplane separating the instances.
- This decision boundary is actually our learned classifier.
- NB: the dataset on the plot is linearly separable.

Best bias value

Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the hyperplane separating the instances.
- This decision boundary is actually our learned classifier.
- NB: the dataset on the plot is linearly separable.
- **Question: lines with 3 values of b are shown. Which is the best?**

Loss

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$

Loss

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$

Loss

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$

Loss

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
4. Binary cross-entropy (logistic loss): $L(\hat{y}, y) = -y \; log\hat{y} - (1 - y)log(1 - \hat{y})$

Loss

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
4. Binary cross-entropy (logistic loss): $L(\hat{y}, y) = -y \ log \hat{y} - (1 - y) log(1 - \hat{y})$
5. Categorical cross-entropy (negative log-likelihood): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_i \boldsymbol{y}_{[i]} log(\hat{\boldsymbol{y}}_{[i]})$

Loss

Common loss functions

1. Hinge (binary): $L(\hat{y}, y) = max(0, 1 - y \cdot \hat{y})$
2. Hinge (multi-class): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = max(0, 1 - (\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
3. Log loss: $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = log(1 + exp(-(\hat{\boldsymbol{y}}_{[t]} - \hat{\boldsymbol{y}}_{[k]}))$
4. Binary cross-entropy (logistic loss): $L(\hat{y}, y) = -y \ log\hat{y} - (1 - y)log(1 - \hat{y})$
5. Categorical cross-entropy (negative log-likelihood): $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_i \boldsymbol{y}_{[i]} log(\hat{\boldsymbol{y}}_{[i]})$
6. Ranking losses, etc, etc...

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.

Regularization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

Why do you think this is? Pause the video and think.

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

  Why do you think this is? Pause the video and think.
- We can live with some errors on the training data, if it gives more generalization power.

Regularization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

  Why do you think this is? Pause the video and think.
- We can live with some errors on the training data, if it gives more generalization power.
- For that, we minimize both the loss and the regularization term $R(\theta)$:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) + \lambda R(\theta) \tag{1}$$

- The hyperparameter λ is regularization weight (how important is it).

Regularization

Regularization

- Sometimes, so as not to overfit, we pose restrictions on the possible θ.
- We would like θ to be not only good in predictions, but also not too complex; it should be 'lean' and avoid large weights.

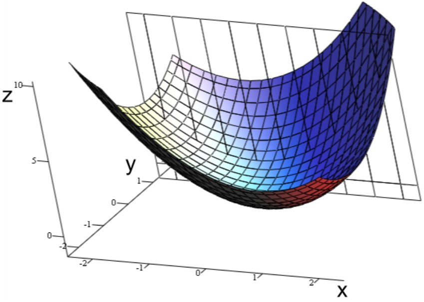
  Why do you think this is? Pause the video and think.
- We can live with some errors on the training data, if it gives more generalization power.
- For that, we minimize both the loss and the regularization term $R(\theta)$:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta) + \lambda R(\theta) \tag{1}$$

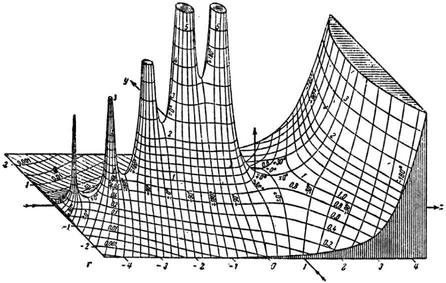
- The hyperparameter λ is regularization weight (how important is it).
- Common regularization terms:
  1. L_2 norm (*Gaussian prior* or *weight decay*);
  2. L_1 norm (*sparse prior* or *lasso*)

Error surface

Error surfaces of convex and not-convex functions:



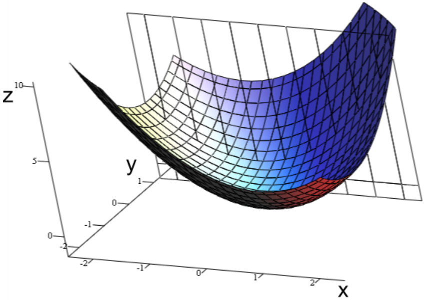
Convex function



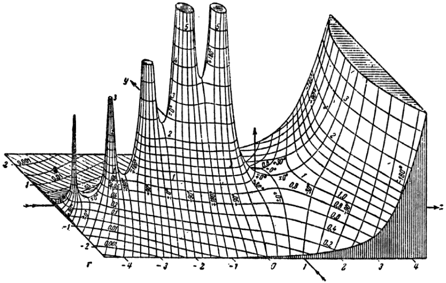
Non-convex function

Error surface

Error surfaces of convex and not-convex functions:



Convex function

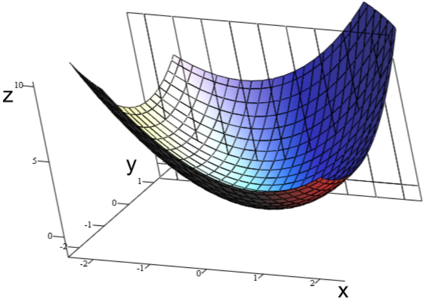


Non-convex function

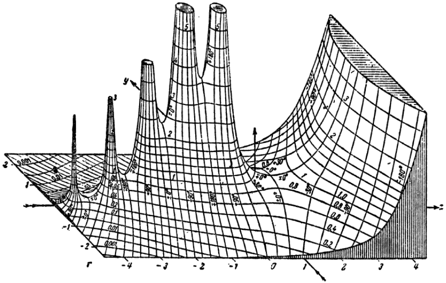
- Convex functions can be easily minimized with gradient methods, reaching the global optimum.
- With non-convex functions, optimization can end up in a local optimum.

Error surface

Error surfaces of convex and not-convex functions:



Convex function



Non-convex function

- Convex functions can be easily minimized with gradient methods, reaching the global optimum.
- With non-convex functions, optimization can end up in a local optimum.
- Linear and log-linear models as a rule have convex error functions.

XOR

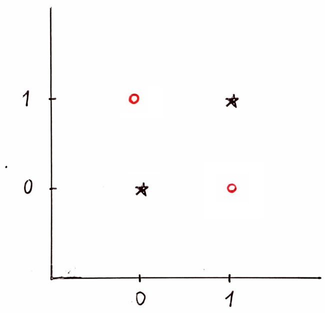
- Are there non-linear functions that linear models can't deal with?

XOR

- Are there non-linear functions that linear models can't deal with?
- Yes, there are.

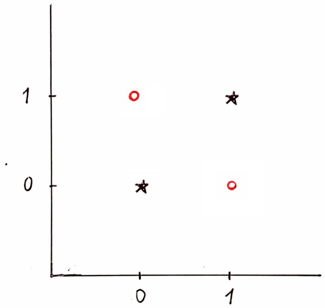
XOR

- Are there non-linear functions that linear models can't deal with?
- Yes, there are.
- One example is the XOR ('excluding OR') function:



XOR

- Are there non-linear functions that linear models can't deal with?
- Yes, there are.
- One example is the XOR ('excluding OR') function:



It is clearly not linearly separable.

XOR

Possible solutions

- We can transform the input so that it becomes linearly separable.

XOR

Possible solutions

- We can transform the input so that it becomes linearly separable.
- Linear transformations will not be able to do this.

XOR

Possible solutions

- We can transform the input so that it becomes linearly separable.
- Linear transformations will not be able to do this.
- We need non-linear transformations.

Possible solutions

- We can transform the input so that it becomes linearly separable.
- Linear transformations will not be able to do this.
- We need non-linear transformations.

For example, $\phi(x_1, x_2) = [x_1 + x_2, x_1 \times x_2]$ maps the instances to another representation and makes the XOR problem linearly separable: