Minimum Mean Square Error Estimation

Logistic Regression

Minimum Mean Square Error Estimation

Minimum Mean Squared Error

recap . . .

$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$

$\widehat{y} = \arg\max_y p(y|\boldsymbol{x})$ MAP estimate

$\hat{y} = \arg\max_y p(\boldsymbol{x}|y)$ ML (maximum likelihood) estimate

- MAP minimizes risk under 0/1 loss.
- ML minimizes risk under 0/1 loss and uniform prior.

$$E[\ell(\widehat{y}, y)] = E[\mathbb{1}_{\{\widehat{y} \neq y\}}] = \mathbb{P}(\widehat{y} \neq y)$$

MAP by definition picks the most probable y.

- MSE (mean squared error):

$$E[\ell(\widehat{y}, y)] = E[(\widehat{y} - y)^2]$$

- **claim**: minimum mean squared error estimate is given by

$\widehat{y} = f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$ MMSE (minimum mean squared error estimate)

(we need knowledge of $p(y|\boldsymbol{x})$ to find MMSE)

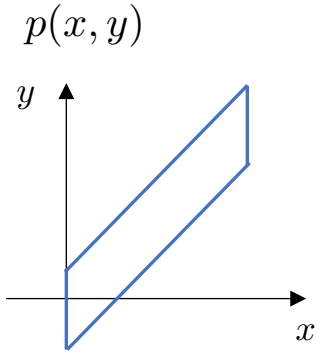
- note: MMSE is basis for denoising techniques such as Wiener filter, Kalman Filter.

MMSE Example

- claim: the MMSE estimate of y given the features $\boldsymbol{x}$ and complete knowledge of $p(\boldsymbol{x}, y)$ is:

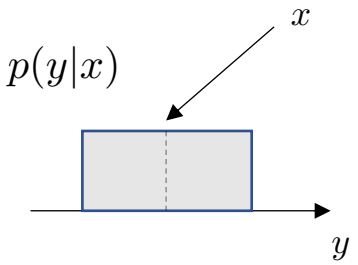
$$\widehat{y} = f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$$

- example:



- given x, what is your best guess for y?

$$y|x \sim U[x - \Delta, x + \Delta]$$



- MMSE: $\widehat{y} = E[y|x] = x$
- MAP: $\widehat{y}$ can be anything between $x - \Delta$ and $x + \Delta$

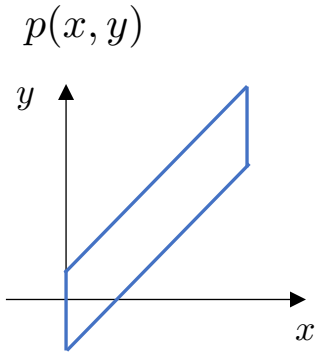
MMSE

- claim: the MMSE estimate of y given the features $\boldsymbol{x}$ and complete knowledge of $p(\boldsymbol{x}, y)$ is:

$$\widehat{y} = f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$$ proof on canvas.

- claim: the minimum mean squared error of the MMSE estimate of y given the features $\boldsymbol{x}$ and $p(\boldsymbol{x}, y)$ is:

$$E[(y - E[y|\boldsymbol{x}])^2] = E_{\boldsymbol{x}}[\text{var}(y|\boldsymbol{x})]$$



Logistic Regression

- logistic regression
- convex functions

Generative and Discriminative Classifiers

- if we have a good approximation of $p(\boldsymbol{x}, y)$, we know what to do:

$$\widehat{y} = \arg\max_y p(y|\boldsymbol{x}) \qquad \text{MAP estimate}$$

$$\widehat{y} = \arg\max_y p(\boldsymbol{x}|y) \qquad \text{ML (maximum likelihood) estimate}$$

$$\widehat{y} = E[y|\boldsymbol{x}] \qquad \text{MMSE estimate}$$

- probabilistic machine learning: we need to estimate $p(\boldsymbol{x}, y)$ from $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$
- most of the approaches we've studied have been *generative*
  - we imagine some distribution $p(\boldsymbol{x}|y)$ that *generates* our data

    example: imagine that $p(\boldsymbol{x}|y) \sim \mathcal{N}(\boldsymbol{u}_i, \Sigma_i)$
- logistic regression is a *discriminative* classifier
  - discriminative classifiers directly find $p(y|\boldsymbol{x})$ to *discriminate* between classes.

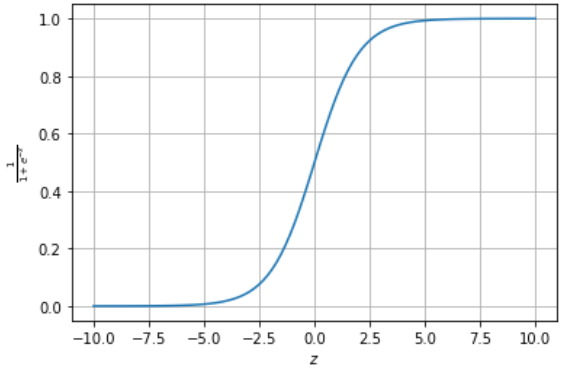
    directly fit $p(y|\boldsymbol{x})$ to data by treating $\boldsymbol{x}$ as non-random
  - we don't learn anything about the distribution of x

(Binary) Logistic Regression

- logistic regression is a *discriminative* classifier
- we imagine that y is a coin flip with bias that depends on the features $\boldsymbol{x}$

$$y_i = \begin{cases} 1 & \text{with probability } \theta_i \\ -1 & \text{with probability } 1 - \theta_i \end{cases} \qquad \theta_i = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \boldsymbol{w}}}$$

$$p(y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}^T \boldsymbol{w}}} \qquad p(y = -1|\boldsymbol{x}) = \frac{1}{1 + e^{\boldsymbol{x}^T \boldsymbol{w}}}$$



- testing: which is more probable?

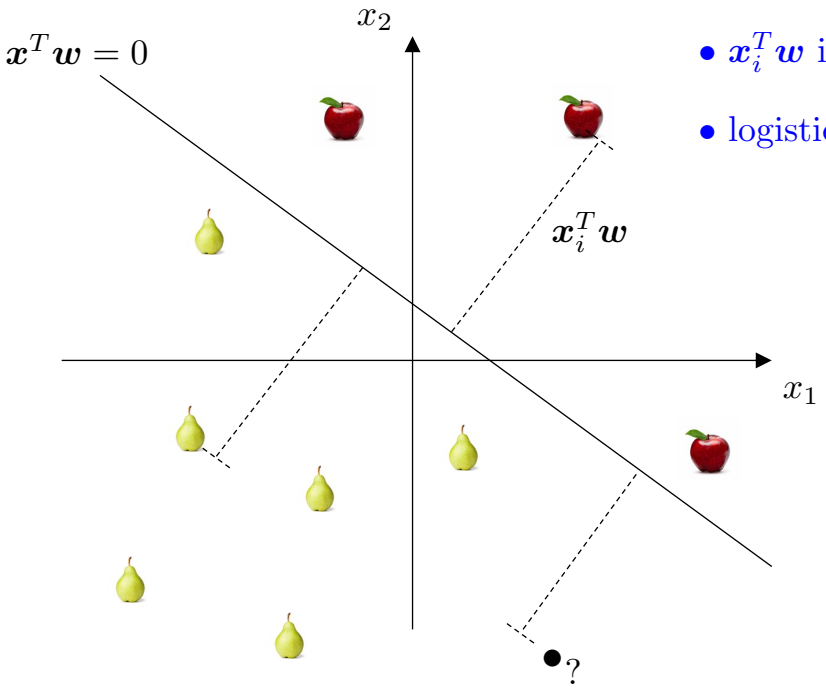
$$p(y = 1|\boldsymbol{x}) \text{ or } p(y = -1|\boldsymbol{x})$$ MAP estimate

- decision boundary is $\boldsymbol{x}^T \boldsymbol{w} = 0$

$$\frac{1}{1 + e^{-z}} = \frac{1}{2} \qquad \text{when } z = 0$$

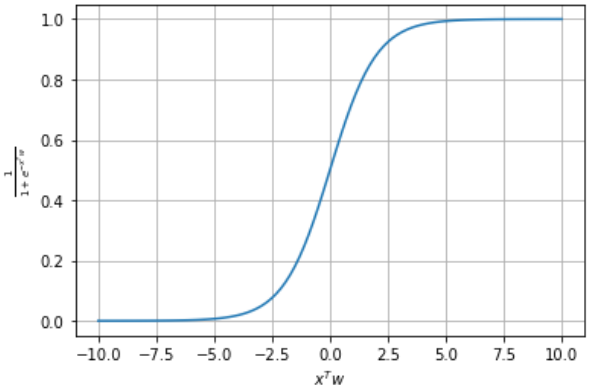
Logistic Regression

$$\boldsymbol{x}^T\boldsymbol{w} = \begin{bmatrix}1 & x_1 & x_2\end{bmatrix}\begin{bmatrix}w_0\\ w_1\\ w_2\end{bmatrix} = w_0 + x_1w_1 + x_2w_2$$



- decision boundary is $\boldsymbol{x}^T\boldsymbol{w} = 0$
- $\boldsymbol{x}_i^T\boldsymbol{w}$ is the signed distance from the decision boundary
- logistic function takes $\boldsymbol{x}_i^T\boldsymbol{w}$ and squishes it to make it a probability

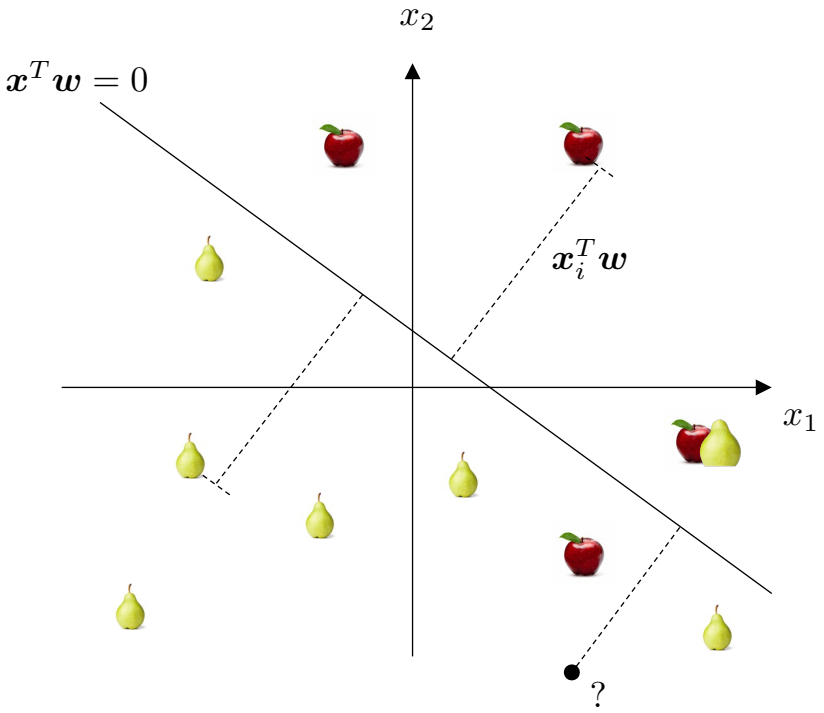
$$\frac{1}{1+e^{-\boldsymbol{x}_i^T\boldsymbol{w}}}$$



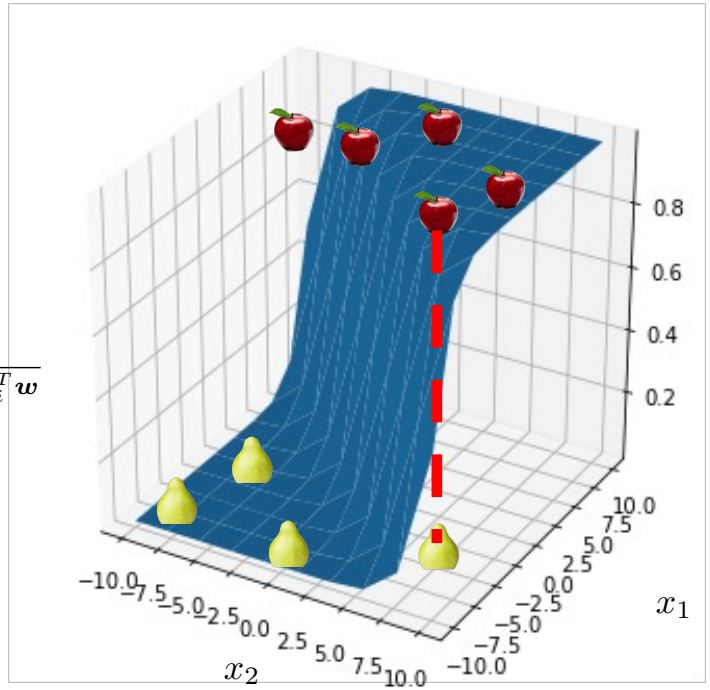
- probability that ? is a pear/apple depends distance from boundary

Logistic Regression

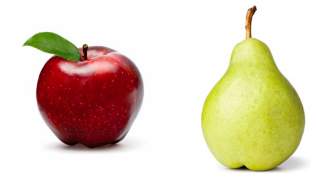
$$\boldsymbol{x}^T\boldsymbol{w} = \begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = w_0 + x_1 w_1 + x_2 w_2$$



$$\frac{1}{1 + e^{-\boldsymbol{x}_i^T\boldsymbol{w}}}$$



Training

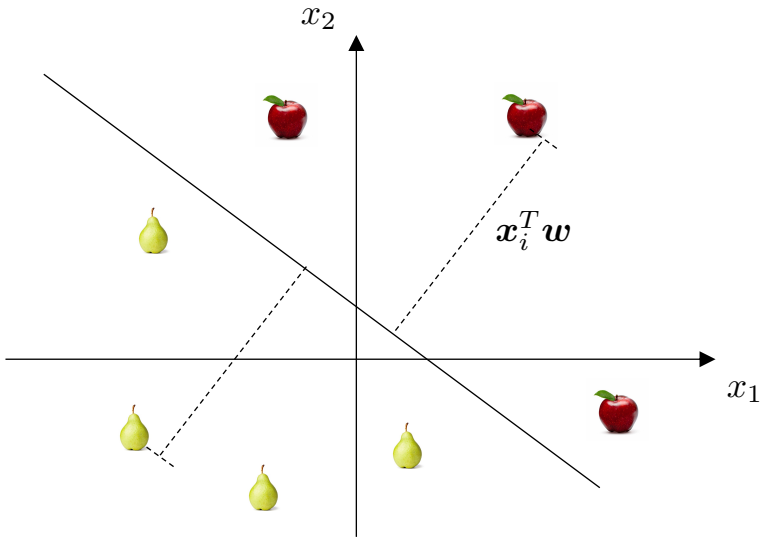


$y = -1$ if pear, $y = 1$ if apple

$$y_i = \begin{cases} 1 & \text{with probability } \theta_i \\ -1 & \text{with probability } 1 - \theta_i \end{cases} \qquad \theta_i = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \boldsymbol{w}}}$$

$$\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^m$$

- training: how do we find a good $\boldsymbol{w}$?
  - maximum likelihood principle
  - given data (and our model) what's the most likely $\boldsymbol{w}$?



$$p_{\boldsymbol{w}}(y_1, y_2, \ldots, y_n) = \prod_i \theta_i^{\mathbb{I}\{y_i=1\}} (1 - \theta_i)^{\mathbb{I}\{y_i=-1\}}$$

Training

- maximum likelihood principle

$$p_{\boldsymbol{w}}(y_1, y_2, \ldots, y_n) = \prod_i \theta_i^{\mathbb{I}\{y_i=1\}} (1-\theta_i)^{\mathbb{I}\{y_i=-1\}} \qquad \theta_i = \frac{1}{1+e^{-\boldsymbol{x}_i^T \boldsymbol{w}}}$$

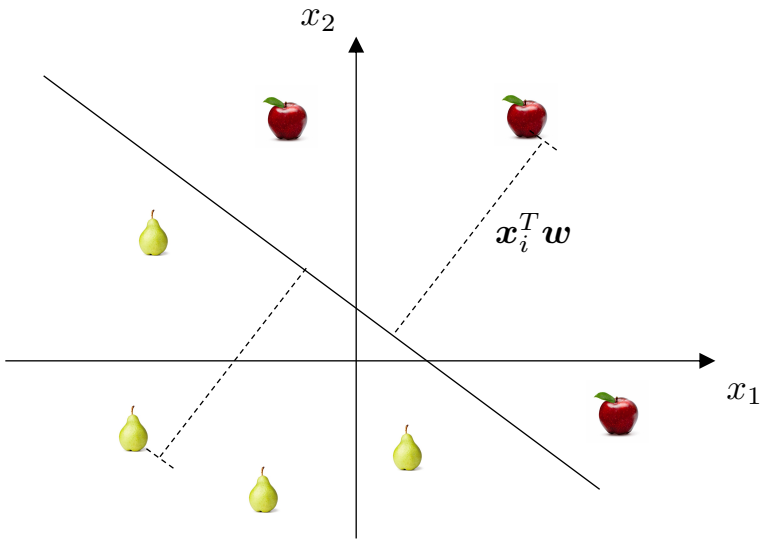
$$\log L(\boldsymbol{w}) = -\sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\max_{\boldsymbol{w}} \log L(\boldsymbol{w})$$

Training

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\max_{\boldsymbol{w}} -\sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

$$= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$



- training a classifier in general:

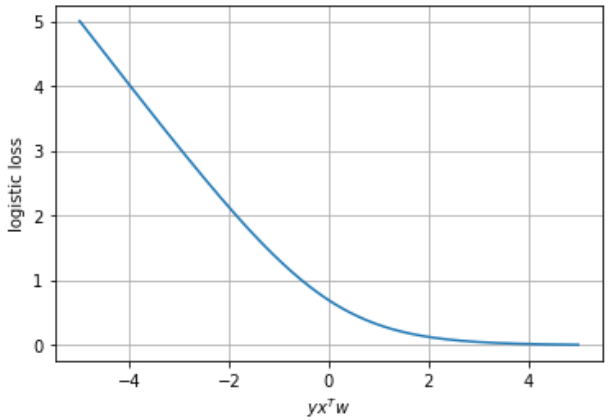
$$\min_{\boldsymbol{w}} \sum_{i} \ell_i(\boldsymbol{w})$$

- hinge loss $(1 - y_i \boldsymbol{x}_i^T \boldsymbol{w})_+$
- squared error $(\boldsymbol{x}^T \boldsymbol{w} - y_i)^2$
- absolute loss $|y_i - \boldsymbol{x}_i^T \boldsymbol{w}|$
- logistic loss $\log(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}})$

Training

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\max_{\boldsymbol{w}} -\sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$
$$= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

$\log(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}})$



- training a classifier in general:

$$\min_{\boldsymbol{w}} \sum_{i} \ell_i(\boldsymbol{w})$$

- hinge loss $(1 - y_i \boldsymbol{x}_i^T \boldsymbol{w})_+$
- squared error $(\boldsymbol{x}^T \boldsymbol{w} - y_i)^2$
- absolute loss $|y_i - \boldsymbol{x}_i^T \boldsymbol{w}|$
- logistic loss $\log(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}})$

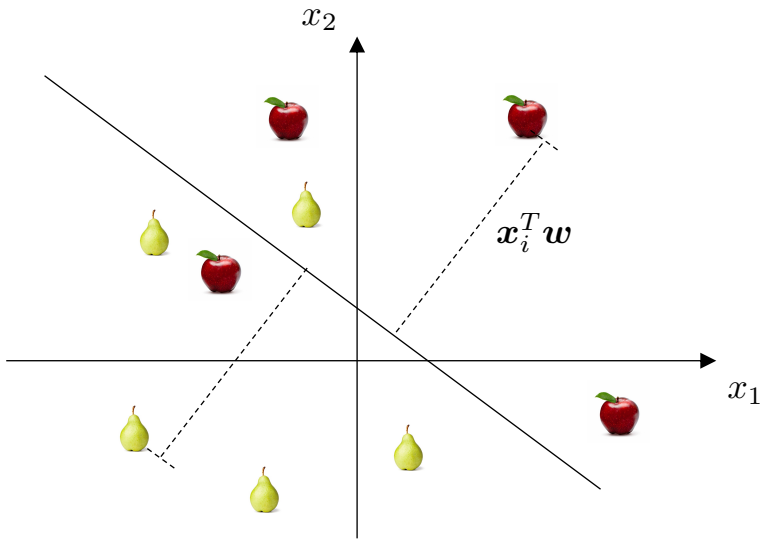
- if $y_i \boldsymbol{x}^T \boldsymbol{w}$ is positive, loss is small
- if sign of $\boldsymbol{x}^T \boldsymbol{w}$ and y_i agree, $\log(1 + \epsilon) \approx 0$

Optimizing

- training reduces to solving the following optimization

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

- changing $\boldsymbol{w}$ change decision boundary orientation and steepness of logistic function



- no closed form solution
- iterative methods for finding $\boldsymbol{w}$
  - logistic loss is easy to optimize . . .
  - convex and differentiable

Convex Functions

- definition: a function $f(x)$ is convex if

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \qquad \text{for all } \lambda \in [0,1], \text{ all } x_1, x_2$$

- interpretation: the line connecting two points on the surface of the function is *always* above the function
- equivalently, a function is convex if the second derivative is non-negative
- examples:

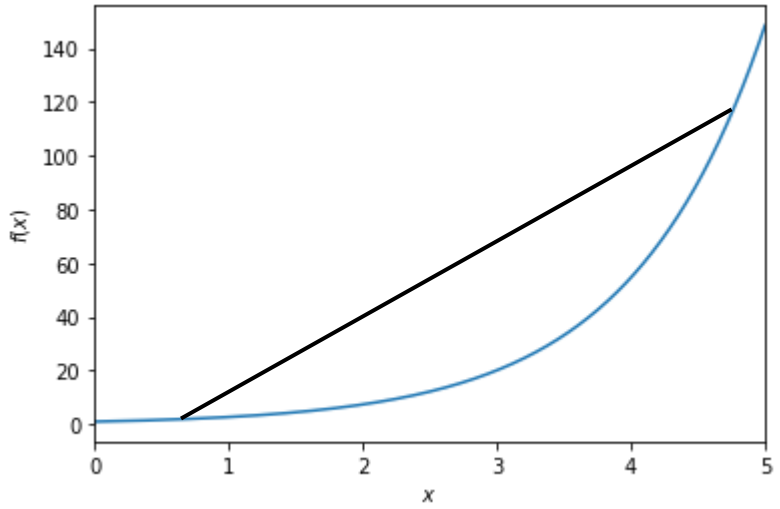
$$f(x) = x^2$$

$$f(x) = e^x$$

$$f(x) = |x|$$

$$f(x) = \sin(x) \text{ for } \frac{\pi}{2} \leq x \leq \frac{3\pi}{2}$$

- a local minimum is a global minimum



Convex Functions in Multiple Dimensions

- definition: a function $f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^n$ is convex if

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2) \text{ for all } \lambda \in [0,1], \text{ all } \boldsymbol{x}_1, \boldsymbol{x}_2$$

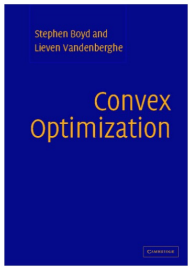
- interpretation: line connecting any two points on the surface of the function is *always* above the function

- equivalently, a function is convex if the Hessian, $\nabla^2 f(\boldsymbol{x})$ is positive semi-definite

- examples:

$$f(\boldsymbol{x}) = ||\boldsymbol{x}||_p \qquad p \geq 1$$

$$f(\boldsymbol{x}) = \max\{x_1, \ldots, x_n\}$$

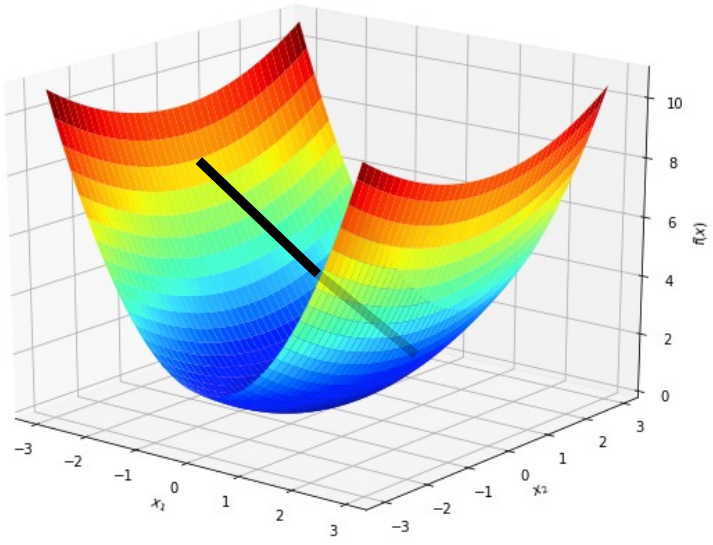


- important because computers can solve basically any optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

if $f(\boldsymbol{x})$ is a convex function over a convex set $\mathcal{X}$

- a local minimum is a global minimum



Optimizing

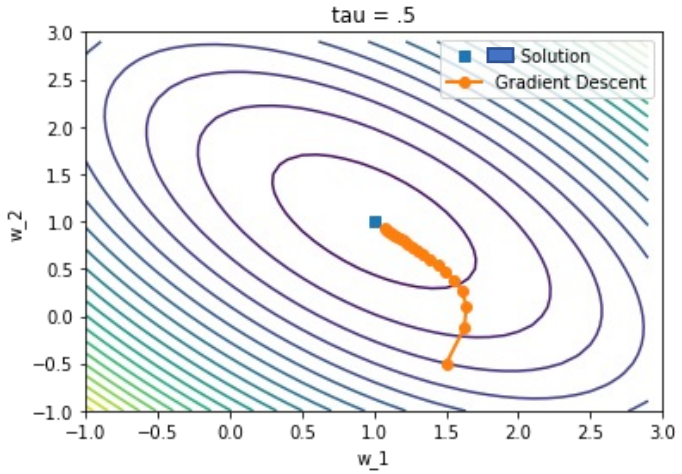
$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

- logistic loss is easy to optimize . . .
  - convex and differentiable
  - we can do gradient descent and are guaranteed to find the minimum

gradient descent

for $k = 1 \ldots$
$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \tau \nabla f(\boldsymbol{w})$

$$\nabla_w f(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$



Finding the gradient

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

$$\nabla_w f(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$

$$\nabla_w f(\boldsymbol{w}) = \sum_i \frac{-y_i}{1 + e^{y_i \boldsymbol{w}^T \boldsymbol{x}_i}} \boldsymbol{x}_i$$

gradient descent

for $k = 1 \ldots$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \tau \nabla f(\boldsymbol{w})$$

Showing Convexity

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i \boldsymbol{x}_i^T \boldsymbol{w}}\right)$$

- option one: show Hessian is positive semi-definite

- option two: use some rules about checking convexity

1) $\frac{d^2}{dz^2} \log(1 + e^{-z}) = \frac{e^z}{(e^z+1)^2}$

2) composition with a linear transform preserves convexity

3) non-negative sum of convex functions is convex

[Boyd Vandenberghe, p.79]

See link BV on Canvas to download the book