The linear model for regression and classification, Gradient Descent, Properties of the Gradient

Alexander Binder

University of Oslo (UiO)

January 25, 2021

UiO : Department of Informatics
University of Oslo

Learning goals

- the four components for structuring many types of machine learning problems
- loss functions: ℓ_2-loss, zero$-$1-loss, Hinge-loss
- generalization as goal: low loss on unseen data points.
- **?!** relationship between generalization and central limit theorem
- be able to reproduce the explicit solution for linear regression and ridge regression
- **?!** directional derivatives and their relationship to the gradient
- **?!** multilinear algebra: derivatives of multi-linear functions and the product rule

Learning goals

- **?!** be able to explain how the set of constant points $\{x : f(x) = c\}$ looks like for f being the linear model
- gradient descent as a vanilla solution strategy to find parameters in machine learning
- be able to explain the impact of large or small learning rates on the performance of gradient descent
- be able to explain the impact of different initialization points on the performance of gradient descent, and for what problems different initializations do not matter
- **if enough time, otherwise lecture 5** Batch vs Stochastic gradient descent

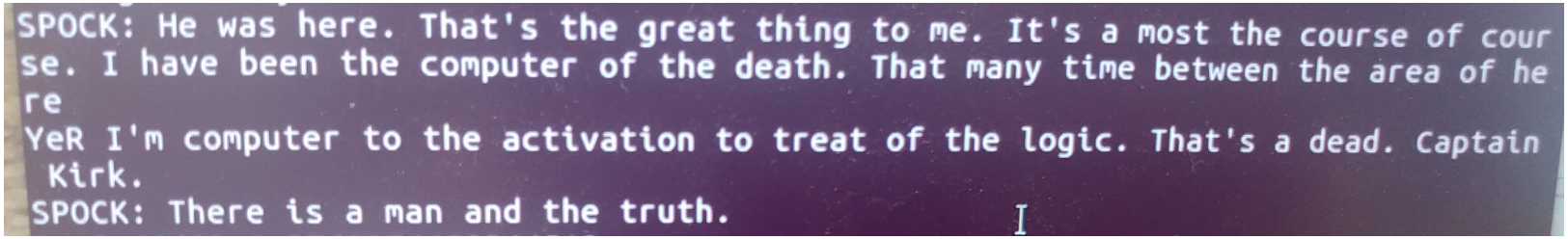
General discriminative ML

Predictor f : Input space $\longrightarrow$ Output space.

- model has trainable parameters w: $f = f_w$
- Use (labeled! ...) training data and a loss function to optimize parameters w
- Prediction is correct with a certain probability of mistake.

Detection in Images: class label and position

- Input: image.
- Output: a number of bounding boxes



- Input: sequence of words $(w_1, \ldots, w_K)$
- Output: sequence of words $(w_1, \ldots, w_L)$
  (Example: "The text is about Star Trek. It mentions Spock and James T. Kirk.")
- Input: sequence of words $(w_1, \ldots, w_K)$
- Output: set of words $\{w_1, \ldots, w_L\}$ – setup can be multi-label classification or sequential outputs from a RNN
  (Example: "Star Trek, RNN-generated nonsense")

D3: Battle D4: Battle 2

- Input: sequence of states (health, ammo, images of view)^K
- Output: next action (move left, fire, ...)
- Dosovitsky et al, ICLR 2017, https://arxiv.org/pdf/1611.01779.pdf

Visual Question answering, Image captioning

credit: visualqa.org

- Input: Image (+ sequence of words for VQA)
- Output: set of words – setup can be multi-label classification or sequential outputs from a RNN

What does one need for defining a discriminative machine learning problem?

four basic components of a machine learning problem

- I/O: Model for Input space, Model for output space
- Model: define a class of prediction models (deep neural network??)
- Loss: define a loss function to measure difference: prediction of the model versus ground truth
- Optimizer: define an algorithm for updating model parameters using (labelled) training data

What input and output space can be used for the examples above?

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

A. http://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set

- $X1$ = the transaction date
- $X2$ = the house age (unit: year)
- $X3$ = the distance to the nearest MRT station (unit: meter)
- $X4$ = the number of convenience stores in the living circle on foot (integer)
- $X5$ = the geographic coordinate, latitude. (unit: degree)
- $X6$ = the geographic coordinate, longitude. (unit: degree)
- The output is as follows Y = house price per unit of area

Input and output space in regression problems?

http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime
http://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test
https://en.wikipedia.org/wiki/Concrete_slump_test
http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

- commonly: $\mathcal{X} = \mathbb{R}^d$
- single regression target: $\mathcal{Y} = \mathbb{R}^1$
- multiple regression targets: $\mathcal{Y} = \mathbb{R}^k$

Linear function without/with a bias

$$f_w(x) = x \cdot w \qquad = \sum_{d=1}^{D} x_d w_d \qquad , \; w \in \mathbb{R}^{D \times 1}$$

$$f_{w,b}(x) = x \cdot w + b \quad = \sum_{d=1}^{D} x_d w_d + b \quad , \; w \in \mathbb{R}^{D \times 1}, b \in \mathbb{R}^1$$

weighted sum of features x_d: $X5 =$ (Cumulated wind speed) $\leftrightarrow x_5$

Loss function for regression

Loss function to measure quality of prediction for a data sample, here a pair (x, y):

$$\ell(f(x), y) = (f(x) - y)^2$$

Reasons:

- $\ell(f(x), y) = 0 \Leftrightarrow f(x) = y$
- capture deviations on both sides of the real ground truth value y
- simple derivative

the goal in regression (and machine learning in general)

Find parameters w, b which generalize well to new, unseen data points. Here: an informal explanation.

We need a way to model *new, unseen data points*

Assumption 1: We can draw test data sets T_n from your data source. A test data set consists of n pairs (x_i, y_i):

$$T_n = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} = \{(x_i, y_i), i = 1, \ldots, n\}$$

x_i is the input feature, y_i is the ground truth label to it (here: the regression value which x_i is expected to have.)

Assumption 2: We cannot predict which samples (x_i, y_i) we will obtain from your data source as new, unseen data points. Therefore, we model the uncertainty by drawing from a probability for the (x_i, y_i).

$$(x_i, y_i) \sim P_{test}.$$

An Example for such a generating probability

for input samples $x \in [0,1] \times [0,1] \subset \mathbb{R}^2$:

$$
\begin{aligned}
p(x) &= \mathit{Unif}([0,1] \times [0,1]) \\
p(y|x) &= \mathit{Normal}(\mu(x), \sigma^2 = 0.1) \\
\mu(x) &= 2x_1 - 3x_2 \\
P_{test}(x,y) &= p(x)p(y|x)
\end{aligned}
$$

Drawing in practice:

- draw $x \sim p(x)$
- draw $y \sim p(y|x)$
- this implies: have $(x,y) \sim p(x)p(y|x) = P_{test}(x,y)$

By drawing n times in this way, one can obtain a training dataset $D_n = \{(x_i, y_i), i = 1, \ldots, n\}$ or a test dataset T_n of independent samples.

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

Generalization (qualitative view)

A mapping generalizes well, if it gives low prediction errors on unseen data samples.

To generalize well in qualitative form:

- we find parameters (w^*, b^*) defining a mapping f which for most draws of test datasets T_n produces predictions with low average errors on test sets T_n:

$$\hat{L}(f, T_n) = \frac{1}{n} \sum_{(x_i, y_i) \in T_n} \ell(f(x_i), y_i) \to \text{low}$$

Generalization for regression with ℓ_2-loss (short form)

Plug in the above:

$$\hat{L}(f, T_n) = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \to \text{low}$$

on new unseen test data for most draws of test data sets.

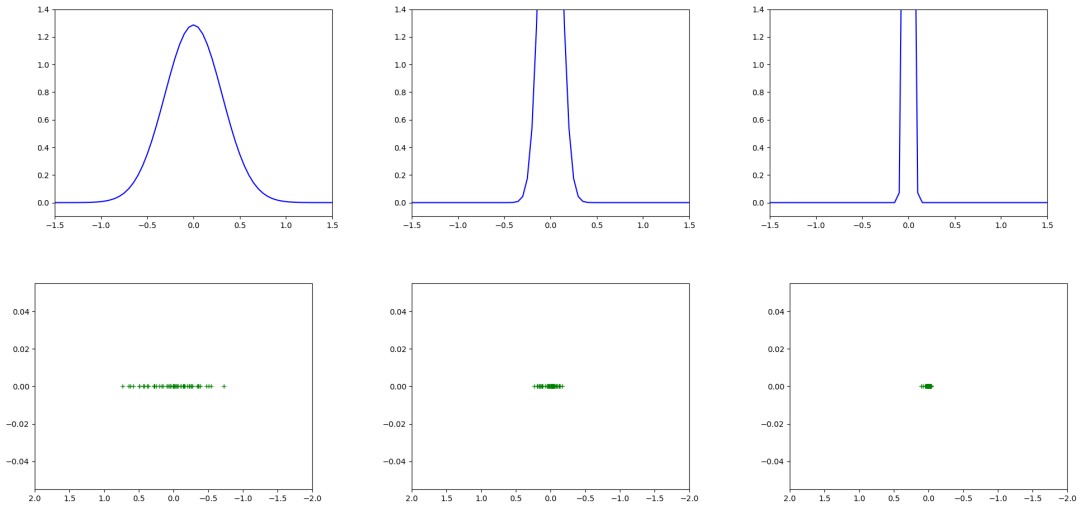
We do not specify here what "low" means. From a practitioners perspective it would be predictions, with errors such that the effort to correct them is low enough, so that using this predictor is productive.

Why do we have a chance to succeed in achieving low errors for most test sets?

Recap on central limit theorem

If you have a population with mean μ and standard deviation σ and you draw n random samples from the population - statistically independently, then the distribution of the sample means will be **approximately** normally distributed **with mean** μ **and standard deviation** $\frac{\sigma}{\sqrt{n}}$

We draw 100 times a testset with $n = 900$ samples. We compute 100 times an average loss.



- a number of the 100 testset losses will be close to μ, as they are approximately on a distribution with variance $\frac{\sigma}{\sqrt{n}}$ around the mean.
- have a probability that a loss estimate on training data is similar to the estimate on test data
- number of "good" testsets ($\hat{\mu}$ close to μ) with sample size n increases as $n \to \infty$

What is the probability to be outside $[\mu - k\sigma_n, \mu + k\sigma_n]$ with your training set or test set average loss ?

https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule

if n large enough and i.i.d drawn data. Note $\sigma_n = \frac{\sigma^2}{\sqrt{n}}$

Why do we have a chance to succeed in achieving average losses being close to the expectation μ under $P_{train/test}$ for most test sets?

consequence CLT

$$E_{(x,y)\sim P_{test}}[L(f(X), Y)] \approx \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \ell(f_w(x_i), y_i)$$

with a certain probability, provided that

1. the samples are drawn from P_{test}
2. the samples are drawn independently

in the sense that $\frac{1}{n}\sum_{(x_i,y_i)\in D_n} \ell(f_w(x_i), y_i)$ will be one sample point as the green points on the last slide, drawn from an approximately normal with variance $\frac{\sigma}{\sqrt{n}}$

Notes on generalization

Search for model parameters is done on a training dataset D_n, and involves minimizing a training loss $\hat{L}(f, D_n)$ computed on the training dataset.

$$(w^*, b^*) = \mathrm{argmin}_{(w,b)} \hat{L}(f, D_n) = \mathrm{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \ell(f(x_i), y_i)$$

$$= \mathrm{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} (f(x_i) - y_i)^2$$

- Question: Why the goal is not: to have low loss on one test dataset $T_{50} = \{(x_i, y_i), i = 1, \ldots, 50\}$?

That is important to understand, because often errors are measured only on a single test dataset ...

A first method to solve linear regression

Assumption here: no bias. Parameters are w.

Goal: to find parameters which minimize the loss on this dataset:

$$(w^*) = \mathrm{argmin}_w \sum_{i=1}^{n} L(f(x_i), y_i) = \mathrm{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2$$

for a given dataset $D_n = \{(x_i, y_i)\}$. Then $f_{w^*}(x) = x \cdot w^*$ is the selected mapping. Rare case: Can be solved explicitly for w.

- consider the loss L as function of w
- compute $\nabla L(w)$ - the gradient of the loss with respect to w
- solve $\nabla L(w) = 0$ for w.
- Solution can be maximum, minimum or saddle point. Verify that the Hessian in the solution point is positive definite. That is: the function has positive curvature in every direction. implies: must be a minimum.

Write in matrix form:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \dots, x_1^{(D)} \\ x_2^{(1)}, \dots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \dots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

then:

$$L(w) = \sum_{i=1}^{n} (x_i \cdot w - y_i)^2 = (X \cdot w - Y)^T \cdot (X \cdot w - Y)$$

Solve the minimization problem by computing the gradient for w and setting it to zero.

$$\begin{aligned} D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) &= 2X^T \cdot (X \cdot w - Y) \\ 2X^T \cdot (X \cdot w - Y) &= 0 \\ \Rightarrow (X^T \cdot X) \cdot w &= X^T \cdot Y \\ w &= (X^T \cdot X)^{-1} X^T \cdot Y \end{aligned}$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

A first method to solve linear regression

explicit solution to linear regression without bias

Let X be the training data matrix.

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1^{(1)}, \dots, x_1^{(D)} \\ x_2^{(1)}, \dots, x_2^{(D)} \\ \vdots \\ x_n^{(1)}, \dots, x_n^{(D)} \end{pmatrix} \in \mathbb{R}^{n \times D}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^{n \times 1}$$

The model is $f(x) = w \cdot x$. Then a solution is given as

$$w^* = (X^T \cdot X)^{-1} X^T \cdot Y$$

if the matrix inverse $(X^T \cdot X)^{-1}$ exists.

Prediction is done using: $f_{w^*}(x) = w^* \cdot x$

What kind of extremum is the solution? Compute the Hessian

$$D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) = 2X^T \cdot (X \cdot w - Y)$$
$$D_w D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y)) = 2X^T \cdot X$$

a matrix $A^T A$ is always non-negative definite. Thus the solution is either a minimum or a saddle point, but not maximum.

- How to extend this solution to the case with a bias?

$$x = \left(x^{(1)}, \ldots, x^{(D)}\right) \rightarrow \hat{x} = \left(x^{(1)}, \ldots, x^{(D)}, 1\right)$$

Then the parameter w also gets an additional dimension

$$\hat{w} = \left(w^{(1)}, \ldots, w^{(D)}, w^{(D+1)}\right)$$

then:

$$\hat{x} \cdot \hat{w} = w \cdot x + w^{(D+1)} = w \cdot x + b$$

$w^{(D+1)}$ acts as bias.

From Linear to Ridge regression

From Linear to Ridge regression

Overfitting: low training error, high test error. Reason: during learning one picks up too much of the noise in the training data, and learns weights that listen to noise signals.

One way to deal with it: avoiding weights w getting too large: add a penalty on the euclidean length of w

$$\mathrm{argmin}_w \sum_{i=1}^{n} (x_i \cdot w - y_i)^2 + \lambda \|w\|^2$$

$$\sum_{i=1}^{n}(x_i \cdot w - y_i)^2 + \lambda \|w\|^2$$
$$= (X \cdot w - Y)^T \cdot (X \cdot w - Y) + \lambda w^T \cdot w$$
$$D_w((X \cdot w - Y)^T \cdot (X \cdot w - Y) + \lambda w^T \cdot w)$$
$$= 2X^T \cdot (X \cdot w - Y) + 2\lambda w$$

$$2X^T \cdot (X \cdot w - Y) + 2\lambda w = 0$$
$$X^T \cdot X \cdot w + \lambda I \cdot w = X^T \cdot Y$$
$$(X^T \cdot X + \lambda I) \cdot w = X^T \cdot Y$$
$$\Rightarrow w = (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y$$

Ridge regression

Ridge regression is Linear regression with an added squared-ℓ_2 penalty term on weights

$$\lambda \|w\|^2$$

λ is a hyperparameter in this approach. The solution changes to

$$w = (X^T \cdot X + \lambda I)^{-1} X^T \cdot Y$$

In practice, one needs to find a good value for the hyperparameter λ on a validation set, before measuring the performance on the test set. The effect of this regularization will be discussed later.

Advantages of ridge regression over least squares:

- for any $\lambda > 0$ a solution always exists: $(X^T \cdot X + \lambda I)$ is always invertible because it is positive definite
- the Hessian of L is positive definite, thus one finds always a minimum and not a saddle point (or maximum)
- when in least squares a solution does not exist?

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

Gradients

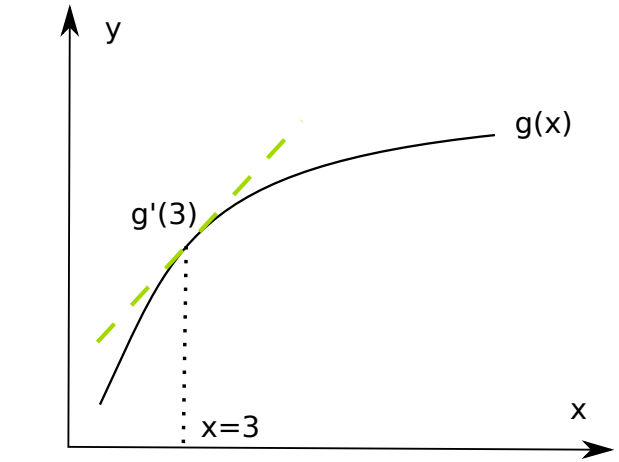
- $g : \mathbb{R}^1 \to \mathbb{R}^1$ is differentiable in input x if the limit exists:

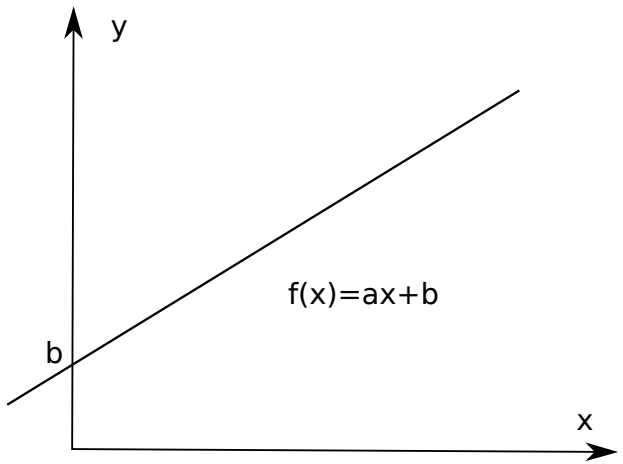
$$\lim_{\epsilon \to 0} \frac{g(x+\epsilon) - g(x)}{\epsilon} \; (=: g'(x))$$

- example: $f(x) = ax + b$ (affine with slope a),then

$$\begin{aligned} & \frac{f(x+\epsilon) - f(x)}{\epsilon} \\ = & \frac{a(x+\epsilon) + b - (ax+b)}{\epsilon} \\ = & \frac{a\epsilon}{\epsilon} = a \\ \Rightarrow & \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon} = a \end{aligned}$$

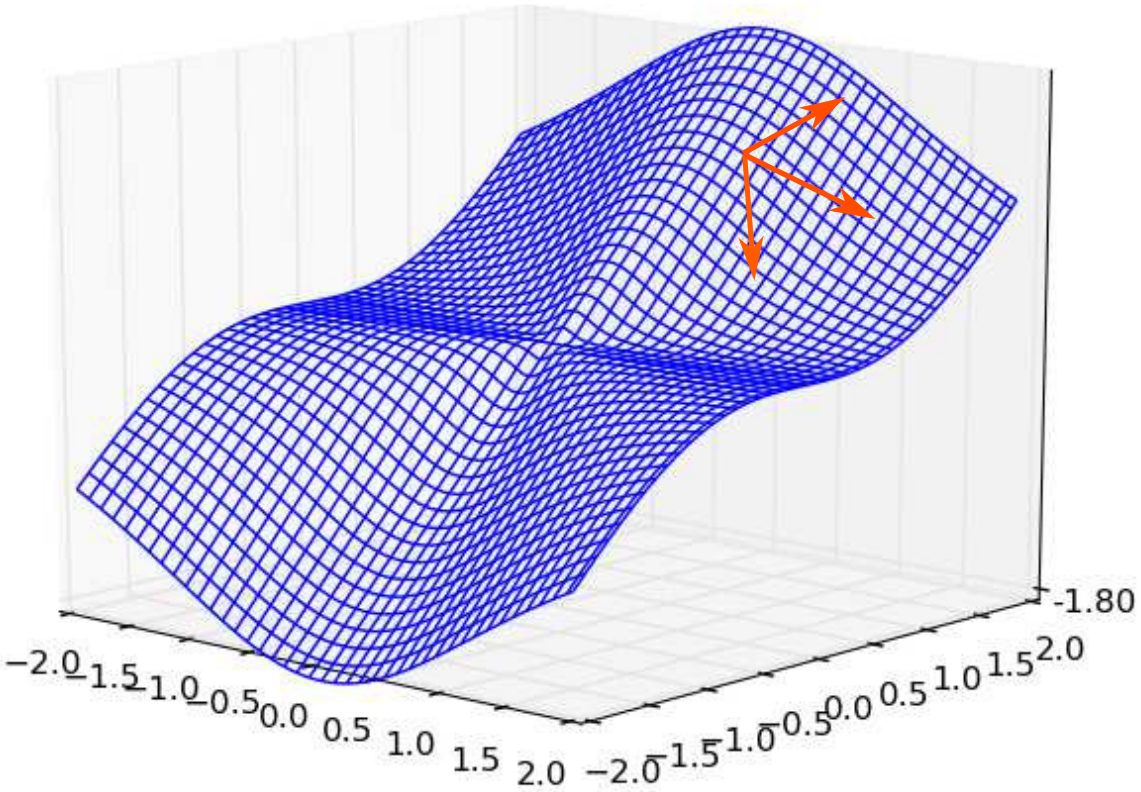
- intuition: slope of the function g at point x





2-dimensions: directional derivatives

Function of 2 input variables: $f(x_1, x_2) \in \mathbb{R}^1$

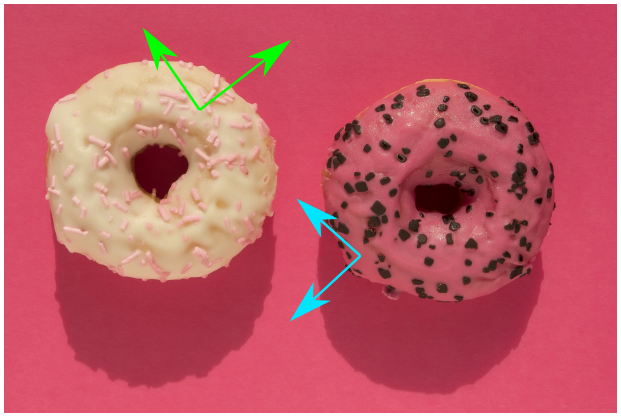


in every point (x_1, x_2): a two dimensional vector space of directions to move from it

in every direction there is a slope – the directional derivative

Function of n input variables: $f(x_1, x_2, \ldots, x_n) \in \mathbb{R}^1$

Example:



The surface of a donut is two dimensional at every point. At every point there is a two-dimensional space of directions to move, each with a slope.

Now take the product space of two donut surfaces. It consists of all pairs (p_1, p_2) such that $p_1 \in$ white donut surface, $p_2 \in$ red donut surface. At every pair (p_1, p_2) - the set of all directions is $n = 4$-dimensional!

in every point $(x_1, x_2, \ldots, x_n)$: a n-dimensional vector space of directions to move from it. In every direction there is a slope – the directional derivative – provides information about function value change in this direction

n-dimensions: directional derivatives and gradient

The directional derivative of function f in point x in direction v is defined as:

$$\delta_{\mathbf{v}} f(\mathbf{x}) = \lim_{\epsilon \to 0} \frac{f(\mathbf{x} + \epsilon \mathbf{v}) - f(\mathbf{x})}{\epsilon}$$

Fact: If the function is differentiable in x, then the directional derivative in $\mathbf{x}$ in direction $\mathbf{v}$ is the inner product of the gradient of x in v:

$$\delta_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v}$$

- directional derivatives tell you how the function grows from x in direction v when you take an infinitely small step
- the gradient contains information about all directional derivatives, if differentiable in x

- next step: define gradient via partial derivatives

$$\begin{aligned}\nabla f(\mathbf{x}) \cdot e_i &= \lim_{\epsilon \to 0} \frac{f(\mathbf{x} + \epsilon e_i) - f(\mathbf{x})}{\epsilon} \\ &= \lim_{\epsilon \to 0} \frac{f(x_1, \ldots, x_i + \epsilon, \ldots, x_D) - f(x_1, \ldots, x_i, \ldots, x_D)}{\epsilon} \\ &= \frac{\partial f}{\partial x_i}(\mathbf{x})\end{aligned}$$

therefore:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_D}(\mathbf{x}) \end{pmatrix}$$

therefore:

$$\delta_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v} = \sum_d \frac{\partial f}{\partial x_d}(\mathbf{x}) v_d$$

Further consequence from:

$$\delta_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v}$$

Which v maximizes $\nabla f(\mathbf{x}) \cdot \mathbf{v}$?

the optimization problem

$$\mathrm{argmax}_{v:\|v\|=1} w \cdot v$$

is solved by $v := \frac{w}{\|w\|}$.

Therefore: The gradient is the direction where the function increases maximally when taking an infinitesimally small step.

Analogously: the negative gradient is the direction where the function **decreases maximally when taking an infinitesimally small step**.

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

examples:

$$X \in \mathbb{R}^{d\times k},\ D_w(X \cdot w) = X \text{ ok, easy}$$

$$D_w((Xw)^T \cdot (Xw)) = X^T Xw + w^T X^T X \text{ ?? shape mismatch !}$$

$$= (k, 1) + (1, k) = ???$$

$$A \in \mathbb{R}^{k\times d}, W \in \mathbb{R}^{d\times d}, C \in \mathbb{R}^{d\times m},$$

$$D_w(AWC) = A\,[\text{a hole here?}]\,C = \text{ nonsense ???}$$

Why do you need that

Consider viewpoint directional derivatives:

$$Df(x)[h] = \nabla f(x) \cdot h$$

The derivative of f in x is a linear mapping of directions onto directional derivatives.
Rule 1: the derivative of a linear function is the linear function itself.

$$f(x) \text{ linear} \Rightarrow Df(x)[h] = f(h)$$
$$f(x) = Ax \Rightarrow Df(x)[h] = Ah$$

This holds also if f maps into vectors or matrices.

Why do you need that

Rule 1: the derivative of a linear function is the linear function itself.

$$f(x) \text{ linear} \Rightarrow Df(x)[h] = f(h)$$

This holds also if f maps into vectors or matrices. Why?

$$\begin{aligned}
f(x) &= Ah \\
\Rightarrow f_j(x) &= (Ax)_j = \sum_k A_{jk} x_k \\
\frac{\partial f_j}{\partial x_m} &= A_{jm} \\
\nabla f_j(x) &= (A_{j1}, A_{j2}, \ldots, A_{jm}, \ldots A_{jd}) \\
Df_j(x)[h] &= \nabla f_j(x) \cdot h = \sum_k A_{jk} h_k = (Ah)_j \\
\Rightarrow Df(x)[h] &= Ah
\end{aligned}$$

Why do you need that

$$D_w(AWC)[H] = ???$$

Why do you need that

$$f(W) = AWC \text{ is linear in } W \text{ so:}$$
$$D_w(AWC)[H] = f(H) = AHC$$

its a no brainer!

You need a partial derivative?

$$W \in \mathbb{R}^{d \times d}, \Rightarrow \frac{\partial f}{\partial W_{ij}}(W) = A\mathbb{1}_{ij}C$$

$\mathbb{1}_{ij}$ is the matrix which is 1 in entry (i,j) and zero everywhere else

Rule 2: product rule, applicable for any vector/matrix-vector/matrix multiplication

$$
\begin{aligned}
D_x(f(x) \cdot g(x))[h] = (Df(x)[h]) \cdot g(x) + f(x) \cdot (Dg(x)[h]) \\
= & Df(x)[h] \cdot g(x) \\
+ & f(x) \cdot Dg(x)[h]
\end{aligned}
$$

This extends to multiplications of more than two terms:

$$
\begin{aligned}
D_x(f(x) \cdot g(x) \cdot r(x))[h] = \\
&= Df(x)[h] && \cdot g(x) && \cdot r(x) \\
&+ f(x) && \cdot Dg(x)[h] && \cdot r(x) \\
&+ f(x) && \cdot g(x) && \cdot Dr(x)[h]
\end{aligned}
$$

Why do you need that

$$D_w(w^T A w)[h] = ?$$

Why do you need that

$$
\begin{aligned}
D_w(w^T A w)[h] &= D_w(w^T \cdot Aw)[h] \\
&= D_w(w^T)[h] \cdot Aw + w^T \cdot D(Aw)[h] \\
&\text{both are linear mappings!} \\
D_w(w^T)[h] &= h^T \\
D(Aw)[h] &= Ah \\
\Rightarrow D_w(w^T A w)[h] &= h^T A w + w^T A h \\
D_w(w^T A w)[h] &= D_w(w^T A \cdot w)[h] \text{ same solution}
\end{aligned}
$$

Why do you need that

$$D_w((Xw - Y)^T(Xw - Y)) = ?$$

$$\begin{aligned} D_w((Xw - Y)^T \cdot (Xw - Y))[h] &= D_w((Xw - Y)^T)[h] \cdot (Xw - Y) \\ &\quad + (Xw - Y)^T \cdot D_w(Xw - Y)[h] \end{aligned}$$

$$\begin{aligned} D_w((Xw - Y)^T)[h] &= D_w((Xw)^T)[h] + D_w((-b)^T)[h] \\ &= (Xh)^T + 0 \\ \Rightarrow D_w((Xw - b)^T \cdot (Xw - Y))[h] &= (Xh)^T \cdot (Xw - Y) \\ &\quad + (Xw - Y)^T \cdot (Xh) \end{aligned}$$

Note both terms are real numbers:
$X \sim (n, d), w \sim (d, 1), Y \sim (n \times 1)$
$\Rightarrow Xw - Y \sim (n \times 1), (Xw - Y)^T \cdot (Xw - Y) \sim (1, n) \cdot (n, 1) = (1, 1)$

$$
\begin{aligned}
D_w((Xw - Y)^T(Xw - Y)) &= ? \\
D_w((Xw - b)^T \cdot (Xw - Y))[h] &= (Xh)^T \cdot (Xw - Y) \\
&\quad + (Xw - Y)^T \cdot (Xh)
\end{aligned}
$$

Note both terms are real numbers: $X \sim (n, d), w \sim (d, 1), Y \sim (n \times 1)$ therefore they are their own transpose!

$$
\begin{aligned}
D_w((Xw - b)^T \cdot (Xw - Y))[h] &= (Xh)^T \cdot (Xw - Y) \\
&\quad + ((Xw - Y)^T \cdot (Xh))^T \\
&= (Xh)^T \cdot (Xw - Y) \\
&\quad + (Xh)^T \cdot (Xw - Y) \\
= 2(Xh)^T \cdot (Xw - Y) &= h^T \cdot 2X^T(Xw - Y)
\end{aligned}
$$

Why do you need that

$$
\begin{aligned}
D_w((Xw-b)^T \cdot (Xw-Y))[h] &= (Xh)^T \cdot (Xw-Y) \\
&\quad + ((Xw-Y)^T \cdot (Xh))^T \\
= 2(Xh)^T \cdot (Xw-Y) &= h^T \cdot 2X^T(Xw-Y)
\end{aligned}
$$

This is a linear operation in h: $h \mapsto h^T \cdot 2X^T(Xw-Y)$,

Therefore the gradient is the transpose of $2X^T(Xw-Y)$

The Linear model for classification

Input and output space in classification problems?

2 classes: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, 1\}$

C classes: $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{0, \ldots, C-1\}$

Consider the case of two classes. Linear function without/with a bias. Thresholded by a sign

$$f_w(x) = x \cdot w \qquad = \sum_{d=1}^{D} x_d w_d,\ w \in \mathbb{R}^{D \times 1}$$

$$f_{w,b}(x) = x \cdot w + b \qquad = \sum_{d=1}^{D} x_d w_d + b,\ w \in \mathbb{R}^{D \times 1}, b \in \mathbb{R}^1$$

$$h(x) = \mathrm{sgn}(f_{w,b}(x)) \qquad \in \{-1, +1\}$$

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

What does a linear mapping represent?

Goal: understand what the mapping $f(\cdot)$ does.
Approach: characterizing the set of points x with a constant output $f(x) = c$.

Thinking task

What is the set of points $x = (x_1, x_2) \in \mathbb{R}^2$ such that

$$3x_1 - 2x_2 + 3 = 0 \ ?$$

What is the set of points $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ such that

$$x_1 - x_2 - 2x_3 + 2 = 0 \ ?$$

A. What is the set of points x: $f_{w,b}(x) = 0$?
B. What is the set of points x the prediction is a constant c, that is $f_{w,b}(x) = c$?

What is the set of points x: $f_{w,b}(x) = 0$?

$$f_{w,b}(x) = 0 \Leftrightarrow x \cdot w = -b$$

To understand how the bias b influences the zero set, lets consider three cases:

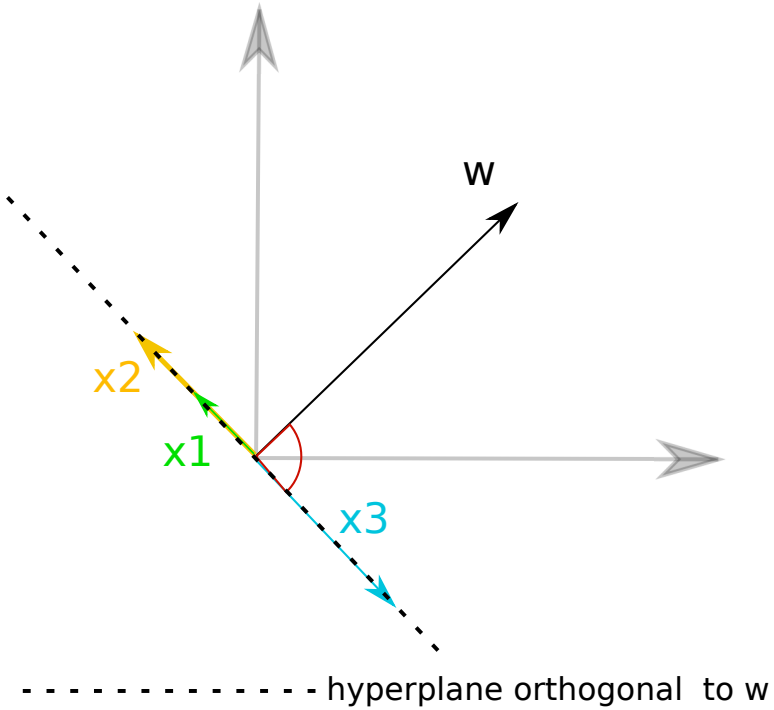
- $b = 0$
- $b > 0$
- $b < 0$

The set of points x: $f_{w,b}(x) = 0$ – The case $b = 0$

- We know that for $b = 0$: $f_{w,0}(x) = w \cdot x = 0$ holds for the zero vector $x = 0$.
- The set of points x such that the inner product

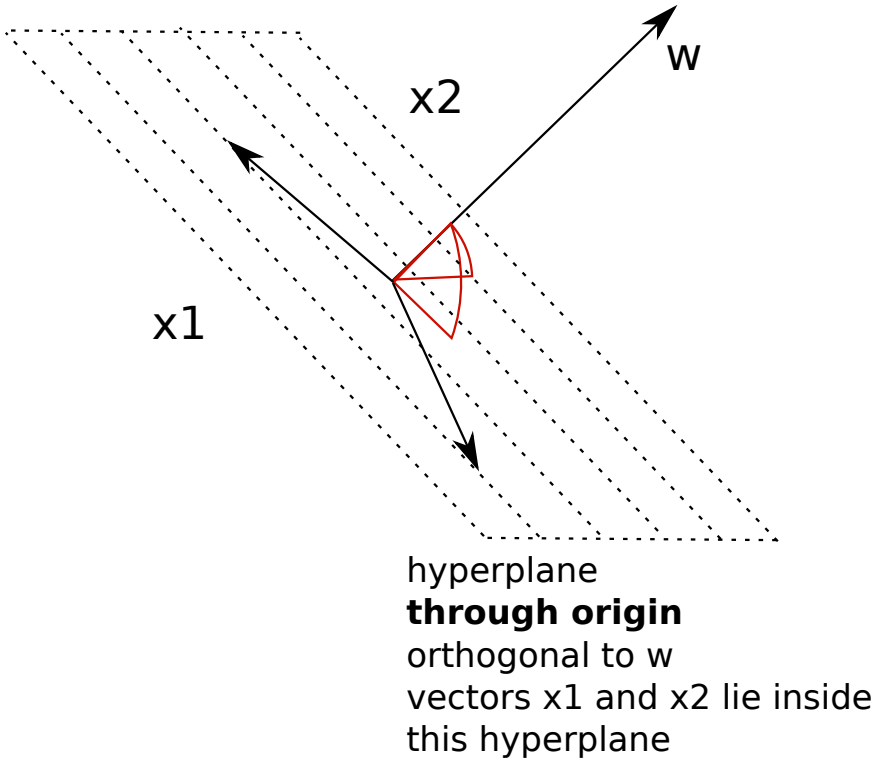
$$x \cdot w = 0$$

is in 2 dims a one-dimensional line, which goes through the origin $(x_1, x_2) = (0, 0)$, and which is orthogonal to w.



x_1,x_2,x_3 are all orthogonal to the vector w

The analogy also holds for 3 or more dimensions. So for 3 dims it is a two-dimensional plane, which goes through the origin $(x_1, x_2, x_3) = (0, 0, 0)$.



For n dimensions the plane of orthogonal vectors has $n - 1$ dimensions (+ goes through the origin), but is still a hyperplane

Recap hyperplane of dimension $n-1$

- P is a hyperplane (linear space) if it holds: $x_1 \in P, x_2 \in P \Rightarrow a_1 x_1 + a_2 x_2 \in P$ (space closed under linear operations)
- can find $n-1$ basis vectors such that each point of P can be represented as a linear combination of the basis vectors)

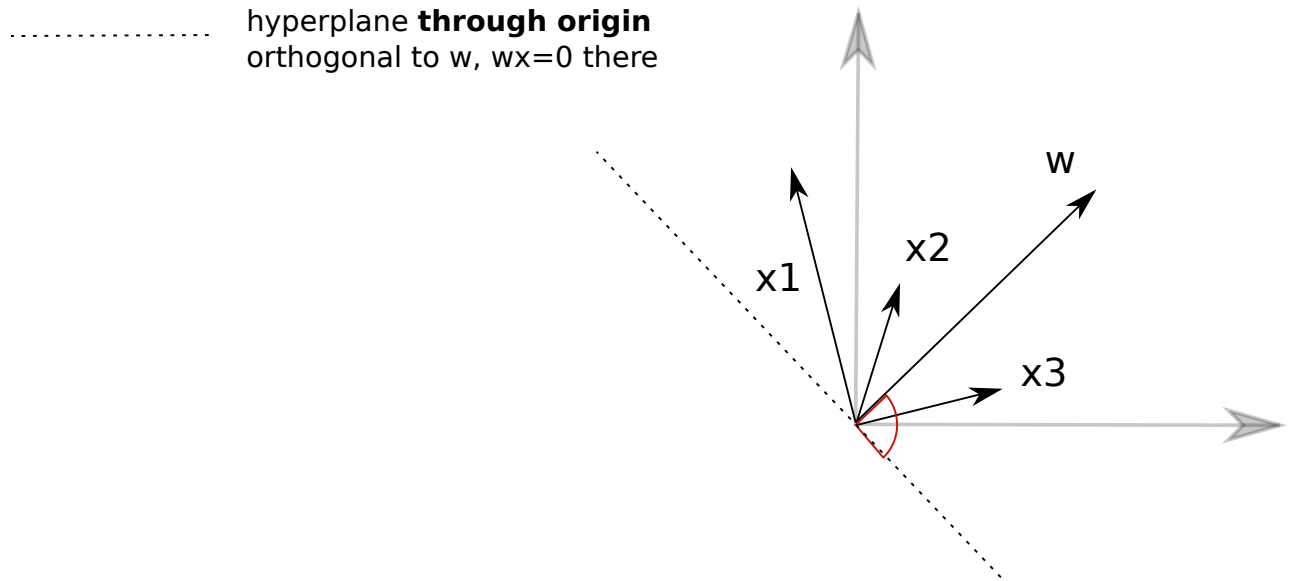
$$\exists\, v_1, \ldots, v_{n-1} \text{ such that}$$

$$\forall x \in P\ \exists\, a_1, \ldots, a_{n-1} \text{ such that } x = \sum_{i=1}^{n-1} a_i v_i = \boldsymbol{a} \cdot \boldsymbol{V}$$

The set of points x: $f_{w,b}(x) = 0$ – The case $b < 0$

$$b < 0, \;\; w \cdot x + b = 0 \Rightarrow w \cdot x = -b > 0$$

We know: $w \cdot x > 0$ for all points x that are on that side of the **hyperplane through the origin**, in which w points to.



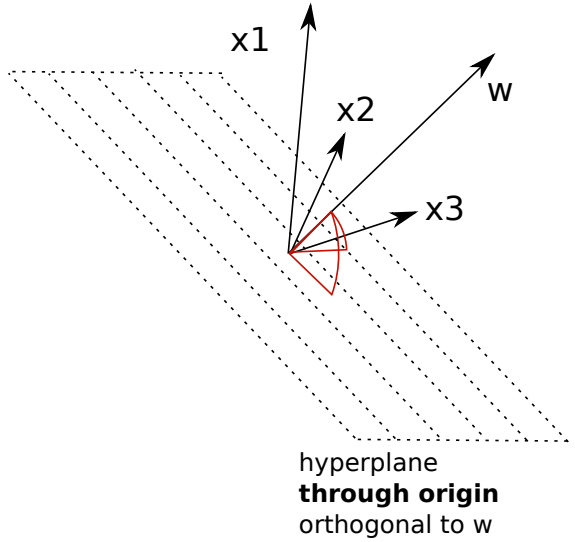
In the above figure $\mathbf{x_1}$,$\mathbf{x_2}$,$\mathbf{x_3}$ all have $w \cdot x_i > 0$ because relative to the hyperplane orthogonal to w which goes through the origin, they all point into the direction of w

The set of points x: $f_{w,b}(x) = 0$ – The case $b < 0$

$$b < 0, \;\; w \cdot x + b = 0 \Rightarrow w \cdot x = -b > 0$$

We know: $w \cdot x > 0$ for all points x that are on that side of the **hyperplane through the origin**, in which w points to.

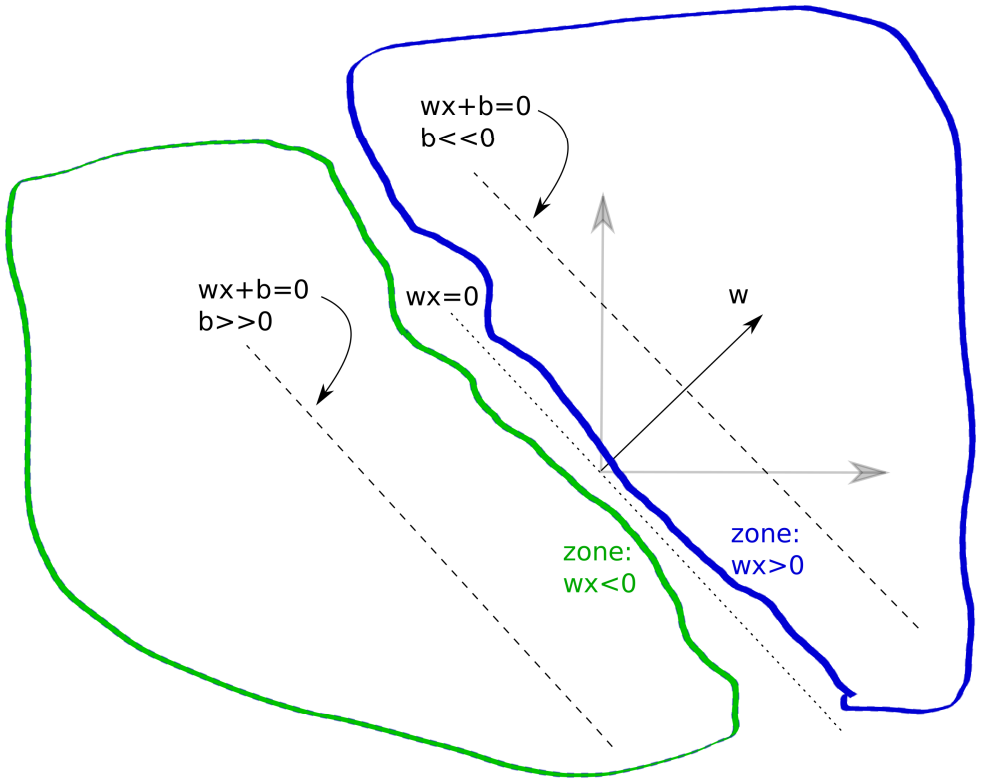
The same also holds for 3 or more dimensions. All the vectors below solve $wx + b = 0$ for some bias $b < 0$!



In above figure x_1,x_2,x_3 all have $w \cdot x_i > 0$

The set of points x: $f_{w,b}(x) = 0$ – any bias

The bias b shifts the zero set corresponding to $w \cdot x + b = 0$ parallel/anti-parallel to the direction of w.



the hyperplane is parallel to the hyperplane orthogonal to w which goes through the origin

The set of points x: $f_{w,b}(x) = 0$ – The case $b < 0$

The set of points x such that

$$\{x : \ wx + b = 0\}$$

is a hyperplane which is parallel to the hyperplane $\{x : x \cdot w = 0\}$ orthogonal to w going through the origin, and which is shifted towards the direction of w

The set of points x: $f_{w,b}(x) = 0$ – The case $b > 0$

The set of points x such that

$$\{x : \ wx + b = 0\}$$

is a hyperplane which is parallel to the hyperplane $\{x : x \cdot w = 0\}$ orthogonal to w going through the origin, and which is shifted opposite to the direction of w

The set of points x: $f_{w,b}(x) = 0$ – any bias

hyperplane dependency on bias b

- $w \cdot x = 0$ is a hyperplane orthogonal to w.
- Negative $b < 0$ shift the hyperplane $\{x : wx + b = 0\}$ into the direction of w,
- positive $b > 0$ shift the hyperplane $\{x : wx + b = 0\}$ against the direction of w.
- Large values of $|b|$ shift it far away.

The set of points x: $f_{w,b}(x) = 0$ – any bias

hyperplane explicit

The linear mapping $f(x) = w \cdot x + b$ has a zero set which is the plane of points

$$\{x : x = u + -b\frac{w}{\|w\|^2}, u \text{ such that } w \cdot u = 0\}$$

In this representation:

- u such that $w \cdot u = 0$ is the hyperplane of vectors u orthogonal to w.
- the vector $-b\frac{w}{\|w\|^2}$ shifts the hyperplane antiparallel to direction of w.

That holds because

$$w \cdot x + b = w \cdot \left(u + -b\frac{w}{\|w\|^2}\right) + b = 0$$

How to find a vector u such that $w \cdot u = 0$?

By subtracting its component parallel to w!

Be x any vector. Subtract its component parallel to w.

Important: use length-normalized w: $\frac{w}{\|w\|}$

$\left(x \cdot \frac{w}{\|w\|}\right)$ is the component of x in direction of $\frac{w}{\|w\|}$

$$u = x - \left(x \cdot \frac{w}{\|w\|}\right) \frac{w}{\|w\|} = x - (x \cdot w) \frac{1}{\|w\|^2} w$$

$$\Rightarrow u \cdot w = 0$$

cf. Gram-Schmid Orthonormalization to find an orthonormal basis of such vectors.

What is the set of points x where the prediction is a constant, that is $g_{w,b}(x) = c$?

Answered by reducing it to a zero set:

$$\begin{aligned} g_{w,b}(x) = wx + b &= c \\ wx + (b - c) &= 0 \end{aligned}$$

The set of points x such that $g_{w,b}(x) = c$ is just the set $x : \ g_{w,b-c}(x) = 0$.

thinking task

What is the set of points $x = (x_1, x_2) \in \mathbb{R}^2$ such that

$$3x_1 - 2x_2 + 3 = 0 \ ?$$

What is the set of points $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ such that

$$x_1 - x_2 - 2x_3 + 2 = 0 \ ?$$

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

First loss function for a pair (x, y):

zero-one-loss

$$y \in \{-1, +1\} \Rightarrow \ell(f(x), y) = 1[\mathrm{sgn}(f_{w,b}(x)) \neq y]$$

Problem: sign is unsuitable for gradient optimization.

Note here the possibility to rewrite the condition for $y \in \{-1, +1\}$:

$1[sign(f(x)) \neq y]$ as $1[f(x)y < 0]$ without any sign – that is: we have an error if the prediction function $f(x)$ has opposite sign of the ground truth label y.

$$f(x)y = \begin{cases} f(x) > 0, y > 0 & \text{no error} \\ f(x) < 0, y < 0 & \text{no error} \\ f(x) > 0, y < 0 & \text{error} \\ f(x) < 0, y > 0 & \text{error} \end{cases}$$

hinge-loss

$y \in \{1, +1\}, \ \ell(f(x), y) = \max(0, 1 - f_{w,b}(x)y)$

The hinge-loss is an upper bound on the zero-one-loss (insight: fix $y = +1$ or $y = -1$. plot $1[\mathrm{sgn}(z)y < 0]$ and $\max(0, 1 - zy)$).

Idea: If the hinge loss is low, the zero-one loss must also become low. Therefore minimize an average over the hinge loss over training samples

$$L(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in D} \max(0, 1 - y_i f_{w,b}(x_i))$$

$$\text{solve } (w^*, b^*) = \mathrm{argmin}_{w,b} L(w, b)$$

Then you predict using f_{w^*, b^*}.

from classification with hingeloss to support vector machines (SVM)

now add again a quadratic penalty on the weights when learning parameters over a training set

$$\mathrm{argmin}_{(w,b)} \frac{1}{n} \sum_{(x_i,y_i)\in D_n} \max(0, 1 - g_{w,b}(x_i)y_i) + \lambda \|w\|^2$$

SVM:

- averaged hinge loss $\max(0, 1 - f(x_i)y_i)$
- with a linear/affine model $f(x_i) = w \cdot x_i + b$
- with quadratic penalty $\lambda \|w\|^2$ on the weights.

A different way to arrive at an SVM (remember $\frac{1}{\|w\|}$ is the margin to be maximized)

Generalize well means again the same as for regression:

generalization

Generalize well means in short form:

- we find parameters (w^*, b^*) defining a mapping f which for most draws of test datasets T_n produces predictions with low errors on them:

$$\hat{L}(f, T_n) = \frac{1}{n} \sum_{(x_i, y_i) \in T_n} \ell(f(x_i), y_i) \rightarrow low$$

Take away: generalization

Generalization is the same idea for many different setups: learn parameters on training data, so that the loss on newly drawn test datasets will be low on average – for most draws of such datasets from a data source.

See also that the linear model is suitable for both classification and regression. What makes the difference whether we have a classification or a regression model?

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

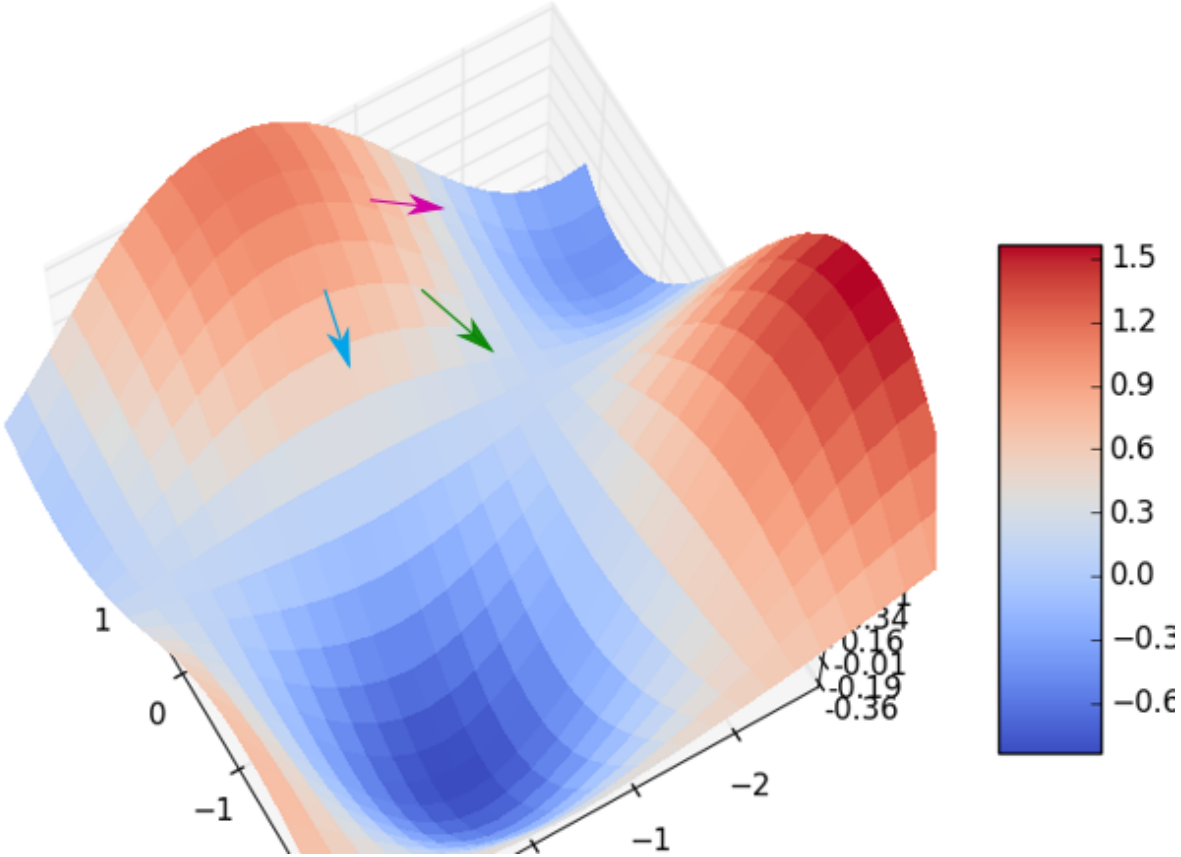
How to find these parameters (w, b) in classification or regression?

Problem setting for application of gradient-based minimization

The Problem Setting, in which gradient methods can be used:

- given some function $g(w)$
- goal: find $w^* = \mathrm{argmin}_w g(w)$
- assumption: can compute $\nabla g(w)$ - the gradient in point w.

Idea: negative gradient at a point w is the direction of locally steepest function decrease from w.



Note: the direction of locally steepest decrease does not point to a global or local minimum.

Gradient Descent

Basic Algorithm: name: **Gradient Descent**:

- given: step size parameter η, initialize start vector w_0 as something
- run while loop, until function value changes very little (δ_g), do at iteration t:
  - $w_{t+1} = w_t - \eta \nabla_w g(w_t)$
  - compute change to last value: $\delta_g = \|g(w_{t+1}) - g(w_t)\|$

Consequences I:

- minimizing the gradient on training data ensures low loss on training data
- does not guarantee low losses on new unseen test data
- a gap between training and test loss is known as overfitting (training loss below test loss).

a method – for linear classification – to select a prediction mapping $f_{w,b}$ / or its parameters from a dataset

Minimize

$$L(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in D} \max(0, 1 - y_i f_{w,b}(x_i))$$

Use $L(w, b)$ with the algorithm box above ... gradient descent with respect to (w, b) for obtaining $(w^*, b^*) = \mathrm{argmin}_{w,b} L(w, b)$

Gradient Descent

Basic Algorithm: name: **Gradient Descent**:

- given: step size parameter η, initialize start vector w_0 as something
- run while loop, until function value changes very little (δ_g), do at iteration t:
  - $w_{t+1} = w_t - \eta \nabla_w g(w_t)$
  - compute change to last value: $\delta_g = \|g(w_{t+1}) - g(w_t)\|$

- the initialization of w_0 <u>matters a lot</u> in deep neural networks! see later exercise. E.g. https://arxiv.org/abs/1502.01852 "Kaiming-He"-initialization ... related to vanishing gradients problem, symmetry breaking

Gradient Descent

Basic Algorithm: name: **Gradient Descent**:

- given: step size parameter η, initialize start vector w_0 as something
- run while loop, until function value changes very little (δ_g), do at iteration t:
  - $w_{t+1} = w_t - \eta \nabla_w g(w_t)$
  - compute change to last value: $\delta_g = \|g(w_{t+1}) - g(w_t)\|$

When does it converge ? When the change δ_g is small, that is when $\eta \|\nabla_w g(w_t)\|$ becomes small. That means $\nabla_w g(w_t) \approx 0$, so a local optimum. Since we always went down, it must be local minimum, or a saddle point.

possible problems of gradient descent:

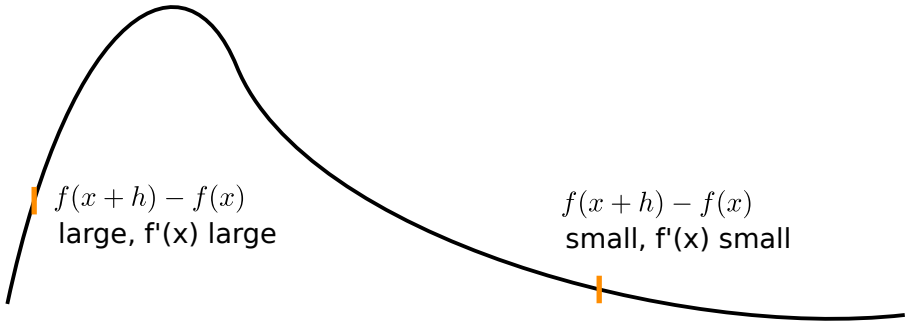
- we find a local minimum, not the global minimum of a function, can be good or bad.
- effects of bad stepsize: divergence – no solution, or slow convergence
- effects of starting point – which ones?

Lets explore these effects:

- in `learnThu8.py` run `tGD([stepsize])` to see the effect of different stepsizes. Why a too large stepsize can lead to numeric overflows? This is a common effect in deep learning training: too large stepsize, then training error will not go down.
- run `tGD2([initvalue])` with *initvalue* $\in [-4, +4]$ to see the effect of a constant stepsize, but different starting points – see in what minimum you end up.

Two Properties and how to deal with them:

1. convergence to global optimum only if the function is convex and the stepsize is sufficiently small. In general one reaches only local minima, sometimes saddle points.
2. the size of the update step $w_{t+1} = w_t - \eta \nabla_w g(w_t)$ depends on the norm of the gradient, too. So when starting in a steep region, even a small stepsize can bring trouble.



$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

the gradient stepsize depends not only on the stepsize parameter but also on the norm of the gradient

learning rate adjustment schemes / learning rate annealing schemes

In practice: one starts with a learning rate, and decreases it over time, either with a polynomial decrease, or by a factor every N iterations.

$$\text{polynomial: } \lambda(t) = c_0 * \quad (t+1)^{-\alpha}, \ \alpha > 0$$

$$\text{regular step at each } T\text{: } \lambda(t) = c_0 * \quad c^{\lfloor t/T \rfloor}, \ c \in (0,1)$$

This enforces convergence (not necessarily to a good point).[a]

in code:
pytorch: `torch.optim.lr_scheduler`

[a] What happens if one decreases the learning rate very fast?

Outline

1. Ordinary Least Squares (Linear regression)
2. Generalization and CLT
3. A recap on gradients, part I
4. Gradients for multilinear Algebra
5. What does a linear mapping represent?
6. Linear model for classification continues
7. Gradient Descent and its properties
8. Stochastic gradient descent

Consider setting with an average of losses over training samples:

$$\frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i), y_i)$$

The application of vanilla gradient descent to this function results in the following algorithm:

$$w_{t+1} = w_t - \eta\nabla_w\left(\frac{1}{n}\sum_{i=1}^{n}\ell(f_{w_t}(x_i), y_i)\right)$$

This is called **batch gradient descent** because it uses the set of **all training data samples** to compute the gradient in each step.

The alternative is **stochastic gradient descent** (SGD).

stochastic gradient descent computes in every iteration a gradient using only one sample every iteration, or, it uses a mini-batch like $k = 64$ samples – such that this set is chosen randomly from the set of all training samples. This is the default in deep learning.

stochastic gradient descent for an average of losses

The core idea of **Stochastic gradient descent** is to compute the gradient only over a randomly selected subset of samples. Stochastic gradient descent when starting at index m and using the next k samples is:

$$\nabla_w \frac{1}{k} \sum_{i=m+0}^{m+k-1} \ell(f_w(x_i), y_i)$$

stochastic gradient descent for an average of losses

use a randomly selected subset of samples:

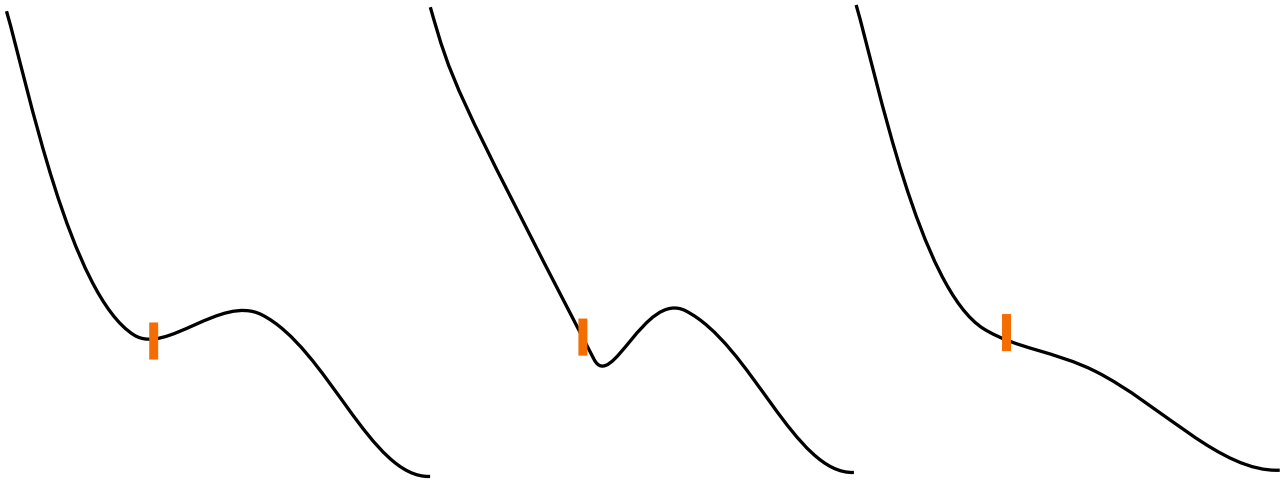
$$\nabla_w \frac{1}{k} \sum_{i=m+0}^{m+k-1} \ell(f_w(x_i), y_i)$$

- initialize start vector w_0 as something, choose step size parameter η
- run while loop, until function value changes very little (δ), do at iteration t:
  - select a random subset of k samples (usually by a random ordering of all training samples)
  - $w_{t+1} = w_t - \eta \nabla_w (\frac{1}{k} \sum_{i=m+0}^{m+k-1} \ell(f_w(x_i), y_i))$
  - compute change to last value: $\delta = \frac{1}{k} \| \sum_{i=m+0}^{m+k-1} \ell(f_{w_{t+1}}(x_i), y_i)) - \ell(f_{w_t}(x_i), y_i)) \|$

Advantages of stochastic gradient descent

- Full-batch is often too costly to compute a gradient using all samples when its more than tens of thousands
- SGD is a noisy, approximated version of the batch gradient.
- injecting small noise is one way to prevent overfitting! Sometimes SGD can be better than full batch gradient descent in finding *good* local optima.

- An illustration why noise to the loss function (e.g. by randomized sampling of training batches) may help to jump out of bad local optima



Left: a loss surfaces at some point (orange). Middle and Right: changes in the loss surfaces as different training data subsets are used. In the middle the gradient norm is much larger compared to the left case – allows to jump out of the local minimum.

SGD as noisy approximation

- Insight: stochastic gradient descent is a noisy approximation (subset of all samples) to the batch gradient.

The batch gradient can be seen as an expected value: non-zero probability only for our training data points $(x_i, y_i) \in D_n$

$$P_{D_n}(x, y) = \begin{cases} \frac{1}{n} & \text{if } (x, y) = (x_i, y_i) \text{ for } (x_i, y_i) \in D_n \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \nabla_w \frac{1}{n} \sum_{(x_i, y_i) \in D_n} \ell(f_w(x_i), y_i) &= \sum_{(x_i, y_i) \in D_n} \nabla_w \ell(f_w(x_i), y_i) \ \frac{1}{n} \\ &= \sum_{(x_i, y_i) \in D_n} \nabla_w \ell(f_w(x_i), y_i) P_{D_n}((x_i, y_i)) \\ &= E_{(x_i, y_i) \sim P_{D_n}}[\nabla_w \ell(f_w(x_i), y_i)] \end{aligned}$$

This is an expectation of a gradient function $\nabla_w \ell(f_w(x_i), y_i)$.

Remember here for a discrete set of values (x_i, y_i)

$$\sum_i r(x_i, y_i) P((x_i, y_i)) = E[r(x, y)]$$

When performing stochastic gradient descent, we use a subset of samples from D_n in every step for computing the gradient. Using only a subset of samples from D_n is an approximation to the expectation shown in the last slide!

$$E_{(x_i,y_i)\sim P_{D_n}}[\nabla_w \ell(f_w(x_i), y_i)] \approx \sum_{i=m+0}^{m+k-1} \nabla_w \ell(f_w(x_i), y_i) \frac{1}{k}$$

Approximation holds due to the central limit theorem! No convergence here[1]

[1] Why there is none?

$$E_{(x_i,y_i)\sim P_{D_n}}[\nabla_w \ell(f_w(x_i), y_i)] \approx \sum_{i=m+0}^{m+k-1} \nabla_w \ell(f_w(x_i), y_i)\frac{1}{k}$$

Approximation holds due to central limit theorem! No convergence here[2]

Recap on central limit theorem

If you have a population with mean μ and standard deviation σ and you draw n random samples from the population - statistically independently, then the distribution of the sample means will be **approximately** normally distributed **with mean** μ **and standard deviation** $\frac{\sigma}{\sqrt{n}}$

[2] Why there is none?

SGD as noisy approximation

The only thing that is necessary for the expectation to hold is that

- drawing pairs (x_i, y_i), (x_k, y_k) is statistically independent

This noise from approximation can sometimes act as regularization – prevents to look at the data too closely.

Questions?!