# Linear Regression

# Linear model for regression

Regression problem $\mathbb{Y} = \mathbb{R}$

ML model $a : \mathbb{X} \to \mathbb{Y}$

Family of ML models $a \in \mathcal{A}$

$$\mathcal{A} = \{a(\boldsymbol{x}) = w_0 + w_1 x_1 + \cdots + w_D x_D \mid w_0, w_1, \ldots, w_D \in \mathbb{R}\}$$

$w_0, w_1, \ldots, w_D$ - parameters/weights of linear model

$w_0$ - bias

In total D+1 parameters. Linear models require small memory, are fast in inference and interpretable.

# Linear model for regression

ML model $a(\boldsymbol{x}) = w_0 + \sum_{j=1}^{D} w_j x_j = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle$

Sometimes it is convenient to augment feature vectors with constant feature =1

$$\boldsymbol{x} \to \tilde{\boldsymbol{x}} = [\boldsymbol{x}, 1]$$

$$a(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + w_0 = \langle \tilde{\boldsymbol{w}}, \tilde{\boldsymbol{x}} \rangle \qquad \tilde{\boldsymbol{w}} = [\boldsymbol{w}, w_0]$$

# Example: apartment cost prediction

x – apartment, y – its cost

$$a(\boldsymbol{x}) = w_0 + w_1 * (area) + w_2 * (number\_of\_rooms) + \\ + w_3 * (distance\_to\_train\_station) + \dots$$

Important property of linear models: each feature influences independently on prediction result

Sometimes it is reasonable to consider different combinations of initial features and feature transformations

# OHE for nominal features

Linear models can work only with numerical features

Let's consider nominal feature "apartment district"

$$x \in C,\ C = \{c_1, \ldots, c_m\}$$

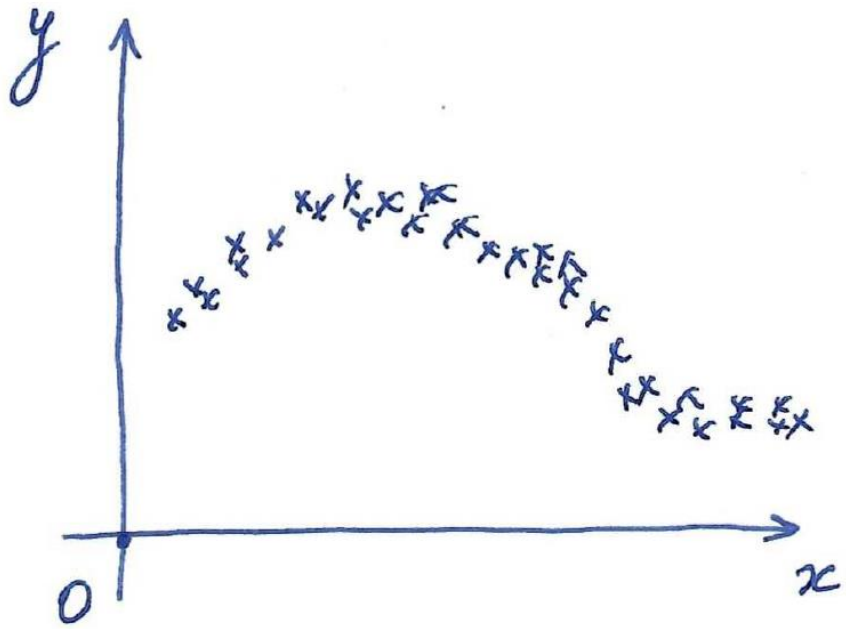
One-hot-encoding

$$x \to [b_1(x), b_2(x), \ldots, b_m(x)]$$

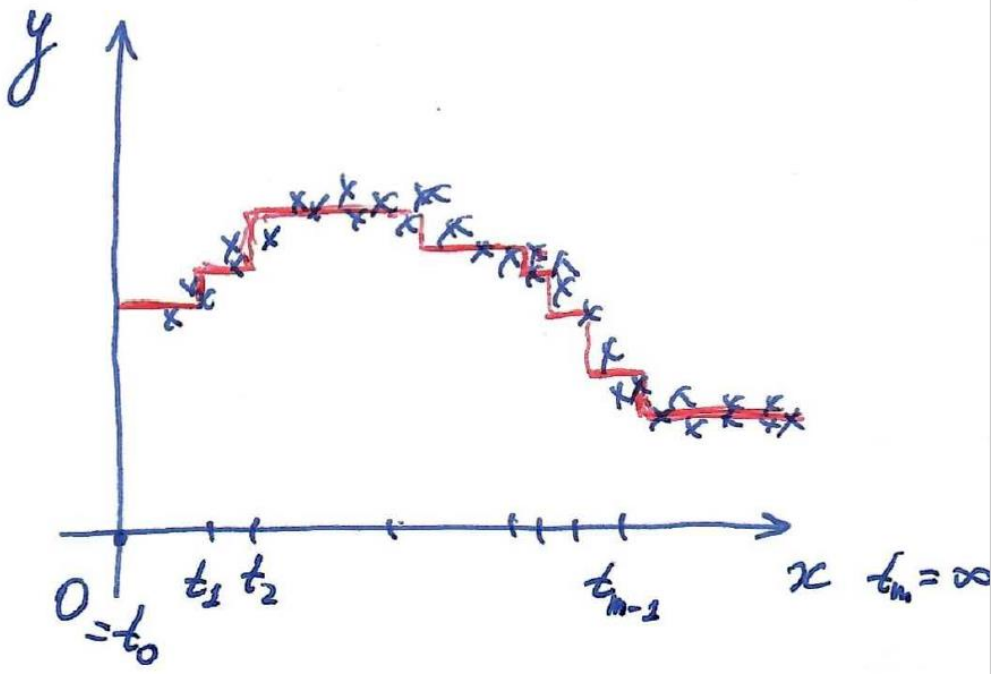
$$b_j(x) = [x = c_j] \in \{0, 1\}$$

$$a(x) = w_0 + w_1[x = c_1] + w_2[x = c_2] + \ldots$$

# Binarization of numerical features



# Binarization of numerical features




$$x \to [b_1(x), \ldots, b_m(x)]$$

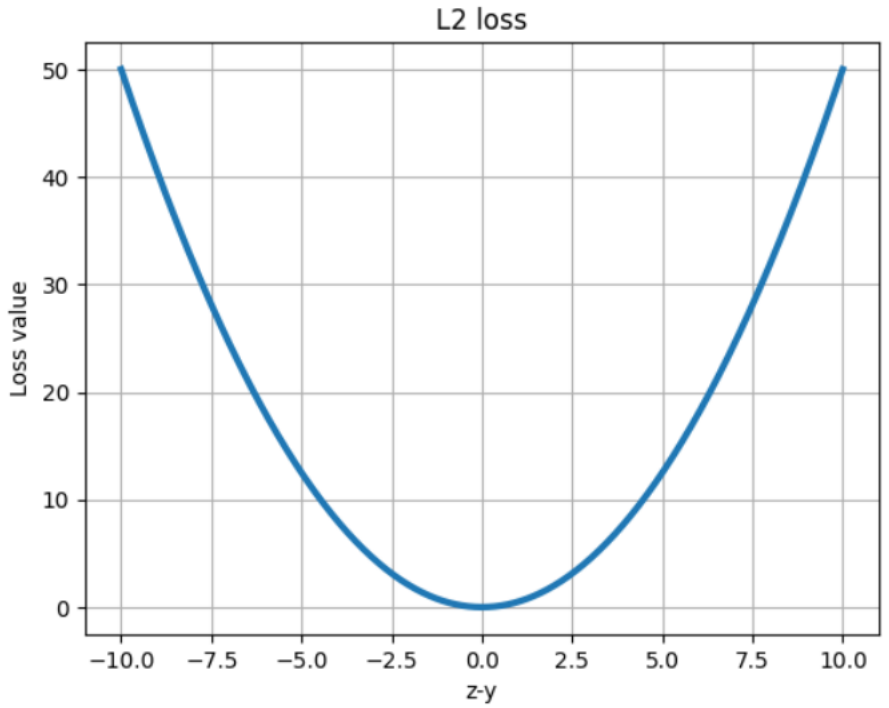
$$b_j(x) = [t_{j-1} \leq x < t_j]$$

For linear models it is important to prepare a reasonable set of features from original ones!

# Loss functions for regression

Squared error loss / Quadratic loss / L2 loss $L(y, z) = (y - z)^2$

Mean squared error $MSE(a, X) = \frac{1}{N} \sum_{i=1}^{N} (a(\boldsymbol{x}_i) - y_i)^2$



# Loss functions for regression

Squared error loss / Quadratic loss / L2 loss $L(y, z) = (y - z)^2$

Mean squared error $MSE(a, X) = \frac{1}{N} \sum_{i=1}^{N} (a(\boldsymbol{x}_i) - y_i)^2$

Root mean squared error $RMSE(a, X) = \sqrt{MSE(a, X)}$

Coefficient of determination

$$R^2(a, X) = 1 - \frac{\sum_{i=1}^{N} (a(\boldsymbol{x}_i) - y_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}$$

For ideal a(x) R2=1, for constant a(x) R2=0

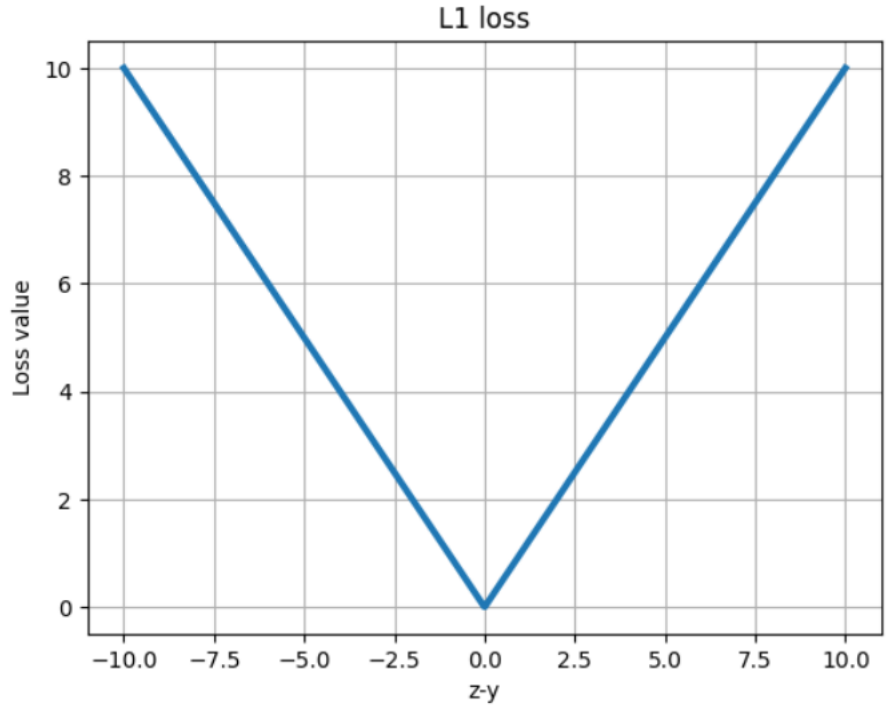
# Mean Absolute Error loss

L1 loss $L(y, z) = |y - z|$

$$MAE(a, X) = \frac{1}{N} \sum_{i=1}^{N} |a(\boldsymbol{x}_i) - y_i|$$

MAE is non-differentiable

MAE is more robust to outliers

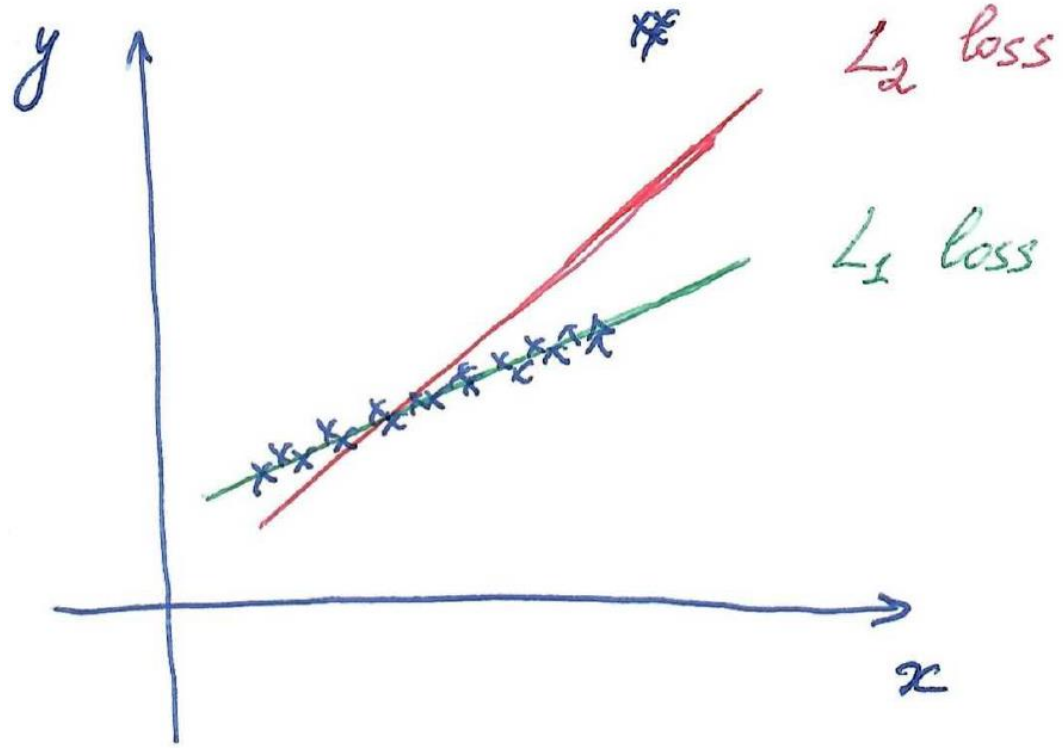


# Mean Absolute Error loss

L1 loss $L(y, z) = |y - z|$

$$MAE(a, X) = \frac{1}{N} \sum_{i=1}^{N} |a(\boldsymbol{x}_i) - y_i|$$

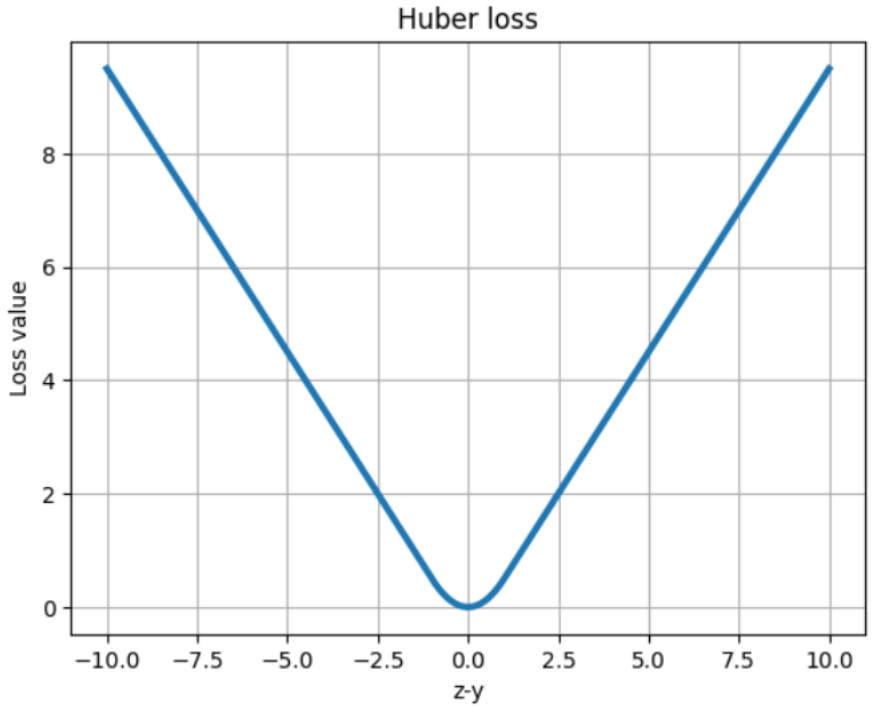
| $y$ | $z$ | $(y-z)^2$ | $\lvert y-z \rvert$ |
|---|---|---|---|
| 1 | 2 | 1 | 1 |
| 1000 | 2 | 996004 | 998 |
| 1 | 3 | 4 | 2 |
| 1000 | 3 | 994009 | 997 |



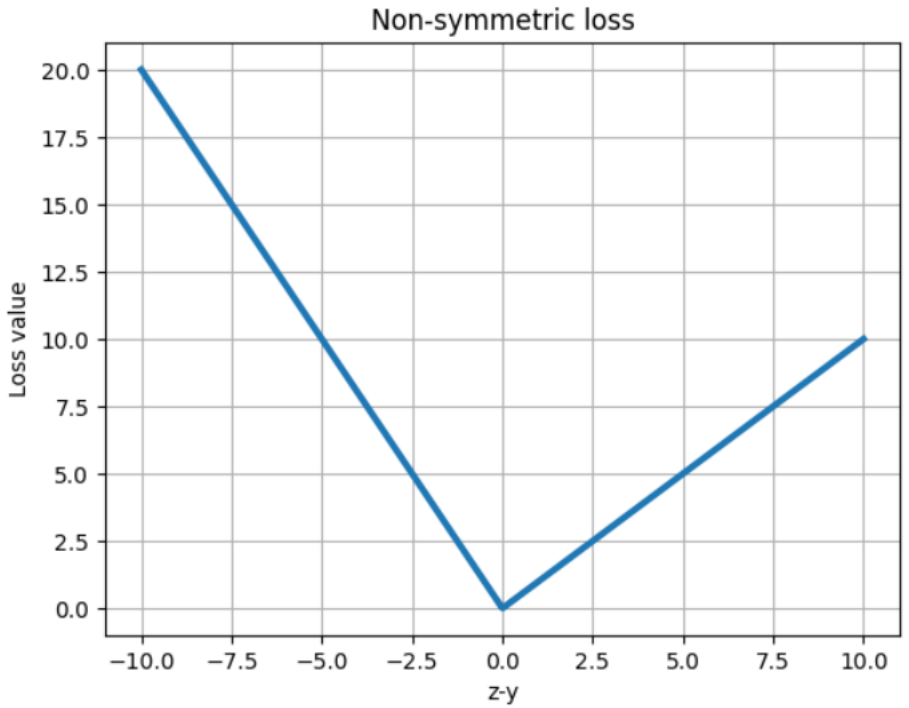
# Huber loss

$$L(y,z) = \begin{cases} \frac{1}{2}(y-z)^2, & \text{if } |y-z| \leq \delta, \\ \delta|y-z| - \frac{1}{2}\delta^2, & \text{otherwise.} \end{cases}$$

Huber loss is continuously differentiable everywhere



# Non-symmetric loss




Loss function should be chosen according to business requirements

# Other losses

Mean Squared Logarithmic Error (MSLE)

$$L(y, z) = (\log(z + 1) - \log(y + 1))^2$$

Mean Absolute Percentage Error (MAPE)

$$L(y, z) = \left| \frac{y - z}{y} \right|$$

And many others…

# Linear regression training

$$Q(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - y_i)^2 \to \min_{\boldsymbol{w}}$$

$$X = \begin{bmatrix} x_{1,1} \dots x_{1,D} \\ x_{2,1}, \dots x_{2,D} \\ \vdots \\ x_{N,1} \dots x_{N,D} \end{bmatrix} \in \mathbb{R}^{N\times D} \qquad \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix} \in \mathbb{R}^{D} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^{N}$$

$$z_i = \langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - y_i$$

$$\boldsymbol{z} = X\boldsymbol{w} - \boldsymbol{y}$$

$$Q(w) = \frac{1}{N}\sum_{i=1}^{N} z_i^2 = \frac{1}{N}\|\boldsymbol{z}\|_2^2 = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2$$

# Linear regression training

$$Q(\boldsymbol{w}) = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 \to \min_{\boldsymbol{w}}$$

$$\nabla_{\boldsymbol{w}} Q(\boldsymbol{w}) = \boldsymbol{0} \;\Rightarrow\; \boldsymbol{w}_{opt} = (X^T X)^{-1} X^T \boldsymbol{y}$$

$\boldsymbol{w}_{opt}$ is solution of linear system $X^T X \boldsymbol{w}_{opt} = X^T \boldsymbol{y}$

Properties of the matrix $X^T X$ :

1) Quadratic and symmetric: $(X^T X)^T = X^T X$
2) Non-negatively defined: $\boldsymbol{u}^T X^T \underbrace{X\boldsymbol{u}}_{\boldsymbol{v}} = \boldsymbol{v}^T \boldsymbol{v} \geq 0$

Recall: matrix $A$ is non-negatively definite, if $\boldsymbol{u}^T A \boldsymbol{u} \geq 0 \;\forall \boldsymbol{u}$

Equivalently: all eigenvalues are non-negative.

# Linear regression training

$\boldsymbol{w}_{opt}$ is solution of linear system $X^T X \boldsymbol{w}_{opt} = X^T \boldsymbol{y}$

For analytical solution the matrix $X^T X$ should be non-singular.

$$\text{rank}(X^T X) \leq \min(\text{rank}(X^T), \text{rank}(X))$$

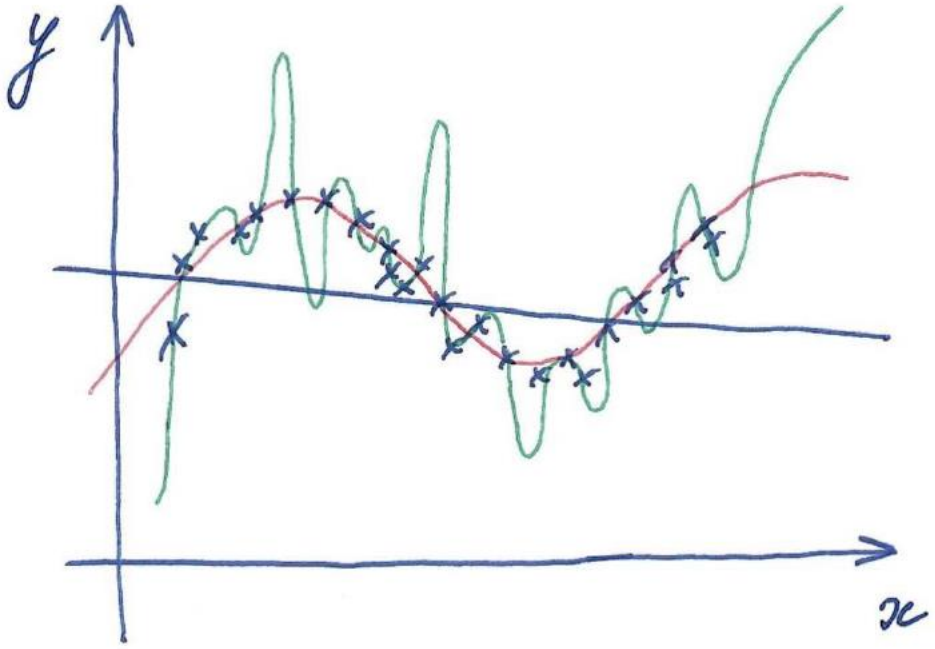
So the matrix is singular if N < D or there are duplicating features

Computational complexity of analytical solution:

$O(ND^2)$ for computing $X^T X$

$O(D^3)$ for linear system solving

# Underfitting and overfitting




$$a(x) = w_0 + w_1 x$$

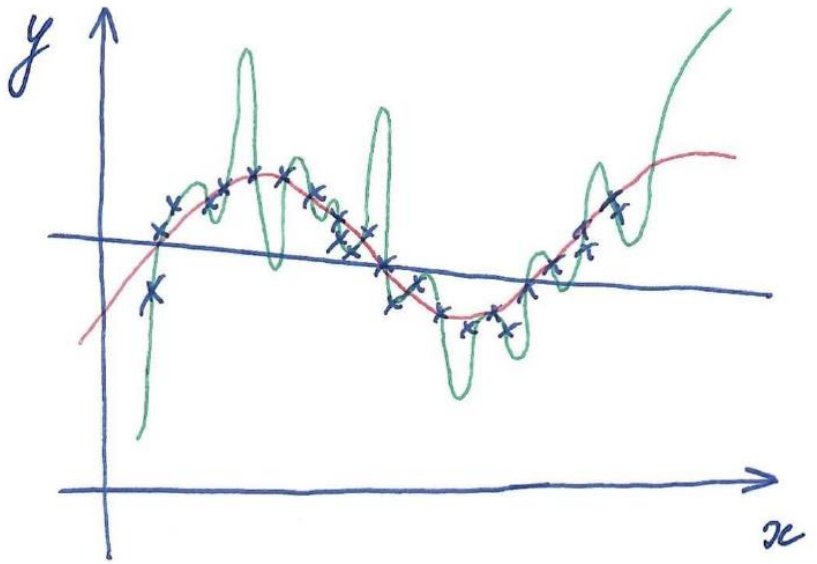
$$a(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

$$a(x) = w_0 + w_1 x + \cdots + w_n x^n$$

Blue model is too simple and underfitted, green model is too complicated and overfitted, red model has good generalization ability

# Regularization

Overfitted model usually has oscillating effect – small change of inputs may lead to large change of outputs




$$a(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D$$

For linear models oscillating happens with high absolute values of some weights

Idea: let's additionally require during training that our ML model should have weights with low absolute values

$$Q(\boldsymbol{w}, X, \boldsymbol{y}) + \lambda \|\boldsymbol{w}\|_2^2 \to \min_{\boldsymbol{w}}, \ \lambda > 0$$

# Regularization

$$F(\boldsymbol{w}) = Q(\boldsymbol{w}, X, \boldsymbol{y}) + \lambda \|\boldsymbol{w}\|_2^2 = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2 \to \min_{\boldsymbol{w}}$$

This is known as L2 regularization, weight decay, ridge regularization

This minimization problem can be solved analytically

$$\boldsymbol{w}_{opt} = \left(X^T X + \lambda N I\right)^{-1} X^T \boldsymbol{y}$$

# Regularization

Non-regularized solution $\boldsymbol{w}_{opt} = \left(X^T X\right)^{-1} X^T \boldsymbol{y}$

This solution can’ t be computed if N<D or there are duplicating features

Regularized solution $\boldsymbol{w}_{opt} = \left(X^T X + \lambda N I\right)^{-1} X^T \boldsymbol{y}$

This solution can always be computed because the matrix $X^T X + \lambda N I$ is strictly positively definite

Important: bias parameter is usually not regularized!

# Eigenvalue Decomposition

Let's consider arbitrary symmetric matrix $A^T = A \in \mathbb{R}^{n \times n}$
and all its eigenvalues $\lambda_1, \ldots, \lambda_n$ and eigenvectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$

Eigenvalue decomposition in matrix form: $A = Q \Lambda Q^T$

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \qquad Q = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n],\ QQ^T = I$$

$$X^T X = Q \Lambda Q^T \qquad\qquad X^T X + \lambda N I = Q \Lambda Q^T + \lambda N Q Q^T = Q[\Lambda + \lambda N I] Q^T$$

$$\mathrm{eigenvalue}(X^T X + \lambda N I) = \mathrm{eigenvalue}(X^T X) + \lambda N$$

# Choosing regularization coefficient

$$F(\boldsymbol{w}, \lambda) = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2 \to \min_{\boldsymbol{w},\lambda} \;\Rightarrow\; \lambda_{opt} = 0$$

Regularization coefficient can't be optimized on the training set. Parameters with this property are called hyperparameters.

Hyperparameter can be chosen by brute force – estimating cross-validation (CV) value of ML model for some set of values

(1/N) coefficient in loss function helps making optimal value of regularization parameter less sensitive to the training set size N.

# Underfitting and overfitting

In case of overfitting increase regularization coefficient

In case of underfitting decrease regularization coefficient or increase ML model complexity

# L1 regularization / Lasso regression

$$F(\boldsymbol{w}) = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_1 = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\sum_{j=1}^{D}|w_j| \to \min_{\boldsymbol{w}}$$

L1 norm penalizes high absolute values of weights
and thus prevents overfitting.

L1 norm allows to find sparse solution when some of weights w_j exactly equal zero. In such a way we may find relevant/irrelevant features.

$$a(\boldsymbol{x}) = w_0 + w_1 * (area) + w_2 * (number\_of\_rooms) +$$
$$+ w_3 * (distance\_to\_train\_station) + \dots$$

# L1 regularization / Lasso regression

From optimization theory it is known that the optimization problem

$$f(\boldsymbol{w}) + \lambda g(\boldsymbol{w}) \to \min_{\boldsymbol{w}}$$

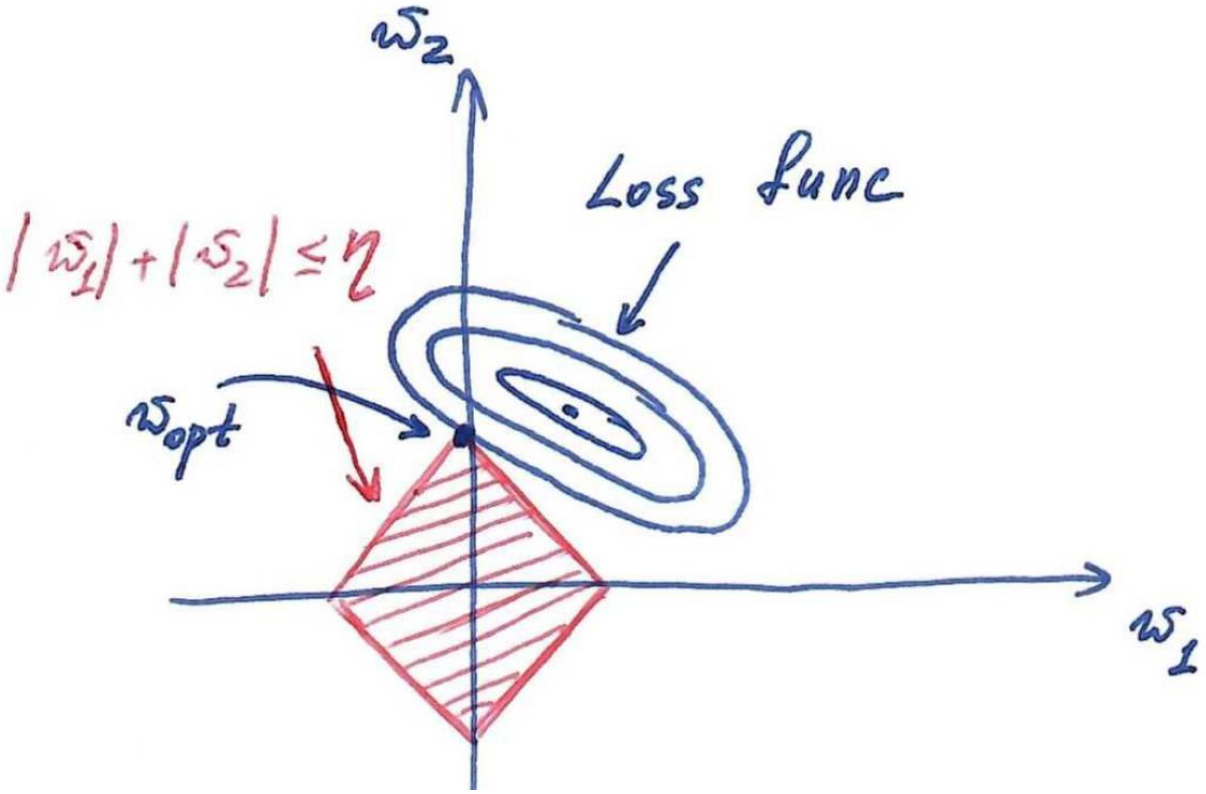
is equivalent to the following constrained optimization problem:

$$f(\boldsymbol{w}) \to \min_{\boldsymbol{w}},$$
$$\text{s.t. } g(\boldsymbol{w}) \leq \eta$$

# L1 regularization / Lasso regression

$$f(\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_1 \to \min_{\boldsymbol{w}}$$

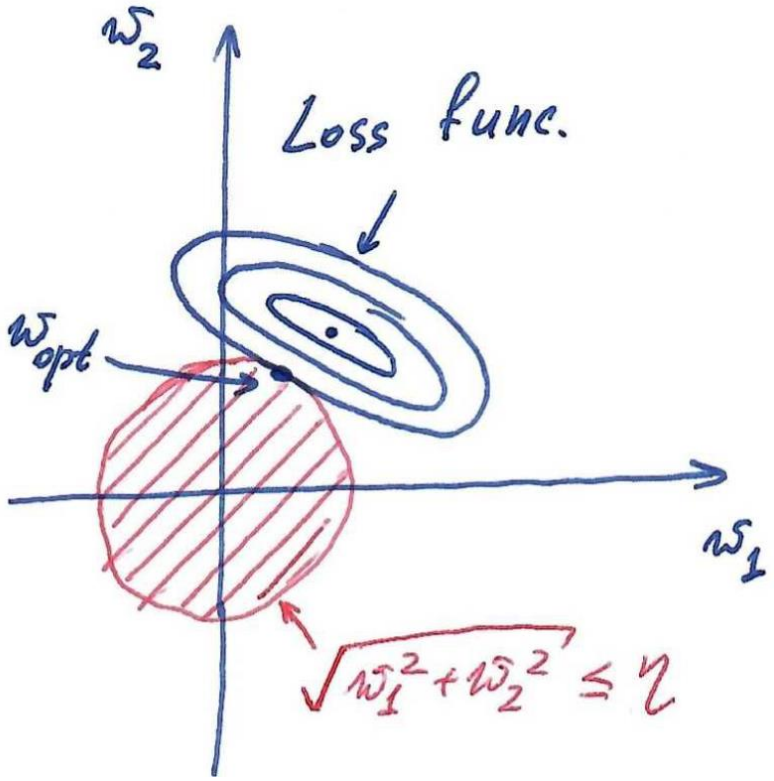
$$f(\boldsymbol{w}) \to \min_{\boldsymbol{w}},$$
$$\text{s.t. } \|\boldsymbol{w}\|_1 \leq \eta$$



# L2 regularization / Ridge regression

$$f(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2 \to \min_{\boldsymbol{w}}$$

$$f(\boldsymbol{w}) \to \min_{\boldsymbol{w}},$$
$$\text{s.t. } \|\boldsymbol{w}\|_2^2 \leq \eta$$



# Elastic Net regularization

$$\frac{1}{N}\|X\boldsymbol{w}-\boldsymbol{y}\|_2^2+\lambda_1\|\boldsymbol{w}\|_2^2+\lambda_2\|\boldsymbol{w}\|_1=\frac{1}{N}\|X\boldsymbol{w}-\boldsymbol{y}\|_2^2+\lambda_1((1-\lambda_2)\|\boldsymbol{w}\|_2^2+\lambda_2\|\boldsymbol{w}\|_1)$$

# Matrix/vector differentiation

# Vector expressions

In mathematical expressions a column-wise vector notation is convenient

**Scalar**

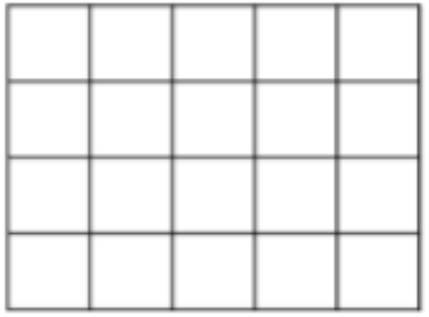
$$\alpha \in \mathbb{R} = \mathbb{R}^1 = \mathbb{R}^{1\times 1}$$

**Vector**

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \in \mathbb{R}^n = \mathbb{R}^{n\times 1}$$

**Matrix**

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \in \mathbb{R}^{n\times m}$$



# Vector expressions

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \quad \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i = \boldsymbol{x}^T \boldsymbol{y}$$

$$\boldsymbol{x}\boldsymbol{y}^T \in \mathbb{R}^{n \times n} \quad (\boldsymbol{x}\boldsymbol{y}^T)_{ij} = x_i y_j$$

$$\|\boldsymbol{x}\|_2^2 = \sum_{i=1}^{n} x_i^2 = \boldsymbol{x}^T \boldsymbol{x}$$

# Matrix expressions

$$A, B \in \mathbb{R}^{n \times m} \qquad \langle A, B \rangle = \sum_{i,j=1}^{n,m} A_{ij} B_{ij} = \operatorname{tr}(A^T B) =$$

$$= \sum_{j=1}^{m} (A^T B)_{jj} = \sum_{j=1}^{m} \sum_{i=1}^{n} (A^T)_{ji} B_{ij} = \sum_{j=1}^{m} \sum_{i=1}^{n} A_{ij} B_{ij}$$

$$\|A\|_F^2 = \sum_{i,j} A_{ij}^2 = \operatorname{tr}(A^T A)$$

Properties of transposition:

$$(A + B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

# Summation and product

Summation of two vector/matrix objects $\boldsymbol{x} + \boldsymbol{y}$
is valid only if two objects have the same size

Product of two objects $AB$
is valid only if the last dimension of the first object coincides with the first dimension of the second object.
Exception: multiplication by scalar is allowed. $\alpha A$

In general, the order of arguments in product is important because

$$AB \neq BA$$

# Summation and product

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n,\ \boldsymbol{z} \in \mathbb{R}^m,\ A \in \mathbb{R}^{n \times m},\ b \in \mathbb{R}$$

Correct examples:

1. $\boldsymbol{x} + \boldsymbol{y}$
2. $\boldsymbol{x}^T \boldsymbol{y} + b$
3. $A\boldsymbol{z}$
4. $A + \boldsymbol{x}\boldsymbol{z}^T$
5. $AA^T$
6. $bA$

Incorrect examples:

1. $\boldsymbol{x} + \boldsymbol{z}$
2. $\boldsymbol{x} + b$
3. $\boldsymbol{x} + \boldsymbol{y}^T$
4. $A\boldsymbol{x}$
5. $AA$
6. $A + \boldsymbol{x}\boldsymbol{y}^T$

# Inversion (division)

It is possible to multiply both parts of equation either by inversed scalar $\alpha^{-1}$ or by inversed matrix $A^{-1}$, where $A$ is quadratic and non-degenerate matrix.

Division by vector $x^{-1}$ is not possible!

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \; A \in \mathbb{R}^{n \times n}, \; \alpha \in \mathbb{R}$$

Correct examples:

1. $\alpha \boldsymbol{x} = \boldsymbol{y} \;\Rightarrow\; \boldsymbol{x} = \alpha^{-1} \boldsymbol{y}$
2. $A\boldsymbol{x} = \boldsymbol{y} \;\Rightarrow\; \boldsymbol{x} = A^{-1} \boldsymbol{y}$

Incorrect examples:

1. $\boldsymbol{x}^T \boldsymbol{y} = \alpha \;\Rightarrow\; \boldsymbol{y} = \boldsymbol{x}^{-T} \alpha$
2. $X^T X \boldsymbol{w} = X^T \boldsymbol{y} \;\Rightarrow\; X\boldsymbol{w} = X^{-T} X^T \boldsymbol{y} = \boldsymbol{y}$

# Manipulating with vector expression

$$Q(\boldsymbol{w}) = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 = \frac{1}{N}(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y}) = \frac{1}{N}((X\boldsymbol{w})^T - \boldsymbol{y}^T)(X\boldsymbol{w} - \boldsymbol{y}) =$$

$$= \frac{1}{N}(\boldsymbol{w}^T X^T - \boldsymbol{y}^T)(X\boldsymbol{w} - \boldsymbol{y}) = \frac{1}{N}(\boldsymbol{w}^T X^T X\boldsymbol{w} - \boldsymbol{y}^T X\boldsymbol{w} - \boldsymbol{w}^T X^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y})$$

Check yourself by dimensionality! $Q(\boldsymbol{w}) \in \mathbb{R}$ $\underbrace{\boldsymbol{y}^T}_{1\times N}\underbrace{X}_{N\times D}\underbrace{\boldsymbol{w}}_{D\times 1} \in \mathbb{R}$

$$\underbrace{\boldsymbol{y}^T X\boldsymbol{w}}_{\in\mathbb{R}} = (\boldsymbol{y}^T X\boldsymbol{w})^T = \boldsymbol{w}^T X^T \boldsymbol{y}$$

$$Q(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{w}^T X^T X\boldsymbol{w} - 2\boldsymbol{w}^T X^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y})$$

# Manipulating with vector expression

$$\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n \qquad \|\boldsymbol{u}\boldsymbol{v}^T\|_F^2 = \mathrm{tr}((\boldsymbol{u}\boldsymbol{v}^T)^T\boldsymbol{u}\boldsymbol{v}^T) = \mathrm{tr}(\boldsymbol{v}\boldsymbol{u}^T\boldsymbol{u}\boldsymbol{v}^T)$$

Circular property of trace operation:

$$\mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA)$$

$$\mathrm{tr}(\boldsymbol{v}\boldsymbol{u}^T\boldsymbol{u}\boldsymbol{v}^T) = \mathrm{tr}(\underbrace{\boldsymbol{v}^T\boldsymbol{v}}_{\in\mathbb{R}}\underbrace{\boldsymbol{u}^T\boldsymbol{u}}_{\in\mathbb{R}}) = \boldsymbol{v}^T\boldsymbol{v}\boldsymbol{u}^T\boldsymbol{u} = \|\boldsymbol{v}\|_2^2\|\boldsymbol{u}\|_2^2$$

# Differential calculus in scalar case

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

Table of standard derivatives: $(x^n)' = nx^{n-1},\ (\sin(x))' = \cos(x),\ (\exp(x))' = \exp(x), \ldots$

Differentiation rules: $(f+g)' = f' + g',\ (fg)' = f'g + fg',\ (f(g))' = f'(g'), \ldots$

# Differential calculus in vector/matrix case

$$f(X) = \mathrm{tr}(AX^{-1}B) : \mathbb{R}^{n\times n} \to \mathbb{R}$$

$$\frac{\partial f(X)}{\partial X} = \left\{ \frac{\partial f}{\partial X_{ij}} \right\}_{i,j=1}^{n,n} = \frac{\partial}{\partial X} \mathrm{tr}( \underbrace{AX^{-1}B}_{\text{2D matrix}} ) = \mathrm{tr}\left[ \underbrace{\frac{\partial}{\partial X} AX^{-1}B}_{\text{4D tensor}} \right]$$

Goal: Make differentiation without dealing with higher dimensional tensors in intermediate computations

# Differential calculus in vector/matrix case

$f : \mathcal{U} \to \mathcal{V}$ $\mathcal{U}, \mathcal{V}$ - set of scalars, vectors, matrices

**Def.** Function $f$ is called differentiable at point $x$ if

$f(x+h) - f(x) = df(x)[h] + \bar{o}(\|h\|)$, $df(x)[h]$ - linear function w.r.t. $h$

$df(x)[h] = df(x)[dx] = df$ - differential

For fixed x $df(x)[h] : \mathcal{U} \to \mathcal{V}$

For scalar functions $f : \mathbb{R} \to \mathbb{R}$ differential is just a product of derivative and increment $df = df(x)[dx] = f'(x)dx$

For functions $f : \mathbb{R}^n \to \mathbb{R}$ differential is $df = df(\boldsymbol{x})[d\boldsymbol{x}] = \nabla f(\boldsymbol{x})^T d\boldsymbol{x}$

# Differential calculus in vector/matrix case

Differentials of standard functions:

$$d(\boldsymbol{c}^T\boldsymbol{x}) = \boldsymbol{c}^T d\boldsymbol{x}$$

$$d(\boldsymbol{x}^T A\boldsymbol{x}) = \boldsymbol{x}^T(A + A^T)d\boldsymbol{x} =$$

$$= \{A^T = A\} = 2\boldsymbol{x}^T A d\boldsymbol{x}$$

$$d\mathrm{tr}(X) = \mathrm{tr}(dX)$$

$$d\det(X) = \det(X)\mathrm{tr}(X^{-1}dX)$$

$$dX^{-1} = -X^{-1}dXX^{-1}$$

Differentiation rules:

$$dA = 0$$

$$d(AXB) = A \cdot dX \cdot B$$

$$d(X + Y) = dX + dY$$

$$d(XY) = dX \cdot Y + X \cdot dY$$

$$d(X/\varphi) = \frac{dX\varphi - Xd\varphi}{\varphi^2}, \ \varphi \in \mathbb{R}$$

$$f(x) = h(g(x))$$

$$df(x)[dx] = dh(g(x))[dg] =$$

$$= dh(g(x))[dg(x)[dx]]$$

# Differential calculus in vector/matrix case

$$f(\boldsymbol{x}) = \frac{1}{3}\|\boldsymbol{x}\|_2^3 = \frac{1}{3}(\boldsymbol{x}^T\boldsymbol{x})^{\frac{3}{2}} : \mathbb{R}^n \to \mathbb{R} \qquad \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \text{ - ?}$$

$$f(\boldsymbol{x}) = h(g(\boldsymbol{x})), \; g(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{x}, \; h(g) = \frac{1}{3}g^{\frac{3}{2}}$$

$$dg = dg(\boldsymbol{x})[d\boldsymbol{x}] = 2\boldsymbol{x}^T d\boldsymbol{x}$$

$$dh = dh(g)[dg] = h'(g)dg = \frac{1}{3}\cdot\frac{3}{2}\cdot g^{\frac{1}{2}}dg = \frac{1}{2}g^{\frac{1}{2}}dg$$

$$df = df(\boldsymbol{x})[d\boldsymbol{x}] = dh(g(\boldsymbol{x}))[dg] = \frac{1}{2}g^{\frac{1}{2}}2\boldsymbol{x}^T d\boldsymbol{x} = (\boldsymbol{x}^T\boldsymbol{x})^{\frac{1}{2}}\boldsymbol{x}^T d\boldsymbol{x}$$

# Differential calculus in vector/matrix case

Connection between differential and gradient:

$f : \mathcal{U} \to \mathbb{R}$

| $\mathcal{U}$ | $\mathbb{R}$ | $\mathbb{R}^n$ | $\mathbb{R}^{n\times m}$ |
|---|---|---|---|
| Canonical form of $df$ | $f'(x)dx$ | $\nabla f(\boldsymbol{x})^T d\boldsymbol{x}$ | $\mathrm{tr}(\nabla f(X)^T dX)$ |

$$df = (\boldsymbol{x}^T\boldsymbol{x})^{\frac{1}{2}}\boldsymbol{x}^T d\boldsymbol{x} = \nabla f(\boldsymbol{x})^T d\boldsymbol{x}$$

$$\nabla f(\boldsymbol{x}) = (\boldsymbol{x}^T\boldsymbol{x})^{\frac{1}{2}}\boldsymbol{x} = \|\boldsymbol{x}\|_2\boldsymbol{x}$$

# Differential calculus in vector/matrix case

General scheme:

1. Compute differential of your function using table of standard differentials and differentiation rules
2. Transform expression for differential into canonical form
3. Extract gradient value

# Differential calculus for linear regression

$$Q(\boldsymbol{w}) = \frac{1}{N}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 = \frac{1}{N}(\boldsymbol{w}^T X^T X\boldsymbol{w} - 2\boldsymbol{w}^T X^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y})$$

$$dQ = \frac{1}{N}(2\boldsymbol{w}^T X^T X d\boldsymbol{w} - 2d\boldsymbol{w}^T X^T \boldsymbol{y}) = \nabla Q(\boldsymbol{w})^T d\boldsymbol{w}$$

$$d\boldsymbol{w}^T X^T \boldsymbol{y} = \boldsymbol{y}^T X d\boldsymbol{w}$$

$$\nabla Q(\boldsymbol{w}) = \frac{1}{N}(2X^T X\boldsymbol{w} - 2X^T \boldsymbol{y}) = \frac{2}{N}X^T(X\boldsymbol{w} - \boldsymbol{y}) = \boldsymbol{0}$$

$$\Rightarrow \; X^T X\boldsymbol{w} = X^T \boldsymbol{y} \; \Rightarrow \; \boldsymbol{w}_{opt} = (X^T X)^{-1} X^T \boldsymbol{y}$$

# Differential calculus for linear regression

$$Q(\boldsymbol{w}) = \frac{1}{N}\| \underbrace{X\boldsymbol{w} - \boldsymbol{y}}_{\boldsymbol{z}} \|_2^2 = \frac{1}{N}\|\boldsymbol{z}\|_2^2 = \boldsymbol{z}^T\boldsymbol{z}$$

$$dQ = \frac{1}{N}2\boldsymbol{z}^T d\boldsymbol{z} = \frac{2}{N}\boldsymbol{z}^T(X d\boldsymbol{w}) = \frac{2}{N}(X\boldsymbol{w} - \boldsymbol{y})^T X d\boldsymbol{w}$$

$$\Rightarrow \ \nabla Q(\boldsymbol{w}) = \frac{2}{N}X^T(X\boldsymbol{w} - \boldsymbol{y})$$

# Differential calculus for linear regression

$$Q(\boldsymbol{z}) = \frac{1}{N}\boldsymbol{z}^T\boldsymbol{z},\ \boldsymbol{z} = X\boldsymbol{w} - \boldsymbol{y} \qquad \nabla_X Q(X) \text{ - ?}$$

$$dQ = \frac{1}{N}2\boldsymbol{z}^T d\boldsymbol{z} = \frac{2}{N}\boldsymbol{z}^T(dX\boldsymbol{w}) = \mathrm{tr}(\nabla_X Q(X)^T dX) =$$

$$= \frac{2}{N}\mathrm{tr}(\boldsymbol{z}^T dX\boldsymbol{w}) = \frac{2}{N}\mathrm{tr}(\boldsymbol{w}\boldsymbol{z}^T dX)$$

$$\Rightarrow\ \nabla_X Q(X) = \frac{2}{N}\boldsymbol{z}\boldsymbol{w}^T = \frac{2}{N}(X\boldsymbol{w} - \boldsymbol{y})\boldsymbol{w}^T$$

$$X \in \mathbb{R}^{N\times D},\ \nabla_X Q(X) = \left\{\frac{\partial Q(X)}{\partial X_{ij}}\right\}_{i,j=1}^{N,D} \in \mathbb{R}^{N\times D},\ \boldsymbol{z} \in \mathbb{R}^N,\ \boldsymbol{w} \in \mathbb{R}^D,\ \boldsymbol{z}\boldsymbol{w}^T \in \mathbb{R}^{N\times D}$$

# Matrix differential example

$$f(X) = \det(AX^{-1}B) : \mathbb{R}^{n\times n} \to \mathbb{R}$$

$$df = \det(AX^{-1}B)\mathrm{tr}((AX^{-1}B)^{-1}d(AX^{-1}B)) =$$

$$= \det(AX^{-1}B)\mathrm{tr}(-(AX^{-1}B)^{-1}AX^{-1}dXX^{-1}B) =$$

$$= -\det(AX^{-1}B)\mathrm{tr}(X^{-1}B(AX^{-1}B)^{-1}AX^{-1}dX)$$

$$\nabla_X f(X) = -\det(AX^{-1}B)X^{-T}A^T(AX^{-1}B)^{-T}B^TX^{-T}$$