Zbrani zapiski za 1. letnik magistrskega študija

Patrik Žnidaršič

November 15, 2024

Opomba k notaciji: Včasih je produkt vektorja s skalarjem napisan kot $\lambda.x$, s piko. Tudi oprator $\vec{\nabla}$ včasih obravnavamo kot vektor, in pišemo gradient s piko spodaj, $\vec{\nabla}.f$. Divergenco označimo z $\vec{\nabla} \cdot f$.

Contents

1 Moderna fizika

1.1 Electromagnetism

1.1.1 A note on fields

A FIELD is defined as a quantity that takes a value at every point in space and time, the value being some element of a particular vector space. Fields are usually LOCAL, which means they only depend on one point in space-time. They influence particles, and particles influence fields.

There are four known interactions:

- ELECTROMAGNETISM is the force between charged particles. Quantum mechanically, it is carried by photons, and it is responsible for most of our experience of the world.
- The WEAK and STRONG NUCLEAR FORCES govern the behaviour inside atomic cores.
- GRAVITY is a geometric force, which we model slightly differently. Compared to the other forces, it is extremely weak.

Electromagnetism is a unified theory connecting the electric and magnetic fields. There is also a successful unification of EM and the weak force, called ELECTROWEAK THEORY. We may further unify this with the strong force, giving the GRAND UNIFICATION THEORY (GUT). It is mathematically consistent, but none of its predictions have ever been measured. This is a real shame, since among other things, it predicts magnetic monopoles and proton decay.

All these unifications are performed with the same techniques, but these fail when trying to unify GUT with gravity.

1.1.2 A Note on Notation

We use Einstein's summation convention: Whenever there is a repeated index in the expression, assume that we are summing over it. For example, the dot product of two vectors $\vec{w}$ and $\vec{v}$ can be written as

$$\langle \vec{v}, \vec{w} \rangle = v^i w^i = \sum_{i=1}^{3} v^i w^i,$$

and similarly, the divergence of a vector field $\vec{E}$ is

$$\vec{\nabla} \cdot \vec{E} = \partial_i E^i = \sum_{i=1}^{3} \partial_i E^i.$$

Note that here, v^i denotes the i-th component of vector $\vec{v}$.

We also define two symbols, the KRONECKER DELTA

$$\delta_{ij} = \begin{cases} 1 & i = j, \\ 0 & \text{otherwise} \end{cases}$$

and the LEVI-CHIVITA SYMBOL

$$\varepsilon_{ijk} = \begin{cases} 0 & \text{one of the indices is repeated,} \\ \operatorname{sgn}(i\,j\,k) & \text{all indices are distinct.} \end{cases}$$

Here, $\operatorname{sgn}\pi$ denotes the sign of permutation π as an element of $\{\pm 1\}$. With these definitions, the following useful identities hold:

- for two vectors $\vec{v}, \vec{w}$, the dot product is

  $$\langle \vec{v}, \vec{w} \rangle = \delta_{ij} v^i w^j$$

  and the cross product is, component-wise

  $$(\vec{v} \times \vec{w})^i = \varepsilon_{ijk} v^j w^k,$$

- $\delta_{ii} = 3$,
- $\varepsilon_{ijk}\varepsilon_{imn} = \delta_{jm}\delta kn - \delta_{jn}\delta_{km}$,
- $\varepsilon_{ijk}\varepsilon_{ijn} = 2\delta_{kn}$,
- $\varepsilon_{ijk}\varepsilon_{ijk} = 6$.

1.1.3 Maxwell's Equations

We define two fields, the electric field $\vec{E}(\vec{x}, t)$ and the magnetic field $\vec{B}(\vec{x}, t)$. For these fields, we assume the following equations to be true:

$$\begin{aligned} \vec{\nabla} \cdot \vec{E} &= \frac{\rho}{\varepsilon_0} & \vec{\nabla} \times \vec{E} &= -\partial_t \vec{B} \\ \vec{\nabla} \cdot \vec{B} &= 0 & \vec{\nabla} \times \vec{B} &= \varepsilon_0 \mu_0 \partial_t \vec{E} + \mu_0 \vec{J} \end{aligned}$$

These are called MAXWELL'S EQUATIONS. Along with the aforementioned $\vec{E}$ and $\vec{B}$, they also contain the following fields;

- the electric charge density ρ is a scalar field,
- the electric current density $\vec{J}$ is a vector field,

and two constants, the PERMITTIVITY OF FREE SPACE ε_0 and the PERMEABILITY OF FREE SPACE μ_0.

1.1.4 Charge and current

Particles have properties, one of which is the electric charge $q \in \mathbb{R}$. Usually, the electric charge is an integer multiple of the basic charge e, but fractional values are also possible for quarks. We can define the DENSITY OF CHARGE ρ as the amount of charge in a given volume, which we think of as a scalar field. The total charge of the universe

$$Q = \iiint_{\mathbb{R}^3} \rho dV$$

is constant in time by the law of the preservation of charge.

We can also define the CURRENT DENSITY $\vec{J}$ as a vector field signifying the flow of current though some area, and the TOTAL CURRENT through a surface as

$$I = \iint_S \vec{J} \cdot d\vec{S}.$$

Finally, there is a force exerted on every charged particle, equal to

$$\vec{F} = q(\vec{E} + \partial_t \vec{r} \times \vec{B}).$$

1.1.5 Electrostatics

By definition, electrostatics is the conditions in which $\vec{J} = 0$ and $\rho \neq 0$. In this case, and additionally setting $\vec{B} = 0$, Maxwell's equations reduce to

$$\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\varepsilon_0},$$
$$\vec{\nabla} \times \vec{E} = 0.$$

Integrating Gauss' law gives

$$\frac{Q}{\varepsilon_0} = \iiint_V \vec{\nabla} \cdot \vec{E} dV = \iint_{\partial V} \vec{E} \cdot d\vec{s}.$$

Since we must satisfy $\vec{\nabla} \times \vec{E} = 0$, it is useful to introduce the electrostatic potential $\vec{E} = -\vec{\nabla}.\phi$. This reduces Gauss' law to $\Delta\phi = -\frac{\rho}{\varepsilon_0}$. Note that ϕ is only defined up to an additive constant. We call this concept GAUGE INVARIANCE.

For a general distribution of point charges, we can find ϕ using the method of Green's functions. The Green's function $G(\vec{r}, \vec{r}')$ is the function which solves $\Delta G(\vec{r}, \vec{r}') = \delta(\vec{r} - \vec{r}')$. For Poisson's equation, we then have

$$\Delta\phi(\vec{r}) = -\frac{1}{\varepsilon_0} \iiint_V \rho(\vec{r}')\delta(\vec{r} - \vec{r}')d\vec{r}' = \Delta_{\vec{r}} \left(-\frac{1}{\varepsilon_0} \iiint_V \rho(\vec{r}')G(\vec{r}, \vec{r}')d\vec{r}' \right)$$

so

$$\phi(\vec{r}) = -\frac{1}{\varepsilon_0} \iiint_V \rho(\vec{r}\,')G(\vec{r},\vec{r}\,')d\vec{r}\,'.$$

We have

$$G(\vec{r},\vec{r}\,') = -\frac{1}{4\pi}\frac{1}{|\vec{r}-\vec{r}\,'|}$$

which gives

$$\phi(\vec{r}) = \frac{1}{4\pi\varepsilon_0} \iiint_V \frac{\rho(\vec{r}\,')}{|\vec{r}-\vec{r}\,'|}d\vec{r}\,'$$

and

$$\vec{E} = \frac{1}{4\pi\varepsilon_0} \iiint_V \rho(\vec{r}\,')\frac{\vec{r}-\vec{r}\,'}{|\vec{r}-\vec{r}\,'|^3}d\vec{r}\,'$$

Take a particle with a charge q in ϕ. The potential energy is the work required to bring this particle from ∞ to $\vec{r}$,

$$U(\vec{r}) = -\int_\infty^r q\vec{E}\cdot d\vec{r} = q\phi(\vec{r}).$$

For more than one charge, we can compute

$$U = \frac{1}{2}\iiint_V \rho(\vec{r})\phi(\vec{r})d\vec{r} = \frac{\varepsilon_0}{2}\iiint_V \vec{\nabla}\cdot\vec{E}\phi d\vec{r} = \frac{\varepsilon_0}{2}\iiint_V \left(\vec{\nabla}\cdot(\vec{E}\phi) - \vec{E}\cdot\vec{\nabla}.\phi\right)d\vec{r}.$$

Setting $\phi(\infty) = 0$ then gives

$$U = \frac{\varepsilon_0}{2}\iiint_V \vec{E}\cdot\vec{E}d\vec{r}.$$

2 Teorija grafov

2.1 Matchings

Definition 2.1.1. A vertex set $S \subseteq V$ is an INDEPENDENT SET of the graph $G = (V, E)$ if the induced subgraph $G[S]$ is empty. The maximum cardinality of an independent set is the INDEPENDENCE NUMBER $\alpha(G)$.

Definition 2.1.2. A vertex set $T \subseteq V$ is a VERTEX COVER if every edge has at least one of its endings in T. The maximum cardinality of a vertex cover is the VERTEX COVER NUMBER $\beta(G)$.

Definition 2.1.3. An edge set $M \subseteq E$ is a MATCHING if for every distinct $e_1, e_2 \in M$, edges e_1 and e_2 have no common ending. The maximum cardinality of a matching is the MATCHING NUMBER $\alpha'(G)$.

Definition 2.1.4. An edge set $C \subseteq E$ is an EDGE COVER if every vertex of G is covered by at least one edge from C. If the minimum degree $\delta(G)$ is at least 1, we can define the EDGE COVER NUMBER $\beta'(G)$ as the minimum cardinality of an edge cover.

Question 1. Define the independence number, the vertex and edge cover number, and the matching number.

Remark. The complement of an independent set is a vertex cover, so $\alpha(G)+\beta(G) = n(G)$ in every graph G. In a maximum matching, every edge must be covered by different vertices, so $\alpha'(G) \leq \beta(G)$. We can similarly argue that $\alpha'(G) \leq {}^{n(G)}/_{2} \leq \beta'(G)$ and $\alpha(G) \leq \beta'(G)$.

Theorem 2.1.5 (Gallai)**.** *If $\delta(G) \geq 1$, then $\alpha'(G) + \beta'(G) = n(G)$.*

Proof. Take a maximum matching M in G and let $V(M)$ be the vertices covered by M. For every vertex not covered by M, we can take an incident edge and add it to M. This gives an edge cover with

$$|M| + \left|\overline{V(M)}\right| = |M| + (n - 2\,|M|) = n - |M|$$

edges. Since $|M| = \alpha'(G)$, this implies $\beta'(G) + \alpha'(G) \leq n(G)$.

Now take a minimum edge cover C. We claim that for every edge in C, at least one end is covered only once by C. For suppose that $uv \in C$ is an edge and both u and v are covered by other edges in C. If we remove uv, then $C \setminus \{uv\}$ is a smaller cover, which is a contradiction.

The induced subgraph $G[C]$ is then a forest of stars. Suppose it consists of k components. We get $|C| = n - k$, since in a tree, the number of vertices is 1 more than the number of edges. A matching is obtained by choosing one edge from every star, which gives $\alpha'(G) + \beta'(G) \geq n(g)$, thus completing the proof. ☐

Question 2. State and prove Gallai's theorem.

Definition 2.1.6. Let M be a matching. A path $v_1 v_2 \dots v_k$ is an M-ALTERNATING PATH if the edges along the path alternate between M and $\overline{M}$. An M-alternating path is M-AUGMENTING if neither end of the path is covered by M.

Remark. Such a path cannot start or end with an edge from M, and the endpoints cannot be part of an edge in M.

Proposition 2.1.7. *If G is a graph, M is a matching and there exists an M-augmenting path P, then M is not a maximum matching.*

Proof. Suppose $P = v_1 \dots v_k$. We know the first and last edge are not in M, so $\left|E(P) \cap \overline{M}\right| = |E(P) \cap M| + 1$. Now let $M' = M \oplus E(P)$ be the symmetric difference of M and $E(P)$. This is clearly a matching. We know that $|M'| = |M| + 1$, so M cannot be maximum. □

Question 3. How can you construct a larger matching from an augmenting path?

Definition 2.1.8. A KÖNIG-EGERVÁRY graph is a graph G with $\alpha'(G) = \beta(G)$.

Theorem 2.1.9 (König)**.** *Let G be a bipartite graph. Then $\alpha'(G) = \beta(G)$. Additionally, if M is a matching in G and there is no M-augmenting path, M is a maximum matching.*

Proof. Let the partite classes of G be A and B. Suppose that M is a matching for which there is no M-augmenting path in G. Define X as the set of all vertices in A that are not covered by the matching, and Y the vertices in B not covered by M. Additionally, let B_1 be the set of vertices in B that can be reached by an M-alternating path from X, and similarly, let A_1 be the set of vertices of A which can be reached from X by an M-alternating path. Finally, define $B_2 = B \setminus (B_1 \cup Y)$ and $A_2 = A \setminus (A_1 \cup X)$.

Observe that on an M-alternating path from X, any edge from A to B is in $\overline{M}$ and any edge from B to A is in M. Out matching provides a one-to-one mapping between A_1 and B_1 and between A_2 and B_2, so $|A_1| = |B_1|$ and $|A_2| = |B_2|$. We also know that $|A_1| + |A_2| = |M|$. Now consider the possible edges between the defined vertex sets.

There is no edge between X and Y, since that would be a (trivial) M-augmenting path. There are also no edges between X and B_2, since an edge xb is an M-alternating path, implying $b \in B_1$. So the only edges from X lead to B_1.

There are also no edges between A_1 and Y, because we can construct an M-augmenting path with such an edge. If $a \in A_1$, then there is an alternating path from X to a, which we could extend with a a-to-Y edge to get an augmenting path. Finally, there are no edges between A_1 and B_2, since that would give an alternating path from X to the B_2-vertex as before.

Then $T = B_1 \cup A_2$ is a vertex cover with $|T| = |B_1| + |A_2| = |A_1| + |A_2| = |M|$. We have thus constructed a vertex cover with $|M|$ vertices, giving

$$\beta(G) \leq |T| = |M| \leq \alpha'(G).$$

The other inequality holds in the general case, so this completes the proof. □

Question 4. State and prove König's theorem.

Corollary 2.1.10. *If G is a bipartite graph, then $\alpha(G) = \beta'(G)$.*

Definition 2.1.11. Let G be a bipartite graph with partite classes A and B. HALL'S CONDITION holds for the set A if for every $S \subseteq A$,

$$|S| \leq |N(S)| = \left| \bigcup_{u \in S} N(u) \right|.$$

Theorem 2.1.12 (Hall)**.** *If G is a bipartite graph with partite classes A and B, then there exists a matching that covers A if and only if Hall's condition holds for A.*

Proof. Let M be a matching covering A and $S \subseteq A$. We can take the pairs matched by M

$$B_S = \{v \in B \mid v \text{ is covered by an edge in } M\}.$$

Clearly, $|S| = |B_S|$ and $B_S \subseteq N(S)$, so $|S| \leq |N(S)|$.

For the other implication, suppose there is no matching covering A. Divide the sets A and B as in the proof of König's theorem, using some matching M. Since the matching doesn't cover A, X is not empty. Now consider $S = A_1 \cup X$. All edges from S lead into B_1, so $N(S) = B_1$, but $|S| = |A_1| + |X| > |B_1| = |N(S)|$. □

Question 5. State and prove Hall's theorem.

Definition 2.1.13. A matching M is a PERFECT MATCHING if it covers all vertices.

Corollary 2.1.14. *In a bipartite graph G, there is a perfect matching if and only if $|A| = |B|$ and A satisfies Hall's condition.*

Definition 2.1.15. Let G be a bipartite graph with partite classes A, B, and $S \subseteq A$. The DEFICIENCY of S is $\operatorname{def}(S) = |S| - |N(S)|$.

Theorem 2.1.16. *Let G be a bipartite graph with partite classes A and B. If M is a maximum matching in G, it covers*

$$\alpha'(G) = |A| - \max_{S \subseteq A} \operatorname{def}(S)$$

vertices of A.

Theorem 2.1.17. *If G is a regular bipartite graph, then G has a perfect matching.*

Proof. The number of edges in the graph is $k \cdot |A| = k \cdot |B|$, so $|A| = |B|$. Let $S \subseteq A$. The number of edges between S and $N(S)$ is exactly $k \cdot |S|$. Every neighbour $u \in N(S)$ has exactly k neighbours, at most k are in S. So at most $k \cdot |N(S)|$ edges are between S and $N(S)$, implying $|S| \leq |N(S)|$. □

Question 6. Show that a regular bipartite graph has a perfect matching.

Theorem 2.1.18. *Let M be a matching in G. Then there is an M-augmenting path in G if and only if M is not a maximum matching in G.*

Proof. We've already proved the right implication. Suppose there is a matching M' with $|M'| > |M|$. Consider the symmetric difference $M \triangle M'$ and denote $G' = G[M \triangle M']$. Clearly the maximum degree $\Delta(G') \leq 2$, from which we know that the components of G' are all paths or cycles.

Any cycle must alternate between edges from M and M', so it is an even cycle, and contains the same number of edges from the two matchings. In any path of even length, there must be the same number of edges in M and in M'. And finally, in a path of odd length, one of the two sets has an extra edge compared to the other. Since $|M'| > |M|$, there must be a path with more edges in M' than in M. Label it G'_1.

This component is an M-augmenting path in G, since if either of its endpoints are covered by M, they must also be covered by the same edge in M', but then M' wouldn't be a matching. $\square$

We can find the maximum matching in polynomial time, with so-called "Blossom algorithms", which find an augmenting path in $O(m\sqrt{n})$. This also means we can determine the edge-cover number $\beta'(G)$ in polynomial time.

2.1.1 Tutte's theorem

Definition 2.1.19. A component of a graph G is ODD if it has an odd number of vertices. We denote the number of odd components in G with $o(G)$.

Theorem 2.1.20 (Tutte). *A graph G has a perfect matching if and only if for any $S \subseteq V(G)$, $|S| \geq o(G - S)$ holds. This is called Tutte's condition.*

Proof. Left to right: Let $S \subseteq V(G)$ and M be a perfect matching in G. Let $H_1, \dots, H_k$ be the components of $G - S$. If H_i is an odd component, then there exists at least one M-edge between $V(H_i)$ and S. Therefore $|S| \geq o(G - S)$.

Right to left: Suppose that Tutte's condition holds for a graph F but there is no perfect matching in F. Now add edges to F so that this still holds after the addition, and let G be a maximal such graph on $n(F)$ vertices. If we consider Tutte's condition on $S = \varnothing$, we see that there are no odd components in G, which means $n(G)$ is even.

Now take any edge in the complement of G. Since G is maximal, $G + e$ is not a counterexample, so it either has a perfect matching, or Tutte's condition does not hold for it. We know that for any $S \subseteq V(G)$, $|S| \geq o(G - S)$. After having added an edge, at most one pair of components in $G - S$ is joined. If this was a pair of odd components, $o(G - S)$ has reduced, and in all other cases, it has remained the same. This means that

$G + e$ must satisfy Tutte's condition, so it must have a perfect matching as it is not a counterexample.

We will consider two cases. Let U be the set of universal vertices in G, that is, the vertices adjacent to every other vertex, and let $H_1, \dots, H_k$ be the components of $G - U$. In the first case, suppose every H_i induces a complete graph. We will construct a perfect matching. For the even components, we may create a matching within the component, and for the odd components, we may connect the one remaining vertex with U. We are left over with an even number of vertices in U (since n is even), which we may form a matching with, as $G[U]$ is a complete graph. So in this case, G is not a counterexample, which is a contradiction.

Now suppose there is a non-complete component H_i. Then there exist vertices x, y, z for which $xy \notin E(H_1)$ but both xz and yz are in $E(H_1)$. There is also another vertex $w \in V(G)$ for which $zw \notin E(G)$, since z is not a universal vertex. Let $G_1 = G + xy$ and $G_2 = G + zw$. We know both these graphs have perfect matchings, which must include xy and zw respectively. Denote the perfect matchings with M_1 and M_2 and consider $G[M_1 \triangle M_2]$. Note that every non-isolated vertex in this graph is of degree 2, since it is covered by both perfect matchings. These vertices must of course appear in even cycles.

If xy and zw belong to different components, then we may choose edges in each component, and the edges deleted by the symmetric difference, to form a perfect matching, and we can avoid taking xy or zw. This is a contradiction. Alternatively, if xy and zw belong to the same component, they appear in the same cycle. Without loss of generality they appear in the order $zwxy$ (but not necessarily adjacent). To form a new perfect matching, we will take the edge xz, and one edge set from each side of the cycle. This avoids both new edges xy and zw, so we have a contradiction. $\square$

We also have the Berge-Tutte formula, which states that a maximum matching in G leaves uncovered exactly

$$\max_{S \subseteq V(G)} \{o(G - S) - |S|\}$$

vertices.

3 Teorija izračunljivosti

3.1 Introduction

A TURING MACHINE is defined to consist of the following components. There is an infinite tape divided into cells, each of which contains a symbol from the chosen alphabet Γ. This alphabet must include a `blank` symbol. At the start, only a finite number of cells in the tape have a character different than `blank`. The machine also possesses a read-write head positioned at some cell, and an internal control state, which determines the instruction to be followed.

Instructions are given as a TRANSITION (partial) function f, which maps

$$(\text{state}, \text{character}) \mapsto (\text{new state}, \text{new character}, \text{motion}).$$

To perform an action, a TM will look for rules matching its current control state and the character currently written at the position of the read-write head. When a matching rule is found, the machine switches to the defined new state, writes the specified character on the tape and moves according to the instruction. We limit its motion to three possibilities: one cell to the left or right, or no motion at all. More formally, we may restate the definition as follows:

Definition 3.1.1. A TURING MACHINE is specified by the following:

- a finite TAPE ALPHABET Γ with $\boxdot \in \Gamma$,
- a finite set Q of STATES with $\texttt{start} \in Q$,
- a transition partial function

$$\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{-1, 0, +1\}.$$

For a given input alphabet $\Sigma_1 \subseteq \Gamma$ and output alphabet $\Sigma_2 \subseteq \Gamma$, a TM specifies a partial function $f : \Sigma_1^* \rightharpoonup \Sigma_2^*$ if for any $w \in \Sigma_1^*$, running the TM on the input w results in the machine halting in the `halt` state and it has $f(w)$ as the word on the tape, with the head at the leftmost character. We also require $\boxdot \notin \Sigma_1$ or Σ_2. If the machine does not halt in the `halt` state, we say $f(w)$ is not defined.

Definition 3.1.2. A partial function $f : \Sigma_1^* \rightharpoonup \Sigma_2^*$ is COMPUTABLE if there exists a TM that computes it.

Question 1. Describe the basic working of a Turing machine. What does it mean for a function to be computable?

Definition 3.1.3. A LANGUAGE over an alphabet Σ is a subset $L \subseteq \Sigma^*$.

For language recognition we require a subset $\Sigma \subseteq \Gamma$ (the INPUT ALPHABET) and two distinguished states, `accept` and `reject`. A language-recognizing Turing machine accepts a word w if, when the machine is run on the input w, the computation halts in the `accept` state. Similarly, w is rejected if the machine halts in the `reject` state. We say that a Turing machine M DECIDES or COMPUTES L if for any $w \in \Sigma^*$, $w \in L$ implies that M accepts w and $w \notin L$ implies that M rejects w.

If M correctly accepts words of the language, and does not accept words that are not part of the language, we say that M SEMIDECIDES, SEMICOMPUTES or RECOGNIZES L. A language is DECIDABLE or COMPUTABLE if there is a TM which decides it, and SEMIDECIDABLE, SEMICOMPUTABLE or COMPUTABLY INNUMERABLE if there is a TM which recognizes it. Clearly, every decidable language is also semidecidable.

Question 2. What is a decidable and what is a semidecidable language?

Given a k-tape machine (Γ, Q, δ), we can simulate it by a single-tape machine. We will encode the different tapes by separating them with a special symbol $| \in \tilde{\Gamma}$ and encoding the positions of the read-write heads with another special symbol $\Delta \in \tilde{\Gamma}$. When simulating a computational step of the multi-tape machine, we scan for transitions and implement them manually.

Question 3. How can you simulate a multi-tape Turing machine with a single-tape machine?

Proposition 3.1.4. *There are languages that are not semidecidable.*

Proof. There are only countably many non-equivalent Turing machines, but an uncountable number of languages on any alphabet. □

We can also give an example of a language that is semidecidable but not decidable. To construct it, we will encode a Turing machine $M = (\Gamma, Q, \delta)$ into a string. We encode it in $\langle M \rangle \in \Sigma_u^*$ for the alphabet

$$\Sigma_u = \{0, 1, -1, [,], \|, \cdot\}.$$

We encode every state $q \in Q$ as a word $\langle q \rangle \in \{0, 1, -1\}^l$, where $l \geq \log_3 |Q|$. We require that the encoding of the start state is 0^l, the encoding of the accepting state is 1^l, and the encoding of the rejecting state is $(-1)^l$. Every symbol $a \in \Gamma$ is encoded as $\langle a \rangle \in \{-1, 0, 1\}^m$ for $m \geq \log_3 |\Gamma|$. We require that the encoding of the blank symbol is 0^m.

Finally, to encode δ, consider an instruction $(q, a) \mapsto (q', b, d)$. This will be encoded as a word

$$[\langle q \rangle \cdot \langle a \rangle \parallel \langle q' \rangle \cdot \langle b \rangle \cdot d]$$

with $d \in \{0, 1, -1\}$. This encoding has length $2l + 2m + 7$, and allows us to encode the full Turing machine as the encoding of the start state, followed by a dot $\cdot$, followed by the encoding of the blank symbol, and then followed by the encodings of all transitions one after another. We can encode any word $w \in \Gamma^*$ as a sequence of characters, delimited by $\cdot$, so that we may define a language L_{accept} as follows: the language includes words of the form $\langle M \rangle \cdot \langle w \rangle$, where M is a single-tape Turing machine with tape alphabet $\Gamma \supseteq \Sigma_u$, and $w \in \Sigma_u^*$ is a word which M accepts.

Question 4. How is L_{accept} defined? Describe the universal encoding of a Turing machine.

Theorem 3.1.5. *The language L_{accept} is undecidable.*

Proof. Suppose that it is decidable, so there exists a Turing machine D which decides it. We define a new Turing machine N with input alphabet Σ_u. This machine reads its input string v and converts it to the string $v \cdot \langle v \rangle$. It then proceeds as D on this input, except it switches the accept and reject states.

Consider what N does when given an input of the form $v = \langle M \rangle$ for some Turing machine M. If D rejects $\langle M \rangle \cdot \langle \langle M \rangle \rangle$, then N terminates on the accepting state, and vice versa. Because D decides L_{accept}, we get from N:

- `accept` iff M does not accept $\langle M \rangle$, and
- `reject` iff M accepts $\langle M \rangle$.

Now run N on $\langle N \rangle$. We get `accept` iff N does not accept $\langle N \rangle$, and `reject` iff N accepts $\langle N \rangle$. This is a contradiction. □

Question 5. Show that L_{accept} is undecidable.

Using Σ_u, we can also construct the UNIVERSAL TURING MACHINE.

Theorem 3.1.6. *There exists a Turing machine U over the tape alphabet $\Sigma_u \cup \{\boxdot\}$ that exhibits the following behavior: If we run U on the string $\langle M \rangle \cdot \langle w \rangle$, then the resulting execution satisfies the following.*

- *It terminates if and only if M terminates on input w.*
- *It accepts if and only if M accepts w.*
- *It rejects if and only if M rejects w.*

The idea of the proof is to use a three-tape machine, putting the encoding of M on the first tape, the encoding of the state on the second, and the input on the third. Then simulate the execution of M.

Theorem 3.1.7. *The language L_{accept} is semidecidable.*

Proof. We construct a machine S that does the following. It first reads its input word $v \in \Sigma_u^*$ and checks whether v is of the form $\langle M \rangle \cdot \langle w \rangle$. If v is not of this form, then we reject immediately. Otherwise, we run the universal machine U on the input v and end in the end state of U. □

Question 6. Show that L_{accept} is semidecidable.

Proposition 3.1.8. *If $f : \Sigma_1^* \rightharpoonup \Sigma_2^*$ and $g : \Sigma_2^* \rightharpoonup \Sigma_3^*$ are computable, then so if $g \circ f$.*

Note that the composite of two partial functions is defined as

$$(g \circ f)(w) \simeq \begin{cases} g(f(w)) & f(w) \downarrow \\ \uparrow & f(w) \uparrow \end{cases}$$

Definition 3.1.9. A k-tape Turing machine COMPUTES $f : \Sigma_1^* \times \cdots \times \Sigma_k^* \rightharpoonup \Sigma^*$ if when we run the Turing machine on the configuration with a word on each tape, the machine terminates in the halt state if and only if for all $(w_1, \ldots, w_k)$ in the domain of f, it halts in the configuration with $f(w_1, \ldots, w_k)$ on the first tape and all other tapes blank.

Example. To define the computability for partial functions $f : \mathbb{N} \rightharpoonup \mathbb{N}$, we can represent numbers using words over $\Sigma_b = \{0, 1\}$ and the representation

$$\gamma_{\mathbb{N}}(w) = \sum_{i=0}^{|w|-1} 2^{|w|-i-1} w_i.$$

If we allow for leading zeros, every number is represented by an infinite number of words. Additionally, zero is also represented by the empty word ε. A Turing machine M computes $f : \mathbb{N} \rightharpoonup \mathbb{N}$ if it computes $g : \Sigma_b^* \rightharpoonup \Sigma_b^*$ such that for all words w for which $\gamma_{\mathbb{N}}(g(w)) \simeq f(\gamma_{\mathbb{N}}(w))$. We say that f is COMPUTABLE if there is a Turing machine which computes it.

We could also restrict our representation, for example requiring words to begin with a 1 (we also allow the empty word). Alternatively, we could use e.g. a unary representation.

Definition 3.1.10. A REPRESENTATION of a set X by words over an alphabet Σ is a surjective partial function $\gamma : \Sigma^* \rightharpoonup X$.

Remark. Only countable sets can be represented.

Definition 3.1.11. Given representations $\gamma_1 : \Sigma_1^* \rightharpoonup X_1$ and $\gamma_2 : \Sigma_2^* \rightharpoonup X_2$, a partial function $f : X_1 \rightharpoonup X_2$ is $(\gamma_1 \to \gamma_2)$-COMPUTABLE if there exists a computable partial function $g : \Sigma_1^* \rightharpoonup \Sigma_2^*$ such that for all words w in the domain of γ_1, if $g(w)$ is defined, then $\gamma_2(g(w)) \simeq f(\gamma_1(w))$.

Definition 3.1.12. Two representations γ_1 and γ_2 of the same set X are EQUIVALENT if the identity function id_X is $(\gamma_1 \to \gamma_2)$-computable and $(\gamma_2 \to \gamma_1)$-computable.

Given representations $(\gamma_i : \Sigma_i^* \rightharpoonup X_i)_{i=1,\ldots,k}$, we construct a product representation $\gamma : \Sigma^* \rightharpoonup X_1 \times \cdots \times X_k$, where

$$\Sigma = (\Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_k) \amalg \{,\}$$

and

$$\gamma(w_1, \ldots, w_k) = (\gamma_1(w_1), \ldots, \gamma_k(w_k)).$$

Definition 3.1.13. A partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is COMPUTABLE if it is $(\gamma_{\mathbb{N}} \times \cdots \times \gamma_{\mathbb{N}} \to \gamma_{\mathbb{N}})$-computable.

Definition 3.1.14. Given a representation $\gamma : \Sigma^* \rightharpoonup X$, a subset $A \subseteq X$ is γ-DECIDABLE if there exists a Turing machine M such that for all w in the domain of γ, if $\gamma(w) \in A$, then M accepts w, and if $\gamma(w) \notin A$, then M rejects w. The same subset is γ-SEMIDECIDABLE if there exists a Turing machine M such that for all w in the domain of γ, M accepts w if and only if $\gamma(w) \in A$.

Proposition 3.1.15. *Given a representation γ of X and $A \subseteq X$, A is γ-decidable if and only if its characteristic function is $(\gamma \to \gamma_b)$-computable, and A is γ-semidecidable if and only if its partial characteristic function is $(\gamma \to \gamma_b)$-computable.*

Remark. Above, γ_b is the representation of $\{0,1\}$ which maps $0 \mapsto 0$ and $1 \mapsto 1$.

3.2 Computability of natural numbers

Definition 3.2.1. The COMPUTABLE PARTIAL FUNCTIONS is the set

$$\{f : \mathbb{N}^k \rightharpoonup \mathbb{N} \mid k \geq 0, f \text{ is computable}\}.$$

Definition 3.2.2. The PRIMITIVE RECURSIVE FUNCTIONS are the smallest collection $\mathcal{F} \subseteq \{\mathbb{N}^k \rightharpoonup \mathbb{N} \mid k \geq 0\}$ of partial functions that satisfies the following properties:

- $\mathcal{F}$ contains the zero function $Z : \{()\} \to \mathbb{N}$, which maps $Z() = 0$, the successor function $S : \mathbb{N} \to \mathbb{N}$, which maps $x \mapsto x+1$ and the projection functions: for any $k \geq 1$ and $1 \leq i \leq k$, $U_i^k : \mathbb{N}^k \to \mathbb{N}$ is defined as

$$U_i^k(x_1, \ldots, x_k) = x_i.$$

- $\mathcal{F}$ is closed under composition: if $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ and $g_1, \ldots, g_k : \mathbb{N}^l \rightharpoonup \mathbb{N}$ are in $\mathcal{F}$, then $f \circ (g_1, \ldots, g_k)$ is in $\mathcal{F}$.
- Primitive recursion: If $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ and $g : \mathbb{N}^{k+2} \rightharpoonup \mathbb{N}$ are both in $\mathcal{F}$, then so is $R_{fg} : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$, defined with $R_{rg}(x_1, \ldots, x_k, 0) \simeq f(x_1, \ldots, x_k)$ and $R_{fg}(x_1, \ldots, x_k, x+1) \simeq g(x_1, \ldots, x_k, x, R_{fg}(x_1, \ldots, x_k, x))$.

Remark. Since all basic functions are total, every function in $\mathcal{F}$ is total.

Remark. Not every total computable function is primitive recursive. We can show for example that the Ackermann function grows faster than any primitive recursive function.

Definition 3.2.3. The PARTIAL RECURSIVE FUNCTIONS are the smallest collection of partial functions $\mathcal{F} \subseteq \{\mathbb{N}^k \rightharpoonup \mathbb{N}\}_{k \geq 0}$, which satisfies the axioms of partial recursive functions, with an additional one:

- Minimization: If $f : \mathbb{N}^{k+1} \rightharpoonup \mathbb{N}$ is in $\mathcal{F}$, then so is $\mu f : \mathbb{N}^k \rightharpoonup \mathbb{N}$, with $\mu f(x_1, \ldots, x_k)$ equal to the smallest number $n \in \mathbb{N}$ such that $f(x_1, \ldots, x_k, n) = 0$ if it exists and $f(x_1, \ldots, x_k, m)$ is defined for all $m < n$, and $\mu f(x_1, \ldots, x_k)$ undefined otherwise.

Proposition 3.2.4. *A partial function $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ is computable if and only if there exists a $(k+1)$-tape Turing machine such that for all $x_1, \ldots, x_k \in \mathbb{N}$ and binary words $w_1, \ldots, w_k$ of $x_1, \ldots, x_k$, if we run the Turing machine with $w_1, \ldots, w_k$ on the first k tapes, then it halts if and only if $f(x_1, \ldots, x_k)$ is defined and it halts with the representation of $f(x_1, \ldots, x_k)$ on the last tape, and $w_1, \ldots, w_k$ on the first k tapes.*

Theorem 3.2.5. *The partial recursive functions coincide with the computable partial functions.*

Proof. A partial recursive function is computable: We will show that the family of computable functions satisfies the properties of partial computable functions. Since the partial recursive functions are the smallest such family, every partial recursive function is computable.

Clearly, computable partial functions satisfy composition and include the basic functions. Let's show they are closed under primitive recursion. Suppose we have a $(k+1)$-tape Turing machine M_f computing f and a $(k+3)$-tape Turing machine M_g computing g. We will find a $(k+4)$-tape Turing machine computing R_{fg}, which can then be compressed. On the first $k+1$ tapes, put the arguments to R_{fg}. On the $(k+2)$-th tape, put a counter. On the next tape, put the result of R_{fg} on the current iteration, and finally, on the last tape, put the result being computed. The Turing machine then proceeds as follows:

1. initialize tape $k+2$ to 0
2. use M_f to compute $f(x_1, \ldots, x_k)$ and write the result on tape $k+4$
3. if the numbers on tapes $k+1$ and $k+2$ are equal, halt
4. copy tape $k+4$ to tape $k+3$
5. apply M_g on tapes $1, \ldots, k, k+2, k+3$ and write the result on tape $k+4$
6. increment tape $k+2$
7. go to step 3

We may similarly construct a machine which performs minimization, which finishes this inclusion.

Before starting the proof of the other inclusion, consider the following. We can encode $\mathbb{N} \times \mathbb{N}$ using $\mathbb{N}$ with the bijection $p(x, y) = \frac{1}{2}(x+y)(x+y+1)+x$, which is also primitive recursive. The functions $q_1, q_2 : \mathbb{N} \to \mathbb{N}$ which reverse p (such that $q_1(p(x, y)) = x$ and $q_2(p(x, y)) = y$) are also primitive recursive.

We can also encode finite sequences with $\lceil \cdot \rceil : \mathbb{N}^* \to \mathbb{N}$, defined as $\lceil n_0, \ldots, n_{k-1} \rceil = 2^{n_0} + 2^{n_0+n_1+1} + \ldots + 2^{n_0+\ldots+n_{k-1}+k-1}$, so that numbers are encoded in a binary sequence with the number of zeros between a pair of ones denoting the sequence. This is clearly a bijection. Additionally, the functions $\sigma : \mathbb{N}^2 \to \mathbb{N}$ mapping

$$\sigma(\lceil w \rceil, i) = \begin{cases} 0 & i \geq |w| \\ w_i + 1 & i < |w| \end{cases}$$

and $l : \mathbb{N} \to \mathbb{N}$ mapping

$$l(\lceil w \rceil) = |w|$$

are primitive recursive.

Now we can prove the other inclusion. Suppose $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ is computable and that it is computed by a Turing machine M. We assume M is a single-tape machine computing f via representations. Suppose M has a tape alphabet $\Gamma \supseteq \{\boxdot, 0, 1, ,\}$ and state set $Q \supseteq \{\text{start}, \text{halt}\}$. Choose injective functions $r : \Gamma \to \{\text{odd numbers}\}$ and $s : Q \to \{\text{even numbers}\}$. Now suppose we are in a configuration C with a finitely many not necessarily blank symbols $a_0, \dots, a_{k-1}$, the tape head at a_i, and the current state equal to q. Define an encoding

$$\lceil C \rceil = \lceil r(a_0) \dots r(a_{i-1}) s(q) r(a_i) \dots r(a_{k-1}) \rceil.$$

The following functions are primitive recursive:

- step : $\mathbb{N} \to \mathbb{N}$, mapping $\lceil C \rceil$ to $\lceil C' \rceil$, which is the configuration we obtain by taking one step of M on configuration C. If the input of the function is not a valid configuration, it returns 0.
- run : $\mathbb{N}^2 \to \mathbb{N}$, mapping (n, x) to $\text{step}^n(x)$.
- extract : $\mathbb{N} \to \mathbb{N}$, mapping $\lceil s(\text{halt}) r(w_0) \dots r(w_k) \rceil$ to n if w is a binary representation of n, and any other input to 0.
- halt? : $\mathbb{N} \to \mathbb{N}$ mapping $\lceil s(\text{halt}) r(w_0) \dots r(w_{k+1}) \rceil$ to 0 and all other inputs to 1.
- init : $\mathbb{N}^k \to \mathbb{N}$, mapping $(x_1, \dots, x_k)$ to $\lceil s(\text{start}) y_1 \dots y_m \rceil$, where y_i is equal to the result of r on the i-th character of the sequence $(\text{bin}(x_1), \dots, \text{bin}(x_k))$.

Then, f is partial recursive because

$$f(x_1, \dots, x_n) \simeq \text{extract}(\text{run}(\mu(n \mapsto \text{halt?}(\text{run}(n, \text{init}(x_1, \dots, x_k)))), \text{init}(x_1, \dots, x_k)))$$

□

4 Uvod v funkcionalno analizo

4.1 Normirani in Banachovi prostori

Definicija 4.1.1. Naj bo X vektorski prostor nad poljem $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$. Preslikava $\|\cdot\| : X \to \mathbb{R}$ je NORMA, če velja:

- $\|x\| \geq 0$,
- $\|x\| = 0 \Leftrightarrow x = 0$,
- $\|\lambda x\| = |\lambda| \, \|x\|$,
- $\|x + y\| \leq \|x\| + \|y\|$.

Opomba. Velja $|\|x\| - \|y\|| \leq \|x - y\|$, iz česar sledi, da je norma zvezna (celo Lipschitzova za $L = 1$).

Norma porodi metriko $d(x, y) = \|x - y\|$ na prostoru X, ki je invariantna na translacije, in za katero velja

$$d(\lambda x, \lambda y) = |\lambda| \, d(x, y).$$

Zaprto kroglo radija r s središčem v točki x označimo z $B(x, r)$, odprto kroglo pa z $\mathring{B}(x, r)$. Zaradi zveznosti norme je zaprtje odprte krogle natanko pripadajoča zaprta krogla.

Definicija 4.1.2. Normiran prostor je BANACHOV, če je poln za inducirano metriko.

Trditev 4.1.3. *Seštevanje in množenje vektorjev s skalarjem sta zvezni operaciji.*

Definicija 4.1.4. Algebra A je NORMIRANA ALGEBRA, če je normiran vektorski prostor in če velja $\|xy\| \leq \|x\| \, \|y\|$. Če ima normirana algebra enoto, zahtevamo še $\|e\| = 1$.

Primer. Naj bo X Hausdorffov topološki prostor in $\mathcal{C}_b(X)$ množica zveznih omejenih funkcij $X \to \mathbb{F}$. Če jo opremimo s supremum normo, postane normirana algebra za seštevanje in množenje po točkah. Preverimo lahko, da je celo Banachov prostor.

Posledica 4.1.5. *Če je X kompakten Hausdorffov prostor, je $\mathcal{C}(X)$ Banachova algebra.*

Trditev 4.1.6. *Naj bo X normiran prostor in Y (vektorski) podprostor v X. Veljata naslednji točki.*

- *Če je Y poln, potem je Y zaprt v X.*
- *Če je X Banachov prostor, potem je Y Banachov natanko tedaj, ko je Y zaprt v X.*

Dokaz. Prva točka: Naj bo $y \in \overline{Y}$. Obstaja zaporedje $(y_n)_n$ v Y, ki konvergira k y. To zaporedje je Cauchyjevo, torej ima limito, ki je enaka y. Sledi $y \in Y$, zato $Y = \overline{Y}$.

Druga točka: V desno smo ravno dokazali. V levo naj bo $(y_n)_n$ Cauchyjevo zaporedje v Y. Potem je Cauchyjevo tudi v X, kjer ima limito, saj je X poln. Ker je Y zaprt, je $y \in Y$, torej $y_n \to y$ v Y. □

Primer. Naj bo X lokalno kompakten Hausdorffov prostor ter $\mathcal{C}_0(X)$ množica vseh funkcij v $\mathcal{C}(X)$, za katere za vsak $\varepsilon > 0$ obstaja kompakt $K_\varepsilon \subseteq X$, da je $|f|$ zunaj K_ε strogo manjša od ε.

Pri $X = \mathbb{R}$ so to natanko vse funkcije, katerih limita v obeh neskončnostih je enaka 0, pri $X = \mathbb{N}$ pa je to natanko prostor c_0 zaporedij, ki konvergirajo k 0.

Dokažemo lahko, da je $\mathcal{C}_0(X)$ zaprt dvostranski ideal v $\mathcal{C}_b(X)$ in zato Banachova algebra.

Primer. Množica c vseh konvergentnih zaporedij je Banachov prostor za supremum normo.

4.1.1 Napolnitve normiranih prostorov

5 Statistika 2

5.1 Ocenjevanje v linearnih modelih

Splošni linearni model je oblike $X = Z\beta + \varepsilon$, kjer je $Z \in \mathbb{R}^{n\times d}$ znana konstantna matrika, $\beta \in \mathbb{R}^d$ neznani parameter, ε pa je neopazljiv slučajni šum. Privzamemo, da velja $E(\varepsilon) = 0$. V splošnem za varianco ε ne privzamemo ničesar, v standardnih linearnih regresijskih modelih pa privzamemo, da je diagonalna.

Privzemimo splošni linearni model in naj bo B vektorski podprostor v $\mathbb{R}^d$. Naj bo $x \in \mathbb{R}^n$ realizacija slučajnega vektorja X. Restringirana ocena za $\beta \in B$ na podlagi x po metodi najmanjših kvadratov je tak vektor $\hat{\beta}_B$, za katerega je

$$\left\| x - Z\hat{\beta}_B \right\|^2 = \min_{b\in B} \|x - Zb\|^2 .$$

Pišimo $\hat{\beta} = \hat{\beta}_B$. Vemo, da je $Z\hat{\beta}$ ravno pravokotna projekcija vektorja x na podprostor $ZB \subseteq \mathbb{R}^n$. Določena je z zahtevo $x - Z\hat{\beta} \perp ZB$. Za $B = \mathbb{R}^d$ to velja natanko v primeru $Z^T(X - Z\hat{\beta}) = 0$, torej $Z^T Z\hat{\beta} = Z^T x$. V primeru, ko je Z polnega ranga, je $Z^T Z$ obrnljiva in $\hat{\beta} = (Z^T Z)^{-1} Z^T x$. Če pa je jedro Z netrivialno, imamo rešitev več.

Če na stolpcih Z izvedemo prirejeno Gram-Schmidtovo ortogonalizacijo, lahko kljub temu poiščemo rešitev. Označimo s S_i rezultat ortogonalizacije na i-tem stolpcu Z. V primeru, ko je ta stolpec v linearni ogrinjači prejšnjih, nastavimo $S_i = 0$. S tem dobimo ortogonalne vektorje $S_1, \ldots, S_d$. Dobimo razcep $Z = SP$, kjer je S matrika iz zloženih stolpcev S_i, P pa zgornje trikotna matrika s pozitivnimi števili na diagonali, in zato obrnljiva. Velja $Z^T Z = P^T JP$, kjer je J diagonalna matrika, na diagonali katere so kvadrati norm stolpcev S_i. S tem lahko definiramo posplošen inverz $(Z^T Z)' = P^{-1} JP^{-T}$.