## Konvexní množina (Convex set)

* Se řeší proto, že v konvexní množině vždy z bodu A do bodu B (z této množiny) najdu přímou cestu, která taky bude obsažena v této množině (každý bod této cesty)
* To je pravděpodobně důležité pro iterační algoritmy
* Obsah obrázku kresba, skica, Dětské kresby

  Popis byl vytvořen automatickyObsah obrázku skica, kresba, Perokresba, Dětské kresby

  Popis byl vytvořen automaticky
* Pokud v konvexní množině při nějaké iteraci/hledání optimálního řešení vyrazím nějakým směrem a ,,narazím do zdi“, mám nejlepší řešení
* U nekonvexní množiny (viz obr. vpravo) se mi toto stát nemusí
* Konvexní množiny lze ohraničit pomocí série poloprostorů (half space), nekonvexní ne
* Obsah obrázku rukopis, text, inkoust, Písmo

  Popis byl vytvořen automaticky

## Cones

Obsah obrázku rukopis, text, řada/pruh, Písmo

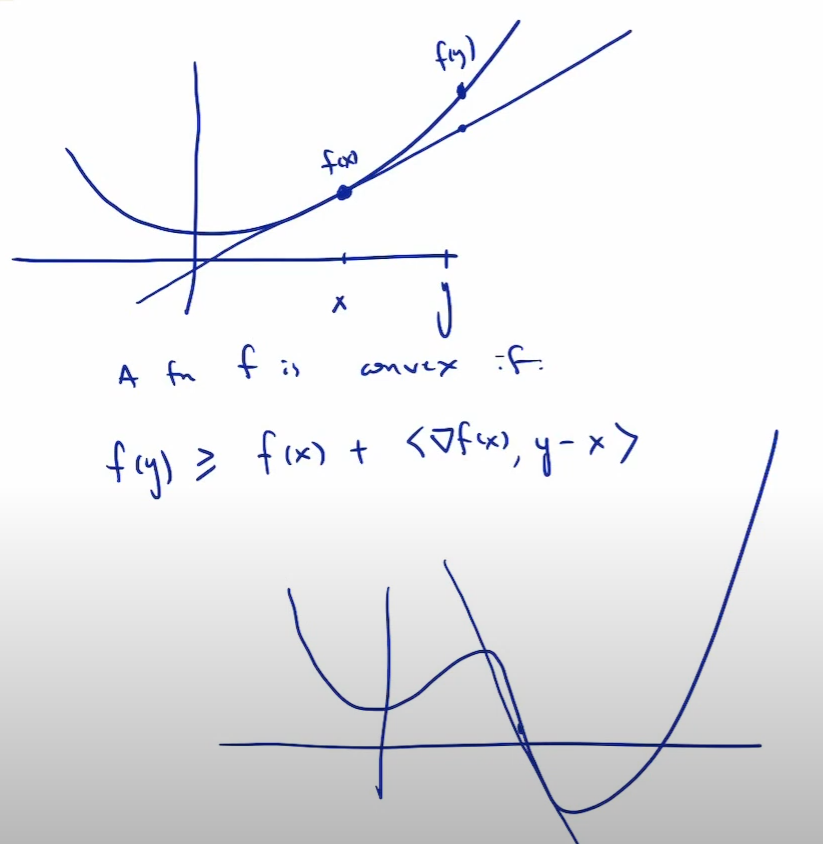
Popis byl vytvořen automaticky

* Zatím netušim, k čemu to je a proč se o tom mluví
* Možná se o tangent cone mluví proto, že v něm konverguju k jednomu řešení?
* Obsah obrázku rukopis, řada/pruh, diagram, skica

  Popis byl vytvořen automaticky

## Konvexní funkce (Convex function)

* Pokud v konvexní funkci spojím 2 body úsečkou, bude tato úsečka vždy ležet nad hodnotami této funkce
* Obsah obrázku text, rukopis, řada/pruh, diagram

  Popis byl vytvořen automaticky
* Lambda je způsob, jak definovat úsečku mezi 2 body
* Proč se konvexní funkce řeší?
  + **Konvexní funkce má gradient!**
* 
* Tangenta k funkci v libovolném bodě vždy leží ,,pod“ touto funkcí
  + U nekonvexní funkce to tak není
* **Konvexní funkce má i druhou derivaci!** (Hessian)

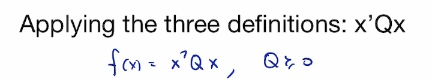
## Subgradient a subdiferenciál

* Obsah obrázku řada/pruh, skica, Dětské kresby

  Popis byl vytvořen automaticky
* Mám funkci, kde je bod, ve kterém derivace neexistuje 🡪 není tam jedna tangenta, ale je jich tam nekonečně mnoho možných
* Pro mě to pravděpodobně znamená, že chci dodefinovat i případy, kdy neexistuje derivace, ale funkce je spojitá
* Tzn. pravděpodobně si picknu jednu z těch nekonečně možností a jdu dál

Obsah obrázku text, rukopis, dopis, Písmo

Popis byl vytvořen automaticky



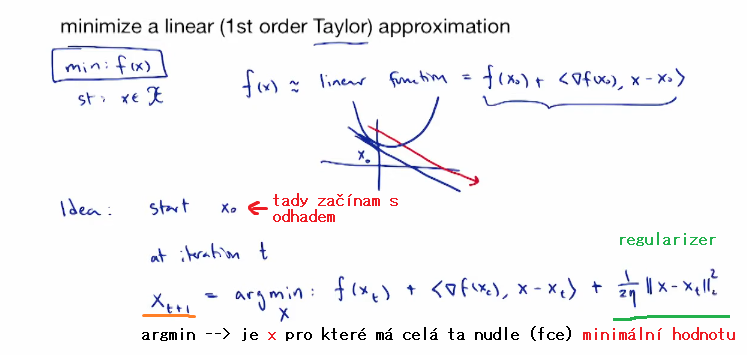
* Pak se z nějakého důvodu řeší tento problém
* Q je **pozitivně semidefinitn**í matice =
  + Je symetrická
  + Všechny vlastní čísla λ ≥ 0

## Gradient Descent

* Lze k němu přistupovat takto:
  + **Máme složitý problém, co kdybychom ho aproximovali jednodušším a ten řešili?**
* Chceme například řešit minimalizaci lineární funkce

Obsah obrázku text, Písmo, snímek obrazovky, rukopis

Popis byl vytvořen automaticky

* To je v tomhle případě těžké, protože funkce neustále klesá a minimum nemá
* Chtěli bychom tedy udělat iterativní algoritmus, který najde po N krocích ,,nějaké minimum“, takové, které si řeknem, že nám stačí
  + Tzn. v každém kroku minimalizujeme lineární aproximaci a přidáme si ošetřující podmínku, že nebudeme donekonečna klesat
* Regularizer mi brání v tom, abych se ,,moc vzdálil“ 🡪 udělal velkej ústřel
* **Argmin se dá snadno řešit**! 🡪 prostě vezmu druhou derivaci té funkce v argmin a položim jí rovnou nule
  + **Když je funkce konvexní, pak má minimum tam, kde je druhá derivace nulová**
* Řešením té rovnice s argmin, když to zderivuju je totok:

Obsah obrázku rukopis, text, Písmo, řada/pruh

Popis byl vytvořen automaticky

* Modelový příklad:
* Obsah obrázku text, rukopis, dopis, Písmo

  Popis byl vytvořen automaticky
* Tohle jde řešit samozřejmě přímo 🡪 řešením jsou -2/3 jako je patrný nahoře
* Úpravama dál pod zelenou šipkou dostanu tenhle výraz

Obsah obrázku text, rukopis, Písmo, řada/pruh

Popis byl vytvořen automaticky

* To **červeně** je podmínka, která musí platit, aby řešení konvergovalo k 0.
  + Tzn éta má omezení pro to aby Gradient Descent fungoval
* Obsah obrázku text, rukopis, Písmo, kaligrafie

  Popis byl vytvořen automaticky
* „Konverguje rychle“ – co to znamená?
* Error (estimation error) je odchylka od řešení 🡪 ta chceme, aby se zmenšovala s každou iterací
* Obsah obrázku text, rukopis, Písmo, snímek obrazovky

  Popis byl vytvořen automaticky

## Subgradient descent

Obsah obrázku rukopis, text, řada/pruh, diagram

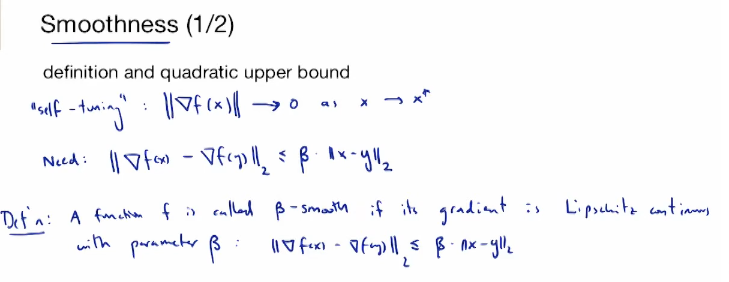
Popis byl vytvořen automaticky

* Musíme brát velmi malé eta, abychom se dostali k řešení a ,,nepřeskočili“ ho
  + Tím se ale zpomalí výpočet a sníží efektivita
* Rozdíl mezi gradient a subgradient descent je ten, **že subgradient někdy může mojí hodnotu x zvětšit,** gradient ne

## Gradient descent vs subgradient

Obsah obrázku text, rukopis, Písmo, řada/pruh

Popis byl vytvořen automaticky



Z tohohle plyne, že každá funkce která je **hladká** může být **aproximována pomocí kvadratické funkce** (asi se tim myslí, že v nějaké části)

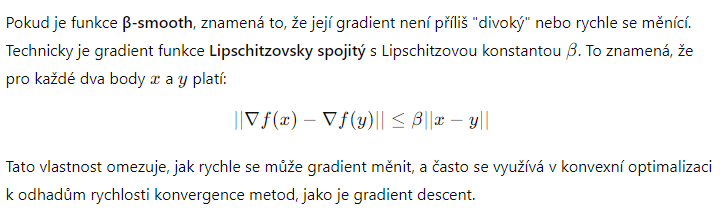


* A
* Platí, že **hodnota skutečné funkce f(x) je vyšší**, než hodnota její aproximace
* Z toho všeho co říká o hladkých funkcích v přednášce plyne totok:
  + Pro hladká funkci platí, že v každým jejím bodě může „nad ní“ vrazit kvadratickou aproximaci:
  + Obsah obrázku skica, řada/pruh, Dětské kresby, kresba

    Popis byl vytvořen automaticky
* Důležité pojmy
  + Smoot function (hladká fce)
  + Strongly convex function (silně konvexní funkce)
* Z nějakého důvodu se nejspíš snažim uzavřít funkce lokálně shora a zdola kvadratickými funkcemi. Nejspíš pak pomocí vyšetření těch kvadratických funkcí dokážu říct něco o funkci f(x)
* Pokud chci něco minimalizovat, obecnej postup co mě vždycky napadne je, **zderivuj to a polož rovný 0!** 🡪 tam většinou najdu minimum. Když najdu minimum, můžu zjistit i jeho argument, což je to co chci
* U neznámých fcí nemůžu spočítat gradient, proto se gradient nahrazuje subdiferenciálem

Obsah obrázku text, rukopis, Písmo, kaligrafie

Popis byl vytvořen automaticky

* V prvním bodu vidim, že oproti gradientu tady **éta** tady závisí na počtu iterací. U obyč gradientu to bylo prostě nějaký neměnný číslo. V subgradientu je závislý na počtu iterací
* Chyba po **T** iteracích odpovídá , tedy je taky závislá na počtu iterací!
* Nezáleží na dimenzi mýho vstupního X. Pořád budou platit ty stejný vlastnosti vázaný na počet iterací
* Hladkost a konvexnost obecně dává lepší výsledky (point z videa)
* 
* Obsah obrázku text, rukopis, Písmo, řada/pruh

  Popis byl vytvořen automaticky
* Tzn funkci, která je beta-smooth (hladká) definuju éta jako 1/beta!!

Obsah obrázku text, rukopis, kaligrafie, Písmo

Popis byl vytvořen automaticky

* Suboptimalitu určuje konstanta C^T, definovaná vejš
* Abych získal chybu epsilon, musim provést log(1/epsilon) iterací
* Pro subgradient potřebuju **1/epsilon^2** iterací, abych získal chybu **epsilon**
* Pro hladkou fci potřebuju **1/epsilon** iterací, abych získal chybu **epsilon**
* Pro silně konvexní a hladkou funkci mi stačí **log(1/epsilon)** iterací, abych získal chybu **epsilon**

Obsah obrázku text, rukopis, Písmo, řada/pruh

Popis byl vytvořen automaticky

* Rozdíl mezi gradient descent a subgradient method
* Subgradient slibuje tu složitost a chybovost s epsilon pro **jakoukoliv funkci** – je to tak?

## Proximální operátor a projekce

Obsah obrázku text, rukopis, Písmo, řada/pruh

Popis byl vytvořen automaticky

* Proximální operátor tedy záleží hlavně na hodnotě kroku éta!!!

Obsah obrázku text, rukopis, Písmo, řada/pruh

Popis byl vytvořen automaticky

Tohle už se blíží teorii, která se omílá v článcích!!

Obsah obrázku text, rukopis, Písmo, snímek obrazovky

Popis byl vytvořen automaticky