Helsinki University of Technology
Laboratory of Telecommunications Technology

S-38.211 Signaalinkäsittely tietoliikenteessä I
Signal Processing in Communications (2 ov)

Syksy 1998

7. Luento: Adaptiiviset korjaimet II

prof. Timo Laakso

Vastaanotto torstaisin klo 10-11

Huone G210, puh. 451 2473

Sähköposti: timo.laakso@hut.fi

Stokastinen gradienttialgoritmi (LM11.2)

- Edellisessä luvussa päästiin FIR-suotimen adaptointiin MSE-gradienttialgoritmilla (MSEG) joka korjaa askelparametrin määräämän verran kerroinvektoria kohti optimiratkaisua.
- Valitettavasti MSEG perustuu myös siihen, että kanavan ominaisuudet eli autokorrelaatiomatriisi ja ristikorrelaatio-vektori tunnetaan.
- Näin ei yleensä ole, eli käytännön tuntemattomissa ja muuttuvissa kanavissa tarvitaan vielä lisää yksinkertaistuksia.

...Stokastinen gradienttialgoritmi

- Lopullinen algoritmi saadaan ottamalla käyttöön *stokastinen* ('kohinainen') *gradientti*estimaatti tarkan asemesta.
- Stokastiseen gradienttiin perustuva adaptointi tunnetaan yleensä *LMS (Least-Mean Square)-algoritmina*, vaikka nimi onkin epätarkka (MSE-virhe minimoituu vain approksimatiivisesti). Näin saadaan kuitenkin käyttökelpoinen ja ehkä maailman eniten käytetty adaptiivinen suodinalgoritmi.

Gradientti ennen odotusarvoa

- MSEG-gradientti johdettiin ottamalla neliövirheestä *ensin* odotusarvo ja *sitten* gradientti:

$$\begin{aligned} \mathrm{E}\left[|e_k|^2\right] &= \mathrm{E}\left[|a_k|^2\right] - 2\,\mathrm{Re}\left\{\mathbf{c}^{\mathrm{H}}\mathrm{E}\left[a_k\mathbf{r}_k^*\right]\right\} + \mathbf{c}^{\mathrm{H}}\mathrm{E}\left[\mathbf{r}_k^*\mathbf{r}_k^{\mathrm{T}}\right]\mathbf{c} \\ &= \mathrm{E}\left[|a_k|^2\right] - 2\,\mathrm{Re}\left\{\mathbf{c}^{\mathrm{H}}\mathbf{a}\right\} + \mathbf{c}^{\mathrm{H}}\mathbf{\Phi}\mathbf{c} \qquad (11.8) \end{aligned}$$

- Tarkastellaan nyt ensin neliövirheen gradienttia *ilman* odotusarvoa. Koska gradientti ja odotusarvo ovat lineaarisia operaattoreita, niiden järjestystä voidaan vaihtaa:

$$|e_k|^2 = |a_k|^2 - 2\,\mathrm{Re}\left\{a_k\mathbf{c}^{\mathrm{H}}\mathbf{r}_k^*\right\} + \mathbf{c}^{\mathrm{H}}\mathbf{r}_k^*\mathbf{r}_k{}^{\mathrm{T}}\mathbf{c} \qquad (11.49)$$

...Gradientti ennen odotusarvoa

- Tälle saadaan gradientiksi (ks. **Harj. 11-5** edellä)

$$\nabla_{\mathbf{c}} |e_k|^2 = -2\mathbf{r}_k^*(a_k - \mathbf{r}_k^{\mathrm{T}}\mathbf{c}) = -2e_k\mathbf{r}_k^* \qquad (11.50)$$

- Tästä *voitaisiin* nyt ottaa odotusarvo ja saada tuttu tulos

$$\nabla_{\mathbf{c}_j} E\left[|e_k|^2\right] = 2\Phi\mathbf{c} - 2\mathbf{a} \qquad (11.25)$$

- mutta (11.50) onkin itse asiassa toteutuksen kannalta parempi sellaisenaan. Voidaan osoittaa, että se antaa *harhattoman* (unbiased) estimaatin todelliselle gradientille. Estimaatti on *stokastinen* eli *kohinainen*.

Stokastinen gradienttialgoritmi (SG)

- SG-algoritmin mukainen kertoimien adaptiivinen päivitys on muotoa

$$\mathbf{c}_{k+1} = \mathbf{c}_k - \frac{\beta}{2}\nabla_{\mathbf{c}} \mathrm{E}\left[|e_k|^2\right]_{\mathbf{c}=\mathbf{c}_k} \qquad (11.52)$$

- ja sijoittamalla tähän (11.50) saadaan *SG-algoritmi* muotoon

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \beta e_k \mathbf{r}_k^* = \left[\mathbf{I} - \beta\mathbf{r}_k^*\mathbf{r}_k^{\mathrm{T}}\right]\mathbf{c}_k + \beta a_k \mathbf{r}_k^* \qquad (11.53)$$

- Tätä voidaan nyt verrata *MSEG-algoritmiin*

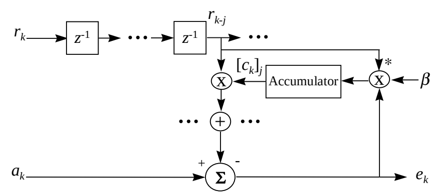
$$\mathbf{c}_{j+1} = \mathbf{c}_j + \beta(\mathbf{a} - \Phi\mathbf{c}_j) = (\mathbf{I} - \beta\Phi)\mathbf{c}_j + \beta\mathbf{a} \qquad (11.28)$$

...Stokastinen gradienttialgoritmi

SG-algoritmista tehdään seuraavat havainnot:

- 1) SG-algoritmi sisältää (valittavaa askelparametria β lukuunottamatta) *vain havaintoarvoja ja muita tunnettuja suureita*, eli *on suoraan laskettavissa* ilman muuta estimointia
- 2) Kun MSEG-algoritmissa iteraatioindeksi j oli aikaindeksistä k riippumaton, SG-algoritmissa $j = k$. Gradientin estimointi toteutetaan siis *aikakeskiarvoistamalla* tulevaa dataa.

SG-algoritmin laskentatoteutus



- SG-algoritmin laskenta yhtä päivitettävää kerrointa kohden on esitetty **Kuvassa 11-7.**

...SG-algoritmin laskentatoteutus

- Adaptaatiokaava (11.53) voidaan kirjoittaa komponenttimuodossa

$$\left[\mathbf{c}_{k+1}\right]_j = \left[\mathbf{c}_k\right]_j + \beta e_k r^*_{k-j}, \quad -L \le j \le L \qquad (11.54)$$

- eli laskenta etenee modulaarisesti komponentti kerrallaan. Uusi virhetermi lasketaan kierroksen päätteeksi seuraavaa päivitystä varten.

SG-algoritmin suppeneminen (LM11.2.1)

- Perusero MSEG- ja SG-algoritmissa on että *MSEG on deterministinen*, eli (kunhan auto- ja ristikorrelaatiodata tunnetaan) suppeneminen etenee ennaltamäärättyä polkua pitkin. Sen sijaan SG-algoritmin suppenemisessa on aina stokastista variaatiota, eli joillain iteraatiokerralla tulos voi huonotakin. Suppeneminen on *keskimääräistä*.
- SG-algoritmin suppenemista voidaan tarkastella ottamalla odotusarvo (11.53):sta:

$$\mathbf{c}_{k+1} = E\left[\left[\mathbf{I} - \beta \mathbf{r}^*_k \mathbf{r}^{\mathrm{T}}_k\right]\mathbf{c}_k + \beta a_k \mathbf{r}^*_k\right] = (\mathbf{I} - \beta \Phi)\mathbf{c}_k + \beta \mathbf{a} \qquad (11.57)$$

...SG-algoritmin suppeneminen (LM11.2.1)

- Ottamalla odotusarvo vielä *kerroinprosessin* suhteen saadaan

$$\mathrm{E}\left[\mathbf{c}_{k+1}\right] \approx (\mathbf{I} - \beta\Phi)\mathrm{E}\left[\mathbf{c}_k\right] + \beta\mathbf{a} \qquad (11.58)$$

- SG-algoritmin suppenemispolku on siis *keskimäärin* sama kuin MSEG-algoritmilla. Tästä seuraa, että myös *suppenemisehdot* (askelparametrin valinta) ovat pääpiirteissään samat molemmille algoritmeille.

...SG-algoritmin suppeneminen

- Suppenemisen tarkemman analyysin helpottamiseksi määritellään vanhaan tapaan virhevektori

$$\mathbf{q}_k = \mathbf{c}_k - \mathbf{c}_{opt} \qquad (11.59)$$

- **(Harj. 11-10.)** Johda SG-algoritmin päivityskaava muotoon

$$\mathbf{q}_{k+1} = \Gamma_k \mathbf{q}_k + \beta d_k \mathbf{r}^*_k \qquad (11.60)$$

missä

$$\Gamma_k = \mathbf{I} - \beta \mathbf{r}^*_k \mathbf{r}^{\mathrm{T}}_k \qquad (11.61)$$

ja d_k on virhesignaali optimisuotimelle

$$d_k = a_k - \mathbf{r}^{\mathrm{T}}_k \mathbf{c}_{opt} \qquad (11.62)$$

...SG-algoritmin suppeneminen

- Koska on nolla vain keskimäärin, *SG-algoritmi (11.60) ei koskaan suppene tarkasti kohti optimiratkaisua* vaan jää keikkumaan sen ympärille. Ratkaisu on vain 'keskimäärin' optimaalinen!
- SG-algoritmin hyvyyttä voidaankin arvioida *kerroinvektorin* Euklidisen (neliöllisen) *virhemitan* avulla

$$\left\|\mathbf{q}_k\right\|^2 = \mathbf{q}_k^{\mathrm{H}}\mathbf{q}_k \qquad (11.63)$$

joka kertoo kuinka lähellä kertoimet ovat optimia.

SG-algoritmin jäännosvirhe

- Viime kädessä kiinnostaa kuitenkin mikä on suodatetun ja halutun signaalin ero (virhesignaali e_k). Tämän neliövirhe ja kerroinvirhe (11.63) ovat yhteydessä seuraavasti kuten edellä (11.15) johdettiin:

$$\mathrm{E}\left[\left|e_k\right|^2\right] = \xi_{\min} + \mathbf{q}_k^{\mathrm{H}}\Phi\mathbf{q}_k \qquad (11.65)$$

- Keskiarvoistamalla tämä myös kerroinvektorien suhteen saadaan

$$\mathrm{E}\left[\left|e_k\right|^2\right] = \xi_{\min} + \mathrm{E}\left[\mathbf{q}_k^{\mathrm{H}}\Phi\mathbf{q}_k\right] \qquad (11.66)$$

- Jälkimmäistä termiä sanotaan *jäännösvirheeksi* (excess MSE) ja se kertoo kuinka paljon kertoimien stokastisesta vaihtelusta aiheutuu lisävirhettä.

SG-algoritmin suppenemisehto

- Tarkastellaan jäännösvirheen lauseketta siinä erikois-tapauksessa että input-prosessi r_k on korreloimatonta (valkoista kohinaa), eli

$$\Phi = \Phi_0\mathbf{I}, \quad \Phi_0 = \mathrm{E}\left[\left|r_k\right|^2\right] \qquad (11.67)$$

- Tällöin keskimääräinen MSE saadaan muotoon

$$\mathrm{E}\left[\left|e_k\right|^2\right] = \xi_{\min} + \Phi_0\left\|\mathbf{q}_k\right\|^2 \qquad (11.68)$$

- Voidaan johtaa (ks. kirjan **Appendix 11-A**) tulos:

$$\mathrm{E}\left[\left\|\mathbf{q}_{k+1}\right\|^2\right] = \gamma\mathrm{E}\left[\left\|\mathbf{q}_k\right\|^2\right] + \beta^2 N\Phi_0\xi_{\min} \qquad (11.69)$$

missä $\gamma = 1 - 2\beta\Phi_0 + \beta^2 N\Phi_0^2$

...SG-algoritmin suppenemisehto

- Valkoisen kohinan tapauksessa on vain yksi adaptaatiomoodi (ominaisarvot ovat samat). Kaavasta (11.69) näkyy, kuinka jäännösvirhe riippuu askelparametrista β.
- Tuloksesta voidaan myös johtaa ehto algoritmin suppenemiselle, eli

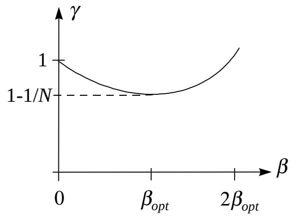
$$\left|\gamma\right| = \left|1 - 2\beta\Phi_0 + \beta^2 N\Phi_0^2\right| < 1 \qquad (11.70)$$

- Nopeimpaan suppenemiseen johtaa pienin γ:n arvo, josta saadan optimi-β:ksi

$$\beta_{opt} = \frac{1}{N\Phi_0} \qquad (11.71)$$

...SG-algoritmin suppenemisehto

- Suppenemisehtoa (11.71) havainnollistaa **Kuva 11-8**:



...SG-algoritmin suppenemisehto

- SG-algoritmin stabiilisuusehdoksi saadaan (-1<γ<1):

$$0 < \beta < \frac{2}{N\Phi_0} = 2\beta_{opt} \qquad (11.72)$$

- Tätä voidaan verrata MSEG-algoritmin stabiilisuusehtoon

$$0 < \beta < \frac{2}{\lambda_{\max}} \qquad (11.39)$$

- SG-algoritmin stabiilisuusehto on usein paljon tiukempi - huomaa että se on kääntäen verrannollinen suotimen pituuteen *N*. Algoritmin stokastinen luonne vaatii siis ylimääräistä pelivaraa.

SG-algoritmin modifikaatioita (LM 11.2.2)

Askelparametrin normalisointi

- Epästationäärisessä ympäristössä signaalin teho voi vaihdella paljonkin, mikä vaikuttaa suoraan algoritmin suppenemisnopeuteen jos askelparametri β pidetään vakiona. Tämä voidaan helposti eliminoida *normalisoimalla* β seuraavasti

$$\beta_k = \frac{a}{\sigma_k^2 + b} \qquad (11.77)$$

missä *a* ja *b* ovat sopivia parametreja ja σ_k^2 on input-signaalin tehon estimaatti aikaindeksillä *k*. Tehon estimointiin voidaan käyttää *eksponentiaalisesti painotettua aikakeskiarvoa*

$$\sigma_k^2 = (1-\alpha)\sum_{j=0}^{\infty} \alpha^j \left| r_{k-j} \right|^2 \qquad (11.78)$$

...SG-algoritmin modifikaatioita

- Kaavassa (11.78) tietysti 0<α<1. Tällöin tehoestimaatille saadaan näppärä rekursiokaava

$$\sigma_k^2 = \alpha\sigma_{k-1}^2 + (1-\alpha)\left| r_k \right|^2 \qquad (11.79)$$

Kaksimoodialgoritmit (Gear-Shift Algorithms)

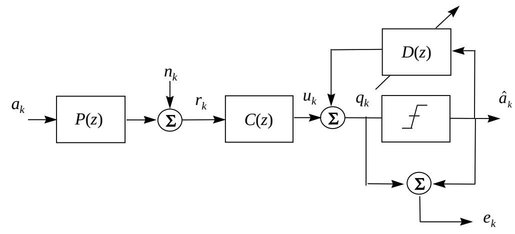
- Johtotransmissiossa (esim. äänitaajuusmodeemit) kanava muuttuu vain vähän yhteyden aikana, jolloin alkuestimoinnin jälkeen askelparametria voidaan pienentää.

Adaptiivinen DFE

- Edellä saatiin lineaarisen FIR-korjaimen adaptointiin kehitettyä lopultakin käyttökelpoinen menetelmä: *stokastinen gradientti(SG- eli LMS-)algoritmi.*
- Seuraavaksi kehitellään vastaava *algoritmi päätöstakaisinkytketylle eli DFE-korjaimelle*, jossa on lineaarisen korjaimen lisäksi osuus joka vähentää ISIä edellisiin päätöksiin perustuen. Osoittautuu, että lineaarisen FIR-korjaimen SG-algoritmia voidaan käyttää pienin muutoksin lähes sellaisenaan DFE-korjaimen adaptointiin.

Adaptiivinen DFE (LM11.3)

- DFE-korjaimen rakenne oli jo esillä aiemmin **Kuvassa 11-2**:



...Adaptiivinen DFE

- DFE-korjaimessa on siis lineaarinen FIR-osa (*prekursori*) joka suodattaa tulevia näytteitä kuten pelkkä LE.
- Lisäksi rekursiivinen osa (*postkursori*) poistaa ISIä vähentämällä vanhoja symboliarvoja (estimaatteja) sopivilla painokertoimilla. Postkursori vähentää siis *vain* ISIä eikä kohinaa. Se vaikuttaa kuitenkin koko korjaimen kohinavahvistukseen helpottamalla prekursorin toimintaa.
- Käytettäessä MSE-virheen minimointikriteeriä postkursorin lisääminen vaikuttaa siis myös prekursorin optimaalisiin kertoimiin. Adaptointi on siis tehtävä yhteisesti.

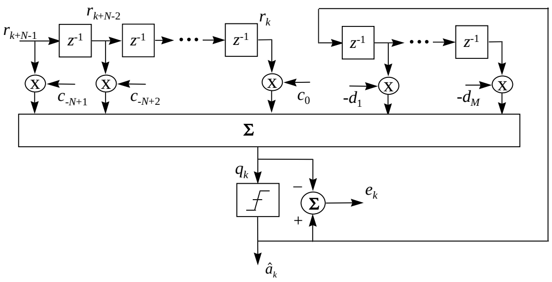
Adaptiivisen DFE:n FIR-rakenne

- Oletamme nyt, kuten edellä, että prekursori on antikausaalinen N-kertoiminen FIR-suodin ja postkursori on M-kertoiminen ja kausaalinen (ei-kausaalisuus poistuu käytännössä viivettä lisäämällä). Laskenta voidaan esittää differenssiyhtälöllä

$$q_k = \sum_{i=-(N-1)}^{0} c_i r_{k-i} - \sum_{i=1}^{M} d_i \hat{a}_{k-i} \qquad (11.80)$$

- eli se on painotettu keskiarvo uusista input-näytteistä ja vanhoista päätöksistä. Vaikka järjestelmä sisältää takaisinkytkentää, se voidaan esittää formaalisti **Kuvan 11-10** FIR-rakenteella.

Adaptiivisen DFE:n FIR-rakenne



Kuva 11-10. DFE-korjaimen rakenne.

Adaptiivisen DFE:n MSE-ratkaisu

- Input-näytteet koostuvat paitsi halutusta symboleista myös keskinäisvaikutuksesta ja kohinasta, eli ne ovat muotoa

$$r_k = p_0 a_k + \sum_{n \neq 0} p_n a_{k-n} + n_k = \sum_n p_n a_{k-n} + n_k \qquad (11.80b)$$

jossa summausindeksit ja kertoimet p_n eivät yleensä ole tarkkaan tiedossa. Sijoittamalla kaavaan (11.80) saadaan

$$q_k = \sum_{i=-(N-1)}^{0} c_i \sum_n p_n a_{(k-i)-n} - \sum_{i=1}^{M} d_i \hat{a}_{k-i} + \sum_{i=-(N-1)}^{0} c_i n_{k-i}$$

$$= \sum_m v_m a_{k-m} + \sum_{i=-(N-1)}^{0} c_i n_{k-i} \qquad (11.80c)$$

jossa päätökset on oletettu oikeiksi.

...Adaptiivisen DFE:n MSE-ratkaisu

- Tämä voidaan jakaa kolmeen termiin:

$$q_k = v_0 a_k + i_k + n_k'$$

- Ensimmäinen on haluttu symboli. Toinen on ISI-termi joka on muotoa

$$i_k = \sum_{m \neq 0} v_m a_{k-m} \qquad (11.81)$$

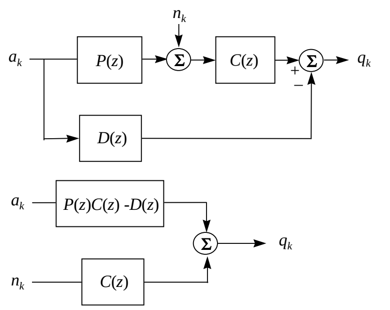
jossa

$$v_k = \begin{cases} \sum_{i=-(N-1)}^{0} c_i p_{k-i} - d_k, & 1 \le k \le M \\ \sum_{i=-(N-1)}^{0} c_i p_{k-i}, & \text{muulloin} \end{cases} \qquad (11.82)$$

- Kolmas termi edustaa kohinaa.

...Adaptiivisen DFE:n MSE-ratkaisu

- Tätä havainnollistaa **Kuva 11-11:**



...Adaptiivisen DFE:n MSE-ratkaisu

- Kun oletetaan että kohina ja lähetetyt symbolit (ja siten myös ISI) ovat riippumattomia, niitä voidaan MSE:n minimoinnin kannalta tarkastella erikseen. Kun lisäksi symbolit keskenään ovat riippumattomia, on ilmeistä että kokonaisvirheen odotusarvo voidaan minimoida *valitsemalla postkursorin kertoimet* d_k *siten, että M ISI-termiä nollautuu*. Näin saadaan MMSE-ratkaisu postkursorikertoimille muodossa

$$d_m = \sum_{i=(-N-1)}^{0} c_i p_{m-i} \qquad 1 \le m \le M \qquad (11.83)$$

- Olettaen että ISI-termit tunnetaan, postkursorikertoimet riippuvat suoraan prekursorista.

...Adaptiivisen DFE:n MSE-ratkaisu

- Toteutuksessa voitaisiin optimoida suoraan prekursorikertoimet sopivalla algoritmilla ja laskea postkursorikertoimet kaavasta (11.83).
- Käytännössä on kuitenkin helpompaa käyttää **Kuvan 11-11** rakennetta ja signaaleja suoraan adaptointiin, kuten seuraavaksi johdetaan.

SG-algoritmi DFE:lle

- Formaalisti SG-algoritmi voidaan johtaa DFE:n tapaukselle suoraviivaisesti. Määritellään *kerroinvektori*

$$\mathbf{v} = \begin{bmatrix} c_{-(N-1)} & \cdots & c_0 & -d_1 & \cdots & -d_M \end{bmatrix}^{\mathrm{T}} \qquad (11.86)$$

- joka sisältää siis prekursorin ja postkursorin kertoimet peräkkäin, *N*+*M* elementtiä. Vastaava pidennetty *datavektori* on muotoa

$$\mathbf{w}_k = \begin{bmatrix} r_{k+(N-1)} & \cdots & r_k & a_{k-1} & \cdots & a_{k-M} \end{bmatrix}^{\mathrm{T}} \qquad (11.87)$$

...SG-algoritmi DFE:lle

- Virhesignaali voidaan nyt esittää muodossa

$$e_k = a_k - \mathbf{v}^{\mathrm{T}}\mathbf{w}_k \qquad (11.88)$$

joka on formaalisti samanlainen kuin LE-korjaimen virheen lauseke. Analogiaan vedoten voidaan katsoa mallia SG-algoritmille kaavasta (11.53):

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \beta e_k \mathbf{r}_k^* \qquad (11.53)$$

- Ottamalla kerroinvektorillekin päivitysindeksi *k* saadaan DFE-korjaimen SG-algoritmi muotoon

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \beta e_k \mathbf{w}_k^* \qquad (11.89)$$

DFE-SG-algoritmin ominaisuuksia

- Aiemmin analysoitiin *ideaalisen* DFE-korjaimen ominaisuuksia stationäärisessä tilanteessa. Olennaista oli *kohinavahvistuksen pieneneminen* koska ISIn poisto helpottuu.
- Lähteessä *Bingham: The Theory and Practice of Modem Design* (Wiley 1988, s. 289) todetaan, että SG-DFE *suppenee nopeammin* kuin SG-LE, koska kanavaa ei tarvitse ‘kääntää’ kokonaan. Näin SG-DFE on vähemmän herkkä input-signaalin autokorrelaatiomatriisin ominaisarvohajeelle.
- Koska pre- ja postkursorien kerrointen laskenta perustuu samaan virhesignaaliin, kompleksisuus ja *suppeneminen* ovat suunnilleen samanlaista kuin (*N*+*M*)-tappisella SG-LE-FIR-korjaimella. *Toteutus* on siis yksinkertainen mutta tehokas.