Vurdering av *features* for steganalyse i JPEG

Hans Georg Schaathun
⟨georg@schaathun.net⟩
Høgskolen i Ålesund, Institutt for Ingeniør- og Realfag
Postboks 1517, 6025 Ålesund

Samandrag

Steganografi er teknikkar for hemmeleg kommunikasjon, der sjølve eksistensen av den hemmelege meldinga må haldast løynd. Steganalyse dreier seg om teknikkar for å påvisa slike meldingar. Ei rekkje moderne steganografiteknikkar finst for å modulera digital informasjon i digitale bilete. Dei mest lovande steganalyseteknikkane byggjer på maskinlæring. Literaturen omfattar ei lang rekkje *feature vectors* som kan brukast til å påvisa steganografi saman med vanlege maskinlæringsalgoritmar som t.d. *support vector machines* (SVM).

I denne artikkelen går me systematisk gjennom yteevna til kjende *features*, både dei kjende *feature*-vektorane og delvektorar. Ved å ha reimplementert ca. 15 kjende teknikkar, kan me gjera ei langt meir omfattande samanlikning enn tidlegare forfattarar. Me har òg brukt tre ulike *feature selection*–teknikkar for å identifisera lovande einskild-*features*.

1 Bakgrunn

Behovet for å kunna senda hemmelege meldingar har eksistert sidan oldtida. Kryptografi er velkjend som løysing og står i dag som svært moden teknologi, med ei rekkje gode og pålitelege løysingar. Ulempa med kryptografi, på somme bruksområde, er at ei kryptert melding openbert er kryptert, og det er difor ikkje hemmeleg at avsendar og mottakar har noko på gong, med løyndomar å utveksla.

Steganografi søkjer å halda sjølve eksistensen av løyndomen hemmeleg. Eit klassisk døme finst hjå den greske historikaren Herodotus. Histiæus ville senda ei melding til Aristagoras for å gje ordre om opprør. Ei kryptert melding ville nesten heilt sikkert vorte stogga på grensa, sjølv om fienden ikkje kunne forstå innhaldet. Han barberte ein tru slave på hovudet, tatoverte meldinga på skallen og lot håret gro tilbake før slaven vart send til Aristogoras.

I klassisk tid fanst ikkje noko klart skilje mellom kryptering og steganografi. Båe handlar om hemmelege meldingar. Då Francis Bacon skreiv om kryptering i 1605 nemnde han den steganografiske eigenskapen (*that they be without suspicion*) som ein viktig eigenskap for gode krypteringssystem. Likevel var steganografi i stor grad gløymt då moderne kryptografi vaks fram på 1900-talet.

Idéen til moderne steganografi finst hjå Simmons [21], som formulerte *The Prisoners' Problem*, og framla steganografi som eit kryptografisk problem. På 1990-talet og

Denne artikkelen vart presentert på konferansen NIK-2011; sjå http://www.nik.no/.

framover har me sett omfattande forsking innanfor steganografi, hovudsakleg bygd på signalprosessering, og ei lang rekkje algoritmar og dataprogram er tilgjengelege. Steganalyse vaks fram parallelt for å knekka steganografisystem.

Hovudmålet i denne artikkelen er å evaluera eksisterande steganalyseteknikkar, og vurdera relevante metodar for slik evaluering. For å avgrensa problemet vil me kun sjå på JPEG-bilete som medium. I avsnitt 2 vil me gje ei kort og enkel introduksjon til aktuelle teknikkar for steganografi og steganalyse, samt maskinlæring generelt, og me håper at denne er tilgjengeleg for dei utan bakgrunn korkje i maskinlæring eller i steganografi. Avsnitt 3 presenterer og drøftar me dei empiriske resultata våre, som omfattar både samanlikning av eksisterande system og nye og betre *feature vectors*. Til slutt konkluderer og oppsummerer me med avsnitt 4.

For å mogleggjera dei omfattande eksperimenta som me byggjer på, har me implementert ei lang rekkje kjende algoritmar og teknikkar i eitt system. Denne programvaren er gjort tilgjengeleg for vidare forsking [18].

2 Grunnleggjande teknikkar

Steganografi i JPEG

Eit *stegosystem* definerer me som eit par med algoritmar med fylgjande eigenskapar[1]. Innkodingsalgoritmen tek tre argument inn, eit bilete som me kallar *vertsbiletet*, ei løynd melding som skal skjulast og, som regel, ein løynd nykel som er kjend av avsendar og mottakar. Utdata er eit modifisert bilete som me kallar eit *steganogram*. Dekodingsalgoritmen tek to argument inn: steganogrammet og den same løynde nykelen som vart brukt ved innkoding. Utdata er den løynde meldinga som var skjult. For at systemet skal vera sikkert, må det vera uråd for ein tredjepart å kunne skilja vertsbilete frå steganogram.

JPEG er eit svært populært format for biletkoding, og det er godt egna for fotografiar. Med store mengder biletdata tilgjengeleg er det òg eit godt medium for steganografi. Dersom ein startar med eit vanleg fullfargebilete (RGB), so består JPEG-kodinga av fire steg:

1. *Fargekonvertering.* Dette er ein enkel, lineær (1-1) transformasjon av ein fargevektor (RGB). Vanlegvis bruker ein YCbCr som skil ut gråtoneinformasjon i éin komponent (Y).

2. *Blokkvis DCT-transform.* Biletet vert delt i blokkar på 8×8 pixlar. Kvar blokk vert transformert med ein diskret kosinustransform (DCT) for å få ein frekvensrepresentasjon av biletet. Resultatet av dette kallar me DCT-matrisa til biletet.

3. *Kvantifisering (komprimering med tap).* Kvar 8×8-blokk i DCT-matrise vert so dividert, elementvis, med ei heiltalsmatrise og so runda til eit heiltal. Slik vert kvar DCT-koeffisient skriven på ein meir grovkorna skala og me sparer plass. Kvantifiseringa er sjølvsagt ikkje 1–1. Koeffisientar med høg frekvens vert komprimert meir enn dei med låg frekvens fordi dei har mindre å seia for kvaliteten på biletet. Resultatet av dette steget kallar me JPEG-matrisa.

4. *Komprimering utan tap.* Det siste steget er igjen 1–1. JPEG-matrisa vert koda med ein generell komprimeringsalgoritme utan tap, som oftast Huffman-koding.

[1] Der finst meir generelle definisjonar, men me har medvite vald definisjonen for å innskrenka problemet.

Steganografi i JPEG vil i dei fleste tilfelle gå ut frå JPEG-matrisa. Det er viktig at me bruker ein representasjon etter komprimering med tap, for å unngå at skjult informasjon forsvinn i komprimeringa. Samstundes treng me ei forståing av biletet som bilete for å unngå at den steganografiske meldinga vert synleg i biletet, og denne forståinga forsvinn når biletet vert vidare komprimert. Det gjer JPEG-matrisa til det naturlege utgangspunktet for steganografi. Det er vanleg i litteraturen å bruka berre gråtonekomponenten (Y) i steganografi. Dette gjer at ein ikkje treng å ta omsyn til dei ulike statistiske eigenskapar i dei ulike komponentane, og ein får eit system som kan brukast både på svart/kvitt og fargebilete.

Den fyrste kjende algoritmen for steganografi i JPEG heiter *Jsteg*. Dekodingsalgoritmen tek JPEG-matrisa og skriv ho som ein sekvens der ein hoppar over alle koeffisientar som er 0 eller $+1$. Dei resterande koeffisientane vert reduserte modulo 2, og resultatet er den løynde meldinga i binærform. Innkodingsalgoritmen vil tilsvarande erstatta den minst signifikante bitten i kvar koeffisient (bortsett frå dei som er 0 eller $+1$) med ein bit frå meldinga.

Eksperimenta bak denne artikkelen har hovudsakleg brukt steganogrammar frå F5 [24], som er ein av dei mest kjende algoritmane for steganografi i JPEG. Dekodinga i F5 ignorerer 0, og dekodar negative koeffisientar x som $(x+1) \mod 2$ og positive som $x \mod 2$. Dette gjer at F5, i motsetnad til Jsteg, held på symmetrien i koeffisienthistogrammet. I tillegg nyttar F5 sokalla *matrix coding*, som i bunn og grunn er det same som koding for defekt minne kjend frå kodeterien. Tanken i *matrix coding* er at ein bruker $n > k$ koeffisientar for å koda k meldingsbittar, men der Jsteg treng modifisera $k/2$ koeffisientar i gjennomsnitt, klarer F5 seg med færre.

Mange publiserte program for steganografi vil alltid komprimera biletet før meldinga vert lagt inn. Dersom inputtbiletet allereie er komprimert, vert det dekomprimert og rekomprimert, ofte med nye parametrar. Denne dobbelkomprimeringa skaper artifaktar som svært ofte er enklare å oppdaga enn sjølve steganografien, og det er rimeleg å rekna det som ein *bug*. I dette arbeidet har me difor reimplementert Jsteg og F5 utan dobbelkomprimering, slik at skjulte meldingar vert vanskelegare å oppdaga.

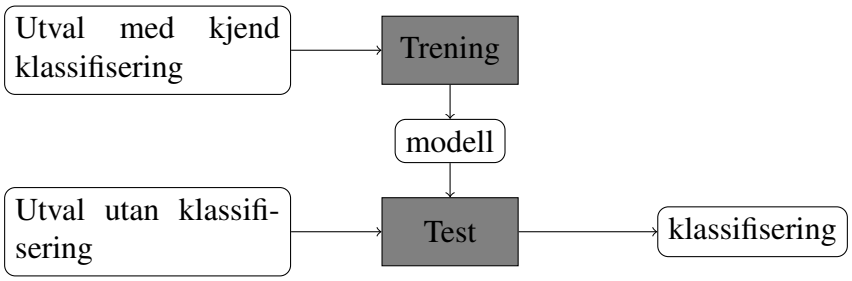
Maskinlæring

Klassifisering er eit vanleg problem i fleire vitskapar. Ein har ein populasjon som er delt i to eller fleire disjunkte delmengder (klasser), og ein ynskjer å kunne ta eit tilfeldig eksemplar frå populasjonen og fortelja kva delmengd det høyrer til. Populasjonen kan t.d. vera mengda av moglege røntgenbilete av ein bestemt kroppsdel, der dei to delmengdene er dei som viser symptom på ein gjeven sjukdom og dei som ikkje gjer det. I steganalyse kan det vera mengda av alle moglege bilete, der klassene er steganogram og naturlege bilete.

Slike klassifiseringsproblem kan løysast ved å byggja statistiske modellar for kvar klasse. Datamengdene me har med å gjera er derimot slik at det er sjelden overkommeleg å gjera dette analytisk. Maskinlæring er ein vanleg løysing på slike problem.

Eit typisk maskinlæringssystem er vist i figur 1. I treningsfasen får algoritmen inn eit utval med objekt frå populasjonen, *med* kjend klassifisering, og bruker dette til å byggja ein modell, meir eller mindre med *brute force*. I testfasen, og ogso i praktisk bruk, får algoritmen inn modellen frå trening saman med objekt *utan* kjend klassifisering. Utdata er klassifiseringa, dvs. delmengda som kvart objekt høyrer til.

For å evaluera systemet køyrer ein simuleringar der objekta har ei kjend *sann* klassifisering som systemet ikkje får tilgang til. Då kan ein samanlikna den *sanne* klassifiseringa



Figur 1: Maskinlæringssystem

med klassifiseringa frå systemet og identifisera feil. Prinsippa for å estimera feilsannsyn fylgjer simulering på andre område, som t.d. kommunikasjonssystem.

I praksis vil me sjelden bruka heile objektet som skal klasifiserast i systemet, fordi det er for mykje data. I staden dreg me ut statistikkar, so-kalla *features*, frå kvart objekt. Kvar *feature* er normalt eit flyttal, og med ei rekkje *features* får med ein *feature vector*. Me skal sjå i neste avsnitt kva me typisk kan bruka som *features*.

Når systemet er trent, slik at me har ein modell og kan klassifisera ukjende objekt, kallar me systemet ein *klassifikator*. Det er viktig at denne vert testa med eit utval som er heilt uavhengig av det som vart brukt til trening, for å kunne estimera ytinga på ukjende data. Dvs. at me samanliknar den klassifiseringa som me får ut frå teststeget, med ei kjend, sann klassifisering som algoritmen ikkje fekk tilgang til. Det enklaste målet på ytinga, og det som vert mest brukt i steganalyse, er *accuracy* (nøyaktigheit), som kan definerast som raten av korrekt klassifiserte objekt over utvalsstorleiken.

Der finst ei lang rekkje maskinlæringsalgoritmar som kan brukast i eit system slik som me har skissert. Kunstige nevrale nettverk er kanskje dei mest kjende. Ei av dei mest populære og nøyaktige algoritmane er *support vector machines* (SVM). Ein vesentleg føremon med SVM er at algoritmen svært enkel å bruka som ein svart boks. I eksperimenta våre har me nytta libsvm [6] og fylgjer prosedyren frå [13] der parametrane vert optimaliserte vha. *grid search*.

Steganalyse

Steganalyse viser til ein kvar teknikk som kan brukast til å skilja mellom steganogram og naturlege bilete. Slike teknikkar kan vera målretta (*targeted*), dvs. at dei er utvikla spesielt for å identifisera steganogram frå eit spesielt stegosystem. Steganalyse basert på maskinlæring er i større eller mindre grad universell; dvs. at ein kan trena steganalysesystemet for ulike stegosystem og få ein brukbar, trent klassifikator. Desse teknikkane er difor ikkje avgrensa til eit einskild stegosystem. Derimot er det ikkje sannsynleg at eit slikt universelt system fungerer like bra på alle stegosystem, og det er ingen grunn til å tru at ein klassifikator kan påvisa steganogram frå stegosystem som ikkje vart brukte i treningsutvalet.

Universell steganalyse vert stundom kalla *blind*, medan somme forfattarar stiller strengare krav for å kalla systemet blindt. Idéelt sett ynskjer me oss ein ferdig trent klassifikator som kan påvisa steganogram frå vilkårleg stegoalgoritmar, inkl. algoritmar som var ukjende ved trening. Slike blinde system kan til ein viss grad realiserast ved å bruka såkalla éinklasseklassifisering, men det har vore svært lite forsking på dette.

Dette arbeidet er avgrensa til toklasseklassifisering der me skil mellom steganogram frå eit bestemt, kjend stegosystem på den eine sida, og naturlege bilete på den andre.

Utfordringa i steganalyse er å velja gode *features*. Medan klassifiseringsalgoritmen brukast som ein svart boks som fungerer like bra om me skal driva steganalyse, medisinsk biletanalyse eller noko anna, so må *features* veljast ut frå problemet.

Feature Selection og Fusjon

Eit hovudproblem i maskinlæring er kva *features* ein lyt velja, og der er to motstridande omsyn å ta. Det fyrste som slår oss er gjerne *flest mogleg*. Di fleire *features*, di meir informasjon, og dermed skulle ein venta betre klassifisering. Dette leier gjerne til so-kalla fusjon på *feature*-nivå; me slår saman alle *feature vectors* som me kjenner til éin stor vektor.

Mykje av tanken i maskinlæring er at maskina skal gjera jobben, og då er det rimeleg å gje alle kjende *features* til maskina og lata ho ta jobben. Då slepp me å tenkja på kvifor kvar *feature* er nyttig.

Problemet me ukritisk inklusjon av *features* er kjend som *the curse of dimensionality*. Laust sagt veit me at når dimensjonaliteten aukar, so aukar òg avstanden mellom punkt i rommet. Ser me konkret på klassifiseringsmodellar, so vil me sjå at med ein *feature vector* med tilrekkeleg høg dimensjon, får me so mange fridomsgradar at modellen kan tilpassast perfekt til treningsutvalet. Då vil modellen òg fanga opp støyelement som finst i treningsutvalet men som er irrelevant for testutvalet. Dette vert kalla overtilpassing (*overfitting*).

Ei rekkje teknikkar har vorte utvikla for å redusera dimensjonaliteten i datasett. *Feature selection* søkjer å identifisera ein optimal delvektor frå ein gjeven *feature*-vektor. *Feature extraction* er meir generelt, og projiserer *feature*-rommet ned i lågare dimensjon. Me har fokusert på *feature selection* av di dette har ein ekstra føremon: Ved å identifisera *features* som er særleg nyttige for klassifikasjonen får me ei betre intuitiv forståing av modellen; noko som t.d. er nyttig for å utvikla betre stegosystem.

Feature Selection *vha. informasjonsteori*

Informasjonsteori vart fyrst utvikla i samband med telekommunikasjon, men har kome til nytte på fleire område. Sentralt i informasjonsteorien er gjensidig informasjon $I(X;Y)$, som måler kor mykje informasjon ein observasjon av ein stokastisk variabel X gjev oss om ein annan stokastisk variabel Y. I maskinlæring kan X svara til ein eller fleire *features* medan Y er klassa. Då vil $I(X;Y)$ vera eit mål på kor mykje klassifiseringssystemet i teorien *kan* læra om klassifiseringa ut frå gjevne *features* X. Informasjonen er definert ut frå entropi $H(X)$, som

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X), \tag{1}$$

$$H(X) = -\sum_{x\in\mathcal{X}} P(X=x)\log P(X=x), \tag{2}$$

$$H(X|Y) = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} P(X=x,Y=y)\log\frac{P(Y=y)}{P(X=x,Y=y)}. \tag{3}$$

Gavin Brown [4] hev utvikla eit generelt rammeverk for å bruka informasjonsteori til *feature selection*, og ei lang rekkje tidlegare metodar vert inkorporerte i rammeverket. Intutivt, når ein har vald N *features* X_i og vurderer å leggja til éin *feature* til, er det fornuftig å velja den *feature* X_n som har høgast gjensidig informasjon med $I(X_n;Y|X_1,\ldots,X_N)$ klassa Y, gjeve dei allereie valde *features* $X_{i_1},\ldots,X_{i_N}$. Brown viser at denne informasjonen

kan tilnærmast med

$$J_n = H(X_n) - H(X_n|Y) - \beta \sum_{i=1}^{N} (H(X_n) + H(X_i) - H(X_n, X_i)) + \gamma \sum_{i=1}^{N} (H(X_n|Y) + H(X_i|Y) - H(X_n, X_i|Y)). \quad (4)$$

Den enkleste tilnærminga, ein rein trunkering av høgareordens ledd, gjev $\beta = \gamma = 1$ og vert kalla FOU (*first-order utility*), men han viser òg at ein kan oppnå betre resultat ved å variera desse parametrane. Han anbefaler *Joint Mutual Information* (JMI),

$$J_{JMI} \approx \sum_{k=1}^{n-1} I(X_n X_k; Y),$$

og viser at dette er ekvivalent med J_n dersom $\beta = \gamma = 1/(n-1)$.

Innanfor maskinlæring er det vanleg å bruka diskret entropi, dvs. at flyttals-*features* må diskretiseras ved å dela tallina i båsar og identifisera kvar verdi med båsen. Val av båsvidd er ikkje trivielt. For å omgå dette problemet har me vald å bruka kontinuerlig entropi, der me bruker estimeringsteknikken etter Ahmad og Lin [1]. For å unngå ekstreme utslag vert kvar *feature* skalert lineært til å ha gjennomsnitt 0 og varians 1 før seleksjonen.

Boosting Feature Selection

Boosting Feature Selection (BFS) vart introdusert av [23] og føreslege for steganalyse i [9]. BFS byggjer på ein tidlegare idé der *boosting* tyder at ein lineært kombinerer fleire klassifikatorar til ein ny og betre klassifikator. I BFS kombinerer klassifikatorar som kvar bruker éin einskild *feature*.

Lat $\vec{x} = (x_1, \ldots, x_n)$ være *feature*-vektoren, og klassene $\vec{y} = (y_1, \ldots, y_n)$ der $y_i = \pm 1$. Klassifikatoren skriv me på formen

$$F_m(\vec{x}) = \sum \beta_i x_i,$$

der $\vec{\beta} = (\beta_1, \ldots, \beta_n)$ har (høgst) m element ulik 0.

Me startar med ein tom klassifikator F_0. I kvar runde har med ein klassifikator F_{m-1} og definerer ein ny F_m ved å velja ein ny *feature* x_γ og tilhøyrande vekt $\beta = \beta_\gamma$. Kvar *feature* vert vald for å minimera kvadratfeilen, dvs. at me løyser optimeringsproblemet

$$\min_{\beta,\gamma} \sum_i w_i \cdot [y_i - F_{m-1}(x_i) - \beta x_\gamma]^2.$$

Ein itererer til resultatet konvergerer.

Sjølvsagt er $F = F_m$ ein trent klassifikator som kan brukast direkte. Det har me førebels ikkje heve kapasitet til å testa. I staden har me brukt dei utvalde *features* (dei som har $\beta_\gamma \neq 0$) i SVM som elles.

3 Analyse og resultat

Samanlikning av kjende system

I eksperimenta våre har me brukt dei 5000 fyrste bileta frå BOWS-samlinga [10]. Av desse 5000 har me brukt halvparten, tilfeldig vald, som steganogram og resten som naturlege

Feature Vector	Ref.	F5/QF75	Jsteg/QF75	F5/QF85
MPEV-438	[14]	86.675	99.875	77.975
NCPEV-219	[14]	86.300	99.875	79.300
Markov-243	[20]	82.150	98.85	72.175
PEV-219	[17]	81.375	99.0	75.300
Fridrich	[11]	74.575	95.45	70.175
HCF-390	[7]	67.875	71.9	56.200
SPAM	[16]	65.200	82.075	58.225
WAM-27	[12]	54.400	66.425	53.650
EM-108	[8]	53.775	61.7	49.650
Farid-72	[15]	53.350	71.350	50.425
HCF-39	[25]	52.625	52.375	51.5
AC-10	[26]	51.825	58.650	49.925
BSM-12	[2]	51.725	50.175	48.925
CP-27	[3]	51.150	51.900	49.675
ALE-10	[5]	49.900	51.275	51.075
SMCM-130	[22]	49.400	51.4	49.100

Tabell 1: Samanlikning av kjende *feature vectors*. For kvar *feature vector* har me gjeve eit namn som me bruker i teksta, referanse til opphavsartikkelen, og feilrate i prosent ved eksperiment med tre ulike datasett.

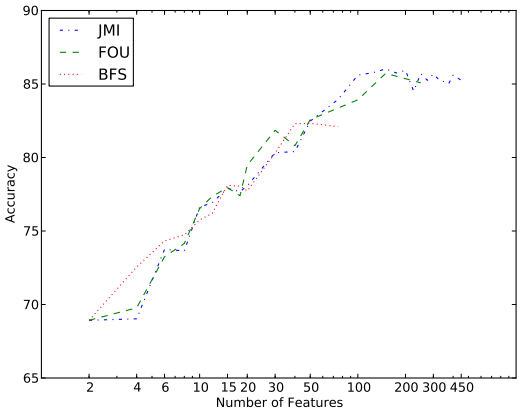
bilete. Frå kvar klasse har me teke 500 tilfeldige bilete til trening, medan dei resterande 2000 er brukte til testing. Arbeidet er hovudsakleg gjort på grunnlag av bilete komprimert med JPEG med kvalitetsfaktor 75 (QF75), med F5 som stegosystem. For å gje eit lite innblikk i generaliseringsevna har me og teke med resultat for Jsteg ved QF75, og F5 ved lågare komprimering (QF85).

Dei skjulte meldingane er tilfeldige bitstrengar på 512 bytes per bilete. BOWS-bileta er 512×512 pixlar, so dette svarer til meldingar på mindre enn 0.016 bitar per pixel. Dette er korte meldingar samanlikna med det som er vanleg i litteraturen, men det er to gode grunnar for at meldingane bør vera so korte. Den eine har med F5 å gjera. Meldingane må vera korte for å få nokon gevinst av *matrix coding*. Den andre grunnen er metodologisk. Dersom meldingane vert lengre, går feilratane i klassikatoren ned, og ein vil trengja større datasett for å få statistisk signifikante resultat.

Samanlikninga av kjende *feature vectors* er vist i tabell 1. Talet som inngår i namna på *feature vectors* indikerer dimensjonaliteten. Merk at me har eliminert 55 dupliserte *features* frå PEV-219/NCPEV-219 og 110 frå MPEV-438. Desse var tidlegare kjende som PEV-274/NCPEV-274 og MPEV-274.

Me legg fyrst merke til at Jsteg vert oppdaga med fleire *feature vectors* som ikkje er signifikant betre enn 50–50 mot F5. Mest påfallande er at pioneren Farid-72 har brukbart resultat mot Jsteg, men hev ca. 50% mot F5. Dei beste *feature vectors* ligg over 99% mot Jsteg, som betyr at ein ikkje kan bruka normaltilnærminga for å vurdera statistisk signifikans. Ein vanleg tommelfingerregel i simulering (generelt) er at ein simulerer til ein får 100 feilfall, og me bør difor ikkje rekna *accuracies* over 97,5% som pålitelege med utvalet vårt på 4000 bilete. Difor vil me se bort frå Jsteg i den vidare analysa.

Dei fem beste vektorane bruker mange felles prinsipp. Sentralt står ein kalibreringsteknikk, der biletet å dekomprimera, klippa fire pixlar på kvar side, og rekomprimera med same kvantisering. Resultatet er at 8×8-nettet for DCT-transformen er forskyve og dis-



Dim.	JMI	FOU	BFS
2	68.925	68.925	68.925
4	69.025	69.775	72.55
8	73.650	74.15	74.75
10	76.625	76.525	75.75
15	77.900	77.95	79.10
20	78.075	79.475	77.725
30	80.325	81.85	80.30
50	82.475	82.575	82.325
100	85.600	83.925	—
150	86.025	85.725	—
200	85.925	86.40	—

Figur 2: Accuracy for SVM using automatically selected features.

kontinuitetar som steogsystemet kan ha innført i blokkovergangane vert glatta ut. PEV-219 tek differansen av NCPEV-219 rekna ut frå det originale og det kalibrerte biletet. Sameleis er MPEV-438 dei to NCPEV-219-vektorane konkatenert.

NCPEV-219 består av ei lang rekkje ulike *features*, inklusive 81 *features* som er tekne som gjennomsnittet av fire delvektorar frå Markov-243. Fridrich er òg nært beslekta med PEV-219, men slår saman *features* for å få lågare dimensjonalitet.

Me merker oss, som òg [14] gjorde, at kalibreringa har lite å seia for steganalyse av F5. PEV-219 er dårlegare enn NCPEV-219, og MPEV-219 er berre marginalt betre. Prinsippet for kalibrering har likevel noko for seg. CP-27 som berre hev 51.15% *accuracy*, får 61% når me tek differansen mot det kalibrerte biletet, og 61.9% ved konkatering.

Den sjette *feature*-vektoren bruker òg data frå JPEG-matrisa. Det er ikkje overraskande at ein får betre *features* ved å studera det same domenet som er brukt til innkoding, og det er eigentleg overraskande at SPAM oppnår 65% *accuracy* med *features* som er tekne frå pixmap-biletet. Dei fleste av vektorane som gjer det dårleg dreg *features* anten frå pixmap-biletet eller frå ein *wavelet*-representasjon. Unntaket er CP-27 som er utvikla for JPEG, men diverre let CP-27 berre til å vera effektiv mot dobbelkomprimering.

Evaluering og optimering av *features* til steganalyse

Me har testa *feature selection* med dei tre nemnde metodane BFS, JMI og FOU. Seleksjonen er gjort basert på 1000 tilfeldig valde bilete frå UCID-basen [19] (JPEG/QF75 og F5 som før), medan trening og test av klassifikatoren er gjort på dei same 5000 bileta som før. Grunnen til at me har brukt to ulike biletbasar er at me byrja med UCID som berre har 1338 bilete totalt, og måtte byta til ein større base for å få tilrekkeleg statistisk signifikans i testane. Pga. køyretida på utrekningane har me ikkje teke oss tid til å gjera seleksjonen opp att basert på det nye treningssettet.

Nøyaktigheita er vist i figur 2. Som me ser er det liten skilnad mellom JMI og FOU. BFS er svært god til å identifisera svært korte vektorar. Dei to fyrste *features* er dei same som frå JMI og BFS, men me får særs gode resultat for 4–8 *features*. Etter det vert nøyaktigheita samanliknbar med JMI, og stagnerer når me går forbi kring 40 *features*. Dette er truleg eit resultat av overtrening, som vert eit problem raskare med den lineære klassifikatoren i BFS enn med SVM.

I seleksjonen er det to grupper med *features* som dominerer: ulike variantar av features basert på Markov-kjeder og *cooccurrence features* frå PEV-219. I tabell 2 ser me at

	QF75		QF85	
Namn	F5	Jsteg	F5	Innhald
HGS-178	89.450	99.725	81.85	HGS-142 + MPEV/L5
HGS-180	89.0	99.725	82.45	HGS-178 + MPEV/Var
HGS-162	88.800	99.975	82.375	HGS-160 + MPEV/Var
HGS-160	88.600	99.975	82.85	HGS-142 + NCPEV/L5
HGS-142	88.575	99.925	80.65	HGS-131 + PEV/global
HGS-212	88.300	99.95	81.575	MPM81 + CCCM50
HGS-131	87.900	99.9	80.875	NCPM81 + CCCM50
NCPM-81	80.725	97.9	71.45	
CCCM-50	74.800	99.95	67.125	
DCCM-25	69.075	98.575	66.35	
NCCM-25	68.550	99.975	62.225	

Tabell 2: Samanlikning av nye *feature vectors* på BOWS-datasettet.

me får korte og brukbare *feature vectors* ved å bruka berre desse to *feature*-gruppene. NCPM-81 er dei *features* frå PEV-219 som er baserte på Markov-kjeder. NCCM-25 er *cooccurence features* frå NCPEV-219, medan DCCM-25 og CCCM-50 er tilsvarande frå PEV-219 og MPEV-438. Desse *features* er likevel dårlegare enn det som me kan oppnå med automatisert seleksjon (figur 2) og same dimensjon.

Cooccurrence features og Markov-kjede-*features* er utrekna på svært forskjellig vis, og det er rimeleg å tru at dei vil vera nær uavhengige. HGS-131 kombinerer CCCM-50 og NCPM-81, og denne gongen får me betre resultat enn med automatisert seleksjon, og ogso betre enn tidlegare kjende *feature vectors*. Me har ikkje gjort nokon formell hypotesetest for å vurdera om forbetringa er statistisk signifikant, men me kan merka oss at med resultata i tabellen ville me ha konkludert med ei forbetring ved eit signifikansnivå på 10%.

Tabell 2 viser fleire andre forbetra *feature vectors* som byggjer på HGS-131. Me har testa ei rekkje variantar som det ikkje er rom for å drøfta her. Dei som er inkluderte i tabellen er eit utval av dei beste for kvart datasett. Dei *features* som er brukte er:

PEV/global det globale histogrammet (11 *features*) frå PEV-219.

NCPEV/L5 båsane for ± 5 frå dei frekvensvise histogramma frå NCPEV-219 (18 *features*).

MPEV/L5 som over, men frå MPEV-438 (36 *features*).

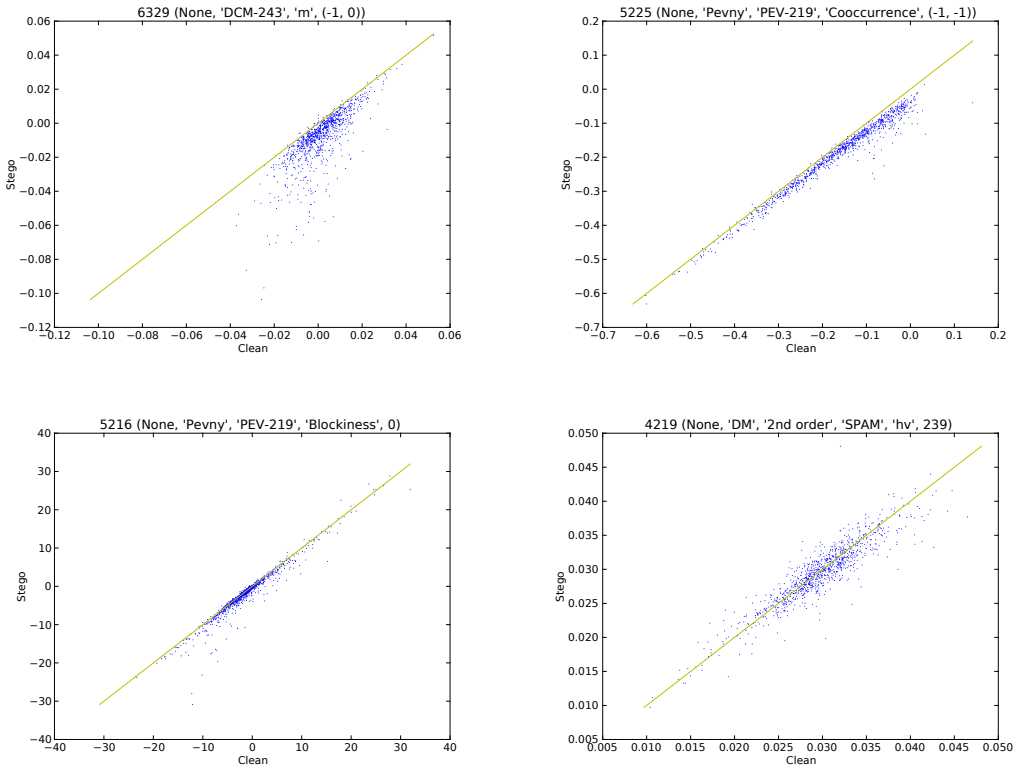
MPEV/Var to *variation features* frå MPEV-438

Som me ser er det fleire *feature vectors* som gjev signifikante forbetringar mot F5 (både QF75 og QF85) i forhold til HGS-131, utan at finst eitt som er optimalt for alle datasett.

Ubrukelege *features*

Me har òg sett nærare på ein del dårlege *features* for å sjå om dei kan eliminerast absolutt. Til dette har me brukt spreideplott der me plottar *feature*-verdien til eit dekkbilete mot eit steganogram med F5 frå same dekkbilete. Fire døme er vist i figur 3.

Dei to øverste plotta viser to av dei aller beste *features* i *feature selection*-forsøket. Dei fleste punkta ligg under diagonalen, som viser at denne *feature* normalt er lågare for



Figur 3: Spreideplott som samanliknar steganogram med dekkbiletet mot utvalde *features*, basert på UCID-datasettet.

steganogrammet enn for dekkbiletet. *Blockiness* (nede til vinstre) er ein *feature* som fekk dårleg heuristikk i *feature selection*-forsøket. Her ser med at punkta stort sett ligg tett på diagonalen, og *blockiness* vert dermed absolutt ubrukeleg for langt dei fleste bilete.

Det siste er ein *feature* som såg middels god ut i *feature selection*-forsøket. Han kjem frå SPAM-vektoren som primært er laga for pixmap-bilete. Her ser me ingen konsekvent forskjell mellom dekkbilete og steganogram, men punkt spreider seg i ein ganske vid sky omkring diagonalen. Det tyder at *feature*n er dårleg som klassifikator i seg sjølv, men kan verka godt i samband med andre.

Blockiness vart opprinneleg brukt som einaste *feature* i eit steganalyse-system, og verka svært bra mot Jsteg og OutGuess. Det er difor interessant at me kan utelukka *blockiness* fullstendig for analyse av F5. Det er mogleg å finna fleire *features* som kan utelukkast, men ein fullstendig gjennomgang kan ikkje gjerast manuelt.

4 Oppsummering

Den største utfordringa innanfor steganalyse er det store talet på variablar og ulike scenario. Sannsynsfordelinga av dekkbilete, komprimering, stegosystem, meldingslengd, osv. kan potensielt påverka klassifikatortreninga. Det finst ein del, men ikkje mykje, forsking på korleis ulike faktorar påverka ytinga. Eksperimentell analyse krev svært omfattande datasett. Dei eksperimenta som me legg fram her er mellom dei mest omfattande i litteraturen; spesielt har me testa og samanlikna fleire ulike *feature*-vektorar enn nokon før oss. Ved å publisera kjeldekoden har me òg opna for at andre kan byggja vidare med eksperiment over ulike biletkjelder og fleire stegosystem.

Me har identifisert ein *feature*-vektor som gjev meir nøyaktig steganalyse enn tidlegare system, med ei forbetring 2,8 prosentpoeng på F5 ved QF75. Dette er statistisk signifikant, men me må òg konkludera med at NCPEV-219 er ein svært god *feature*-vektor som er vanskeleg å forbetra. Kjende *feature selection*-teknikkar er eit godt hjelpemiddel for å finna gode *features*, men for å finna forbetringar tarv ein supplera med manuelle teknikkar og prøving og feiling. Dette stemmer godt med påstandar i [14] som sa at i deira røynsler hev *feature selection* lite for seg.

Denne artikkelen hev berre skrapt i overflata av problemet, men grunnlaget for å gå vidare ligg i den publiserte programvaren [18]. Totalt sett omfattar detter drøyt 13 000 liner kode, fordelt på steganalyse, *feature extraction*, *feature selection*, støtteverkty til SVM, test-suitar, osv. Dette gjer det relativt lett å køyra og automatisera gjentekne testar og samanlikningar. Der tidlegare implementasjonar er allment tilgjengeleg, er dei vanskelegare å integrera og automatisera.

Vidare forsking må sjå på andre kjende stegosystem for JPEG, for å sjå kor godt resultata generaliserer. Vidare utvikling av nye stegosystem må fokusera på dei *features* som er identifiserte som gode, og syta for at dei ikkje vert påverka av løyndmeldinga. Eit anna ope problem er korleis teknikkane vert påverka av ulike meldingskjelder. Me har brukt bilete av god kvalitet; bilete som har vore nedskalert eller komprimert fleire gongar vil ha heilt andre eigenskapar. Til slutt kan ein gjera tilsvarande analyse på pixmap-bilete.

I tillegg til opne spørsmål innanfor steganalyse, har me òg reist eitt innanfor rein maskinlæring. Bruken av kontinuerlig entropi ved *feature selection* er ny, og me tarv no gjera ein samanlikning mellom kontinuerlig og diskret entropi for ulike datasett.

Referansar

[1] I. Ahmad and Pi-Erh Lin. A nonparametric estimation of the entropy for absolutely continuous distributions. *IEEE Transactions on Information Theory*, 22:372–375, 1976.

[2] Ismail Avcibaş, Mehdi Kharrazi, Nasir Memon, and Bulent Sankur. Image steganalysis with binary similarity measures. *EURASIP Journal on Applied Signal Processing*, 17:2749—2757, 2005.

[3] Johann Briffa, Anthony TS Ho, Hans Georg Schaathun, and Ainuddin Wahid Abdul Wahab. Conditional probability based steganalysis for JPEG steganography. In *International Conference on Signal Processing Systems (ICSPS 2009)*, May 2009.

[4] Gavin Brown. A new perspective for information theoretic feature selection. In *Twelfth International Conference on Artificial Intelligence and Statistics*, June 2009. Florida.

[5] G. Cancelli, G. Doërr, I. Cox, and M. Barni. Detection of ± 1 steganography based on the amplitude of histogram local extrema. In *Proceedings IEEE, International Conference on Image Processing (ICIP)*, October 2008.

[6] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] Chunhua Chen, Yun Q. Shi, Wen Chen, and Guorong Xuan. Statistical moments based universal steganalysis using JPEG 2-d array and 2-d characteristic function. In *Proceedings of the International Conference on Image Processing, ICIP 2006, Atlanta, Georgia, USA*, pages 105–108, October 2006.

[8] Xiaochuan Chen, Yunhong Wang, Tieniu Tan, and Lei Guo1. Blind image steganalysis based on statistical analysis of empirical matrix. In *Proc. 18th Int. Conf. on Pattern Recognition*, volume 3, pages 1107–1110, 2006.

[9] Jing Dong, Xiaochuan Chen, Lei Guo, and Tieniu Tan. Fusion based blind image steganalysis by boosting feature selection. In *IWDW '07: Proceedings of the 6th International Workshop on Digital Watermarking*, pages 87–98, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] Tomas Filler, Tomas Pevny, and Patrick Bas. Break our steganographic system, 2008. Includes a 10000-image database intended for Information Hiding experimentation.

[11] Jessica Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81. Springer Berlin / Heidelberg, 2005.

[12] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In E. J. Delp and P. W. Wong, editors, *Proc. SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 1–13, January 2006.

[13] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

[14] Jan Kodovský and Jessica Fridrich. Calibration revisited. In *MM&Sec '09: Proceedings of the 11th ACM workshop on Multimedia and security*, pages 63–74, New York, NY, USA, 2009. ACM.

[15] Siwei Lyu and Hany Farid. Detecting hidden messages using higher-order statistics and support vector machines. In *IH '02: Revised Papers from the 5th International Workshop on Information Hiding*, Lecture Notes in Computer Science, pages 340–354, London, UK, 2003. Springer-Verlag.

[16] Tomáš Pevný, Patrick Bas, and Jessica Fridrich. Steganalysis by subtractive pixel adjacency matrix. In *MM&Sec '09: Proceedings of the 11th ACM workshop on Multimedia and security*, pages 75–84, New York, NY, USA, 2009. ACM.

[17] Tomáš Pevný and Jessica Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In *Proc. SPIE Electronic Imaging*, pages 3–4, January 2007.

[18] Hans Georg Schaathun. pysteg – a python library for steganography and steganalysis, 2011. `http://www.ifs.schaathun.net/pysteg/`.

[19] G. Schaefer and M. Stich. UCID - an uncompressed colour image database. In *Proc. SPIE, Storage and Retrieval Methods and Applications for Multimedia 2004*, pages 472–480, 2004.

[20] Yun Q. Shi, Chunhua Chen, and Wen Chen. A Markov process based approach to effective attacking JPEG steganography. In *Information Hiding*, Lecture Notes in Computer Science, 2006.

[21] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In *CRYPTO*, pages 51–67, 1983.

[22] Kenneth Sullivan, Upamanyu Madhow, Shivkumar Chandrasekaran, and B. S. Manjunath. Steganalysis of spread spectrum data hiding exploiting cover memory. In Edward J. Delp and Ping Wah Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents*, volume 5681 of *Proceedings of SPIE*, pages 38–46. SPIE, 2005.

[23] Kinh Tieu and Paul Viola. Boosting image retrieval. *Int. J. Comput. Vision*, 56(1-2):17–36, 2004.

[24] Andreas Westfeld. F5—a steganographic algorithm. In *IHW '01: Proceedings of the 4th International Workshop on Information Hiding*, pages 289–302, London, UK, 2001. Springer-Verlag.

[25] Guorong Xuan, Jianjiong Gao, Y.Q. Shi, and D. Zou. Image steganalysis based on statistical moments of wavelet subband histograms in DFT domain. In *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, pages 1–4, 30 2005-Nov. 2 2005.

[26] Arezoo Yadollahpour and Hossein Miar Naimi. Attack on LSB steganography in color and grayscale images using autocorrelation coefficients. *European Journal of Scientific Research*, 31(2):172–183, 2009.