Proofs

for the purpose of these proofs, our user prediction is p

, the realized market returns are y

, the meta-model prediction is m

, our optimizer performs O(x) = argmax_x[y.T x – C(x)] = r. SWG[p, fobj(y,m)] is the stake-weighted gradient operator, and norm(x) = sum(x**2)**0.5

Lemma 0.1: SWG[p, fobj(y,m)] = (p – m).T grad_m[fobj(y,m)]

by definition:

SWG[p, fobj(y,m)] = sum(stakes) * grad_stakes[fobj(y, sum(stakes*preds)/sum(stakes))] [p].T

= sum(stakes)*grad_m[fobj(y, m)]grad_stakes[sum(stakes

preds)/sum(stakes)][p].T

= sum(stakes)grad_m[fobj(y, m)]

[preds/sum(stakes) – sum(stakes*preds)/sum(stakes)**2][p].T

= grad_m[fobj(y, m)][preds – sum(stakes

preds)/sum(stakes)][p].T

= grad_m[fobj(y, m)].T*(p – m)

Lemma 0.2:

grad_r[C(r)] = m

By definition r = argmax_x[m.T x – C(x)]

At a maximum the gradient is zero therefore:

grad_r[m.T r – C(r)] = 0

grad_r[m.T r] = grad_r[C(r)]

m = grad_r[C(r)]

Lemma 0.3:

Jacobian_m[r] = H^-1

m = grad_r[C(r)]

(Lemma 0.2)

Jacobian_r[m] = Jacobian_r[grad_r[C(r)]]

Jacobian_r[m] = Hessian_r[C(r)]

Jacobian_r[m] = H

Jacobian_m[r] = H^-1

Lemma 0.4: grad_x[f(x/norm(x))]

= 1/norm(x) (grad_x[f(x/norm(x))] – (x.T grad_x[f(x/norm(x))]) / (x.T x) x)

Proof 1: SWG[p, {fobj(y,m): y.T O(m) - C(O(m))}] = (p - m).T H^-1 (y – m)

Lemma 1.1 : grad_m[y.T O(m) - C(O(m))] = (p - m).T H^-1 (y – m)

grad_m[y.T O(m) - C(O(m))] = grad_m[-y.T r(m) + C(r(m))]

`=

- Jacobian_m[r] y + grad_m[C(r(m))]`

`=

- Jacobian_m[r] y + Jacobian_m[r] grad_r[C(r(m))]`

`=

- H^-1 y + H^-1 grad_r[C(r(m))]` (Lemma 0.3)

`=

- H^-1 y + H^-1 m ` (Lemma 0.2)

= H^-1(m – y)

Proof:

SWG[p, {fobj(y,m): y.T O(m) - C(O(m))}] =

(p – m).T grad_m[y.T O(m) - C(O(m))]` (Lemma 0.1)

(p – m).T H^-1 (y – m)

(Lemma 1.1)

QED

Proof 2: SWG[p, {fobj(y,m): y.T O(m/norm(m)))}] = (p - m).T (H^-1 y – (m.T H^-1 y)/(m.T m) m)

Lemma 2.1: grad_m[y.T O(m)] = H^-1 y

grad_m[y.T O(m)] = grad_m[-y.T r(m)]

`=

- Jacobian_m[r] y `

`=

- Jacobian_m[r] y `

`=

- H^-1 y ` (Lemma 0.3)

`=

- H^-1 y `

= H^-1 y

`Lemma

2.2: grad_p[y.T O(m/norm(m))] = (H^-1 y – (m.T H^-1 y)/(m.T m)

m)``

grad_p[y.T O(m/norm(m))] = (H^-1 y – (m.T H^-1 y)/(m.T m) m)

(Lemma

1.1 and Lemma 0.4)`

Proof:

SWG[p, {fobj(y,m): y.T O(m/norm(m)))}] =

(p – m).T grad_m[y.T O(m) - C(O(m))]`

`SWG[p, {fobj(y,m): y.T O(m/norm(m)))}]

=(p – m).T H^-1

(

H^-1 y – (m.T H^-1 y)/(m.T m) m)`` (Lemma 0.1 and Lemma 2.2)